Karin Anna Hummel
James P.G. Sterbenz (Eds.)

Self-Organizing Systems

Third International Workshop, IWSOS 2008
Vienna, Austria, December 10-12, 2008
Proceedings

Springer

Volume Editors

Karin Anna Hummel
University of Vienna
Department of Distributed and Multimedia Systems
Lenaugasse 2/8, 1080 Vienna, Austria
E-mail: karin.hummel@univie.ac.at

James P.G. Sterbenz
The University of Kansas
Information and Telecommunication Technology Center
Lawrence, Kansas 66045-7612, USA
E-mail:jpgs@ittc.ku.edu
and
Lancaster University
InfoLab21, Lancaster, LA1 4WA, United Kingdom
E-mail: jpgs@comp.lancs.ac.uk

Library of Congress Control Number: 2008940838

CR Subject Classification (1998): C.2, D.4.4, C.4, H.3, H.2.8

LNCS Sublibrary: SL 5 – Computer Communication Networks
and Telecommunications

ISSN 0302-9743
ISBN-10 3-540-92156-7 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-92156-1 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12589364 06/3180 5 4 3 2 1 0

Preface

We welcome you to the proceedings of the Third International Workshop on Self-Organizing Systems (IWSOS 2008) hosted at the University of Vienna, Austria. IWSOS provides an annual forum to present and discuss recent research in self-organization focused on networks and networked systems. Research in self-organizing networked systems has advanced in recent years, but the investigation of its potentials and limits still leaves challenging and appealing open research issues for this and subsequent IWSOS workshops.

Complex and heterogeneous networks make self-organization highly desirable. Benefits envisioned by self-organization are the inherent robustness and adaptability to new dynamic traffic, topology changes, and scaling of networks. In addition to an increasingly complex Global Internet, a number of domain-specific subnetworks benefit from advances in self-organization, including wireless mesh networks, wireless sensor networks, and mobile ad-hoc networks, e.g., vehicular ad-hoc networks. Self-organization in networked systems is often inspired by other domains, such as nature (evolution theory, swarm intelligence), sociology (human cooperation), and economics (game theory). Aspects of controllability, engineering, testing, and monitoring of self-organizing networks remain challenging and are of particular interest to IWSOS.

This year, we received 70 full paper and 24 short paper submissions from authors of 33 different countries. This strong interest in the workshop is very encouraging for research in self-organizing systems and allowed us to provide a strong technical program. Based on the recommendations of the Technical Program Committee and external expert reviewers, we accepted 20 full papers from the full paper submissions and invited 9 as short papers. Of the 24 short paper submissions we accepted 4 for presentation for a total of 13 short papers. Most full papers were reviewed by four experts, and all papers received at least three reviews. A number of papers were shepherded toward publication by the Technical Program Committee and external expert reviewers.

Our technical program consisted of sessions on Peer-to-Peer Systems (4 papers), Overlay Networks (3 papers), Resource and Service Management (4 papers), Theory and General Approaches (3 papers), Wireless Sensor Networks (3 papers), and Fault Detection, Resilience, and Self-Healing (3 papers). Additionally, there were two short paper sessions: Short Papers – Networking Topics (7 papers) and Short Papers – Theory, General and Distributed System Topics (6 papers).

To complement the technical program we invited a discussion paper on open issues of self-organizing systems, which led to a panel discussion on the future application of self-organization to large, complex, robust, and resilient networks. We were delighted to have two keynote addresses: Martha Steenstrup discussed the history, state-of-the-art, and future prospects of self-organizing networks. Kurt Tutschku focused on how network virtualization is facilitated by self-organization and how self-organization can be applied to future virtualized networks.

We are grateful to all Technical Program Committee members and additional reviewers who provided thorough reviews that made the selection of the papers possible. For their mentoring effort, we want to express our particular thanks to Paul Smith, Amine Houyou, Helmut Hlavacs, Mikhail Smirnov, Simon Dobson, Marco Mamei, Christian Bettstetter, Majid I. Khan, Güneş Erçal-Özkaya, and Matthias Hollick. Special thanks go to our IWSOS 2008 General Chair, Helmut Hlavacs, for his outstanding support in all the phases of the workshop organization. Many thanks go to Eugen Mühlvenzl and his team from the Austrian Computer Society for their support in the organization of the workshop. Additionally, without the support of our Viennese Organizing Committee, the workshop would not have been possible. Particular thanks go to Shelley Buchinger for her contributions to the website and invaluable support during paper submission, Harald Meyer for his tireless support in announcing different calls for the workshop, Andrea Hess for her precise technical editing support of the proceedings, and Alexander Adrowitzer for coordinating the side-program of the workshop. Finally, we want to thank the numerous authors for their submissions and contributions to the technical program.

December 2008

Karin Anna Hummel
James P.G. Sterbenz

Organization

IWSOS 2008, the Third International Workshop on Self-Organizing Systems, was organized by the Department of Distributed and Multimedia Systems, University of Vienna, Austria in cooperation with the Austrian Computer Society in Vienna, December 10–12, 2008.

Steering Committee

David Hutchison	Lancaster University, UK
Hermann de Meer	University of Passau, Germany
Randy Katz	UC Berkeley, USA
Bernhard Plattner	ETH Zürich, Switzerland
James P.G. Sterbenz	The University of Kansas, USA and Lancaster University, UK
Georg Carle	TU München, Germany (IFIP TC6 Representative)

General Chair

Helmut Hlavacs	University of Vienna, Austria

Technical Program Co-Chairs

Karin Anna Hummel	University of Vienna, Austria
James P.G. Sterbenz	The University of Kansas, USA and Lancaster University, UK

Technical Program Committee

Marin Bertier	IRISA/INSA-Rennes, France
Sandford Bessler	Forschungszentrum Telekommunikation Wien, Austria
Christian Bettstetter	University of Klagenfurt, Austria
Ernst Biersack	Institute Eurecom, France
Georg Carle	TU München, Germany
Taric Cicic	University of Oslo, Norway
Alexander Clemm	Cisco Systems, USA
Costas Courcoubetis	AUEB, Greece
Simon Dobson	University College Dublin, Ireland

Wilfried Elmenreich	University of Klagenfurt, Austria
Stefan Fischer	University of Lübeck, Germany
Michael Fry	University of Sydney, Australia
Indranil Gupta	University of Illinois at Urbana–Champaign, USA
Hannes Hartenstein	University of Karlsruhe, Germany
Manfred Hauswirth	National University of Ireland, Ireland
Joseph L. Hellerstein	Microsoft Developer Division, USA
Matthias Hollick	Technical University of Darmstadt, Germany
Amine Houyou	University of Passau, Germany
Majid I. Khan	University of Vienna, Austria
Alexander Keller	IBM Global Technology Services, USA
Wolfgang Kellerer	DoCoMo Lab Europe, Germany
Alexander V. Konstantinou	IBM T.J. Watson Research Center, USA
Rajesh Krishnan	Scientific Systems Company, Inc., USA
Guy Leduc	University of Liege, Belgium
Baochun Li	University of Toronto, Canada
Marco Mamei	University of Modena and Reggio Emilia, Italy
Andreas Mauthe	Lancaster University, UK
Paul Mueller	Kaiserslautern University, Germany
Masayuki Murata	Osaka University, Japan
Ben Paechter	Napier University, UK
Manish Parashar	Rutgers University, USA
Dimitrios Pezaros	Lancaster University, UK
Christian Prehofer	Nokia Research, Finland
Lukas Ruf	Consecom AG, Switzerland
Susana Sargento	University of Aveiro, Portugal
Marcus Schoeller	NEC Laboratories Europe, Germany
Caterina Maria Scoglio	Kansas State University, USA
Mikhail Smirnov	Fraunhofer Fokus, Germany
Paul Smith	Lancaster University, UK
Thrasyvoulos Spyropoulos	ETH Zürich, Switzerland
Dirk Staehle	Würzburg University, Germany
Burkhard Stiller	University of Zürich, Switzerland
John Strassner	Motorola Labs, USA
Zhili Sun	University of Surrey, UK
Kurt Tutschku	University of Vienna, Austria
Patrick Wüchner	University of Passau, Germany

Local Organizing Committee

Shelley Buchinger	University of Vienna, Austria
Alexander Adrowitzer	University of Vienna, Austria
Harald Meyer	University of Vienna, Austria
Andrea Hess	University of Vienna, Austria

Reviewers

Alexander Adrowitzer
Panayotis Antoniadis
Remi Badonnel
Tobias Bandh
Marin Bertier
Sandford Bessler
Christian Bettstetter
Ernst Biersack
Guenther Brandner
Carsten Buschmann
Egemen Çetinkaya
Tarik Cicic
Alexander Clemm
Costas Courcoubetis
Bart Craenen
Jochen Dinger
Simon Dobson
Ping Du
Wilfried Elmenreich
Güneş Erçal-Özkaya
Markus Fiedler
Andreas Fischer
Stefan Fischer
Marc Fouquet
Michael Fry
Thomas Galla
Wilfried Gansterer
Indranil Gupta
Hannes Hartenstein
Manfred Hauswirth
Joseph Hellerstein
Robert Henjes
Helmut Hlavacs
Matthias Hollick
Ralph Holz
Richard Holzer
Amine Houyou
Karin Anna Hummel
Abdul Jabbar
Costas Kalogiros
Alexander Keller
Wolfgang Kellerer
Majid I. Khan
Moritz Killat
Alexander Konstantinou
Rajesh Krishnan
Guy Leduc
Baochun Li
Xiaodong Liu
Marco Mamei
Andreas Mauthe
Eduard Mehofer
Harald Meyer
Geyong Min
Gary J. Minden
Mu Mu
Paul Mueller
Masayuki Murata
Heiko Niedermayer
Michael Nussbaumer
Ben Paechter
Marc-Oliver Pahl
Thanasis Papaioannou
Manish Parashar
Dimitrios Pezaros
Jean-Marc Pierson
Christian Prehofer
Rastin Pries
Daniel Prince
Andres Quiroz
Justin Rohrer
Lukas Ruf
Susana Sargento
Petri Savolainen
Udo Schilcher
Felix Schmidt-Eisenlohr
Marcus Schöller
Caterina Maria Scoglio
Steven Simpson
Mikhail Smirnov
Paul Smith
Sergios Soursos
Thrasyvoulos Spyropoulos
Dirk Staehle
James P.G. Sterbenz
Burkhard Stiller
John Strassner
Zhili Sun

Sponsors and Technical Sponsors

University of Vienna
Austrian Computer Society
Telekom Austria
Euro-NF
IEEE Communications Society
Lakeside Labs
IFIP TC 6

Table of Contents

Self-Organizing Networked Systems for Technical Applications: A Discussion on Open Issues

Wilfried Elmenreich[1] and Hermann de Meer[2]

[1] Lakeside Labs, Mobile Systems Group
Institute of Networked and Embedded Systems
University of Klagenfurt, Austria
wilfried.elmenreich@uni-klu.ac.at

[2] Faculty of Informatics and Mathematics
Chair of Computer Networks and Communications
University of Passau, Germany
demeer@uni-passau.de

Abstract. The concept of self-organization has been examined oftentimes for several domains such as physics, chemistry, mathematics, etc. However, the current technical development opens a new field of self-organizing applications by creating systems of networked and massively distributed hardware with self-organized control. Having this view in mind, this papers reviews the questions: *What is a self-organizing system?*, *What is it not?*, *Should there be a separate field of science for self-organizing systems?*, and *What are possible approaches to engineer a self-organizing control system?*.

The presented ideas have been elaborated at the *Lakeside Research Days'08* (University of Klagenfurt, Austria), a workshop that featured guided discussions between invited experts working in the field of self-organizing systems.

1 Introduction

The idea of Self-Organizing Systems (SOSs), although long known from domains such as physics, chemistry, and biology, has recently gained interest to be applied to technical applications. The reason for this is a paradigm shift from monolithic systems or systems with a small number of components to large networked systems. This paradigm shift is driven by the technological advancement and the emergence of pervasive systems integrating information processing into everyday objects and activities. For example, a fieldbus network with accurate but expensive sensors interconnected by a dependable wired communication system might be replaced by a system of hundreds of small, but inexpensive sensors using a wireless ad-hoc network to interconnect. Such a cyber-physical system [1] can use the view of multiple sensors to come to a massively distributed view of a technical process, where the fusion of several sensor measurements potentially leads to a more extensive, more accurate, and more robust observation. However, such an approach requires a control paradigm that copes with the complexity of such a solution. A promising approach to attack this problem is the principle of self-organization, where the control is as well decentralized as the controlled system. Through the definition of the behavior in local interactions, it is expected that the overall system shows emergent behavior such as complex order or properties like robustness, adaptability and scalability.

Designing and controlling an SOS can be very demanding. There is no general methodology yet explaining how to design an SOS and in many cases it is very difficult to provide a concise validation of the system. In order to identify the current problems and to search for possible directions for a solution, the Lakeside Labs at the University of Klagenfurt, a research center focusing on networked self-organizing systems, arranged a one-week event called *Lakeside Research Days'08* by inviting international researchers working in the area of SOSs to discuss the topic of SOSs and its open problems. It is the purpose of this paper to summarize the main results of the workshop in order to give researchers an idea of the open problems and the potential for future research.

The rest of the paper is organized as follows: Section 2 briefly reviews the results of a discussion on the definition of SOSs followed by identifying several misconceptions on the understanding of SOSs. Section 3 approaches the question if SOSs should become a separate field of science having its own experts, terminology, and nomenclature. The question how an SOS can be designed is addressed in Section 4 by sketching three different types of design approaches. Section 5 concludes the paper.

2 Definition and Misconceptions of Self-Organizing Systems

In order to communicate problems, methods and results, it is necessary to have a common understanding of the terms used in scientific communication. The term "self-organization" is used by many researchers, but it has no generally accepted meaning. Instead, there exist a number of definitions from different domains such as from cybernetics [2,3], mathematics [4], information theory [5], etc. Gershenson and Heylighen [6] argument to evade the debate about an exact definition of SOSs and to regard self-organization merely as a *way* of observing systems. Depending on the system type and its applications, understanding a system as being self-organized can be more or less helpful.

Therefore, instead of trying to find a single concise definition of SOSs we worked out a brief sentence explaining the main idea of SOSs that assists in communicating the main idea. Followed by this, we elaborated on the common misconceptions in the definition of SOS. Among several proposals, the following sentence, which was an outcome of the discussions at the Research Days [7] showed an interesting attempt to sketch the concept of SOSs in a nutshell:

A self-organizing system (SOS) consists of a set of entities that obtains an emerging global system behavior via local interactions without centralized control.

In the following, we address the question of common misconceptions in the understanding of SOSs.

2.1 Misconception #1: Self-Organizing Systems Establish a Class of Systems

If a system is considered to be self-organizing or not depends mainly on the way how the system is observed, especially where the borderline between the observed system and its environment is drawn. Gershenson proposes the following perspective to overcome this problem: Instead of thinking of SOSs as an absolute class of systems, self-organization should be understood as a way of observing systems [6]. Depending on the type of problem and the desired solution, the way of observing a system as an SOS can be beneficial or not.

2.2 Misconception #2: Self-Organizing Systems Are Chaotic Systems

There is a relation between chaos theory and self-organization in that a SOS may show chaotic behavior, that is having critical turning points (also known as bifurcations) in the system behavior [8]. However, an SOS does not necessarily have to show such behavior. Instead, some SOSs also might approach their target state without a sensitive dependence on initial conditions. Accordingly, a system with chaotic behavior may be built without employing the typical building blocks of SOSs such as distributed entities and local interactions.

2.3 Misconception #3: The Emerging Structure Is a Primary Property of Self-Organizing Systems

SOSs provide a powerful mechanism to create structure and patterns. This phenomenon can be observed in many physical and biological systems, such as the skin pigmentation of fish, the polygonal pattern of nest territories of fish such as *Tilapia*, or the cathedral-like buildings of termites [9].

However, the emerging pattern should not be seen as a primary property of an SOS. There are SOSs, like homeostatic operational control in living beings, where such a structure is not present or is hidden from the observer. Thus, the emerging structure can be rather seen as a secondary property that indicate self-organization in many cases.

2.4 Misconception #4: Self-Organizing Systems Are Always Based on Evolutionary Processes

Evolutionary processes, as best known from biological examples, are an iterative mechanism of change in the inherited traits of a population of organisms from one generation to the next [10]. Evolutionary processes are driven by mutation, selection and recombination.

Many biological examples of self-organizing systems have emerged from an evolutionary process, which made the term self-organization connected to evolution. Thus, the connection of SOSs to evolutionary processes is not an obligatory one, since many non-biological examples of SOSs have developed without an evolutionary process, thus showing the possibility to design self-organizing without an evolutionary process. However, an interesting research task for future technical systems arises in constructing SOSs, which implement an evolution of their local rules in order to adjust to new situations.

2.5 Misconception #5: A self-Organizing System Never Needs Maintenance

Many SOSs show adaptive behavior, which means that they can operate well within a wide range of input parameters. However, that does not imply that a technical SOS will have a low maintenance effort. Typically, a complex technical system that must operate over a considerable life time will require maintenance in order to provide its service during system lifetime. It is an open question if maintenance of a technical system with self-organizing properties will be easier or more complicated to maintain than a traditionally designed technical application. On the one hand, properties like robustness might make it easier to replace parts of the system without disturbing the overall operation, on the other hand, diagnosis and maintenance of an SOS might turn out to be more complex than in systems built following a more straightforward approach.

3 Self-Organizing Systems Forming an Own Field of Science?

A science field is characterized as a category of specialized expertise within science, often also having its own terminology and nomenclature. In the example of SOSs we cannot speak of a separate field of science today.

As sketched in Figure 1, SOSs are found in multiple disciplines, being thus a highly interdisciplinary field. This does not necessarily hinder the formation of an own field of science, as it is shown by the also highly interdisciplinary research on Artificial Intelligence, which is regarded as a separate field of science. A key question is, if the research on SOSs is likely to converge in a separate field of science with its own experts, terminology, and nomenclature. The other option is that research on SOSs will rather be covered by researches coming from one of the related fields.

The answer of the question is of great interest, since it will influence future decisions on installing academic curricula for the “field” of SOSs. We do not claim that we can give an absolute answer to this question, but we can formulate two subquestions to this issue:

Is there a common vision within the field?

For example, Artificial Intelligence (AI), also consisting of many different methods and viewpoints, came at least with a common vision as it was presented

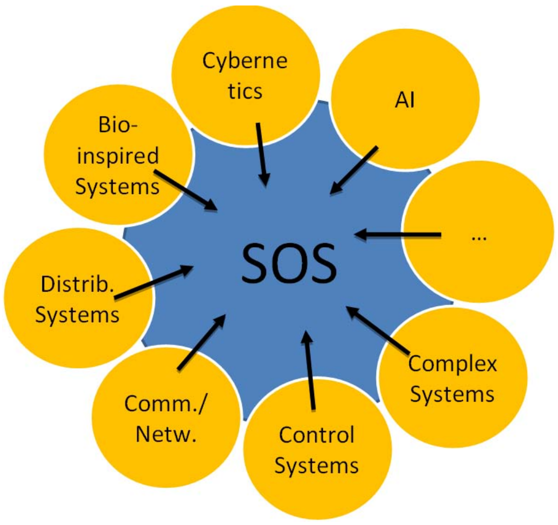


Fig. 1. Interdisciplinary role of self-organizing systems

in the proposal for the Dartmouth Summer Research Project by McCarthy, Minsky, Rochester and Shannon in 1955 [11].

Are there fundamental research questions that would require investigation from dedicated SOSs researchers?

If not, trying to force the creation of a new field with its own terminology and nomenclature would be an effort that can even worsen the communication between other domains since the problem is not discussed in the domain-specific language.

4 Design Approaches for Self-Organizing Systems

In [12], Prehofer and Bettstetter identify the task of finding *local behavior rules that achieve global properties* as a major paradigm to be approached. In the following, we will elaborate on design approaches for solving this problem.

4.1 Bio-inspired Design

In nature, there are several examples of self-organizing behavior, for example, ants cooperatively finding shortest routes to food sources, termites building complex constructions without using a blueprint, fish schools organizing themselves without a leader, and swarms of fireflies in south-east Asia synchronously emitting light flashes.

There are two main paradigms of bio-inspired design: the direct and the indirect approach. In the direct (or top-down) approach, a technical problem is tackled by looking for natural examples solving an equivalent problem. The biological solution and its principles are then analyzed and re-built in a technical application. Examples of the direct approach are the design of aeroplane wings by observing the gliding flight of birds as it was done by Otto Lilienthal in the 19th century, or, after a closer analysis of the up-bent feathers at the wingtips of several birds, the refinement of aeroplane wings by turbulence-reducing and thus fuel-saving winglets [13].

In contrast, the indirect (or bottom-up) approach of bio-inspired design involves first the derivation of principles by analyzing natural systems. This step is done in a basic research effort that is not yet targeted at a specific application. The principle is then abstracted from its biological context and used in particular technical applications where they could be suitable. Examples of the indirect approach are the concept of artificial neural networks or the concept of ant foraging behavior being applied to mesh network packet routing.

An important aspect of bio-inspired design are the notable differences between biological and technical solutions: In many biological systems, especially for lower animals, there is no real counterpart to what we call software in technical systems. For example, protozoa have no mechanism to learn and circulate behavior during lifetime. Instead, new behavior is stored via the genes of the next generation. With the same effort, also physical changes of the next generation are possible. Thus, in such an evolutionary approach, the physical body and the physical abilities typically grow as part of the solution, while in traditional engineering the analogy of the body, that is the hardware, has to be usually fixed early in the design process. For example, a hardware revision comes with considerable little effort in biological systems while in a technical project an unscheduled hardware change likely might cause the project to go over budget. On the other hand, when using the biological approach only with the software part being able to change, the result might be less effective than the biological example. For example, a bio-inspired algorithm optimizing only the software for a mobile robot would not play with optimizing the number and placement of the robot's physical sensors while the biological counterparts co-optimize these aspects as well.

On the other hand, evolution is at a disadvantage when it comes to the creation of radically new designs. All living beings follow more or less a general blueprint, in case of a particular class such as mammals, the design is even more restricted. Thus, for example, nature was never able to design a wheeled vehicle-like animal or other rotating machines.

Thus, bio-inspired mechanisms might be inappropriate when (i) the biological solution is too difficult to rebuild with technical means or (ii) a technical solution can be more efficient by taking advantage of mechanisms that cannot be found in the biological paradigm. A mistake to be avoided is to stick to biological solutions just because of their seeming elegance.

These differences between nature's way to build things and engineering make especially a direct design approach very difficult to apply. For designing self-organizing technical systems, the indirect approach seems more promising, since it allows the assignment of biological ideas in a wider context.

4.2 Trial and Error

This approach requires to have a testbed that allows extensive testing without high cost or possible endangering of persons. Usually, such a testbed consists of a simulation of the target system with a model of the environment and the system itself. However, a simulation always implements an abstraction of the real environment, so after the experiments, a real-world validation is required in order to create trust in the derived solution.

The process of trial and error itself can be refined in several ways.

Gershenson [14] introduces the notion of *friction* to describe the fitness of an interaction between two entities. There exists also a friction of the overall system that should be minimized in order to maximize its performance. By identifying and analyzing points of friction, an engineer can change the local rules in order to mitigate the problem. However, this is not a straightforward approach – in several cases a higher friction for a particular entity might be beneficial for the overall system.

Furthermore, existing search algorithms can be applied to the search for an optimal or sufficiently good set of local rules. However, the search space is typically too large for an exhaustive search and its non-monotonic properties can get a non-exhaustive search algorithm to be stuck in local maxima. In this cases, heuristic search algorithms like genetic algorithms and simulated annealing can be a choice.

4.3 Learning from an Omniscient Solution

This approach requires to have an optimal or at least well-performing solution to the problem beforehand. However this solution might be created using features that cannot be realized in the final solution. An example could be the implementation of an omniscient algorithm for a simulation. In many cases, it might not be possible to use this solution for a real application because the perfect information cannot be provided to the algorithm or the algorithm might be too complex to be implemented with reasonable response times. However, the omniscient algorithm can be used as an example for teaching its behavior to distributed entities that use only local information and interactions.

An example for such an approach is given by Auer, Wüchner and De Meer [15] by designing an agent performing well in the prisoner's dilemma [16]. In the design process, an agent having perfect knowledge is created first, then its behavior is analyzed using Causal State Splitting Reconstruction [17], which is basically a method for time series analysis. The results are then used for designing the local rules for a non-omniscient agent. The resulting agent showed to be an improvement of the standard and already well-performing *tit-for-tat* strategy by implementing also forgiveness.

5 Summary and Conclusion

The connection of self-organization to so many disciplines in science is an advantage and a disadvantage at the same time. In terms of definition and terminology, the many definitions from different domains have blurred the overall idea which is definitely a disadvantage. On the other hand, the many disciplines keep the potential for many ideas and new approaches for creating self-organizing control systems. This possibility will be even more attractive, if the research on SOSs can converge towards a more standardized nomenclature, probably even forming a new field of science some day.

Several positive effects from the interdisciplinarity of self-organization became apparent when discussed possible ways to design the behavior of the particular entities. The local behavior is an integral part of an SOS, since the overall behavior of the system emerges from the local interactions of the entities. We have identified three basic approaches for finding a suitable set of local rules, namely bio-inspired design, trial and error, and learning from an omniscient solution. Bio-inspired design can give promising results, however, due to the differences in natural evolution and traditional engineering, it has to be applied carefully. The approach of learning from an omniscient solution might be also interesting for non-game-theoretic settings. However, the approach relies on the possibility that the omniscient solution can be build in a simulation and that the behavior can be mimicked by an agent with local information in a useful way. Trial and error is definitely the most general approach among the three. However, despite of improvements in identifying friction and search algorithms this approach can be too inefficient so that it may not succeed for a large search space. The right design approach (which may be also a combined approach) for a particular project will be defined by the particular constraints and requirements.

We hope that the discussions and suggestions in this paper will be helpful to future research in the area of SOSs. In the future, we plan on elaborating the design process for building SOSs by combining existing engineering approaches with methods especially tailored to SOSs.

Acknowledgments

This work was supported by the Lakeside Research Days'08 under contract No. KWF-20214/16476/23861, taking place from June 30, 2008 to July 4, 2008 at the University of Klagenfurt, Austria. We would like to thank all participants of Lakeside Research Days'08 for the fruitful discussions on self-organization. Furthermore, we would like to thank Patrick Wüchner for constructive comments on the paper.

References

1. Lee, E.A.: Cyber physical systems: Design challenges. Technical Report UCB/EECS-2008-8, EECS Department, University of California, Berkeley (January 2008)

2. von Foerster, H.: On self-organizing systems and their environments. In: Yovitts, M.C., Cameron, S. (eds.) Self-Organizing Systems, pp. 31–50. Pergamon Press, Oxford (1960)
3. von Foerster, H.: Principles of the self-organizing system. In: von Foerster, H., Zopf Jr., G.W. (eds.) Principles of Self-organization, pp. 255–278. Pergamon Press, Oxford (1962)
4. Lendaris, G.G.: On the definition of self-organizing systems. Proceedings of the IEEE 52(3), 324–325 (1964)
5. Haken, H.: Information and Self-Organization – A Macroscopic Approach. Springer, Heidelberg (1988)
6. Gershenson, C., Heylighen, F.: When can we call a system self-organizing? In: Banzhaf, W., Ziegler, J., Christaller, T., Dittrich, P., Kim, J.T. (eds.) ECAL 2003. LNCS (LNAI), vol. 2801, pp. 606–614. Springer, Heidelberg (2003)
7. Elmenreich, W.: Lakeside Research Days 2008. Technical report, Lakeside Labs, Klagenfurt, Austria (June-July 2008)
8. Bloom, S.L.: Chaos, complexity, self-organization and us. Psychotherapy Review 2(8) (August 2000)
9. Camazine, S., Deneubourg, J.-L., Franks, N.R., Sneyd, J., Theraulaz, G., Bonabeau, E.: Self-Organization in Biological Systems. Princeton University Press, Princeton (2001)
10. Wikipedia, the free Encyclopedia. Evolution. Wikimedia Foundation (August 6, 2008)
11. McCarthy, J., Minsky, M.L., Rochester, N., Shannon, C.E.: A proposal for the dartmouth summer research project on artificial intelligence. Technical report, Dartmouth College (1955)
12. Prehofer, C., Bettstetter, C.: Self-organization in communication networks: Principles and design paradigms. IEEE Communications Magazine, 78–85 (July 2005)
13. Faye, R., Laprete, R., Winter, M.: Blended winglets. Aero, Boeing (17) (January 2002)
14. Gershenson, C.: Design and Control of Self-organizing Systems. PhD thesis, Vrije Universiteit Brussel (2007)
15. Auer, C., Wüchner, P., de Meer, H.: A Method to Derive Local Interaction Strategies for Improving Cooperation in Self-Organizing Systems. In: Proceedings of the Third International Workshop on Self-Organizing Systems, Vienna, Austria (December 2008)
16. Tucker, A.: A two-person dilemma. Stanford University Press (1950)
17. Shalizi, C.R., Shalizi, K.L.: Blind construction of optimal nonlinear recursive predictors for discrete sequences. In: Chickering, M., Halpern, J. (eds.) Uncertainty in Artificial Intelligence: Proceedings of the Twentieth Conference, pp. 504–511 (2004)

Cooperation in P2P Systems through Sociological Incentive Patterns

Sebastian Kaune[1], Konstantin Pussep[1], Gareth Tyson[2], Andreas Mauthe[2], and Ralf Steinmetz[1]

[1] Technische Universität Darmstadt, Germany
[2] Lancaster University, United Kingdom

Abstract. While the performance of peer-to-peer (p2p) systems largely depend on the cooperation of the member nodes, there is an inherent conflict between the individuals' self interest and the communal social welfare. In this regard, many interesting parallels between p2p systems and cooperation in human societies can be drawn. On the one hand, human societies are organized around a certain level of altruistic behavior. Whilst, on the other hand, individuals tend to overuse public goods, if they are free to do so. This paper proposes a new incentive scheme that extracts and modifies sociological incentive patterns, based on the Tragedy of Commons analogy, to work efficiently in a p2p environment. It is shown through simulations that this scheme encourages honest peers whilst successfully blocking non-contributors.

1 Introduction

It has long been understood that the performance of peer-to-peer (p2p) systems rely on the cooperation of the member nodes. This realization creates a social dilemma for the users of such systems, as the necessity to altruistically provide resources goes against the selfish desire to limit one's own personal sacrifice. Consequently, users' self interests results in the free-riding problem [1,2] by trying to exploit others while not contributing themselves. Hence, cooperation amongst peers becomes sparse unless an incentive scheme can encourage participants to contribute their resources.

Considering the tensions between individuality and communal social welfare in human societies, many interesting parallels to p2p systems can be drawn. On the one hand, individuals tend to overuse public goods resulting in the Tragedy of Commons [3]. On the other side, human societies are organized around cooperative interactions and a certain level of altruism. Rich analysis in evolutionary sociology has tried to answer this issue and has largely concluded that indirect reciprocity explains the evolution of cooperation among unrelated individuals [4,5]. In an extensive study, [6] analyzed how the concept of reputation is used in human societies to encourage cooperation. As an outcome, important incentive patterns were identified that are mandatory for the evolution of cooperation.

Inspired by these findings and the similarities observed between p2p systems and human societies, we propose a new reputation-based incentive scheme that aims to encourage honest users to participate in the system whilst successfully blocking free-riders. Our major contribution can be summarized as follows: we present a new point in the design space of reputation systems by using extremely limited, non-local reputation information, amounting to a single bit per participant. By adopting insights of sociologists, we show that the classification of nodes as either good or bad offers high potential to encourage cooperation while still encoding as much information as necessary to prevent rational/malicious attacks. We further introduce a similarity-based approach to filter out false recommendations submitted by dishonest nodes.

The paper is organized as followed: in Section 2 an overview of related work is provided. Section 3 describes the design of our system, highlighting both the representation of reputations and how they are utilized. Section 4 then outlines a number of practical deployment issues and how we resolve them. Subsequently, Section 4 evaluates, using game theoretic modeling, the effectiveness of our approach whilst, Section 5 concludes the paper, outlining future work in the field.

2 Background

Any participant in a p2p system is both a service provider and a service consumer. A *transaction* is the process in which a provider voluntarily grants a service to a consumer. Accordingly, the consumer benefits from this service whilst the provider pays the cost (e.g. upload bandwidth).

In general, a well-designed incentive scheme has to meet several challenges in order to be robust, notably:

- *Different user types*: Users can be classified into two categories: *obedient* and *dishonest*. The former are consistent with the system specifications and thus contribute to the system whereas the latter try to maximize their benefit at the expense of others.
- *Asymmetry of interests*: For example, peer A is interested in receiving a service from peer B whilst not being able to offer a valuable service in return.
- *Newcomers*: In general, it is impossible to distinguish dishonest nodes from so called legitimate newcomers. Thus, a newcomer policy is mandatory.
- *Untraceable actions*: In decentralized systems, it is impossible to monitor all occurred transactions. Thus, decentralized mechanisms are required to prove that two peers were involved in a distinct transaction.

2.1 Prior Incentive Schemes for Cooperation in P2P

In the area of p2p, various incentive schemes have been proposed to encourage users to contribute their own resources. Some of them are based on *monetary payment schemes* in which peers have to pay for the resources they consume [7,8,9]. However, many of these algorithms require a centralized infrastructure to enable micro payments and accounting.

An alternative is *reciprocity-based schemes* in which peers use historical information of past behavior of other peers to decide whether they want to share resources or not. These schemes can be further separated into direct and indirect reciprocity. In *direct reciprocity*, user offer resources only to those who have helped them before based on local observations, e.g., BitTorrent [10]. However, it assumes frequent repeated meetings between the same peers which might not be the case in large, diverse p2p environments.

In contrast, *indirect reciprocity* [11,12] allows peers to claim back their cooperativeness from any peer as each participant is associated with a reputation. Users earn a reputation based on the feedback from others they have interacted with; this, in turn, is used to differentiate between contributors and free-riders. These schemes, accordingly, rely on local observations, and additionally on second-hand information distributed among all nodes in the system [13,14]. However, this share of own experiences enables malicious nodes to disseminate false information about cooperative participants. To tackle this problem, a sound solution is to determine transitive chains of trust among known and reputable nodes [15,16]. On the other side, the share of information additionally introduces the collusion problem in which peers artificially increase each other's reputation. A countermeasure against this threat is to apply the computational expensive min-cut max-flow theorem, as proposed by [17,18].

Different from all studies above, our system design neither relies on transitive trust nor on the often applied min-cut max-flow theorem.

2.2 The Tragedy of Commons Analogy

Many social scientists, as well as psychologists, have tried to explain cooperation in human societies. This problem is also known as the *Tragedy of Commons.* From the societies' point of view, the community does best if all individuals mutually cooperate. However, it can be observed that individuals or groups will exploit the generosity of others, if they are free to do so.

By means of game theory, sociologists try to find evolutionary stable behavioral strategies (ESS) to explain the question of cooperation. In particular, [19] found indirect reciprocity to enable cooperative ESSs in human societies. Further, [6] identified certain key properties of successful reputation schemes to encourage cooperation among unrelated individuals. In particular, these incentive patterns have been proven to be highly robust and stable against different patterns of defection, even in the presence of observational errors concerning individuals' reputation.

In spite of the fact that our work is inspired by these observations, we are aware that the aforementioned studies are carried out in environments that differ from p2p systems in the following points: (*i*) permanent identities (players do not leave the system), and (*ii*) traceable actions (both defection and cooperation). However, both conditions are challenged in p2p systems, and the transfer of these insights to p2p systems must be deliberate.

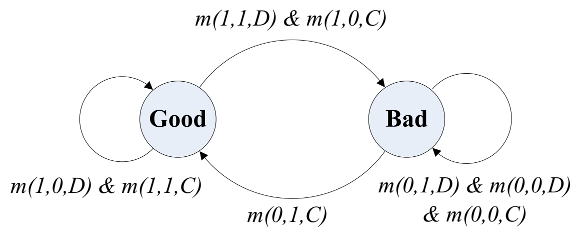


Fig. 1. The 8 reputation transitions

3 Reputation-Based Incentive Scheme

The fundamental aspects of reputation-based p2p systems can mainly be divided into: (*i*) a reputation-based incentive scheme and (*ii*) a distributed reputation infrastructure. The former is of major importance as it describes how reputation is computed within the system. The underlying mechanisms are therefore crucial for the scheme's overall performance and must be carefully designed. The latter, on the other hand, is responsible for implementing (*i*) in a fully distributed manner; it maintains and stores reputation values, and allows peers to access the reputation of the others.

3.1 Representation of Reputation

In our work, reputation values are represented by a globally binary digit that can be either 0 or 1, indicating a good (G) and bad standing (B) respectively. Let N be the population of peers in our system. Then, the global reputation score r of an individual is given by $r : n \rightarrow \{0, 1\},\ n \in N$.

3.2 Assignment of Values

Reputation values are dynamically assigned to peers based on their *last action* when performing the role of service provider. In more detail, if a node takes an action A, either to cooperate (C) or to deny cooperation (D), when there is the option of providing a service, our system assesses the goodness of this action by using *reputation transitions*. In general, each reputation transition m depends on three factors:

– the current reputation value of the service provider r_p,
– the reputation score of the service consumer r_c,
– the taken action A (either C or D) by the service provider.

Thus, each transition is well-defined by a triple m which is mapped to either 0 or 1 as defined in the following:

$$m(r_p, r_c, A) \rightarrow \{0, 1\} \qquad (1)$$

Fig. 1 shows a state diagram of this transition process, highlighting the 8 possible steps between states leading a service provider to either a good or a bad standing[1]. The design of these transitions is inspired by the observations made in [6]. As stated before, this extensive study identified important incentive patterns that are mandatory to encourage cooperation in human societies. The so called "keys to success" have been defined in the following properties: *being nice* (maintenance of cooperation among contributors), *retaliatory* (identification of dishonesty, punishment and justification of punishment), *forgiving*, and *apologetic*. All of them are incorporated in the depicted transitions:

(1) *Maintenance of cooperation: m(1,1,C)=Good.* If two nodes in a good standing cooperate, the donor should maintain its good standing.

(2) *Identification of Dishonesty: m(0,1,D)=Bad, (1,1,D)=Bad.* Nodes not providing services have to fall into bad standing, irrespective of their reputation.

(3) *Apology and Forgiveness: m(0,1,C)=Good.* Once (mistakenly) fallen into bad standing, there should be an opportunity to allow immediate forgiveness to regain a good standing again.

(4) *Punishment and Justification of Punishment: m(1,0,D)=Good.* When a dishonest node is detected and identified, other nodes contributing to the system should refuse to provide services to it, and should not be punished for this.

The remaining three transitions are degrees of freedom, which we fixed running several experiments measuring the impact of each combination.

3.3 Behavior of Nodes

We define the way a peer uses the reputation scores as its *behavioral strategy* denoted by $\vec{s}$. In more detail, each peer uses a decision function f to decide how to behave towards requesting service consumers. f takes as input parameters the reputation score of both itself and the consumer. There are four possible situations in which a peer i would want to assess another peer j with respect to the reputation scores ($f_{ij} := f(i,j)$):

- f_{00}: both peers are in bad standing
- f_{01}: peer i is in bad standing whereas peer j is in good standing
- f_{10}: peer i is in good standing whereas peer j is in bad standing
- f_{11}: both peers are in good standing.

Thus, $\vec{s}$ consist of four components $(=(f_{00}, f_{01}, f_{10}, f_{11}))$ whilst each of them describes whether to cooperate (C) or to deny cooperation (D).

$$\vec{f} : \{0,1\}^2 \rightarrow \{C, D\} \tag{2}$$

For example, altruistic peers would follow behavioral strategy $\vec{s}_{alt}$ = (C, C, C, C) whereas free-riders are described by $\vec{s}_{free}$ = (D, D, D, D). The built-in incentive in our scheme is based on the assumption that cooperative peers

[1] Depending on the application, a good standing is bounded on predefined time interval, in order to encourage nodes to continuously take the role of a provider. However, we will not pursue this issue any further in this paper.

will favor each other. Thus, peers are encouraged to take the role of a service provider in order to gain a good standing. In turn, this greatly enhances the probability of obtaining services provided by others. As shown later on, peers using the *discriminator* strategy $\vec{s}_{disc}$ = (D, C, D, C) can successfully block non-contributors.

3.4 Newcomers

Up to now, we have assumed that nodes already have a standing within the system. Newcomers, however, do not have a transaction history, and are therefore marked as *strangers*. In order generate a good standing, they have initially to cooperate with another stranger or a peer already enjoying a good standing. When requesting services, discriminators will deny to provide services. Accordingly, our system assigns no profit to newcomers.

4 Reputation Infrastructure

Here, we address the practical issues of our approach. In particular, it is specified which nodes are authorized to update reputation values, how reputation values can be globally accessed, and how peers are able to protect themselves against false reports. We assume that users participating in the system are characterized by anonymous identities. Each node owns a public/private key pair suitable for establishing signed messages between nodes. In addition, each participant in the system is identified by a random unique overlay identifier (OId). To ensure that node Ids are chosen randomly from the identifier space, we use trusted certification authorities (CA). These CA's bind a random node id to the node's public key, a process conventionally done offline.

4.1 Replica Set

Due to the lack of a centralized authority, the task to reliably store and update global reputation values is none-trivial and challenging. The peer's reputation must not be stored locally, where it can become subject to manipulation. Storage on a randomly chosen peer similarly does not guarantee that this one is honest. Thus, we assign this task to multiple nodes in the system.

Each peer i is assigned a *replica set* R_i, consisting of a small number of k random peers. To this end, we interconnect all participants in the system using a distributed hash table, e.g. Chord [20]. The members of R_i are then determined by applying a set of k one-way secure hash-functions $h_0(i), h_1(i), ..., h_{k-1}(i)$ to i's overlay id. The hashes derived from these functions constitute the overlay identifiers of the replica set nodes. This ensures that peer i cannot select the members of its own replica set R_i.

If a peer wants to request the reputation of another one, it individually contacts the responsible replica set members. The provided information is *legitimate* if, and only if, more than half of the reports are identical. This implies that the majority of the replica set members must be obedient.

To quantify this, we define a replica set as *reliable* if more than half of the nodes are obedient. Let o and m be the amount of obedient and malicious peers in the system, respectively. The probability to chose an obedient peer for a replica set is $\frac{o}{o+m}$. From this, the probability of obtaining at least $\lceil \frac{k}{2} \rceil$ obedient peers in a replica set is given by $\sum_{n=\lceil \frac{k}{2} \rceil}^{k} \binom{k}{n} \left(\frac{o}{o+m}\right)^n \left(1 - \frac{o}{o+m}\right)^{k-n}$.

For example, the probability of obtaining a reliable replica set in a population consisting of 100.000 nodes, of which $m = 5.000$ nodes are malicious, is 99,88% for $k = 5$. It can easily be verified that k must only be slightly adapted with continuing increase of m.

4.2 Transaction Process

Consumers must submit experiences about the outcomes of transactions (whether a distinct provider p has delivered a service or not) to the provider's replica set R_p. As stated above, this replica set is then authorized to update the reputation value of the provider based on its decision (cf. Sect. 3.2). Since R_p constitutes a third party not directly involved in the transaction process, a mechanism is needed that proves that two distinct peers have interacted with each other. Each transaction therefore consists of five sequential steps:

Step 1. The consumer c creates a service request message containing the following fields $< r_c, pKey_c, OId_c, OId_{lv} >$, where r_c is the consumers current reputation value; $pKey_c$ is its public key; OId_c is its overlay id; and OId_{lv} is the overlay id of the peer that has lastly rated c in the role of provider. Afterwards, c signs the request with its private key and sends it to the chosen provider.

Step 2. Upon receipt, the provider contacts the consumer's replica set R_c to verify the correctness of the information contained in the message.

Step 3. If correct, the provider signs the message and sends it back to the consumer. Thereafter, the service delivery takes place.

Step 4. After the transaction phase is completed, the consumer rates the cooperativeness of the provider (*1*= service received or *0*=service not received) and submits its decision to the provider's replica set R_p.

Step 5a. Each replica set member $R_p(x)$, $\forall x \in [h_0..h_{k-1}]$, first checks whether the service request has been actually signed by provider p. Afterwards, it stores the OId of c as the one of the provider p's last voter, and updates p's reputation value by applying the appropriate reputation transitions.

Step 5b. Additionally, each member of R_p is mapped in the x-th position to its respective counterpart in R_c forming k-pairs. For each pair, the provider's replica set member contacts its counterpart to inform it about the rating c has given on p; this is matter of consequence, as explained in the following.

4.3 Similarity-Based Trustworthiness

To reflect the personal experience a consumer has had with distinct providers, each peer i in the system owns a global vector $\vec{t_i}$, where $\vec{t_i} = (t_{1i}, ..., t_{|N|i})$ for all $n \in N$. Each component of $\vec{t_i}$ contains in the n-th position the arithmetic mean of all ratings peer i has submitted on a distinct peer n. Hence, this value describes the *subjective trust* peer i places in peer n. Since each rating can either be 0 or 1, the component values will also be between 0 and 1. According to Step 5b, these trust vectors are stored and maintained by the peer's replica set and are publicly available.

The purpose of these vectors is to define a notion of trust Peer A places in the recommendations of Peer B. To this end, we introduce a similarity function

$$sim(\vec{t}_A, \vec{t}_B) = \frac{1}{|N|} \sum_{i=1}^{|N|} 1 - |t_A(x_i) - t_B(x_i)| \in [0,1] \tag{3}$$

which in its basic functionality component-wise compares whether both peers have rated the same provider. If so, the deviation between both ratings is calculated and summed up to an overlay similarity S. We define the recommendations of Peer B as *trustworthy* for Peer A, if S exceeds a certain similarity threshold t. In our system, providers apply this function on trust vectors of requesting consumers, in order to determine whether they have maliciously rated obedient providers as bad. Also, it is applied on the last voter of a distinct peer to determine the trustworthiness of his recommendations.

5 Evaluation

In the following, the performance of our scheme is examined against threats of selfish users. Our main goal is to explore which behavioral strategy is the dominant one among a set of chosen strategies. Further, we will examine the effectiveness of our reputation infrastructure against malicious attacks. For that reason, we adopt a game theoretical approach as explained in the following.

5.1 Generalized Prisoner's Dilemma

To model a p2p system by means of game theory, we use the Generalized Prisoner's Dilemma (GPD) that includes two players who interact once in a *one-shot game*, as described in [17]. Unlike the original Prisoner's Dilemma GPD includes the social dilemma and the asymmetry of interests. In particular, each player i follows a *behavioral strategy* by having the choice to cooperate(C_i) or defect(D_i) its opponent. Depending on their actions, each payer receives one of the following *payoffs*: R_i (the reward for mutual cooperation), S_i (the sucker's payoff), T_i (the temptation to defect), and P_i (the punishment for mutual defection). In our context one of the peers acts as provider (P) and the other as consumer (C). The payoff matrix for both consumer and provider is shown in Figure 2(a). To create a social dilemma, the payoffs must fulfill the following criteria:

- Mutual cooperation among peers yields a higher payoff than mutual defection: $R_C + R_P > P_C + P_P$
- Mutual cooperation yields a higher payoff than alternating cooperation-denial cycles: $R_C + R_P > S_C + T_P$ and $R_C + R_P > S_P + T_C$

Payoff-Table *General Form*		**Service Provider**	
		Cooperate	Defect
Service Consumer	Cooperate	R_C / R_P	S_C / T_P
	Defect	T_C / S_P	P_C / P_P

(a) Asymmetric payoff matrix

Payoff-Table *Simulations*		**Service Provider**	
		Provide	Deny
Service Consumer	Request	*2 / -1*	*0 / 0*
	Don't Request	*0 / 0*	*0 / 0*

(b) Payoff matrix used in simulations

Fig. 2.

– In a one shot interaction, defection dominates cooperation as the costs for the service provisioning can be saved: $T_P > R_P$ and $P_P > S_P$

Let $u_{A|B}$ denote the achieved payoff of a behavioral strategy A when interacting with behavioral strategy B.

Definition 1. *Strategy A is said to be dominant if for all B holds $u_{A|A} \geq u_{B|A}$ and $u_{A|B} \geq u_{B|B}$.*

Under this definition, defection would be the dominant strategy for the provider in the one-shot GPD game. Hence, cooperation will never take place and the consumer will only have the choice between the payoffs S_C and P_C.

5.2 Simulations

To assess the performance of our scheme, we have implemented a simulator that corresponds to the above stated game theoretical model. We assume time to be divided into slots, and each slot lasts long enough to allow each peer to provide exactly one service to a requesting consumer. The evaluative scenario we utilize is a file-share application. The assignment of files and queries to peers follows a Zipf distribution ($\alpha = 0.9$). Each file is subdivided into equally sized file segments (chunks) constituting services peers are sharing in the network. Participants fall into two categories: *obedient* and *dishonest*. Obedient nodes follow the discriminator strategy $\vec{s}_{disc}$, and their similarity threshold is set to $t = 0.7$. The strategy of dishonest nodes will be varied as described later on.

In each slot, each peer has the opportunity to simultaneously act as service provider and consumer. Based on the service a consumer is interested in, it selects the most desired provider and sends a request. Each provider, on the other hand, favours the most appropriate consumer from its upload queue that enjoys a good standing and passes the similarity checks mentioned in Section 4.3. The behavioral strategy of a provider then defines the action (either C or D) she will take by considering the reputation of both herself and the consumer. Depending on how the provider acts, both the consumer and the provider will receive a payoff from the matrix depicted in Fig. 2(b). This matrix satisfies the inequalities stated in the previous section. It is assumed that providing a service incurs the same costs c (-1) to all providers, and consumers receive the same benefit b ($+2$), respectively. Finally, the reputation and trust vectors are updated after each time slot, and it is assumed that peers can leave or join the system with a probability of 5%.

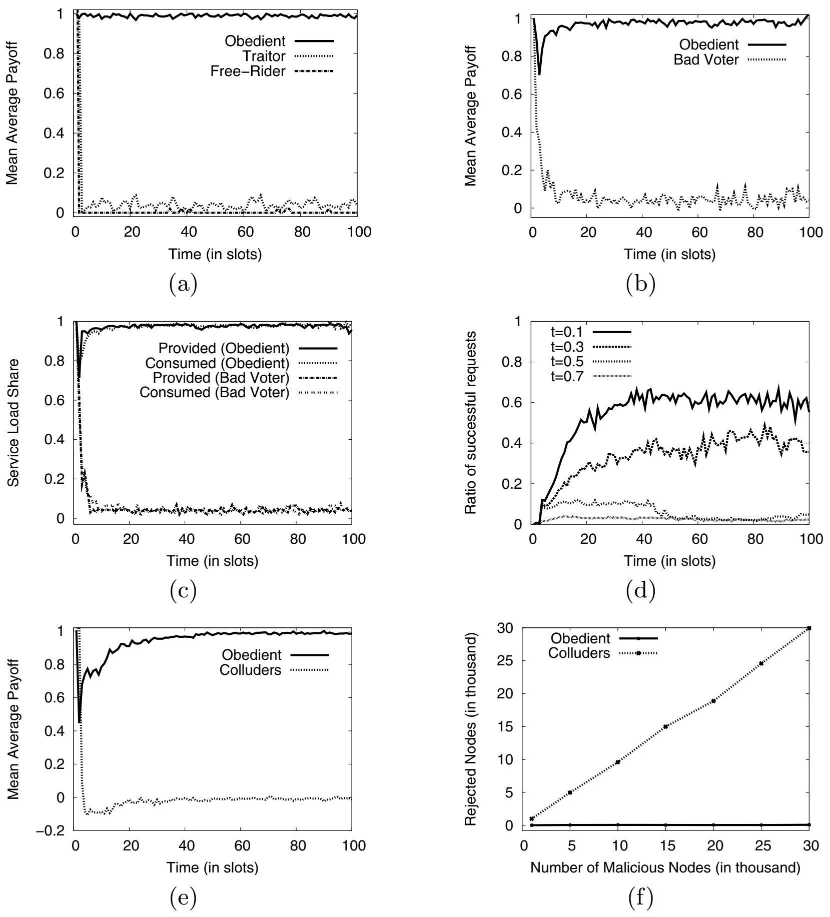


Fig. 3. Simulation results for (a) rational attacks, (b-d) bad voters, and (e-f) colluders

5.3 Performance under Rational Attacks

In our first experiments, we assume that users do not break down the system specifications (e.g. submit false reports), but try to exploit the generosity of obedient nodes by means of two types of selfish attacks. We consider the first type as *traitors* since these nodes acquire a good standing before turning into defectors. The second type is represented by free-riders who never contribute themselves. Accordingly, we equally divided the population in three groups: obedient peers, free-riders $\vec{s}_{free}$ = (D, D, D, D), and traitors $\vec{s}_{trait}$ = (D, C, D, D).

Fig. 3(a) shows the achieved mean payoff of each strategy per time slot. The highest level of cooperation would be 1 indicating that all peers following the respective strategy are contributing to the system and everyone is able to receive a service. It can be seen that the discriminator strategy applied by obedient nodes achieves the highest payoff over time. More precisely, our simulations revealed that this strategy obtains a mean average payoff of 0.98, indicating that nearly all obedient nodes continuously obtain services. In contrast, free-riders are successfully blocked never receiving a service after the first time slot. Traitors acquire a mean average payoff of 0.05. In particular, only 3%-6% of these nodes are able to receive a service. This stems from the fact that traitors deny to provide services after they have generated a positive standing. Accordingly, they will be subsequently ignored by obedient peers when acting in the role of consumer in the next slot.

In conclusion, both types of attackers cannot gain ground in the system as they achieve payoffs close to zero. Obedient nodes, applying the dominant strategy $\vec{s}_{disc}$, self-organize themselves into a robust and cooperative group in which non-contributors are efficiently detected and excluded.

5.4 Effectiveness of Reputation Infrastructure

In the second set of simulations, we study the effectiveness of our reputation infrastructure against malicious nodes falling into two categories: (*i*) bad voters and (*ii*) colluders. *Bad voters* follow the discriminator strategy $\vec{s}_{disc}$, but always rate cooperative providers as bad. Accordingly, they are mainly interested in lowering the providers' reputation to encourage other participants to exclusively use their own services. *Colluders*, instead, form a malicious collective and provide services to obedient nodes only with a probability of 20%. Moreover, they boost the reputation values of all peers in the collective by submitting fake transactions. To study these attacks, we assume that 30% of the population consist of malicious nodes from either of both presented categories.

The results of the *bad voter* experiments are as follows. Fig. 3(b) depicts the mean average payoff achieved by obedient nodes and bad voters. It can be observed that obedient peers achieve the highest mean average payoff over time amounting to 0.97 whilst that of the bad voters is close to zero. Fig. 3(c) measures the *service load share* of both strategy types. This metric determines in each time slot the fraction of peers that provided and were able to receive a service, subject to a distinct user group. It can be seen that nearly all obedient peers are continuously able to receive a service whereas only 5-7% of the bad voters are supplied with data. To explain this, Fig.3(d) plots the mean ratio of successful requests experienced by bad voters while varying the similarity thresholds applied by obedient nodes. That is, this ratio measures how often a bad voter was unrecognized when requesting a service by an obedient provider, related to all send requests. For $t = 0.7$ on average 97% of all request carried out by bad voters are detected when applying the similarity function. Accordingly, bad voters are almost never served by obedient nodes.

The experiments with *colluders* strengthens our findings that the similarity-based comparison of global trust vectors very efficiently detects nodes trying to compromise the system. Fig. 3(e) plots the achieved payoffs of both colluders and obedient peers. As in our previous experiments, the discriminator strategy clearly dominates the attacker strategy. Colluders are quickly detected as the trust vectors of both user groups highly differ from each other. In fact, the determined overall similarity between the trust vectors of both users groups is on average 0.23. Accordingly, obedient peers do not trust ratings submitted by colluding peers but favour honest peers. To confirm this, Fig. 3(f) plots the total amount of consumers that have been rejected by obedient providers after 50 transactions. At this point of time, nearly all malicious consumers are rejected by obedient providers, irrespective of the size of the malicious collective. Instead, the number of rejects to obedient consumers is nearly zero in all simulated scenarios.

We conclude that the usage of the similarity function enables the system to efficiently filter out spurious reports from malicious nodes. Moreover, bad voters are immediately punished when submitting false reports on obedient nodes; since their global trust vectors very quickly deviate to the one of peers conforming to system's norm, they are immediately rejected by these nodes.

6 Conclusion

This paper has investigated the correlations between p2p environments and cooperation in human society. Through this, a new reputation-based incentive scheme has been designed, utilizing extremely limited binary reputation representations. Alongside this, we have also proposed a fully decentralized reputation infrastructure capable of securely managing reputations and protecting against malicious collusion and false reports. This approach was evaluated, through simulation, showing that nearly all peers wishing to gain services must contribute to the system, eliminating free-riding. It was further shown that malicious peers, solely interested in disrupting the network, were also quickly ostracized.

There are a number of areas of future work. Firstly, detailed overhead studies are necessary to investigate the impact that utilizing such a scheme has on the overall system. Further investigation into improving the infrastructure is also planned to protect against extremely high levels of malicious users ($> 50\%$) of the replica set. Lastly, more detailed evaluative scenarios will be performed to investigate the reliability of the infrastructure against bad voters, especially if these nodes selectively or randomly change their misbehavior per transaction.

Acknowledgement

The collaboration on this paper has been funded through the European Network of Excellence CONTENT, FP6-0384239.

References

1. Adar, E., Huberman, B.: Free riding on gnutella. First Monday (2000)
2. Saroiu, S., Gummadi, P., Gribble, S.: A measurement study of p2p file sharing systems. Technical report, Washington University (2002)
3. Harding, G.: Tragedy of commons. Science (1968)
4. Nowak, M.A., Sigmund, K.: Evolution of indirect reciprocity. Nature (2005)
5. Leimar, O., Hammerstein, P.: Evolution of cooperation through indirect reciprocation. Proc. R. Soc. Lond. (2001)

6. Othsuki, H., Iwasa, Y.: How should we define goodness? - reputation dynamics in indirect reciprocity. Journal of Theoretical Biology (2004)
7. Zhang, Z., Chen, S., Yoon, M.: MARCH: A distributed incentive scheme for p2p networks. In: INFOCOM (2007)
8. Jakobsson, M., et al.: A micro-payment scheme encouraging collaboration in multi-hop cellular networks. In: Wright, R.N., et al. (eds.) FC 2003. LNCS, vol. 2742. Springer, Heidelberg (2003)
9. Wilcox-O'Hearn, B.: Experiences deploying a large-scale emergent network. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429. Springer, Heidelberg (2002)
10. Cohen, B.: Incentives build robustness in bittorrent. Technical report (2003)
11. Habib, A., Chuang, J.: Service differentiated peer selection: an incentive mechanism for p2p media streaming. IEEE Transactions on Multimedia (2006)
12. Srivatsa, M., Xiong, L., Liu, L.: Trustguard: countering vulnerabilities in reputation management for decentralized overlay networks. In: WWW (2005)
13. Damiani, E., et al.: A reputation-based approach for choosing reliable resources in peer-to-peer networks. In: CCS (2002)
14. Xiong, L., Liu, L.: Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. IEEE Trans. on Knowledge and Data Engineering (2004)
15. Kamvar, S., Schlosser, M., Garcia-Molina, H.: The eigentrust algorithm for reputation management in P2P networks. In: WWW (2003)
16. Lee, S., Sherwood, R., Bhattacharjee, B.: Cooperative peer groups in nice. In: INFOCOM (2003)
17. Feldman, M., Lai, K., Stoica, I., Chuang, J.: Robust incentive techniques for p2p networks. In: CECOMM (2004)
18. Eftstathiou, E., Francgoudis, P., Polyzos, G.: Stimulating participation in wireless community networks. In: INFOCOM (2006)
19. Nowak, M.A., Sigmund, K.: Evolution of indirect reciprocity by image scoring. Nature (1998)
20. Stoica, I., et al.: Chord: A scalable p2p lookup service for internet applications. In: SIGCOMM (2001)

A Self-Organizing Super-Peer Overlay with a Chord Core for Desktop Grids

Peter Merz, Steffen Wolf, Dennis Schwerdel, and Matthias Priebe

Distributed Algorithms Group
University of Kaiserslautern, Germany
{pmerz,wolf,d_schwe,priebe}@cs.uni-kl.de

Abstract. We present a self-organizing super-peer overlay that suits the communication requirements of a Peer-to-Peer Desktop Grid system well. This overlay combines favorable properties of Chord rings and fully meshed super-peer networks, yielding benefits that include an efficient broadcast scheme and a reduced average message hop count compared to pure Chord. To this end, we introduce a distributed algorithm that sets up such an overlay in a self-organized way. Moreover, we deploy network coordinates to improve Chord's end-to-end message routing delay and build a deterministic gossip mechanism over the Chord ring's fingers. We demonstrate the effectiveness of our concept through simulations using topology information retrieved from PlanetLab.

1 Introduction

Desktop Grids have been proposed to tap the idle computation resources provided by desktop computers [1,2,3,4]. The requirement to support a large number of participants emphasizes the importance of scalability in this context. The construction of a Desktop Grid structure on a Peer-to-Peer (P2P) overlay network improves scalability in large-scale settings [2,5]. The scalability of P2P networks is further improved by the concepts of *super-peers* [6,7,8] and *structured P2P overlays*, commonly associated with Distributed Hash Tables (DHT) [9,10,11]. In this paper, we propose to merge these two concepts for the benefit of Desktop Grid operations to yield a self-organizing super-peer overlay network with a Chord [10,11] ring core that interconnects the super-peers. The resulting overlay is managed by a proximity-aware distributed algorithm. A deterministic gossip mechanism provides all peers with a list of the Chord ring members. Moreover, we improve Chord's ID assignment by allocating IDs to joining peers in a delay-optimized fashion, and modify Chord's message routing scheme to prioritize low-delay finger links, to the end of reducing end-to-end delays for both unicast and broadcast messages. In our concept, we resort to a network coordinates scheme as a provider of low-cost round-trip time (RTT) estimates.

Besides scalability, our concept addresses common requirements of Desktop Grids including low end-to-end message delay and an efficient broadcast for resource discovery. It features robustness due to the absence of a single point of failure. The inclusion of super-peers can reduce the number of hops required to deliver a message compared to pure Chord, generate locality for common peers through clustering them around super-peers, and provide a firewall bypass if common peers cannot communicate directly with each other.

Following the results of [9], we have preferred Chord over other overlay types due to higher flexibility and resilience. Our concept strips Chord from the Distributed Hash Table (DHT) functionality that associates keys with content managed by individual peers, and uses the finger-enhanced ring structure only.

This paper is organized as follows. In Section 2, we outline network coordinates. In Section 3, we introduce our distributed overlay management concept. In Section 4, we propose delay-minimizing modifications to Chord's ID assignment and routing methods. In Section 5, we present simulations conducted to quantify the benefit of our concept. In Section 6, we provide an overview of related work. The paper concludes with directions for future research in Section 7.

2 Network Coordinates

Network coordinates are space coordinates individually assigned to each node in a network such that the distance between any two nodes in the space approximates the measurable delay (RTT) between those nodes in the network. We deploy the Vivaldi [12] system for coordinates generation, a concept which is built on the notion of spring deformation. Vivaldi's authors point out its use for Chord to help build proximity awareness in finger table construction [12].

In the remaining sections, we use the following notation: $d_{RTT}(X,Y)$ refers to the live, actual, RTT-based network delay between nodes X and Y, $d_{NC}(X,Y)$ refers to the delay predicted by network coordinates between nodes X and Y, and $d_{ID}(X,Y)$ refers to the distance between Chord nodes X and Y in Chord's identifier space, measured in the routing direction. By construction, $d_{NC}(X,Y)$ is symmetric, i. e. $d_{NC}(X,Y) = d_{NC}(Y,X)$, while d_{RTT} and d_{ID} are not.

3 Overlay Management

In this section, we present a distributed mechanism to construct a topology in which some nodes will act as super-peers while the remaining nodes, henceforth called edge peers, are assigned to exactly one of the super-peers each [8].

We aim to minimize the number of connections a super-peer needs to keep up. In a super-peer overlay network with a fully meshed super-peer core and n nodes in total, every super-peer maintains one link per remaining super-peer and one link to each of its edge peers. Assuming an even distribution of the edge peers over the super-peers, the optimal number of super-peers equals $\sqrt{n}$, yielding $2 \cdot (\sqrt{n} - 1) \in \mathcal{O}(\sqrt{n})$ connections per super-peer. While such a network bears significant advantages including a small diameter of only 3 hops [13], the load super-peers need to handle in this setting may become prohibitively high in large networks. In contrast, the use of a Chord ring vastly reduces the number of links to other peers, however the hop count rises to $\mathcal{O}(\log_2 n)$ in the average case. Also, Chord has no sense of locality. There is no proximity awareness in overlay construction or routing [14]. The situation can be improved by replacing a super-peer overlay's fully meshed core with a Chord ring. A distributed mechanism can manage an overlay of this type, providing proximity awareness and self-stabilization. The remainder of this section introduces our approach.

3.1 Super-Peer Selection

A distributed super-peer selection algorithm (SPSA) for a fully meshed overlay core has been introduced in [13]. SPSA promotes edge peers to super-peer level and assigns edge peers to their closest super-peers in terms of network delay. It is executed on every peer. We improve SPSA to manage a dynamic Chord core and denote that version *ChordSPSA*. In our model, super-peers become part of the Chord ring, while super-peers falling back to edge peer level instantly leave the Chord ring. Hence, the set of super-peers matches the set of Chord ring members. Messages sent by an edge peer for another edge peer will enter the Chord ring at the sender's super-peer, be routed through the ring, and exit it at the receiver's super-peer. As with SPSA, every peer periodically enters a reorganization phase. During this phase, a peer decides whether to change its state, depending on its role (edge peer or super-peer).

Let n be the total number of peers in the overlay and x the optimal number of super-peers to populate the Chord ring. ChordSPSA operates on the basis of super-peers having $f = \log_2 x$ finger connections to distinct nodes. Given a maximum number $m > 0$ of edge peers a super-peer can handle per distinct finger node, the equation below describes the composition of the overlay:

$$x \cdot f \cdot m + x = n \tag{1}$$

ChordSPSA numerically solves this equation for x in a distributed way. Let g be the number of edge peers currently managed by a particular super-peer. That super-peer is able to handle as many edge peers as its limit, $f \cdot m$, permits. If the super-peer in its reorganization phase finds that $g > f \cdot m$ holds, it promotes one of its edge peers to super-peer level such that a portion of its edge peers will relocate to the new super-peer. That way, the number of super-peers grows as long as unserved edge peers require more super-peers. The growth stops once every edge peer is under management by a super-peer, i. e. the minimum stable number of super-peers is reached when every super-peer administrates the maximum number of edge peers. This reflects the optimal situation as described by (1). A super-peer's edge peer set grows merely logarithmically with the overlay size n as $f \cdot m \in \mathcal{O}(\log_2 n)$ compared to $\mathcal{O}(\sqrt{n})$ in the fully meshed case.

Since the super-peer structure adds two additional hops to message paths when two edge peers communicate, the choice of m determines the average hop count. If m is sufficiently large to allow an x small enough, the average hop count will diminish compared to pure Chord. The super-peer overlay's average hop count h_{avg} equals $2 + \frac{1}{2} \cdot \log_2 x$ when two edge peers communicate. h_{avg} is less when one or both communication endpoints are super-peers. However, we use the given h_{avg} as the worst case average hop count for determining a bound for m. The average hop length in the overlay is less than with pure Chord if

$0.5 \cdot \log_2 n > 2 + 0.5 \cdot \log_2 x$ which can be reshaped to $\log_2 \frac{n}{x} > 4$. Solving (1) for $\frac{n}{x}$ and putting the result into the reshaped inequality yields $f \cdot m > 15$, indicating that already a small m suffices to lower the average hop count.

When edge peers leave the overlay, some super-peers may find themselves underloaded. In their reorganization phase, super-peers downgrade to edge peer level if they find that $g < \frac{1}{k} \cdot f \cdot m$ holds. $k = 2$ represents a lower bound considering the requirement of viable edge peer sets after splitting: when a super-peer finds its load too high (i. e. $g = f \cdot m + 1$), it promotes one of its edge peers to super-peer level and splits the remaining $f \cdot m$ edge peers evenly among itself and the new super-peer such that neither super-peer downgrades due to low load.

The selection that an overloaded super-peer P performs to promote an edge peer to super-peer level assumes that since P is overloaded, its edge peer set E_P is non-empty. Based on the notion of edge peers connecting to their closest super-peers, we filter E_P to create the set E'_P holding all edge peers of E_P which, if selected as a super-peer, would cause the post-promotion edge peer numbers of both itself and P to reach or exceed the lower bound to prevent instant downgrades. P can accomplish this on its own by using its edge peers' network coordinates to attain inter-peer delays. Out of all $e \in E'_P$, P picks e' for which the optimal Chord ID choice yields minimal total finger delay. e' is asked to become a new super-peer. The associated proximity-aware identifier picking scheme will be discussed in detail in Section 4.2. If e' declines or fails to respond, P will retry in its next reorganization phase and may exclude e' that time. Since some edge peers can be unwilling to act as super-peers, announcing this to their super-peers at connect time may prevent them from being asked for promotion.

Edge peers connect to their closest super-peers. We expect this property to lift the actual number of super-peers above the optimal number, x, since some super-peers will not reach their maximum edge peer limit while others will exceed it, creating the need to promote more edge peers to super-peers.

ChordSPSA uses a subset of SPSA's messages [13] plus messages to transport the Chord overlay maintenance signals described in [11]. A super-peer taboo list locally kept with every edge peer prevents the individual peer from consecutively repeating an unsuccessful connection attempt to the same super-peer.

In an overlay of this kind, a broadcast can be efficiently performed. The approach corresponds to efficient broadcast in a Chord ring [15,16] with two extensions. First, an additional hop is taken to deliver the broadcast message to a super-peer's edge peers. Second, a sender should be able to restrict its broadcast message to reach only a limited portion of the identifier space. This is useful for Desktop Grids: a job-submitting peer needs a certain number of worker peers to share the job's computational burden. A regular broadcast covers the entire overlay, probably resulting in an exceedingly large wave of responses.

3.2 Deterministic Gossip

A super-peer list containing all Chord ring members enables edge peers to connect to their closest super-peers. Such a list may contain, for every super-peer, its Chord ID, network address, network coordinates, and a sequence number in lieu of a timestamp. If a super-peer fails, its edge peers may connect to other super-peers picked from the list. Furthermore, a super-peer list enables peers to directly deliver urgent unicast messages to the destination super-peer if need be (at the expense of an additional connection set-up), compute the ring size, and use delay-optimized Chord ID selection as will be set forth in the next section.

To maintain a super-peer list, we deploy deterministic gossiping. Due to network dynamics, super-peers are created and removed, causing the ring to change continuously. Our gossip intends to capture perceived changes and to notify other ring members about them. It is a method to perform anti-entropy super-peer list synchronization [17]. The list is loosely consistent: gossip provides a best-effort approach to continuously keeping the list's contents current.

All peers log recorded ring changes to a locally stored *update set*. Each element in this set contains the affected remote node's Chord ID, its most recently propagated network coordinates, a propagation counter, and a sequence number allocated by a scheme presented in [18]. An entry with a higher sequence number overrides an entry with a lower sequence number, provided that the same peer is concerned. An odd sequence number marks a peer as not belonging to the Chord ring, while an even sequence number marks it as an alive Chord node.

Information is propagated by every super-peer. A Chord node periodically transmits super-peer list updates to one of its finger peers, iterating through its finger set in a round-robin way, and requests the target peer to reply with its own set of recent updates.

To deal with churn, the deterministic gossip scheme deploys redundancy. With x Chord nodes, every update set entry is sent $\log_2 x$ times, resulting in $x \cdot \log_2 x$ messages. While $x - 1$ messages would suffice using regular broadcast, the departure of a single node may severely hamper the propagation effort if broadcast is used. If, for instance, the farthest finger node crashes, the broadcast will fail to reach $\frac{x}{2}$ nodes. To this end, we find the gossip scheme to be more robust while retaining adequate efficiency.

4 Delay Minimization

Originally, Chord considers the average number of hops as its main optimization goal [11]. While the Chord-enhanced super-peer overlay can further reduce the average hop count by shrinking the Chord ring, we concentrate on end-to-end delay (i. e. message latency) [9] as our major optimization objective. This way, messages may travel more hops but ultimately arrive faster. As pointed out in [9], there is a tradeoff between reducing the average hop count and reducing the average end-to-end latency. In a Desktop Grid context, we find latency to be more important. Besides the benefit for Desktop Grids, a low delay will also improve Chord's self-stabilization properties since information about ring composition changes is propagated faster.

In the first place, ChordSPSA provides locality by clustering edge peers around their closest super-peers. Moreover, we present two Chord-related optimizations. We refer to Proximity Route Selection (PRS) and Proximity Identifier Selection

(PIS) as introduced in [9]; similar approaches are discussed in [14]. In line with this scheme, we have designed PRS and PIS mechanisms for Chord to improve Chord's average message delay. They operate autonomously – neither need they be used in combination nor do they need to be used in conjunction with a super-peer structure. Hence, users of pure Chord may also benefit from them. However, as we will show in Section 5, the largest gains are achieved if both mechanisms are jointly deployed with the proposed super-peer overlay.

4.1 Finger Use in Routing

In original Chord, a node X routes a message onward by picking the finger node which covers the largest portion of the identifier space. This makes for a greedy finger selection process. We propose to refine this process as follows. Out of all suitable fingers, we prefer the one with lowest delay per covered identifier space unit. Since the super-peer list contains all finger nodes' network coordinates, the finger can be chosen instantly. Formally, with F_X as X's finger set, we suggest to pick X's finger node Y which satisfies

$$Y = \underset{Y' \in F_X}{\operatorname{argmin}} \frac{d_{NC}(X, Y')}{d_{ID}(X, Y')} \tag{2}$$

as the next message hop. This plan corresponds to a PRS approach. While it may incur additional hops, it reduces the average end-to-end message delay.

4.2 Chord ID Selection

A Chord peer picks a random ID from an identifier space whose size is publicly known. The odds of assigning the same ID twice are negligible due to the sheer size of that space and the uniform probability with which each ID is drawn. However, this approach does not account for proximity awareness. Since a peer's ID choice determines the placement of its fingers, we wish to assign IDs that lead to fast finger connections. With a super-peer list, a new Chord node may pick an ID such that the sum of delays of its outbound finger connections is minimized. Since the list contains network coordinates, the new node can select its optimal position without sending any messages. Apart from the ID choice, ring joining is performed as in original Chord.

In principle, every non-occupied element of the identifier space needs to be checked because every ID leads to different fingers. To limit this effort, we now propose a heuristic approach. The first peer picks an arbitrary ID. The second peer picks its ID such that it places itself at maximum distance in the identifier space from the first peer. All peers joining subsequently perform a *gap check*: in a Chord ring with r peers, there are at most r gaps between peers. Peers evaluate the positions marked by the middle ID of each gap, and choose the ID for which the total finger delay is minimized.

The super-peer list may temporarily contain outdated information. For this reason, two peers that join the ring concurrently or subsequently with only a brief time span in between could choose the same ID, causing a collision. Hence, ring members need to reject prospective newcomers that request already occupied IDs, instead providing them with an up-to-date super-peer list. Also, ring members should periodically check if their super-peer list contains another peer with the same ID. If the check yields a hit, the detecting peer can ask the remote peer for an ID change, or change its own ID.

The delay-based ID selection corresponds to the PIS scheme [9]. It may cause the message routing load to be unevenly distributed. However, since super-peers can promote edge peers to super-peer level to relieve themselves from excessive load, we expect the gains to outweigh the non-uniform load.

5 Experiments

We have conducted experiments to quantify the benefit of our concept. To this end, we have assumed that all peers wish to communicate with each other to an equal extent, and deployed an objective function Z to compute the total all-pairs delay. Concisely, we defined Z as

$$Z = \sum_{i} \sum_{j \neq i} d_{RTT}(i, s(i)) + d_C(s(i), s(j)) + d_{RTT}(s(j), j) \tag{3}$$

where $s(x)$ is the super-peer of x (with $s(x) = x$ if x is a super-peer), and $d_C(a, b)$ the sum of delays of the Chord ring hops required to travel from super-peer a to super-peer b in the routing direction. The average message delay d_{avg} in a n-node network can be obtained from Z by computing $d_{\text{avg}} = \frac{Z}{n \cdot (n-1)}$.

5.1 Setup

Each experiment was implemented through event-oriented simulation that lasted for 2500 seconds of simulated time. We have resorted to publicly accessible live node-to-node delay information from PlanetLab [19], using matrices with all-pairs RTT measurements [20]. We used the two largest matrices available with 419 and 414 nodes, and shall refer to them as *Matrix A* and *Matrix B*, respectively. Since data were flawed, in particular incomplete, we have dealt with the issue of missing distance information by computing a two-hop delay there.

We have measured ChordSPSA's effects in a static environment. With regard to Vivaldi, we have used a 4-dimensional Euclidean space and placed the nodes initially at positions drawn from a uniform random distribution where each coordinate was contained in the range [0; 500], such that in the 4-dimensional space, the initial distance between any two nodes did not exceed 1000. Nodes sent requests for distance estimation to other peers every 2 seconds, routed to random peers via one of their super-peers' fingers in a round-robin way, ultimately establishing their pairwise force using a three-way handshake.

All peers were restricted to a local view of the network and equipped with a possibly outdated super-peer list. We abstracted from the actual node coordinates exchange and assumed that edge peers received super-peers' coordinates

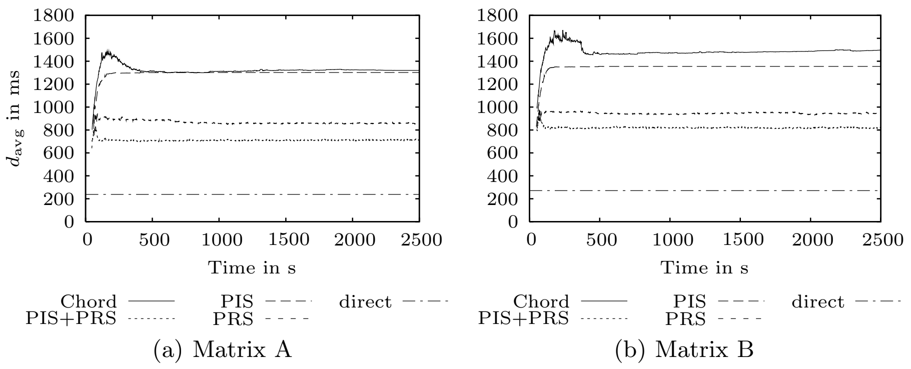


Fig. 1. Impact of Chord improvements (PRS and PIS, on their own and combined) on d_{avg} (Y axis)

via their assigned super-peer, while super-peers knew all remaining super-peers' coordinates. All nodes joined the network at the same time. One node was randomly chosen as the first super-peer. Each peer entered ChordSPSA's periodical reorganization phase with the default interval between cycles being $\gamma = 4 \pm 0.02$ seconds. A gossip cycle was triggered in every reorganization phase. The lower bound for super-peer downgrades was set to $k = 4$. Chord's identifier space contained 2^{20} elements.

We have used the objective function (3) and the average message delay d_{avg} to quantify an overlay's performance. We have also computed a direct connection lower bound which reflects the sum of communication costs in a non-hierarchical fully meshed network, and designated this bound *direct* in our plots.

5.2 Results

All plots display the progress of simulated time on the horizontal axis. Plotted values are averages over 30 runs.

Fig. 1 compares the performance of original Chord with that of enhanced Chord, the latter variant incorporating our PIS and PRS improvements. This experiment is based on $m = 0$, leading the topology to quickly converge to a Chord ring after being a super-peer overlay for a brief span of time in the beginning. While the benefit of delay-optimized ID assignment (PIS) on its own is comparatively small, the delay-optimized finger picking scheme (PRS) achieves a significant gain. The combination of PIS and PRS effectively halves the average message delay compared to original Chord. For all remaining experiments, we have implicitly included our PIS and PRS enhancements.

Fig. 2 depicts the effects of variations of m, the maximum number of edge peers per distinct finger, on d_{avg}. Since $m = 0$ after stabilization equals Chord, the plot labels the respective curve *Chord*. The figure shows the addition of super-peers to the PIS and PRS optimizations to yield even larger savings. With rising m, d_{avg} diminishes. However, an m which is too large would result in excessive load for

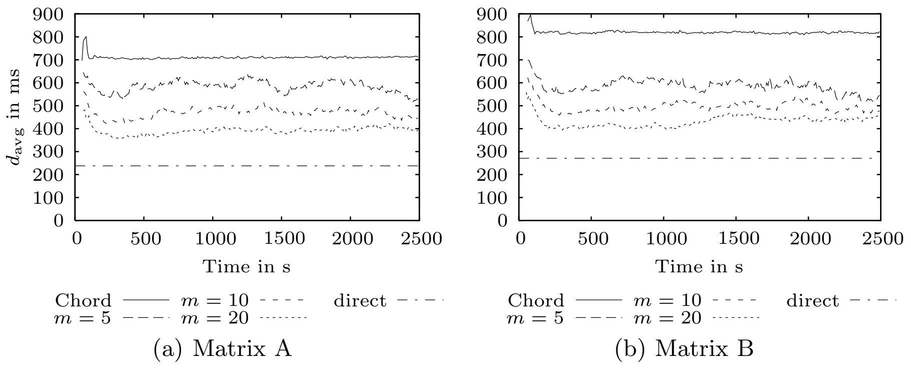


Fig. 2. Effects of varying the maximum number of edge peers per distinct fingers on d_{avg} (Y axis)

the super-peers in large networks where f, the number of distinct fingers, may exceed 10. Hence, we have limited m to $m = 20$. Beyond these observations, both Fig. 1 and Fig. 2 show the rapid convergence of ChordSPSA.

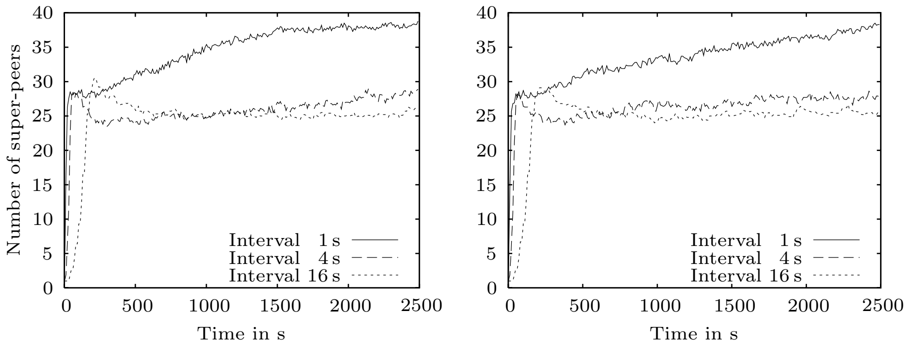
Fig. 3 plots the number of super-peers with reorganization phase interval lengths of 1, 4 and 16 seconds, respectively. While a length of 1 second leads to a growth that prevents the curve from converging within the simulation's timeframe, periods of 4 and 16 seconds both yield the desired numbers of super-peers. Fig. 4 shows the effects of reorganization phase interval changes on the average message delay, d_{avg}. Here, a short interval length of 1 second delivers better results than longer interval lengths. Despite the savings, a shorter interval leads to more reorganization phases, causing more ChordSPSA overlay control messages to be sent as seen in Fig. 5. The goals of swift adaptation to a changing underlay network and a low message amount are conflicting. Therefore, we have opted for a default value of 4 seconds as a compromise between these goals.

6 Related Work

The importance of proximity awareness in P2P overlay management has been stressed in [9,14]. While [14] centers on proximity-aware enhancements for Pastry, [9] concentrates on various DHT geometries and finds the ring geometry, as established by Chord, to be most flexible and resilient.

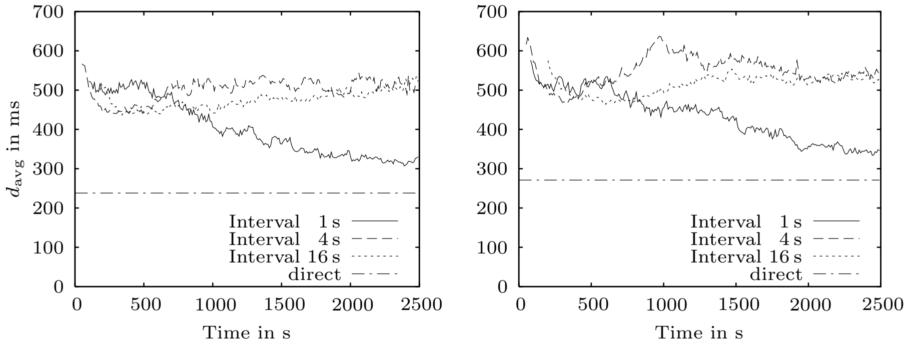
With a focus on performance, properties of gossip variants have been analyzed in a survey [21]. This survey also contains the generic deterministic round-robin scheme which we have modified for use within a Chord ring.

Related approaches to distributed super-peer topology construction include *SG-2* which also proposes the construction of a super-peer overlay topology with the aid of network coordinates [6]. SG-2 mimics the social behavior of insects to promote particular nodes to super-peer level and vice versa. Another procedure to create super-peer topologies is based on Yao graphs and a clustering-based



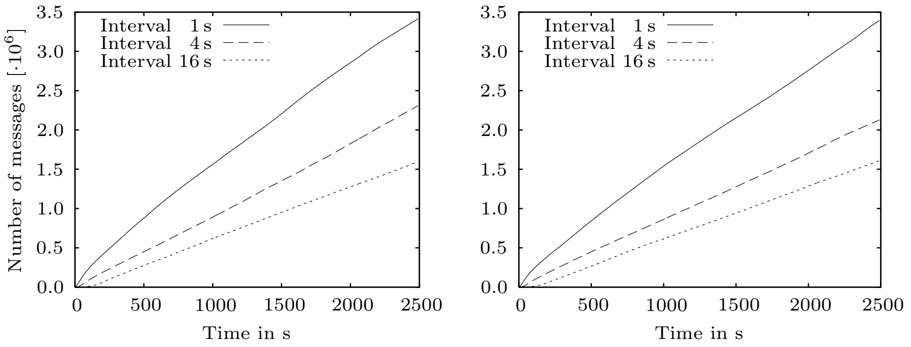
(a) Matrix A (b) Matrix B

Fig. 3. Effects of variations of the reorganization phase interval on the number of super-peers (Y axis)



(a) Matrix A (b) Matrix B

Fig. 4. Effects of variations of the reorganization phase interval on d_{avg} (Y axis)



(a) Matrix A (b) Matrix B

Fig. 5. Effects of variations of the reorganization phase interval on the number of ChordSPSA messages sent (Y axis)

network coordinates generator [22]. It determines the role of a node statically upon entrance into the overlay. However, neither approach has been geared towards the needs of P2P Desktop Grids.

7 Conclusion

Desktop Grids require scalability and swift communication in a dynamic P2P setting. In this context, we have discussed the benefits of a Chord-enhanced super-peer overlay for P2P-based Desktop Grids, and presented ChordSPSA, a distributed overlay management algorithm that maintains a topology of this type in a proximity-aware way. We have introduced a deterministic variant of gossiping that provides peers with a list of all Chord ring members. Moreover, we have improved Chord's ID assignment and routing processes to minimize the average end-to-end message delay. Simulations have quantified the gains our concept achieves, indicating that our Chord modifications alone suffice to cut Chord's average message delay in half and the addition of our proposed super-peer structure to save even more. With these improvements, Desktop Grids may operate more smoothly both in terms of overlay maintenance and job management, delivering results faster.

Future work will continue to explore the aspects of adaptivity, proximity awareness, and self-organization in distributed overlay management. The super-peer list bears additional opportunities, e. g. the detection of ring partitions provided that peers' successor IDs are also stored in the list, the direct delivery of urgent messages, and convenient finger table updates. Another field concerns the effects of churn and the super-peer overlay's practical deployment in PlanetLab.

References

1. Anderson, D.P., Fedak, G.: The Computational and Storage Potential of Volunteer Computing. In: Proceedings of the 6th International Symposium on Cluster Computing and the Grid (CCGrid 2006), pp. 73–80 (2006)
2. Andrade, N., Brasileiro, F.V., Cirne, W., Mowbray, M.: Automatic grid assembly by promoting collaboration in Peer-to-Peer grids. Journal of Parallel and Distributed Computing 67(8), 957–966 (2007)
3. Chakravarti, A.J., Baumgartner, G., Lauria, M.: The organic grid: self-organizing computation on a Peer-to-Peer network. IEEE Transactions on Systems, Man, and Cybernetics, Part A 35(3), 373–384 (2005)
4. Kondo, D., Taufer, M., Brooks, C.L., Casanova, H., Chien, A.A.: Characterizing and Evaluating Desktop Grids: An Empirical Study. In: Proceedings of the 18th International Parallel and Distributed Processing Symposium, IPDPS 2004 (2004)
5. Iamnitchi, A., Foster, I.T.: A Peer-to-Peer Approach to Resource Location in Grid Environments. In: Nabrzynski, J., Schopf, J.M., Weglarz, J. (eds.) Grid resource management: state of the art and future trends, pp. 413–429. Kluwer, Dordrecht (2004)
6. Jesi, G.P., Montresor, A., Babaoglu, Ö.: Proximity-Aware Superpeer Overlay Topologies. In: Keller, A., Martin-Flatin, J.-P. (eds.) SelfMan 2006. LNCS, vol. 3996, pp. 43–57. Springer, Heidelberg (2006)

7. Yang, B., Garcia-Molina, H.: Designing a Super-Peer Network. In: Proceedings of the 19th International Conference on Data Engineering, pp. 49–62 (2003)
8. Zöls, S., Despotovic, Z., Kellerer, W.: On hierarchical DHT systems - An analytical approach for optimal designs. Computer Communications 31(3), 576–590 (2008)
9. Gummadi, P.K., Gummadi, R., Gribble, S.D., Ratnasamy, S., Shenker, S., Stoica, I.: The impact of DHT routing geometry on resilience and proximity. In: Proceedings of SIGCOMM, pp. 381–394 (2003)
10. Lua, E.K., Crowcroft, J., Pias, M., Sharma, R., Lim, S.: A survey and comparison of peer-to-peer overlay network schemes. In: Communications Surveys & Tutorials, vol. 7(2), pp. 72–93. IEEE, Los Alamitos (2005)
11. Stoica, I., Morris, R., Karger, D., Kaashoek, F.M., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: Proceedings of SIGCOMM, pp. 149–160 (2001)
12. Cox, R., Dabek, F., Kaashoek, M.F., Li, J., Morris, R.: Practical, distributed network coordinates. ACM SIGCOMM Computer Communication Review 34(1), 113–118 (2004)
13. Merz, P., Priebe, M., Wolf, S.: Super-Peer Selection in Peer -to- Peer Networks Coordinates. In: Proceedings of the 3rd International Conference on Internet and Web Applications and Services (ICIW 2008), pp. 385–390. IEEE Computer Soceity, Los Alamitos (2008)
14. Castro, M., Druschel, P., Hu, Y.C., Rowstron, A.I.T.: Topology-aware routing in structured peer-to-peer overlay networks. In: Schiper, A., Shvartsman, M.M.A.A., Weatherspoon, H., Zhao, B.Y. (eds.) Future Directions in Distributed Computing. LNCS, vol. 2584, pp. 103–107. Springer, Heidelberg (2003)
15. El-Ansary, S., Alima, L.O., Brand, P., Haridi, S.: Efficient Broadcast in Structured P2P Networks. In: Kaashoek, M.F., Stoica, I. (eds.) IPTPS 2003. LNCS, vol. 2735, pp. 304–314. Springer, Heidelberg (2003)
16. Merz, P., Gorunova, K.: Efficient Broadcast in P2P Grids. In: Proceedings of the 5th International Symposium on Cluster Computing and the Grid (CCGrid 2005), pp. 237–242 (2005)
17. Demers, A.J., Greene, D.H., Hauser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H.E., Swinehart, D.C., Terry, D.B.: Epidemic algorithms for replicated database maintenance. In: Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing (PODC), pp. 1–12 (1987)
18. Perkins, C.E., Bhagwat, P.: Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for mobile computers. In: Proceedings of SIGCOMM, pp. 234–244 (1994)
19. Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M., Bowman, M.: PlanetLab: an overlay testbed for broad-coverage services. ACM SIGCOMM Computer Communication Review 33(3), 3–12 (2003)
20. Banerjee, S., Griffin, T.G., Pias, M.: The Interdomain Connectivity of PlanetLab Nodes. In: Barakat, C., Pratt, I. (eds.) Proceedings of the 5th International Workshop on Passive and Active Network Measurement (2004)
21. Sistla, K., George, A.D., Todd, R.W.: Experimental analysis of a gossip-based service for scalable, distributed failure detection and consensus. Journal of Cluster Computing 6(3), 237–251 (2003)
22. Kleis, M., Lua, E.K., Zhou, X.: Hierarchical Peer-to-Peer networks using lightweight superpeer topologies. In: Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC 2005), pp. 143–148 (June 2005)

Replication in Peer-to-Peer Systems*

Mirko Knoll, Haitham Abbadi, and Torben Weis

University of Duisburg-Essen, 47057 Duisburg, Germany
firstname.lastname@uni-due.de

Abstract. We present a replication algorithm for peer-to-peer networks that automatically adapts to an increase of requests. If some information is suddenly very popular, the algorithm distributes it in the peer-to-peer system and reduces the replicas when the demand decreases. The algorithm is totally self-organizing, i.e. it does not need any administration and is very resilient to node failures. Furthermore, our algorithm uses the concept of geographical proximity. Data is preferably replicated on peers which are geographically close. This is especially useful for location-based information, such as traffic information, tourist data and weather alerts.

1 Introduction

Peer-to-peer systems (p2p) are known to scale well with respect to the amount of data offered by the system, thus they work perfectly for large video files or software downloads. However, current systems fail to adapt to "hot topics", i.e. if a certain information suddenly attracts many users, the system should replicate the data on an increasing number of peers [1]. Once the interest in the topic has passed the peak, the number of replicas can be reduced again. This is a serious issue: an open-source project offering high-resolution pictures on its website "suffers" from a post on a major blog (e.g. engadget.com or slashdot.org). The same may happen to a news website offering videos about special events (e.g. an important soccer game). Both services are rendered unavailable in a matter of hours, as the amount of request shortly surpasses the capabilities of the respective service providers. However, a few days later this information will hardly be requested any more.

The p2p replication mechanism we are presenting in this paper automatically detects and replicates highly requested information. Furthermore, our system is very resilient towards peer failures, because there is no central point of failure. In contrast, systems such as Bittorrent [2] can easily be taken down by stopping the so-called tracker. Our system is totally self-organizing. Thus, a highly requested topic cannot become unavailable because a small set of peers goes offline.

Our approach bases on Geostry [3] considering locality of information. This allows us to store and retrieve information for geographical regions. The replication mechanism presented in this paper always tries to replicate data on peers

* The presented work has been funded by DFG SPP 1183 "Organic Computing" and DFG "Emmy Noether" Program.

geographically close to the origin of the data. In the example of the soccer game, this information will preferably be stored on peers near the game. One reason for this is to reduce overall network load. For example, there is no reason to host game information about Cologne on a peer in Madrid (unless it's Cologne vs. Madrid). Another reason is to avoid free riders [4], i.e. peers who download but do not intend to upload data. If a peer computer hosts data that is relevant to the owner of the computer, the owner has an incentive to offer some of his network capacities for the topic.

Replication in peer-to-peer systems is fundamentally different from replication in server-based environments. Servers are costly and therefore they are ideally under load most of the time. In contrast, a peer-to-peer system consists of thousands of personal or mobile computers which are idle most of the time. Thus, it is relatively easy to find an idle peer. The major challenge in our scenario is the self-organization, i.e. automatically detecting and replicating hot topics and dealing with peers that suddenly leave the network.

Our paper is structured as follows. In the next section, we detail goals and challenges of our algorithm, followed by a brief introduction of the underlying peer-to-peer system Geostry in section three. In section four, we present our algorithm and evaluate it in section five to show how it adapts to various scenarios. Section six gives a detailed overview of the related work and section seven closes with a summary, and an outlook on future works.

2 Replication Goals and Challenges

In this section we describe the requirements that should be met by a replication technique which is p2p-based and location-aware. Thereby, we illustrate the challenges that arise when using replication in peer-to-peer systems. General requirements for peer-to-peer systems [5] (e.g. scalability, efficiency...) are covered by the peer-to-peer system itself and therefore not regarded in this paper.

1. Availability: Objects need to be accessible at all times. Therefore, replicas need to be distributed over the network, thus minimizing the effect of network failures.
2. Durability: The main objective of durability is that information should not get lost. To strengthen this goal, data can be replicated to redundant locations.
3. Flash-Crowd-Proof: When a specific information abruptly receives a burst in popularity, the host often becomes unavailable due to the sudden increase in traffic. The server suffers from the so-called *slashdot-effect* [6,1]. To cope with this effect, counter-measures have to be initiated in time.
4. Fast Discovery: Having many replicas does not necessarily improve the overall system performance. It is crucial to discover replicas efficiently.
5. Economic: A fast discovery can be easily implemented with a high number of replicas. However, an intelligent and economic replication scheme should only generate as much replicas as needed to satisfy all incoming request.

6. Commitment: Peers implicitly have to accept to host information from foreign peers. Without this commitment peer-to-peer networks suffer from the freerider problem. On one hand we rely on the enthusiasm to use our system while on the other hand we refer to [4] to counter freeriders.

3 Geostry

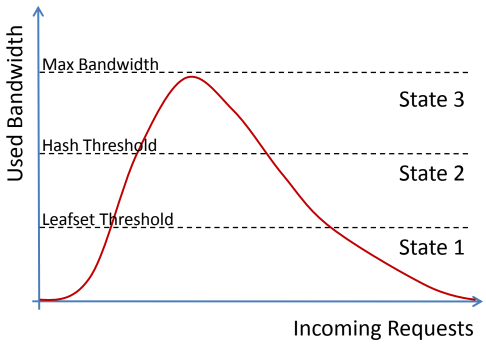
In [3], [7] we presented Geostry, a peer-to-peer system for context-based information. Geostry bases on Pastry [8] and thus belongs to the latest generation of p2p systems. For addressing a value Distributed Hash Table (DHT) systems mostly use a 128-bit identifier. In Pastry, these IDs are generated by applying a hash function to the IP address, which guarantees unique IDs. Given a message and a key, Pastry routes a message to the node with the node ID numerically closest to the key. In each routing step the message reaches a node sharing an ID-prefix (with the destination object) at least one digit longer, therefore reaching the target in $O(log(N))$. As Pastry hashes over the IP address, a numerically small hop may actually lead to a huge jump in terms of geographic distance.

In contrast to Pastry, with Geostry we want to achieve locality of data. Peers which are geographically close to each other in the real world should therefore exhibit numerically close IDs in the node ID space. Therefore, we need to link the node IDs to the physical location of the node in a special way. In [7] we have shown how to use space-filling curves to achieve that goal. Furthermore, nodes in Pastry, as well as in Geostry, use a leafset to keep track of nodes with similar node IDs. As Geostry implements locality, the leafset nodes (*leafs*) are also in the near vicinity of the node. In the next section we will illustrate the use of these nodes for replication purposes.

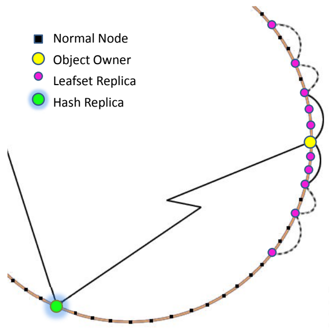
Additionally, the combination of locality and Distributed Hash Tables leads to fast converging routes. That is, the distance per hop decreases in the direction of the target node, quickly putting the query geographically close the original provider of the data. This feature as well may be used for a fast information retrieval, as can be seen in Section 4.5. In Geostry, the majority of peers runs on a rather low load. This derives from the fact, that at one point in time only a few peers offer highly interesting information. The main goal of the replication mechanisms in Geostry therefore lies in dampening the peaks when many peers request data from a single peer. In contrast, data center (e.g. Amazon) administrators usually expect their servers to run on continuously high loads (Google [9] talks about 70-80%). Due to these different assumptions, we cannot adopt the replication mechanisms used in data centers.

4 Replication Design

In Geostry as in other Distributed Hash Table-based p2p systems, nodes use a hash function to calculate which peer is responsible for hosting the desired information. More precisely, a peer applies the hash function on a topic or a keyword, which leads to a *key*. In the next step, the peer generates a query



(a) Replications States



(b) Replication Distribution

Fig. 1. Illustration of Geostry's replication behavior

with this key as a target address. The peer with the node ID closest to that target address - the so called *creator* - is responsible for the sought-after topic respectively keyword. Depending on the state of the peer the query is sent to, different actions need to be taken.

4.1 State 1: Direct Object Access

In our peer-to-peer system, we subdivide the replication process into three states (see Figure 1(a)). At the beginning, the *creator* is able to satisfy all incoming requests req_{in} on its own. Therefore, no replication is needed in state 1.

4.2 State 2: Leafset Redirection

With an increasing interest in the information provided by a specific host, the load increases as well. If no special action is taken, the load will increase until the point where the bandwidth per request is getting lower and lower while new requests cannot be answered at all. Therefore, as soon as the amount of incoming requests reaches the *LeafsetThreshold*, the data is replicated on the leafset nodes indicating state 2 (see Algorithm 1). Further incoming requests are not answered anymore, but redirected to a node from leafset. For fairness reasons we use the round robin method when choosing the redirection peer. In doing so, the *creator* distributes the load to peers in the near geographical surrounding. Wolfson's [10] showed that requests for geographical-related information mostly originate from the immediate vicinity. By distributing replicas to *leafs* - which in Geostry are the closest peers nearby - we take advantage of this theorem. At this state, the *creator* may leave the network as one of his *leafs* will take its place.

4.3 State 3: Timeout - Deterministic Alternatives

In the third state, the amount of incoming requests surpasses the limit of requests that a peer can answer and redirect. Thus, some requesting peers will encounter a

Table 1. Basic procedures during Replication

`UsedBandwith(`req_{in}`)`	Returns the bandwidth for handling all incoming requests
`SendReplica(`*peer*`)`	Sends a snapshot to a given *peer*
`HashKey(`*it, data*`)`	Returns the it^{th} hash value to a given datum
`FreeReplica(`*id, peer*`)`	Allows a *peer* to delete a replica with a specific *ID*
`SendLowUsage(`*peer*`)`	Informs a given *peer* about a decreased request rate
`SendAck(`*bool, peer*`)`	Allows/Denies a *peer* to leave the replica group
`SendDeny(`*peer*`)`	Informs the *creator* about its high load situation

Algorithm 1. Check whether to start Leafset Replication

```
1: procedure CheckLeafsetReplication()
2: if (UsedBandwidth() > LeafsetThreshold)
   & (UsedBandwidth() < HashThreshold) then
3:    for all (Peer in Leafset) do
4:       SendReplica(Peer);
5:    end for
6: end if
```

timeout while waiting for the *creator* to answer. As the requesting peers cannot guess the *creator*'s leafset, they need a well-known algorithm to find another peer with a replica. To do so, the requesting peer applies the hash function: `HashKey(`*1, topic*`)=`*ID1* → `HashKey(`*2, topic*`)=`*ID2* and so on. Each hash operation results in a new key. This peer which is responsible for the new key located far away, we call *remote*. This *remote* itself will start in state 1 and shift to state 2 in case the incoming requests continue to increase.

The amount of replicas (*NrOfRemotes*) that are generated at remote peers can be configured dynamically at the creator (see Algorithm 2), eliminating the need for *remotes* to switch to state 3. To guarantee a fair load distribution, we apply the SHA-1 [11] algorithm for the hash operations. Figure 1(b) illustrates how replicas are distributed on the p2p ring. For this paper, we assume all nodes have homogeneous capabilities. However, we can easily increase *NrOfRemotes* to minimize the probability of choosing a weak node as *remote*.

In the unlikely event, that a chosen *remote* is already running on high load, it may send a `SendDeny(`*creator*`)` message. In doing so, the *creator* simply applies the `Hashkey(`*i+1, topic*`)` again to find the next suitable peer to take the role of a *remote*. However, as we assume that the majority of peers runs on low loads, this case is rather uncommon.

4.4 Replication Clean-Up

As the disk space on all participating peers is finite, we have to limit the degree of replication over time. Without any further limitations, replicas would remain on a multitude of peers though the demand for this specific information has decreased or even ceased completely. On the other hand we want to guarantee

Algorithm 2. Check whether to start Hash Replication

```
1: procedure CheckHashReplication()
2: if UsedBandwidth() $>$ $HashThreshold$ then
3:     for $(i = 0, i < NrOfRemotes, i++)$ do
4:         $remote[i]$ := HashKey($i$, $datum$);
5:         SendReplica($remote[i]$);
6:     end for
7: end if
```

durability. Therefore, we have to find a trade-off between the amount of replicas and the time the replicas are being hold at the foreign peers.

The *creator* solely decides on the amount of *remotes*, depending on the estimated required bandwidth. After this, the *remotes* act autonomously and decide on how many *leafs* they themselves send the replica to. In doing so, the *remotes* fairly distribute the load to their neighbors. Considering the overall load for a specific datum it is the highest at the *creator*, followed by its *leafs* and then the *remotes* with their *leafs*. This makes it easy to adapt the amount of replicas to the actual need in the network.

As the *remotes* decide locally when to forward replicas to their *leafs*, they can also delete these replicas. In case the amount of arriving queries at the *remote* drops under a certain threshold *FreeLeafThreshold*, it sends a `Free(replica)` message to some or all of its *leafs*. This allows them to destroy their replicas and free their memory. Later on, when the amount of queries has dropped under the *FreeRemoteThreshold*, the *remote* informs the *creator* about this circumstance (`SendLowUsage(creator)`) and waits for a reply. If the load at the *creator* is low enough such that the *creator* can handle the *remote's* incoming requests itself, it accepts the *remote*'s request for leaving the replica group (`SendAck(true, remote)`). Then, the *remote* frees its memory and makes it available for other replicas. If the load at the *creator* is still too high, it denies the request and the *remote* stays in the replica group.

4.5 Fast Converging Routes

As we mentioned before, Geostry features fast converging routes [7]. This feature results from the prefix-routing and the locality of data. The first hops will differ greatly in the first few bits, resulting in a large geographical distance being traveled between these hops. As the key of the peers the query is passing gets closer to the key of the target node, the geographical distance between the hops gets lower.

For our replication purposes we can take advantage of this characteristic. After the *creator* of a well-queried information changed to state 2, information is replicated to other nodes of his leafset. These *leafs* are chosen as they are listed in the *creator's* leafset and thus are in its near geographical surrounding. This in turn means, that the ID of those nodes does not differ a lot from the *creator*. In doing so, we replicate information around the location it is about.

This allows the *creator* to leave the network without rendering its information unaccessible. Furthermore, there is the effect of the fast converging routes. As the amount of queries for this specific information increases, queries eventually pass by nodes, which already store a replica. This is due to the routing protocol in Pastry. Peers, who do not know the *creator* (respectively have an entry in their routing table) forward the query to a peer with a node ID which is closer than their own. Thus, the query possibly hits a *leaf* and does not need to be forwarded any more, but can be answered locally. Thereby, in addition to the replication itself, the load on the *creator* can be further reduced.

5 Evaluation

To document the adaptability under varying load situations we implemented a simulator in C#. We consecutively generate 1000 nodes and integrate them into the overlay using Pastry's *Join* method. Thereby, the routing tables of each peer are getting filled and each peer keeps an average of 6 connections to other peers. Incoming requests are answered directly by the responsible peer (called the *creator*) if possible. If more requests arrive, they are redirected to the *leafs* until the peer processes up to 30 requests in parallel. From this point on, queries remain unanswered and their senders thus have to use the `HashKey(topic)` method to find a suitable *remote*. In our evaluation peers need between 30 and 180 ms to process a query, resulting in an average throughput of 9.5 requests/second. For the evaluation we assign eight *leafs* to each peer and calculate four *remotes*.

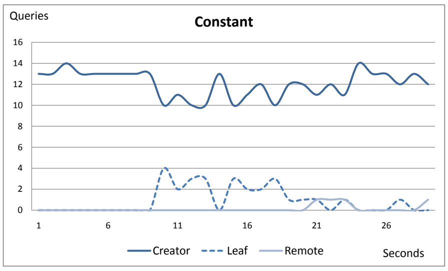
5.1 Scenarios

Our evaluation is based on the three most typical usage patterns.

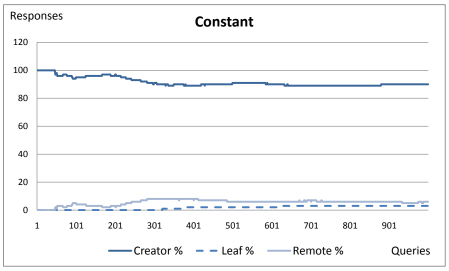
Constant: By simulating a nearly constant rate of interest for a specific piece of information, we can simulate the general behavior for information of average interest. The request rate for a specific piece of information in this scenario is cycling between 12 and 14 requests/second.

Wave: The interest in some data changes periodically. For example, traffic data is most relevant during the rush hour and almost obsolete at midnight. This leads to a wave-like behavior in regular recurrences. We simulated this by using a request rate cycling periodically between 10 and a 100 requests/second.

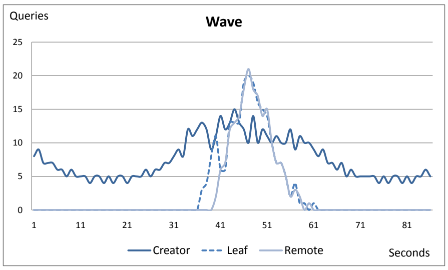
Surge: In case a certain piece of information becomes important for a large group of peers (e.g. an earthquake happened and lots of users want to check for available information about it), the request rate will erratically rise far above the capabilities a peer can handle. Therefore, all states of the replication process will be used. To simulate this behavior, we set the request rate to 8.3 requests/second at the beginning, which is then rapidly rising to a maximum of 50 requests/second.



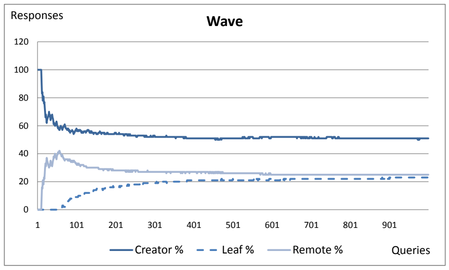
(a) Load Distribution: Constant



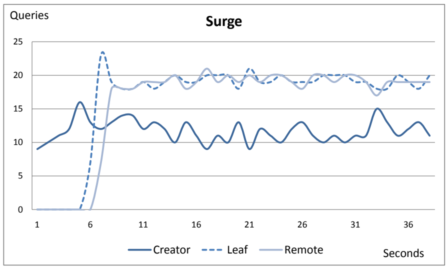
(b) Accumulated Constant



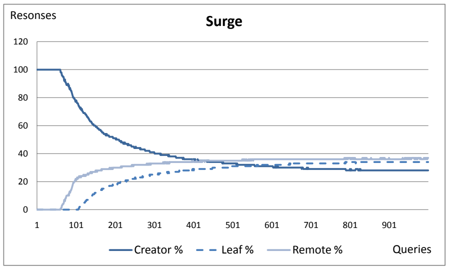
(c) Load Distribution: Wave



(d) Accumulated Wave



(e) Load Distribution: Surge



(f) Accumulated Surge

Fig. 2. Replication simulation using 1000 peers

5.2 Results and Analysis

One of the major challenges we address in this paper is the ability to adapt the replication to various different scenarios. We had six runs per scenario with a duration of 15 minutes each, all showing the same behavior. A measurement as depicted in Fig. 2 illustrates how often a request was handled by each "role".

We have classified these scenarios into three categories according to their stress level. As we can see in Fig. 2(a), a constant low level of queries leads to the fact that the *creator* is able to handle most of the queries on its own. After eight seconds, the time to answer a query takes longer (simulating more complex queries) and thus a few more peers (*leafs* and *remotes*) are needed to compensate the amount of queries. Fig. 2(b) shows that over time, the *creator* is able to handle around 90% of queries on its own. In the wave scenario (see Fig. 2(c)), we see that the *creator* handles all queries until the interest in a topic gets too high. Then, the *leafs* and the *remotes* take part in the replication process and answer queries, too. As the interest decreases, the replica clean-up phase starts and removes replicas from *leafs and remotes*. Thus, the *creator* solely answers all queries, not needing further peers any more. On average, the *creator* answers around 50% itself (see Fig. 2(d)), leaving the remainder for the *leafs* and *remotes*. The surge scenario (see Fig. 2(e)) shows what happens in times of high loads. All participating peers run at their limit to answer the incoming queries. In general (see Fig. 2(f)), the *creator* answers even less queries than the peers hosting the replicas. The kind of peer answering a query changes in relation to the current load. In times of low loads, the *creator* handles all traffic, whereas in times of high interest in its data, the *leafs* and *remotes* add to the *creator's* bandwidth and answer a major part of the queries. This shows, that our replication scheme is able to adapt to the required bandwidth and processing load.

6 Related Work

Research in the area of replication produced some interesting techniques. Our approach does not fall completely into one of the following categories, but rather combines concepts of them with new aspects.

6.1 Cache-Based Replication

In most p2p protocols intermediate nodes participate in the object transfer and query forwarding [12] process (e.g. routing). These protocols include e.g. OceanStore [13] and Freenet [14]. Observing passing objects and queries enables the creation of copies of popular objects for caching reasons. This reduces the response time and transfer overhead and may also reduce the load on the original *creator* of the object. One of the major drawbacks of this approach is the fact that a large hop count between two peers nodes will result in replicas on all nodes along the way. This leads to a large storage overhead, which is especially critical at nodes with low storage capacities.

6.2 Active Content Replication

Active content replication (ACP) was designed to achieve the goal of increasing availability, performance and locality of data. ACP is common, but not limited to unstructured overlays where a non-deterministic search is used. It is used in protocols like Scan [15], Freenet, and Gnutella [16], where it is also known as dynamic or proactive replication. Scan applies the smart dynamic replication strategy, which performs best in terms of bandwidth consumption. The Gnutella protocol [17,18] had a rather statistical approach to assign replication weights to each node. Comparing p2p network performance in case of uniform replication on one hand and linearly proportional replication related to popularity on the other hand, the replication designers have concluded that an intermediate solution through proportional square-root replication will achieve best performance and least replication overhead [19]. The focus of ACP however lies on unstructured overlays and is hard to implement in DHT-based systems.

6.3 OceanStore's Replica Management

OceanStore, a distributed storage architecture built on Tapestry [20], uses intelligent replication management to keep efficient and consistent storage and versioning services. At the same time, it uses security mechanisms to keep object integrity and provide secure access to data.

The focus of OceanStore is on the persistence of data i.e. preventing data loss. However, it does not provide mechanisms to dynamically adapt to changing request rates. Each version of data is stored in a read-only form, which is encoded using erasure-codes and then spread over thousands of servers. In a flash-crowd scenario, OceanStore produces high costs for the generation of these erasure-encoded fragments. To satisfy the requests the consistency restrictions may be lowered, however clients will receive older versions then. Additionally, the created fragments cause an overhead through the additionally needed storage [21], thereby violating the efficiency requirement.

6.4 Initial Factor Assignment

PAST [22] uses location randomization to place copies of its replicas. This results in geographically diverse locations and ownerships for replicas. PAST differs from the other protocols in adding an initial replication factor k to each object. This factor depends on the estimated availability and persistence requirements for a certain object. The k peers having the closest ID to the most significant bits of the object identifier shall maintain replicas of this object. However, the assignment of an initial factor at the insertion time poses to be the drawback of this schema as it is not able to consider future changes for an object's popularity.

6.5 Static Replication Schemes

Chord [23] - not providing a replication by itself - however does provide an infrastructure suitable for replication. These successor lists contain the nodes succeeding the peer's key. The application using the replication can decide on the amount of entries it needs. According to the protocol, the node keeps track of these nodes and update the successor lists as they join and leave the network. This allows for a simple replication of data to the peers in the successor list, however without any geographic relation.

CAN [24] is another well-analyzed p2p system. In CAN, the ID space is divided in so-called zones, where there is at least one peer per zone. CAN offers two mechanisms, which can be exploited for replication. On the one hand, CAN supports realities, allowing in each reality a different peer is responsible for a certain piece of information. On the other hand, CAN can be configured such that it allows multiple peers in one zone, each one responsible for the same data. Though this procedure allows for load-balancing, it is not dynamic and lacks mechanisms to minimize the effect of peaks.

7 Conclusion

Solving the replication problem within peer-to-peer systems is a challenging task. The adaptation to flash-crowds requires a dynamic replication scheme, which also provides robustness in terms of node churn. In this paper we presented a solution - solely based on local decision making - to overcome this challenge. Thereby, we combine existing concepts from PAST and enhance them by an efficient replica distribution and a dynamic replication factor.

In the event of a high interest in a certain topic, data is replicated to nearby nodes at first. If the arising load exceeds the bandwidth, being offered by the*creator* and its leaf nodes, a hash function is used to determine other peers, which will store further replicas. In this case, a peer searching for information about this "hot topic" will possibly receive a timeout when searching for the information at the *creator*. Therefore, it applies a well-known hash function to find other peers, which also provide this information. The amount of replicas self-adapts to the demand, in terms of replicas being generated in times of great demand and deleted when the demand falls. Thus, we can eliminate the effect of peaks and satisfy the replication goals (see Section 2). Though algorithms are optimized for location-based information, due to the underlying Pastry framework it can be easily adapted to standard DHT-based systems.

In the future, we will run more experiments and study the effects of our parameters to optimize the efficiency of the replication process. Furthermore, we plan to overcome the problem of “impatient” peers. In high load situations, these greedy peers send n-queries in parallel instead of waiting for an answer from the first node. We think about using an exponential back-off, growing with the distance from the creator, to cope this problem.

References

1. Elson, J., Howell, J.: Handling flash crowds from your garage. In: USENIX Annual Technical Conference (2008)
2. Cohen, B.: Incentives build robustness in bittorrent. In: Proceedings of the First Workshop on the Economics of Peer-to-Peer Systems, Berkeley, CA, USA (2003)
3. Knoll, M., Weis, T.: A P2P-Framework for Context-based Information. In: 1st International Workshop on Requirements and Solutions for Pervasive Software Infrastructures (RSPSI) at Pervasive 2006, Dublin, Ireland (2006)
4. Porter, R., Shoham, Y.: Addressing the Free-Rider Roblem in File-Sharing Systems: A Mechanism-Design Approach. In: Proceedings of EC 2004, New York, USA (2004)
5. Milojicic, D.S., Kalogeraki, V., Lukose, R., Nagaraja, K., Pruyne, J., Richard, B., Rollins, S., Xu, Z.: Peer-to-Peer Computing. Technical Report HPL-2002-57, HP Laboratories Palo Alto (2002)
6. Homepage, T.S.: What is the slashdot effect? (2000)
7. Knoll, M., Weis, T.: Optimizing Locality for Self-organizing Context-Based Systems. In: de Meer, H., Sterbenz, J.P.G. (eds.) IWSOS 2006. LNCS, vol. 4124. Springer, Heidelberg (2006)

8. Rowstron, A., Druschel, P.: Pastry: Scalable, distributed object loaction for routing for large-scale peer-to-peer systems. In: Proceedings IFIP/ACM Middleware 2001, Heidelberg, Germany (2001)
9. Sanjay Ghemawat, H.G., Leung, S.T.: The google filesystem. In: 19th ACM Symposium on Operating Systems Principles, Lake George, NY, USA (2003)
10. Xu, B., Ouksel, A., Wolfson, O.: Opportunistic resource exchange in inter-vehicle ad hoc networks. In: Proc. of the Fifth IEEE International Conference on Mobile Data Management (MDM), Berkeley, CA, USA, pp. 4–12 (2004)
11. Rijmen, V., Oswald, E.: Update on sha-1. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 58–71. Springer, Heidelberg (2005)
12. Androutsellis-Theotokis, S., Spinellis, D.: A survey of peer-to-peer content distribution technologies. ACM Comput. Surv. 36(4), 335–371 (2004)
13. Kubiatowicz, J., Bindel, D., Chen, Y., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C., Zhao, B.: OceanStore: An Architecture for Global-scale Persistent Storage. In: Proceedings of ACM ASPLOS. ACM, New York (2000)
14. Clarke, I., Sandberg, O., Wiley, B., Hong, T.W.: Freenet: A distributed anonymous information storage and retrieval system. In: Federrath, H. (ed.) Designing Privacy Enhancing Technologies. LNCS, vol. 2009, p. 46. Springer, Heidelberg (2001)
15. Chen, Y., Katz, R.H., Kubiatowicz, J.D.: Scan: A dynamic, scalable and efficient content distribution network. In: Mattern, F., Naghshineh, M. (eds.) PERVASIVE 2002. LNCS, vol. 2414, pp. 145–148. Springer, Heidelberg (2002)
16. Kan, G.: Gnutella. In: Oram, A. (ed.) Peer-to-Peer. Harnessing the Power of Disruptive Technologies, Sebastopol, CA, USA, pp. 94–122. O'Reilly, Sebastopol (2001)
17. Kirk, P.: The annotated gnutella protocol specification v0.4 (2003)
18. Saroiu, S., Gummadi, P.K., Gribble, S.D.: A measurement study of peer-to-peer file sharing systems. In: SPIE/ACM Conference on Multimedia Computing and Networking (MMCN) 2002, San Jose, CA, USA (2002)
19. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer networks. In: International Conference on Supercomputing (ICS 2002). ACM, New York (2002)
20. Zhao, B.Y., Huang, L., Stribling, J., Rhea, S.C., Joseph, A.D., Kubiatowicz, J.D.: Tapestry: a resilient global-scale overlay for service deployment. IEEE Journal on Selected Areas in Communications 22(1), 41–53 (2004)
21. Rhea, S., Eaton, P., Geels, D., Weatherspoon, H., Zhao, B., Kubiatowicz, J.: Pond: The oceanstore prototype. In: FAST 2003: Proceedings of the 2nd USENIX Conference on File and Storage Technologies, pp. 1–14. USENIX Association, Berkeley (2003)
22. Rowstron, A., Druschel, P.: Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. In: 18th ACM SOSP, Lake Louise, Alberta, Canada (2001)
23. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: SIGCOMM 2001: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, San Diego, CA, USA, pp. 149–160. ACM Press, New York (2001)
24. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. In: Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM), San Diego, CA, USA, pp. 161–172. ACM Press, New York (2001)

Validating Peer-to-Peer Storage Audits with Evolutionary Game Theory

Nouha Oualha and Yves Roudier

EURECOM, Sophia Antipolis, France
{nouha.oualha,yves.roudier}@eurecom.fr

Abstract. This paper discusses the efficiency of audits as incentives for reducing free-riding in P2P storage applications, that is, lessening the number of peers that can store their data without contributing to the storage infrastructure. Audits are remote data verifications that can also be used to decide whether to cooperate with a given peer based on its behavior. We demonstrate how an audit-based strategy can dominate or outperform the free-riding strategy by exhibiting the equilibria of our evolutionary game theoretical model of a P2P storage application.

Keywords: evolutionary game, trust establishment, audit, cooperation, P2P storage.

1 Introduction

Peer-to-peer (P2P) is an emerging technology in the storage system area whereby data are distributed in a self-organizing manner to multiple peers instead of using a central storage outsourcing server. Such a distributed storage in particular aims at maintaining a reliable storage without a single point of failure, although without the need for an expensive and energy-consuming storage infrastructure as offered by data centers. Peers volunteer for holding data within their own storage space on a long term basis while they expect a reciprocal behavior from other peers. Just like in P2P file sharing, cooperation incentives have to be used to prevent selfish behaviors whereby peers free-ride the storage system by storing data onto other peers without contributing to the storage infrastructure. Selfishness is however likely more attractive in P2P data storage since unduly gained storage might be used for an extensive period of time contrary to bandwidth which is related to only one file sharing operation. Remote data verification protocols have recently appeared (e.g., [2], [5], [6], [7], [8], [9], [10], and [11]) that aim at auditing peers so as to detect misbehaving ones that claim to hold some data which they in fact destroyed. We claim that such auditing protocols form the basis of an efficient cooperation incentive mechanism. Generally, such a mechanism is proven to be effective if it is demonstrated that any rational peer will always choose to cooperate whenever it interacts with another cooperative peer. We evaluate this mechanism using an evolutionary game model describing the evolution of strategies within large populations as a result of many local interactions, each involving a small number of randomly selected individuals. We propose in this paper an evolutionary game model of a P2P storage system that we use to study under which conditions an audit-based strategy wins over self-interested strategies.

The rest of this paper is organized as follows: in Section 2, an outline of the P2P storage system is first provided. An evolutionary game model of such system is described in Section 3 and the solution of the model, the evolutionary stable strategy, is derived. Results obtained through numerical simulation experiments are finally analyzed in Section 4.

2 A P2P Storage System

We consider a P2P storage system in which peers can store their personal data at other peers. The latter, called *holders*, should store data until the *owner* retrieves them. The availability and correctness of stored data is periodically checked by the owner or its delegates, called *verifiers*. However, holders or verifiers may still misbehave in various ways to optimize their own resource usage.

We assume that the storage application enforces a random selection of holders and verifiers (for instance using the method outlined in [13]). This strategy aims at preventing collusions between peers trying to gain an unfair advantage out of the storage system. Behavior assessment relies on an audit-based verification (see [13] for more details) of the cooperation of peers as holders. This assessment is the basis for deciding whether to accept to store or to audit another holder on behalf of a given peer. Audits result from the execution of a remote data possession protocol (see [4]) periodically performed by verifiers. A peer chooses to cooperate with other peers that have proven to be reliable in the past based on audits.

One-stage or repeated games model interactions that only involve the participants to the storage of one piece of data. Modeling the storage of multiple data as a one-stage game would lead to overly complex games otherwise. In contrast, an evolutionary game makes it easy to model the macroscopic aspects of such interactions occurring within populations of peers. An evolutionary game model describes interactions between randomly chosen players, thus practically portraying the random selection of holders and verifiers in the audit-based approach outlined above.

3 Evolutionary Game

We propose in this paper an evolutionary game model of a cooperative storage system with which we endeavor to evaluate under which conditions peers using the audit-based strategy will dominate the system.

3.1 Game Model

In the proposed system, an owner stores data replicas at r holders. It appoints m verifiers for its data replica that will periodically check storage at holders.

The system is modeled as an evolutionary game. According to Friedman [1], "*an evolutionary game is a dynamic model of strategic interaction with the following characteristics: (a) higher payoff strategies tend over time to displace lower payoff* *strategies; (b) there is inertia; (c) players do not intentionally influence other players' future actions*".

One-stage game: The one-stage game represents an interaction between one data owner, *r* data holders, and *m* verifiers randomly chosen. Thus, the considered game players are an owner, *r* holders, and *m* verifiers. The one-stage interaction consists of several phases:

- *Storage phase:* the owner stores data at the *r* holders. At this phase, holders may decide to keep data stored or to destroy them depending on their strategy (see next paragraph "Evolutionary game"). Holders that crash or leave the system without any notice are considered as defectors contrary to our previous work [14].
- *Delegation phase:* the owner sends verification information to the *m* verifiers in order to be able to periodically check data at holders. Whether to cooperate with the owner in verifying data is determined by each verifier's strategy (see next paragraph "Evolutionary game").
- *Verification phase:* a verifier can decide whether the holder has been cooperative based on the results of a verification protocol such as [2] and take potential action depending on its strategy. A verifier whose strategy is to cooperate will send the owner the results it obtained by auditing the holder. A non-cooperative verifier may mimic a cooperative strategy by sending a bogus result. Verifiers are not more trusted than other peers and may lie about verification, for instance reporting an absence of response to a challenge for a cooperative holder. A verifier might also be framed by a malicious holder trying to make it appear as a non-cooperative verifier. Some verifiers may also crash or leave the system, and be unable to communicate results of verifications. The owner therefore cannot determine with certainty whether a verifier chose to adopt a cooperative strategy. One negative result from a verifier is also not enough for the owner to decide that the holder is non cooperative. Such a notification may however be used as a warning that the holder may have destroyed its data. Based on such a warning, the owner would replicate the endangered data, therefore maintaining or even increasing storage reliability to his advantage.
- *Retrieval phase:* the owner retrieves its data from the *r* holders. If one holder destroyed the data, the owner decides on potential action towards that holder depending on its strategy (see next paragraph "Evolutionary game").

Data storage is a long-term process during which several peers may have been storing data from multiple owners; we define the evolutionary game that models our P2P storage application as a sequence of a random number of such simultaneous one-stage interactions.

Evolutionary game: Our proposed game is similar to the game in [2] where players have either the role of the donor or the role of the recipient. The donor can confer a benefit *b* to the recipient, at a cost *-c* to the donor. We consider three roles in our game: owner, holder, and verifier; any peer may play several of these roles throughout the game. In a one-stage game, the owner is considered a recipient, the *r* holders and *m* verifiers are donors. The owner gains *b* if at least one holder donates at a cost *–c*; however if no holder donates then the owner gains βb if at least one verifier donates at a cost $-\alpha c$ ($\alpha \leq 1$) for each verifier. The latter case corresponds to the situation where the cooperative verifier informs the owner of the data destruction, and then the owner may replicate its data elsewhere in the network thus maintaining the security of its data storage.

Holders and verifiers have the choice between cooperating, which we call interchangeably donate, or defecting:

- Cooperation whereby the peer is expected to keep others' data in its memory and to verify data held by other peers on behalf of the owner.
- Defection whereby the peer destroys the data it has accepted to hold, and also does not verify others' data as it promised to.

The peers' strategies that we consider for study are:

- Always cooperate (*AllC*): the peer always decides to donate, when in the role of the donor.
- Always defect (*AllD*): the peer never donates in the role of the donor.
- Discriminate (*D*): the discriminator donates under conditions: if the discriminator does not know its co-player, it will always donate; however, if it had previously played with its co-player, it will only donate if its co-player donates in the previous game. This strategy resembles Tit-For-Tat but differs from it in that both the owner (the donor) and its verifiers may decide to stop cooperating with the holder in the future.

3.2 Observations

Let's consider a scheme (see Fig. 1) inspired from epidemic models which categorize the population into groups depending on their state [12]. Two states are distinguished: "not known" and "known" states. Because of the random selection of holders and verifiers among all peers and given the presence of churn, there are always nodes potentially in the "not known" state.

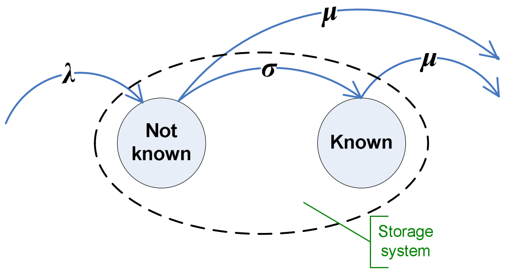


Fig. 1. System dynamics

We denote the number of peers that a given peer in average does not know at a certain time t by D and the number of peers that it knows on average at time t by K. Peers that may join the system are peers who were invited by other members with a fixed invitation rate λ. Peers are leaving the system with a fixed departure rate of μ. The rate σ designates the frequency of encounter between two peers, one of them being the holder (i.e., the probability that a peer knows about the behavior of another peer).

We denote the total number of peers in the storage system - excluding the observing peer - as $n = D + K$. The dynamics of K and D are given by the following equations:

$$\frac{dD}{dt} = \lambda n - (\sigma + \mu)D$$

$$\frac{dK}{dt} = \sigma D - \mu K = \sigma n - (\sigma + \mu)K$$

Since $n = D + K$:

$$\frac{dn}{dt} = (\lambda - \mu)n$$

Let q be the probability that the discriminator knows what a randomly chosen co-player chose as a holder strategy in a previous one-stage game (the discriminator being an owner or verifier in that game). The probability q is equal to K/n, hence:

$$\frac{dq}{dt} = \frac{dK/dt}{n} - \frac{Kdn/dt}{n^2}$$

Thus,

$$\frac{dq}{dt} = \sigma - (\sigma + \lambda)q$$

At time t=0, the set of peers in state K is empty. Over time, peers in state D enter state K with rate σ. A new peer joining the system is assigned state D meaning that initially $q(0)$=0. The result of the above differential equation is thus:

$$q(t) = \frac{\sigma}{\sigma + \lambda}(1 - e^{-(\sigma+\lambda)t})$$

The limit of $q(t)$ when $t \rightarrow \infty$ is $\sigma/(\sigma + \lambda)$. If we consider a system without churn (λ=0), the limit becomes 1.

3.3 Fitness

We respectively denote the frequency (i.e., fraction in the population of playing peers) of strategies *AllC* by x, *AllD* by y, and D by z. The expected values for the total payoff obtained by the three strategies are denoted by U_{AllC}, U_{AllD} and U_D, and the average payoff in the population by:

$$\overline{U} = x \times U_{ALLC} + y \times U_{ALLD} + z \times U_D$$

The average payoffs that are also called fitness for each strategy are defined in the following.

At time t, a participating peer will have r times more chances to be chosen as a holder and m times more chances to be chosen as verifier than to be chosen as an owner.

A peer playing the strategy *ALLC* will always cooperate: it will donate at a cost $-c$ if it is chosen as a holder or at a cost $-\alpha c$ if it is chosen as a verifier. It will gain a benefit b if it is chosen as an owner and at least one of its data holders is not a defector, otherwise, it may gain a benefit βb if at least one of its verifiers is not a defector.

$$U_{ALLC} = -rc - m\alpha c + b(1 - y^r) + \beta b\big(y^r(1 - y^m)\big)$$
$$= -c(r + m\alpha) + b(1 - y^r + \beta y^r(1 - y^m))$$

A peer playing the strategy *ALLD* will never cooperate, so it will never donate. It will gain a benefit b if it is chosen as an owner and at least one of its data holders is not any of these types: a defector (type occurs with frequency, i.e., probability y) or a discriminator that knows the peer (type occurs with probability qz on average). Otherwise, the peer may gain a benefit βb if at least one of its verifiers is not of any of the former two types.

$$U_{ALLD} = b(1 - (y + qz)^r) + \beta b\big((y + qz)^r(1 - (y + qz)^m)\big)$$
$$= b(1 - (y + qz)^r + \beta(y + qz)^r(1 - (y + qz)^m))$$

A peer playing the strategy *D* will always cooperate if it does not know the recipient or the latter was cooperative in a previous interaction. It will donate at a cost $-c$ if it is chosen as a holder or at a cost $-\alpha c$ if it is chosen as a verifier. It will gain a benefit b if it is chosen as an owner and at least one of its data holders is not a defector, otherwise, it may gain a benefit βb if at least one of its verifiers is not a defector.

$$U_D = -c(r + m\alpha)(1 - qy) + b(1 - y^r + \beta y^r(1 - y^m))$$

Strategies with higher fitness are expected to propagate faster in the population and become more common. This process is called *natural selection.*

3.4 Replicator Dynamics

The basic concept of replicator dynamics is that the growth rate of peers taking a strategy is proportional to the fitness acquired by the strategy. Thus, the strategy that yields more fitness than the average fitness of the whole system increases, and vice versa. We will use the well known differential replicator equations:

$$\frac{dx}{dt} = x(U_{ALLC} - \overline{U})$$
$$\frac{dy}{dt} = y(U_{ALLD} - \overline{U})$$
$$\frac{dz}{dt} = z(U_D - \overline{U})$$

3.5 Evolutionary Stable Strategy

A Strategy is said to *invade* a population of strategy players if its fitness when interacting with the other strategy is higher than the fitness of the other strategy when interacting with the same strategy. An evolutionarily stable strategy (ESS) is a strategy which no other strategy can invade if all peers adopt it.

Case $x\neq0$, $y=0$, $z\neq0$: This case corresponds to a fixed point in the replicator dynamics, which means that a mixture of discriminating and altruistic population can coexist and are in equilibrium.

Case $x\neq0$, $y\neq0$, $z=0$: In this case, the replicator dynamics of both altruistic and defector populations are:

$$\frac{dx}{dt} = -xyc(r + m\alpha) \leq 0$$
$$\frac{dy}{dt} = xyc(r + m\alpha) \geq 0$$

The population of defectors wins the game and the ESS is attained at x=0 and y=1.

Case x=0, y≠0, z≠0: There is an equilibrium point for which defectors and discriminators coexist (x=0, y=y_0, z=z_0) which corresponds to:

$$\frac{dy}{dt} = \frac{dz}{dt} = 0$$

The equilibrium point is then solution of the following equation:

$$\begin{aligned} c(r + m\alpha)(1 - qy_0) \\ = b\Big((y_0 + qz_0)^r - {y_0}^r \\ + \beta\big({y_0}^r(1 - {y_0}^m) - (y_0 + qz_0)^r(1 - (y_0 + qz_0)^m)\big)\Big) \end{aligned}$$

Table 1 describes equilibrium values in some particular cases. More cases for equilibrium values will be examined in Section 4.

Table 1. Finding the equilibrium for x=0, y≠0, z≠0

Conditions	y_0	z_0
r=1, m=0, b≠c, $q(t) \xrightarrow[t\to\infty]{} \frac{\sigma}{\sigma+\lambda}$	$\min\left(\max\left(\frac{b\sigma - c(\sigma+\lambda)}{(b-c)\sigma}, 0\right), 1\right)$	$\min\left(\max\left(\frac{c\lambda}{(b-c)\sigma}, 0\right), 1\right)$
r=0, m=1, b≠c, $q(t) \xrightarrow[t\to\infty]{} \frac{\sigma}{\sigma+\lambda}$	$\min\left(\max\left(\frac{\beta b\sigma - \alpha c(\sigma+\lambda)}{(\beta b - \alpha c)\sigma}, 0\right), 1\right)$	$\min\left(\max\left(\frac{\alpha c\lambda}{(\beta b - \alpha c)\sigma}, 0\right), 1\right)$
r=1, m=1, $q(t) \xrightarrow[t\to\infty]{} 1$	$\min\left(\max\left(\frac{c(1+\alpha) - b}{\beta b}, 0\right), 1\right)$	$\min\left(\max\left(\frac{(1+\beta)b - c(1+\alpha)}{\beta b}, 0\right), 1\right)$

Case x≠0, y≠0, z≠0: There is one stationary point (x=0, y=y_0, z=z_0) for which defectors will exploit and eventually deplete all cooperators. The amount of defectors will first increase, and then converges to the equilibrium where there is either coexistence with discriminators, or winning over them, or losing to them depending on storage system parameters.

4 Numerical Evaluation

The evolutionary game is simulated with a custom simulator using the differential equations of subsection 3.4, and within several scenarios varying storage system's parameters.

Initial frequency of strategies: Fig. 2 shows the frequency of cooperators and defectors over time, and demonstrates that with time cooperators will be eliminated from the system by these defectors. The presence of discriminators in the system does not

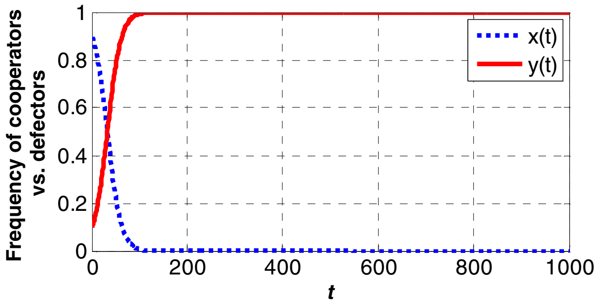


Fig. 2. Frequency of cooperators vs. defectors over time. m=5, r=7, β=0.1, α=0.001, λ=0.01, σ=0.05, b=0.05, c=0.01, x(0)=0.9, y(0)=0.1, and z(0)=0.

prevent cooperators from being evicted from the system; however, discriminators and defectors will converge to an equilibrium where both coexist (see Fig. 3).

This equilibrium is perturbed by the injection of a large population of defectors, as illustrated in Fig. 4 (by varying the initial frequency of z). If discrimination becomes a minor strategy in the population (frequency $\leq$ 0.1), it is completely eliminated from the system. However, if a small population of defectors is injected, discriminators converge to the same equilibrium. So, there are two equilibria that are determined by the initial population of discriminators: (x=0, y=1, z=0) and (x=0, y=y_0, z=z_0).

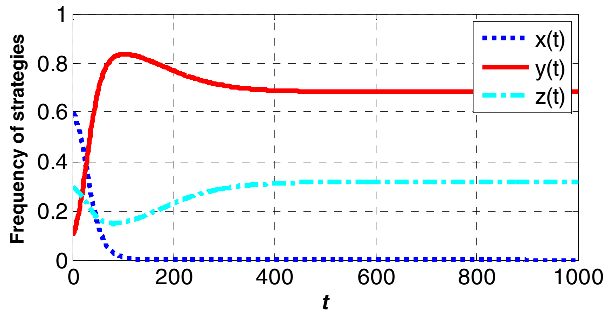


Fig. 3. Frequency of the three strategies over time. m=5, r=7, β=0.1, α=0.001, λ=0.01, σ=0.05, b=0.05, c=0.01, x(0)=0.6, y(0)=0.1, and z(0)=0.3.

The discriminators do not win over defectors, because the latter may still have a good payoff if they interact with some discriminators that do not know them yet, for instance for discriminators that just entered the system, or defectors that just joined in. Additionally, defectors do not always win over the discriminators because there are discriminators that already know them and that always choose to defect with them. The figure shows also a little decrease in the frequency of discriminators before converging to the equilibrium. The decrease is due to the fact that discriminators act as cooperators in the beginning of the game since they do not know the behavior of defectors yet.

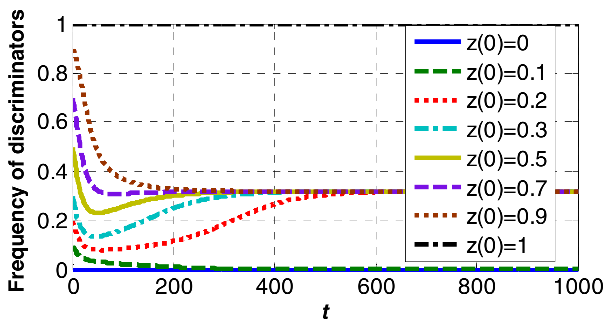


Fig. 4. Frequency of discriminators over time varying $z(0)$. m=5, r=7, β=0.1, α=0.001, λ=0.01, σ=0.05, b=0.05, c=0.01, and $x(0)$=0.

Number of verifiers and replicas: Varying the number r of data replicas or the number m of verifiers changes the equilibrium point. Increasing r favors defectors (see Fig. 5a).

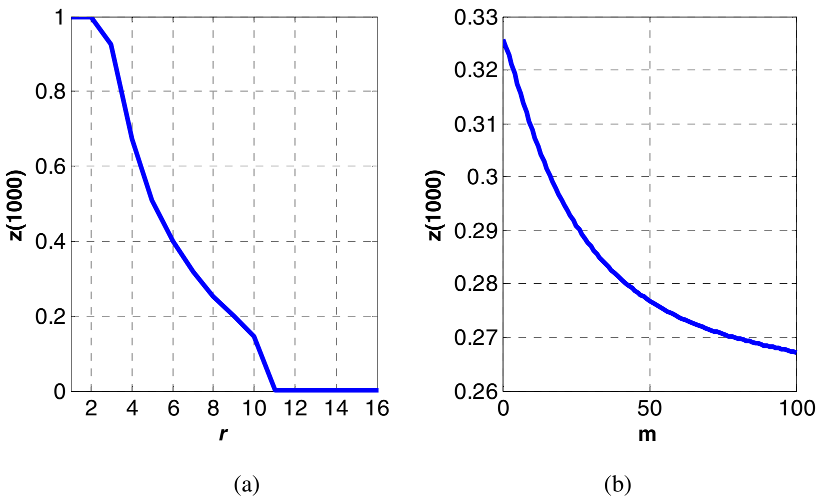


Fig. 5. Frequency of discriminators at equilibrium varying (a) r and m. r=7, m=5, β=0.1, α=0.001, λ=0.01, σ=0.05, b=0.05, c=0.01, $x(0)$=0, $y(0)$=0.5, and $z(0)$=0.5

This is because the fitness gain of discriminating owners is overwhelmed by the fitness loss that results from data storage cost -c that is always paid by discriminating holders. Increasing r increases data reliability, thus increasing chances of having the benefit b. But, this benefit is perceived by both populations of discriminators and defectors without favoring one over the other.

Increasing m decreases the equilibrium value of discriminators frequency (see Fig. 5b). This is due to the fact that increasing m increases the cost of data verification -αc.

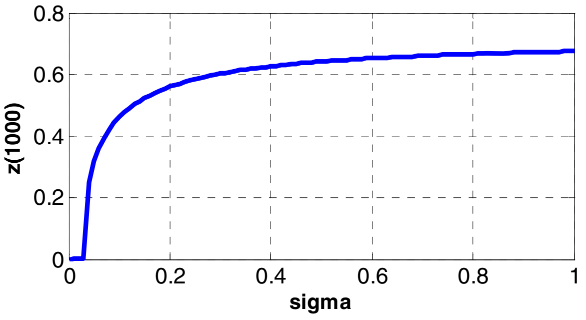


Fig. 6. Frequency of discriminators at equilibrium varying the probability of encounter σ. m=5, r=7, β=0.1, α=0.001, λ=0.01, b=0.05, c=0.01, $x(0)$=0, $y(0)$=0.5, and $z(0)$=0.5

This cost is just paid by discriminating verifiers. That's why increasing m reduces their fitness. Fig. 6 also illustrates the fact that increasing the probability of encounter σ leads to an increase in the equilibrium value of discriminators' frequency because more discriminators get acquainted with more defectors.

Churn: The peer arrival rate λ affects the probability q, and hence the equilibrium point of the game (see Fig. 7). For a low churnout value (small λ), the frequency of discriminators at equilibrium is high; whereas for a high churnout value (large λ) the frequency at equilibrium decreases. For high churnout, peers are not able to get acquainted with all peers since there are always new peers in the system, and defectors may take advantage of the lack of knowledge of discriminators about the system to gain benefit and remain in the game. For a system without churnout (λ=0), discriminators win against defectors that are eliminated from the game.

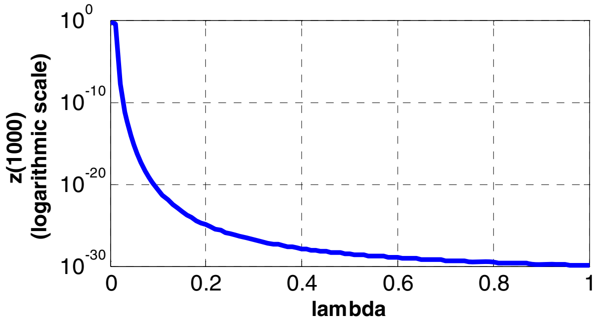


Fig. 7. Frequency of discriminators at equilibrium varying the arrival rate λ. m=5, r=7, β=0.1, α=0.001, σ=0.05, b=0.05, c=0.01, $x(0)$=0, $y(0)$=0.5, and $z(0)$=0.5

Benefit and cost: Fig. 8 depicts the impact of the benefit b and of the cost c on the frequency of discriminators at equilibrium. The figure shows that b and c have opposite effects on the equilibrium frequency of discriminators: increasing b increases the

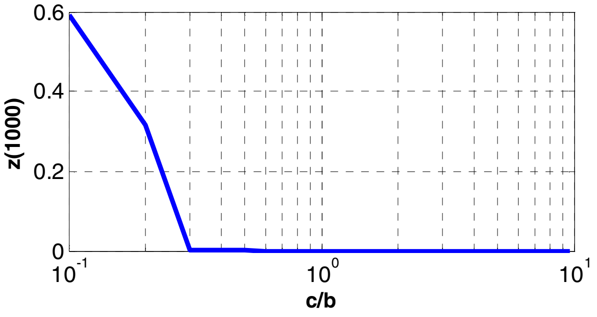


Fig. 8. Frequency of discriminators at equilibrium varying the ratio c/b. m=5, r=7, β=0.1, α=0.001, λ=0.01, σ=0.05, b=0.05, $x(0)$=0, $y(0)$=0.5, and $z(0)$=0.5

frequency whereas increasing c makes it decrease. If the storage cost is small, it will be compensated by the benefit. In contrast, if the storage cost is high ($c \geq 0.3 \times b$), discriminators cannot cope with this high cost and they will be eliminated from the system by defectors.

Summary: Simulation results prove that there exist parameter values for which discriminators, who use an audit-based mechanism, may win against free-riding defectors. Discriminators are not hopeless when confronting defectors, even if the latter may dominate altruists (*always cooperate* strategy). At the equilibrium of the game, both discriminators and defectors may coexist if there is churn in the system otherwise discriminators will dominate. The replication rate r and the number of verifiers m decreases the frequency of discriminators at the equilibrium for a small value for r. Additionally, a costly storage or an increase in the number of verifications performed reduce this frequency.

5 Conclusion

In this paper, we validated an audit-based strategy as an evolutionary stable strategy under some conditions as the basis for a P2P storage system. The results obtained show that such a strategy wins over a free-riding strategy in a closed system. Given reasonable constraints, they also show that this strategy can coexist with free-riders, and even achieve a high frequency. The fact that cryptographic primitives exist that make the implementation of appropriate audit mechanisms possible without unrealistic network bandwidth requirements is noteworthy for practical implementation. We are currently investigating other security issues of P2P storage systems, notably those raised by maliciousness.

References

1. Friedman, D.: On economic applications of evolutionary game theory. Journal of Evolutionary Economics 8, 15–43
2. Brandt, H., Sigmund, K.: The good, the bad and the discriminator–errors in direct and indirect reciprocity. Journal of Theoretical Biology 239(2), 183–194 (2006)

3. Oualha, N., Önen, M., Roudier, Y.: A Security Protocol for Self-Organizing Data Storage. In: Proceedings of 23rd International Information Security Conference (SEC 2008) IFIP Sec 2008, Milan, Italy (September 2008)
4. Oualha, N., Önen, M., Roudier, Y.: A Security Protocol for Self-Organizing Data Storage (extended version) Technical Report N° RR-08-208, EURECOM (January 2008)
5. Oualha, N., Roudier, Y.: Securing ad hoc storage through probabilistic cooperation assessment. In: WCAN 2007, 3rd Workshop on Cryptography for Ad hoc Networks, Wroclaw, Poland, July 8 (2007)
6. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D.: Provable data possession at untrusted stores. In: CCS 2007: Proceedings of the 14th ACM conference on Computer and communications security, pp. 598–609. ACM, New York (2007)
7. Juels, A., Kaliski, B.S.: PORs: Proofs of retrievability for large files., Cryptology ePrint archive, Report 2007/243 (June 2007)
8. Deswarte, Y., Quisquater, J.-J., Saïdane, A.: Remote Integrity Checking. In: Proceedings of Sixth Working Conference on Integrity and Internal Control in Information Systems (IICIS) (2004)
9. Filho, D.G., Barreto, P.S.L.M.: Demonstrating data possession and uncheatable data transfer. In: IACR Cryptology ePrint Archive (2006)
10. Sebe, F., Domingo-Ferrer, J., Martínez-Ballesté, A., Deswarte, Y., Quisquater, J.-J.: Efficient Remote Data Possession Checking in Critical Information Infrastructures. In: IEEE Transactions on Knowledge and Data Engineering, August 06, 2007, IEEE Computer Society Digital Library. IEEE Computer Society (December 6, 2007), `http://doi.ieeecomputersociety.org/10.1109/TKDE.2007.190647`
11. Schwarz, T., Miller, E.L.: Store, forget, and check: Using algebraic signatures to check remotely administered storage. In: Proceedings of the IEEE Int'l Conference on Distributed Computing Systems (ICDCS 2006) (July 2006)
12. Jones, D.S., Sleeman, B.D.: Differential Equations and Mathematical Biology. Allen & Unwin, London (1983)
13. Oualha, N., Roudier, Y.: Reputation and Audits for Self-Organizing Storage. In: Workshop on Security in Opportunistic and SOCial Networks SOSOC 2008, Turkey, Istanbul (September 2008)
14. Oualha, N., Michiardi, P., Roudier, Y.: A game theoretic model of a protocol for data possession verification. In: IEEE International Workshop on Trust, Security, and Privacy for Ubiquitous Computing TSPUC 2007, Helsinki, Finland, June 18 (2007)

A Self-Organized Clustering Scheme for Overlay Networks

Francois Cantin, Bamba Gueye, Mohamed Ali Kaafar, and Guy Leduc

University of Liege, Belgium
{francois.cantin,cabgueye,ma.kaafar,guy.leduc}@ulg.ac.be

Abstract. Hierarchical approaches, where nodes are clustered based on their network distances, have been shown to allow for robust and scalable topology-aware overlays. Moreover, recent research works have shown that cluster-based deployments of Internet Coordinates Systems (*ICS*), where nodes estimate both intra-cluster and inter-cluster distances, do mitigate the impact of Triangle Inequality Violations (*TIVs*) on the distance predictions, and hence offer more accurate internet latency estimations. To allow the construction of such useful clusters we propose a self-organized distributed clustering scheme. For better scalability and efficiency, our algorithm uses the coordinates of a subset of nodes, known by running an ICS system, as first approximations of node positions. We designed and evaluated two variants of this algorithm. The first one, based on some cooperation among nodes, aims at reducing the expected time to construct clusters. The second variant, where nodes are selfish, aims at reducing the induced communication overhead.

Keywords: ICS, Clustering, Triangle Inequality Violations, Performance.

1 Introduction

Recent years have seen the advent of Internet applications which are built upon and benefit from topology-aware overlays. In particular, most if not all of these applications and associated overlays rely on the notion of network proximity, usually defined in terms of network delays or round-trip times (RTTs), for optimal neighbor selection. Recent research has focused on proposing elegant solutions to the problem of proximity retrieval while avoiding algorithms that can prove to be very onerous in terms of measurement overheads, and significant bandwidth consumption (ping storms). In this context, network positioning systems, such as [1,2], were introduced. The key idea is that if each node can be associated with a "virtual" coordinate in an appropriate space, distance between nodes can be trivially computed without the overhead of a direct measurement. Despite the elegance from a theoretical perspective, these systems have some practical limitations. In particular, Internet latencies that do violate triangle inequalities, in the actual Internet [3], degrade both coordinates' accuracy and stability [4,5].

Recent research works, however, have shown that shorter internet paths are less likely victims of severe TIVs. Following these observations, in [5] we evaluated the efficiency of a hierarchical approach for ICS. In this approach, nodes that are near each other are clustered, and an independent ICS runs in each cluster. These independent ICS are used to estimate intra-cluster distances, whereas inter-cluster distances are simply estimated by using an ICS involving all nodes. Since only short paths remain inside the clusters, there are less TIVs in these subspaces. Consequently, the hierarchical ICS offers more accurate latency estimations for intra-cluster distances. More generally, hierarchical overlay approaches, where nodes are clustered based on their network distances, have been shown to allow for robust and scalable network-aware overlays [6,7,8]. In such case, scalability is achieved by drastically reducing the bandwidth requirements and management overhead for overlay maintenance. Moreover, robustness is obtained by mitigating the effect of dynamic environment as most changes are quickly recovered and not seen beyond the clustered set of nodes.

In this paper we propose a self-organized clustering scheme whose goal is twofold. Firstly we address the problem of constructing efficient clusters in an autonomous way by building on an existing ICS system. Secondly our clustering scheme aims at providing a self-managed clustering structure to overlay-based applications, to allow both topology awareness and scalability of these applications. The novelty of our approach lies in simultaneously relying on the partial knowledge of coordinates of nodes involved in ICS operations, and on a distributed clustering algorithm based on adaptive back-off strategy, to construct efficient network topology-aware clusters in a load-balancing way. The main idea is to allow each node to identify a set of clusters in the network, using its own knowledge of a set of nodes' coordinates (as provided by the ICS in which it is involved), and to verify the validity of such clusters using a few measurements towards the identified cluster heads. The distributed algorithm is scheduled using an exponential back-off strategy, where nodes plan their own wake-up time to verify the existence of clusters in their proximity or not. Our main objective behind this strategy is to load balance the clustering process, while adapting to previous clusters creation, and hence optimize the maintenance and measurements overhead.

We provide two variants of our distributed algorithm: a first variant, called "Cooperative", aims at reducing the expected time to construct clusters for the whole network. This approach induces some overhead to inform other nodes that they are likely to belong to a newly created cluster. A second "Selfish variant" is also introduced, where nodes are more selfish and can only form and/or join clusters when they wake up, without any assistance (or guidance) from other nodes that woke up earlier. In both cases nodes use only knowledge provided by a subset of other nodes, in some neighborhood as explained later, and obtain the needed pieces of information (coordinates, existing cluster heads) by piggybacking them in the messages exchanged by the ICS system.

We analyze the performance of our distributed self-clustering algorithm considering clusters efficiency in terms of actual latency existing between members of such clusters, and considering their size. We also observe the time needed to construct efficient clusters, and the induced overhead. By measuring both exchanged messages and measurement rates, we show that our distributed clustering algorithm is fit for purpose, providing a simple, practical and efficient way to build useful topology-aware clusters.

The remainder of the paper is as follows: in section 2, we describe briefly the quality threshold clustering algorithm on which we based our self-organized clustering scheme, and we discuss the reasons that motivate the choice of such an algorithm. In section 3, we introduce the proposed algorithm to allow nodes to identify clusters they belong to. We also discuss the variants of our distributed algorithm, with their pros and cons. Section 4 presents the clustering results in terms of achieved performance, induced overhead and convergence time. Section 5 concludes the paper.

2 QT (Quality Threshold) Clustering Algorithm

Clustering is defined as a process of partitioning a set of elements into a number of groups based on a measure of similarity between the data (distance-based approaches) or relying on the assumption that the data come from a known distribution (model-based approaches). For our self-clustering process, we aim at exploiting nodes' coordinates as a first approximation of the inter-node distances existing in the actual network topology. As nodes' coordinates do not follow any a priori distribution, we will focus on distance-based clustering. Moreover, since we aim at providing a self-clustering process that is performed in a distributed way among all the nodes of the network, the optimal number of clusters that can be created is not known in advance. Approaches that do set a constraint on the number of clusters to be formed (such as K-means, C-Means Fuzzy Clustering, etc.) are thus inappropriate.

Having in mind these facts, we choose to leverage the Quality Threshold algorithm (*QT_clustering*) to propose our self-organized clustering scheme. This algorithm has been initially proposed by Heyer et al. [9] for genetic sequence clustering. It is based on the unique constraint of the cluster diameter, as a user-defined parameter. For the QT_clustering and for the remainder of this paper we define the cluster diameter as the maximal distance existing among any two members of the cluster. The QT_clustering is an iterative algorithm and starts with a global set that includes all the elements (*e.g* node coordinates) of the data set, and then returns a set of clusters that respect the quality threshold. Such threshold is defined in terms of the cluster diameter.

First, for each element, a candidate cluster seeded by this element is formed. Such cluster is iteratively added by other elements. Each iteration adds the element that minimizes the increase in cluster diameter. The process continues until no element can be added without surpassing the diameter threshold. A second candidate cluster is formed by starting with the second element and repeating the procedure. Note that all elements are made available to the second candidate cluster. That is, the elements from the first candidate cluster are not removed from consideration. The process continues for all elements. At the conclusion of this stage, we have a set of candidate clusters. The number of candidate clusters is equal to the number of elements, and many candidate clusters overlap. At this point, the largest candidate cluster is selected and retained. The elements it contains are removed from consideration and the entire procedure is repeated on the smaller set. A possible termination criterion is when the largest remaining cluster has fewer elements than some specified threshold.

3 Self-Clustering Process

In this section we describe how we exploit the QT_clustering algorithm to provide a distributed self-organized clustering process, based on the knowledge of a subset of nodes' coordinates in a metric space, resulting from running a positioning system to estimate network distances. We will denote by (direct) neighbors the set of peer nodes that are used as neighbors in the ICS for the purpose of coordinate computation. We will also denote by *long-sight* neighbors, the union of these (direct) neighbors and the neighbors' neighbors (i.e., node's 2-hop neighbors). For instance, if a node has 32 neighbors in order to estimate its coordinates, its long-sight neighbors will be formed by at most 1024 nodes.

3.1 Description

The general idea of our clustering algorithm is to distribute the clustering tasks among nodes in the network relying not only on measurements towards a potential existing cluster, but also on their knowledge of the coordinates of their long-sight neighbors. In other words, if a node wakes up (with respect to the clusters algorithm) and does not find directly an existing cluster it may belong to, it tries to construct such cluster based on the coordinates as provided by the ICS it is running. In such a way, nodes that do wake up earlier try to create clusters that their peers waking up later may join. Put simply, nodes perform trailblazing of the network conditions, to construct the clusters in a distributed way, while optimizing the needed overhead. Three main advantages could then be considered. Firstly nodes do not need global knowledge of nodes in the network, nor distances between these nodes, nor a common landmark/anchor infrastructure. Secondly the network is not overloaded by measurements performed to obtain the cluster structure. And thirdly the network is able to self-construct the clusters that may exist.

During the cluster forming phase, nodes are initially in a waiting mode. Each node waits for an initiator timer according to an exponential random distribution, computed as described in section 3.2. The clustering process follows the procedure presented in Algorithm 1 and can be described as follows: each time a node wakes up, it gets the list of existing cluster heads in the network. Although such information could be obtained by requesting the set of long-sight neighbors that the node is aware of, we choose to perform this information retrieval by exploiting the communication already established at the level of the ICS. Existing cluster heads are propagated in the network by simply piggybacking in the classical ICS messages the identity of the cluster head(s) of cluster(s) a node belongs to. Considering these already existing clusters, each node verifies its membership to one of them. If the measurement towards the cluster head satisfies the cluster diameter, say D, meaning that such distance is less than $D/2$, the node simply joins such cluster by sending a $JOIN$ message to the cluster head. Our previous study [5] showed that when the cluster diameter does not exceed a fixed threshold (*e.g* 140ms), intra-cluster paths are less likely victims of severe TIVs. Following this observation, for our simulations, we set the upper bound of the cluster diameter D to 140ms. Finding a way to adapt automatically this upper bound to the network is one of our future work. Depending on the maximum number of clusters a node can join, say k, such procedure could be repeated with other cluster heads.

Nevertheless if none of the distances to existing cluster heads satisfies the clustering criterion, the node starts the QT-clustering algorithm on the basis of the coordinates of its long-sight neighbors. It is worth noticing that this clustering is just a first approximation. Indeed coordinates may be subject to distance estimations errors, resulting from inaccuracies in coordinates. However this gives the node an approximate view of its

Algorithm 1. Procedure when a node wakes up

```
1: if The node is already in at least one cluster then
2:     The node goes back to sleep;
3: else
4:     The node gets the list of existing cluster heads (known by its long-sight neighbors);
5:     The node measures RTTs to all existing cluster heads;
6:     Let $C$ be the list of existing cluster heads within a range $RTT < D$;
7:     if $C \neq \emptyset$ then
8:         The node joins at most the $k$ nearest clusters whose heads are in $C$;
9:     else
10:        Let $S$ be the list of coordinates of the node's long-sight (1-hop and 2-hop) neighbors;
11:        The node runs a QT-Clustering on $S \Rightarrow$ This returns a set of clusters;
12:        if The node is in none of these clusters then
13:            The node goes back to sleep;
14:        else
15:            The node selects a cluster head in its cluster;
16:            The node measures the RTT to this new potential cluster head;
17:            if $RTT > D/2$ then
18:                The node goes back to sleep;
19:            else
20:                The node freezes all of its long-sight neighbors (by sending them a message);
21:                if A neighbor answers that it is already frozen by another node then
22:                    The node goes back to sleep;
23:                else
24:                    After a short while the node notifies the selected cluster head and waits for
                       confirmation;
25:                    if Confirmation is positive then
26:                        The node joins the cluster;
27:                    else
28:                        The node goes back to sleep;
```

neighbors positions, and in particular of the clusters that could be formed from this approximation. This first coordinate-based clustering phase allows the node to identify a set of clusters in the metric space of the ICS. This set of clusters is then subject to a verification according to direct measurements.

When a node has verified that its distance to an identified cluster head satisfies the clustering criterion, it decides to inform this potential cluster head that it should create a cluster, and waits for a confirmation. The cluster creation is conditioned by the acceptance of the requested cluster head. In fact, a potential cluster head could refuse to lead a cluster because of load constraints, or more specifically because its actual distance to an already existing cluster head has been considered too short. To this end, when a node is informed that it is a potential cluster head, it measures its distance to the list of cluster heads it is aware of. If at least one of these distances is less than $\alpha \times D/2$, for some $1 < \alpha < 2$, distance between the two cluster heads is considered too short to construct a new cluster, and the request is refused. In this case, the node that identifies this cluster head is informed of this refusal and goes back to sleep. Otherwise, i.e. if the cluster is created, nodes that wake up later follow this decision and consider the cluster head among the list of existing cluster heads.

The algorithm relies on self-organization of nodes. When a node decides to join a cluster, two variants could drive the process of nodes joining the identified clusters. The first variant, with the main goal of speeding the clustering process, is to inform all identified nodes in a cluster of their potential membership to such cluster, and let them check this fact with direct measurements. The second variant trades off the speed of cluster creation against a reduced measurement overhead. In this case nodes never inform others that they may belong to a newly created cluster, and let them discover this fact when they wake up.

Finally it is worth noticing that the wake-up procedure allows also some adaptation to changes in the network. Since distances in the network may evolve over time, including the distances of nodes towards their identified cluster head(s), a node should not stick to any cluster, and should also verify its membership to additional clusters due to new network conditions. Waking up from time to time, following the distributed scheduling as presented in 3.2, allows them to check their membership to existing clusters, and thereby adapt to changes in network conditions.

3.2 Distributed Scheduling of Wake-Up Timers

During the cluster forming phase, nodes are initially in a waiting mode. Each node waits for an initiator timer according to an exponential random distribution, i.e. $f(t_i) = \lambda_i.e^{-\lambda_i.t_i}$, where $\lambda_i = \lambda_0.(n_i/N_i)$; n_i being the number of non already clustered nearby neighbors, and N_i being the total number of known long-sight nearby neighbors. By nearby nodes we refer to nodes whose coordinates indicate that they are (likely to be) within some specified range. To set the timer according to an exponential random distribution, we set p_t = random(0,1), compute λ_i as described above and let $t_i = (-1/\lambda_i).ln(1 - p_t)$. The wake-up timer could then be computed as $timer = min(t_i, MAX\_Timer)$. From the expression of t_i it is obvious that the timer decreases when λ_i increases. Therefore such timer will ensure that the nodes with more residual non-clustered neighbors have more opportunities to (re)initiate the clustering algorithm, since their timer is more likely to elapse before other nodes. The main idea behind this exponential backoff scheduling is to load-balance the clustering process as initiated by nodes in the network, while optimizing the time needed to construct and join the clusters.

We can also expect that in one MAX_Timer period enough nodes initiate the clustering algorithm. To this end the selection of λ should satisfy the following inequation:

$$Prob(t > MAX\_Timer) \leq 1 - p$$

where p is the expected percentage of nodes initiating the algorithm. Therefore, in such a case

$$\int_{MAX\_Timer}^{+\infty} \mathrm{f}(t)dt \leq 1 - p => \lambda \geq -(ln(1 - p)/MAX\_Timer)) \tag{1}$$

Based on (1) we can calculate λ needed to ensure that a percentage of nodes initiate the algorithm, at least in the initial state, when nodes are not clustered yet. From (1) we can conclude that $\lambda_{min} = -(ln(1-p)/MAX\_Timer)$ is sufficient to ensure this.

4 Analyzing the Clusters

In this section we present the results of an extensive simulation study of the self-organised clustering process. We performed a set of simulations using two datasets: the *p2psim* data, a set of 1740 nodes [10] and *Meridian* data, comprising 2500 nodes [11] to model Internet latencies based on real world measurements.

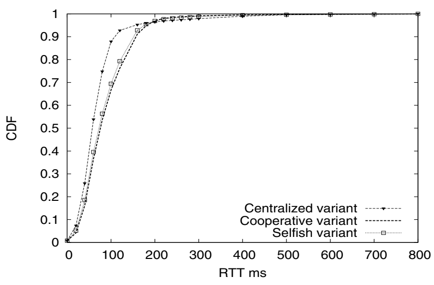
In our simulations, we allowed nodes to join at most two clusters ($k = 2$) and we set the expected maximum cluster diameter D to 140ms for the King dataset and to 80ms for the Meridian dataset, following recommendations in [5]. The maximal timer for a sleeping node is set to 5 minutes and the minimum distance between any two cluster heads is fixed to $3/2 \times D/2$ (*i.e.* $\alpha = 3/2$). As we used coordinates as provided by an ICS in the first step of our self-organised clustering, we deployed the Vivaldi system [2] as a prominent representative of purely P2P coordinate systems. Each node runs the Vivaldi system, setting the number of its neighbors to 32 and our results are obtained for a 2-dimensional coordinate space. In [2], authors show that the more dimensions an Euclidean space has, the more accurate the coordinates are. Similarly, the more neighbors a node has, the more accurate the system is[1]. However, we choose not to illustrate our results by using more accurate coordinates, for ease of deployment and low computing loads, at the cost of some loss in clusters accuracy.

We evaluate the performance of our clustering algorithm with respect to three main indicators. (i) The clusters quality: it is the deviation between the expected cluster diameter and the actual diameter. (ii) The convergence time: it is the time needed by our distributed algorithm to cluster 95% of the nodes in the system. This allows us to differentiate between the initial phase of the algorithm, when clusters are yet in the construction process, and the steady state, when nodes continue to manage their membership to already constructed clusters. Finally, we measured (iii) The overhead: it is the number of exchanged messages and the number of measurements performed. We can further split the overhead during the initial phase and during the steady state. We compare the two variants of our algorithm, and when needed we compare our distributed self-clustering algorithm to a centralized approach. All algorithms used are implemented and evaluated in *R* environment [12].

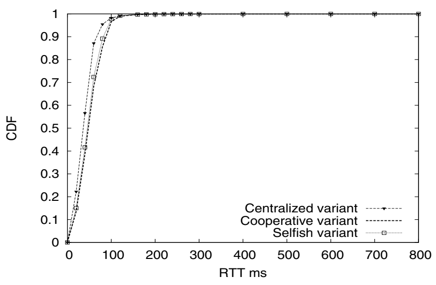
4.1 Clusters Quality

We can evaluate the cluster quality according to the deviation between the expected cluster diameter, as we set it in the QT_clustering and the actual diameter as obtained after our self-organised clustering process reaches its steady state. However, the cluster size is also an important parameter we should mention. A cluster populated with only a few nodes, even though its diameter is optimal, may be of little use.

[1] As shown in [5], the triangle inequality violations phenomena prevents a perfect mapping between latency and coordinates, even for high-dimensional spaces. Coordinates are deemed to be inaccurate.



(a) King dataset



(b) Meridian dataset

Fig. 1. CDF of the RTT of the intra-cluster paths

To evaluate the clustering quality in terms of cluster diameter, we observe in figures 1(a) and 1(b), the Cumulative Distribution of the actual delays (RTTs) between the members of the same identified cluster (called intra-cluster RTTs). These figures show, for both data sets, the proportion of nodes that actually violate the diameter constraint. We compare the proportion of these violations for the two variants of our clustering algorithm, and for a centralized approach. In this case, a centralized approach consists in emulating a centralized entity that collects the coordinates of all nodes in the system, computes in a centralized way clusters resulting from these coordinates using the QT_clustering and then informs all nodes of the identified cluster heads. These nodes verify their membership to these clusters, and join clusters if the diameter constraint with their cluster heads is verified. Otherwise, they are considered as outliers. The main reason why we compare our algorithm to such a centralized algorithm is to evaluate how partial knowledge of neighborhood and coordinates impact our clustering performance.

Figure 1(a) shows that more than 85% of the intra-cluster links satisfy the cluster diameter constraint, with an RTT less than the expected diameter. The same trend is observed in Figure 1(b) for the Meridian dataset, with more than 95% of clustered nodes scattered in delimited clusters, respecting the expected cluster diameter of 140ms. We also note that both variants are achieving the same performance, which is actually not surprising, since the main difference between our two variants is when nodes join a cluster, and not how they join it. The centralized approach creates slightly more accurate clusters. However, this little difference is overwhelmed by onerous cost induced by a centralized approach that needs global knowledge of both coordinates and nodes in the system.

Performing a QT_clustering based on coordinates of long-sight neighbors gives us a first approximation of nodes positioning. Even though nodes measure network distances, as RTTs, towards identified cluster heads, this does not prevent some mutual distances between cluster members to be above the expected diameter, due to TIVs. Using coordinates reduces the proportion of diameter violations, but since coordinates "only" provide distance estimates, with intrinsic errors, errors may always exist.

As shown in Table 1, the number of clusters identified by our algorithm ranges from 9 to 11 for both variants. However, we can in both cases consider 3 main clusters, with an average population of 700 nodes each for the King dataset, and 1260 nodes as an average population of each cluster in the Meridian dataset. The percentage of nodes

Table 1. Characteristics of the clustering process

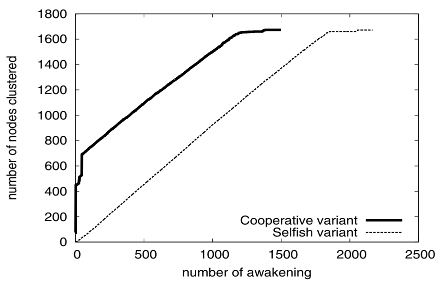
	Cooperative Variant		Selfish Variant	
	King	Meridian	King	Meridian
Number of clusters	9	9	11	9
Number of outliers (unclustered nodes)	67	81	68	102
Total Number of pings	11116	17003	20125	18075
Total Number of messages (excluding pings)	1582	2300	843	246
Convergence time (seconds)	1875	1658	1658	2300
Ping rate before convergence (pings/s)	4.48	8.05	9.95	6.45
Mean ping rate before convergence (pings/node$\times$s)	0.0026	0.003	0.0057	0.0026
Max ping rate before convergence (pings/node$\times$s)	0.027	0.038	0.0398	0.023
Mean msg rate before convergence (msg/node$\times$s)	0.0005	0.0006	0.0003	$4\,10^{-5}$
Max msg rate before convergence (msg/node$\times$s)	0.403	0.635	0.23	0.049
Mean ping rate after convergence (pings/node$\times$s)	0.0002	0.0002	0.0002	0.0002
Max ping rate after convergence (pings/node$\times$s)	0.03	0.032	0.034	0.022
Mean msg rate after convergence (msg/node$\times$s)	10^{-6}	$6\,10^{-7}$	$3\,10^{-7}$	$6\,10^{-7}$
Max msg rate after convergence (msg/node$\times$s)	0.0007	0.0003	0.0002	0.0005

that have not been clustered are roughly 3.8% of nodes existing in the system. The bottom part of Table 1 will be presented later.

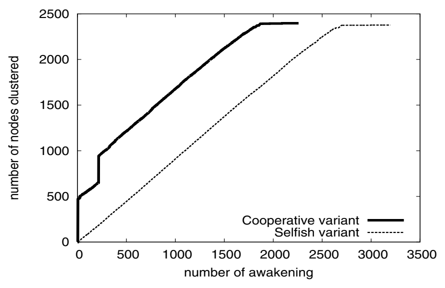
4.2 Convergence Time

To separate the initial phase from steady state, we analyze the evolution of the number of clustered nodes versus the number of awakenings (and hence versus the number of clustering process calls) for both variants. As depicted in Figures 2(a) and 2(b), the curves labeled "Selfish Variant" follow linear evolutions. Such observation is expected since at most one node can join a cluster at each awakening, and then the growth of the number of clustered nodes can be at best linear. Clusters in the "Cooperative" variant may cumulate node membership with each node's awakening, because information of potential membership to a newly created cluster is sent by the creator of a cluster to identified members. In the curves labeled "Cooperative approach", we can then observe for both data sets the steps corresponding to a set of nodes joining simultaneously a defined cluster. Such steps allow this variant to cluster more than 95% of nodes in 1210 awakenings for the King dataset and in 1854 awakenings for the Meridian dataset, whereas the "Selfish" takes more occurrences, up to 2749 awakenings. Such faster clustering comes at the expense of costs of spreading information and exchanged messages, we will discuss in the following section.

Let us consider that our system is in the steady state if at least 95% of existing nodes have been clustered. Although such parameter relies on our a priori knowledge of the number of outliers in the system, and hence on the minimum number of nodes that can be clustered, it gives us however a suitable way to separate the initial phase from the steady state. It is important to notice that the convergence time, although different in terms of number of awakenings, is roughly the same in real time (in seconds) spent to cluster 95% of the nodes, as shown in table 1. This confirms our choice of the



(a) King dataset



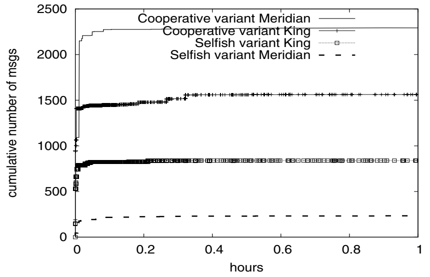
(b) Meridian dataset

Fig. 2. Evolution of the number of clustered nodes

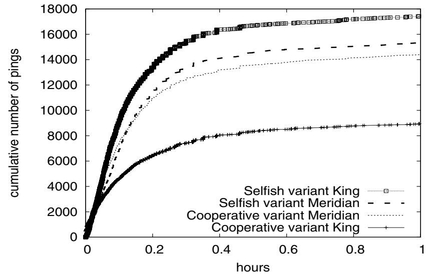
exponential-backoff strategy to set timers. Recall from section 3.2, that our algorithm will ensure that the greater the λ, the lower the timer, giving hence more opportunity to (re)initiate the clustering algorithm. This guarantees a higher probability to initiate the clustering algorithm in a "area" populated by nodes that have not been clustered yet. This gives a way to adjust the awakening rate according to the number of clustered nodes independently from the number of nodes existing in the system. Such trend is emphasized in the "Cooperative variant" where the convergence time is less than 2000 seconds for both data sets. Next we discuss the cost of the self-clustering algorithm.

4.3 Overhead

To observe the control messages and measurement overhead, we differentiate between the two states of the system: the initial phase (when clusters are built) and the steady state. We observe the induced overhead as the number of measurements performed, but also as the number of exchanged messages during the clustering process. Figure 3(a) depicts the cumulative number of exchanged messages versus time (up to one hour). We do not consider the $JOIN$ messages sent by clustered nodes to their cluster heads, but focus on the difference that may exist between the two variants of the algorithm.



(a) Cumulative number of exchanged messages



(b) Cumulative number of performed measurements

Fig. 3. Induced Overhead: exchanged messages and performed measurements

The sharp rise of the curves in the initial phase is due to the fact that, at the beginning of the algorithm, nodes have the same probability to wake-up, since all nearby neighbors are not clustered yet. We manage to resolve such potential conflictual situation using the Freeze messages, sent to the long-sight neighbors, when a node identifies a cluster head, and waits for its confirmation. We are more likely to encounter such situations at the beginning of the algorithm. Moreover as very few nodes are clustered in the initial phase, more clusters are created, leading to more exchanged messages and measurements.

We can observe from figure 3(a) that, as expected, the cumulative number of messages by the "Selfish variant" is less important than the "Cooperative variant". It is however important to observe that the number of exchanged messages induced by newly created clusters is very low after the initial phase. Once the system reaches the steady state, no more messages, are exchanged. The low message exchange rate observed during the initial phase is confirmed by our results in table 1 with very low message rates of the scale of 10^{-4} per node per second as average values, and a maximum of 0.635 message/node$\times$second.

In figure 3(b) we observe the cumulative number of measurements performed towards the identified cluster heads (typically ping messages). We see again that the major overhead is induced during the initial phase. We observe more pings initially, when nodes initiate their clustering process, with a maximum ping rate of 0.034 ping/node$\times$ second. Such very low overall ping rate per node is promising for large scale deployment of our clustering scheme.

5 Conclusion

We have presented a distributed self-organized clustering process suitable for both ICS accuracy enhancement, and scalable network-aware overlay deployment. We have shown that, although ICS systems suffer from inaccuracies resulting from triangle inequality violations in the Internet, they can be exploited to construct efficient clustering schemes in a distributed way. Indeed, the proposed clustering process is based on a first approximation of node positioning using coordinates, along with an adaptive back-off strategy that allows load-balanced construction of clusters. We also present two variants of such clustering process that trade off the convergence time against the induced overhead. However, the two variants have been shown to be effective, enjoying good clustering performance, while achieving a very good trade-off between scalability and convergence time. Indeed, nodes are able to identify and join their clusters in a reasonable amount of time, while rarely violating the diameter constraints, and they still do not trigger too frequent measurements.

Although this paper focused on Vivaldi for distance estimates and experimentations, the algorithm proposed for clustering is independent of the coordinate system used. Our proposed scheme would then be general enough to be applied in the context of coordinates computed by any Internet coordinate systems.

The reader should note though, that we do not yet consider churn situations, when nodes join and leave the system running the Internet Coordinate System in an asynchronous way. If we consider highly-dynamic networks, the differentiation between the initial phase and the steady state may fade away. However, our clustering process would adapt to situations when only few nodes are existing in the system, by simply not creating clusters if they are "useless". Basically, by setting a minimum number of nodes per cluster at the level of the QT_clustering, low populated clusters could be avoided. It is also important to note that in churn situations, ping and message rates would be moderate, and that considering a non-dynamic network in our simulations gives us a worst-case of the cost of the system, at least in its initial phase. Finally, even though this paper does not address the problem of clusters maintenance, we note that different solutions to such issues have been proposed elsewhere (e.g. [13]). Our future work would then consist in integrating such maintenance techniques in our distributed clustering scheme.

References

1. Ng, T.S.E., Zhang, H.: Predicting Internet network distance with coordinates-based approaches. In: Proc. IEEE INFOCOM, New York, NY, USA (June 2002)
2. Dabek, F., Cox, R., Kaashoek, F., Morris, R.: Vivaldi: A decentralized network coordinate system. In: Proc. ACM SIGCOMM, Portland, OR, USA (August 2004)
3. Zheng, H., Lua, E.K., Pias, M., Griffin, T.: Internet Routing Policies and Round-Trip-Times. In: Dovrolis, C. (ed.) PAM 2005. LNCS, vol. 3431. Springer, Heidelberg (2005)
4. Lua, E.K., Griffin, T.: Embeddable overlay networks. In: IEEE Symposium on Computers and Communications, Aveiro, Portugal (2007)
5. Kaafar, M.A., Gueye, B., Cantin, F., Leduc, G., Mathy, L.: Towards a two-tier internet coordinate system to mitigate the impact of triangle inequality violations. In: Das, A., Pung, H.K., Lee, F.B.S., Wong, L.W.C. (eds.) NETWORKING 2008. LNCS, vol. 4982, pp. 397–408. Springer, Heidelberg (2008)
6. Lua, E.K., Zhou, X., Crowcroft, J., Mieghem, P.V.: Scalable multicasting with network-aware geometric overlay. Comput. Commun. 31(3), 464–488 (2008)
7. He, Y., Zhao, Q., Zhang, J., Wu, G.: Topology-aware multi-cluster architecture based on efficient index techniques. In: Jin, H., Reed, D., Jiang, W. (eds.) NPC 2005. LNCS, vol. 3779, pp. 163–171. Springer, Heidelberg (2005)
8. Xue, G., Jiang, Y., You, Y., Li, M.: A topology-aware hierarchical structured overlay network based on locality sensitive hashing scheme. In: UPGRADE, New York, NY, USA, pp. 3–8. ACM, New York (2007)
9. Heyer, L.J., Kruglyak, S., Yooseph, S.: Exploring expression data: Identification and analysis of coexpressed genes. Genome Research 9(11), 1106–1115 (1999)
10. A simulator for peer-to-peer protocols, `http://www.pdos.lcs.mit.edu/p2psim/index.html`
11. Wong, B., Slivkins, A., Sirer, E.: Meridian: A lightweight network location service without virtual coordinates. In: Proc. the ACM SIGCOMM (August 2005)
12. R Development Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria (2008) ISBN 3-900051-07-0
13. Liu, X., Lan, J., Shenoy, P., Ramaritham, K.: Consistency maintenance in dynamic peer-to-peer overlay networks. Comput. Netw. 50(6), 859–876 (2006)

A Practical Approach to Network Size Estimation for Structured Overlays

Tallat M. Shafaat[1], Ali Ghodsi[2], and Seif Haridi[1]

[1] Royal Institute of Technology (KTH),
School of Information and Communication, Stockholm, Sweden
{tallat,haridi}@kth.se
[2] Computer Systems Laboratory,
Swedish Institute of Computer Science, Stockholm, Sweden
ali@sics.se

Abstract. Structured overlay networks have recently received much attention due to their self-* properties under dynamic and decentralized settings. The number of nodes in an overlay fluctuates all the time due to churn. Since knowledge of the size of the overlay is a core requirement for many systems, estimating the size in a decentralized manner is a challenge taken up by recent research activities. Gossip-based Aggregation has been shown to give accurate estimates for the network size, but previous work done is highly sensitive to node failures. In this paper, we present a gossip-based aggregation-style network size estimation algorithm. We discuss shortcomings of existing aggregation-based size estimation algorithms, and give a solution that is highly robust to node failures and is adaptive to network delays. We examine our solution in various scenarios to demonstrate its effectiveness.

1 Introduction

Structured peer-to-peer systems such as Chord [20] and Pastry [18] have received much attention by the research community recently. These systems are typically very scalable and the number of nodes in the system immensely varies. The network size is, however, a global variable which is not accessible to individual nodes in the system as they only know a subset of the other nodes. This information is, nevertheless, of great importance to many structured p2p systems, as it can be used to tune the rates at which the topology is maintained. Moreover, it can be used in structured overlays for load-balancing purposes [4], deciding successor-lists size for resilience to churn [12], choosing a level to determine outgoing links [14], and for designing algorithms that adapt their actions depending on the system size [1].

Due to the importance of knowing the network size, several algorithms have been proposed for this purpose. Out of these, gossip-based aggregation algorithms [8], though having higher overhead, provide the best accuracy [17]. Consequently, we focus on gossip-based aggregation algorithms in this paper. While aggregation algorithms can be used to calculate different aggregates, e.g. average, maximum, minimum, variance etc., our focus is on counting the number of nodes in the system.

Although Aggregation [8] provides accurate estimates, it suffers from a few problems. First, Aggregation is highly sensitive to the *overlay topology* that it is used with. Convergence of the estimate to the real network size is slow for non-random topologies. On the contrary, the majority of structured p2p overlays have non-random topologies. Thus, it is not viable to directly use Aggregation in these systems. Second, Aggregation works in rounds, and the estimate is considered converged after a predefined number of rounds. As we discuss in section 4.1, this can be problematic. Finally, Aggregation is highly sensitive to node failures.

In this paper, we suggest a gossip algorithm based on Aggregation to be executed continously on every node to estimate the total number of nodes in the system. The algorithm is aimed to work on structured overlay networks. Furthermore, the algorithm is robust to failures and adaptive to the network delays in the system.

Outline. Section 2 serves as a background for our work. Section 3 describes our solution and discusses how the proposed solution handles the dynamism in the network. Thereafter, section 4 gives a detailed evaluation of our work. Section 5 discusses the related work, and finally, section 6 concludes.

2 Background

In this section, we briefly define a model of a ring-based structured overlay underlying our algorithm. We also describe the original Aggregation algorithm suggested by Jelasity et. al. [8].

2.1 A Model of a Ring-Based Structured Overlay Network

A ring-based structured overlay network consists of nodes which are assigned unique identifiers belonging to a ring of identifiers $\mathcal{I} = \{0, 1, \cdots, N-1\}$ for some large constant N. This is general enough to encompass many existing structured peer-to-peer systems such as Chord[20], Pastry[18] and many others.

Every node has a pointer to its successor, which is the first node met going clockwise on the ring. Every node also has a pointer to its predecessor, which is first node met going anti-clockwise on the ring. For instance, in a ring of size $N = 1024$ containing the nodes $\mathcal{P} = \{10, 235, 903\}$, we have that $succ_{\mathcal{P}}(10) = 235$, $succ_{\mathcal{P}}(903) = 10$, $pred_{\mathcal{P}}(235) = 10$, and $pred_{\mathcal{P}}(10) = 903$.

In this paper, we assume that there exists an out-of-bound mechanism to make all of the predecessor and successor pointers correct. This can, for example, be achieved by using periodic stabilization[20].

Apart from successor and predecessor pointers, each node has additional long pointers in the ring for efficient routing. Different structured overlays use different schemes to place these extra pointers. Our work is independent of how the extra pointers are placed.

While this model looks specific to ring topologies, other structured topologies use similar metrics, for instance, the XOR-metric [16] or butterfly networks [14] Our work can be extended to incorporate metrics other then that employed in ring-based overlays.

2.2 Gossip-Based Aggregation

The Aggregation algorithm suggested by Jelasity et. al. [8] is based on push-pull gossiping, shown in Algorithm 1.

Algorithm 1. Push-pull gossip executed by node p in Aggregation [8]

```
1: do periodically every $\delta$ time units
2:     q ← getneighbour()
3:     send $s_p$ to q
4:     $s_q$ ← receive(q)
5:     $s_q$ ← update($s_p$, $s_q$)
```

(a) Active thread

```
do forever
    $s_q$ ← receive(*)
    send $s_p$ to sender($s_q$)
    $s_q$ ← update($s_p$, $s_q$)
```

(b) Passive thread

The method GetNeighbour returns a uniform random sampled node over the entire set of nodes provided by an underlying sampling service like Newscast [7]. The method Update computes a new local state based on the node p's current local state s_p and the remote node's state s_q.

The time interval δ after which the active thread initiates an exchange is called a *cycle*. Given that all nodes use the same value of δ, each node roughly participates in two exchanges in each cycle, one as an initiator and the other as a receipient of an exchange request. Thus, the total number of exchanges in a cycle are roughly equal to $2 \cdot n$, where n is the network size.

For network size estimation, one random node sets its local state to 1 while all other nodes set their local states to 0. The global average is thus $\frac{1}{n}$, where n is the number of nodes. Executing the aggregation algorithm for a number of cycles decreases the variance of local states of nodes but keeps the global average the same. Thus, after convergence, a node p can estimate the network size as $\frac{1}{s_p}$. For network size estimation, Update(s_p, s_q) returns $\frac{s_p+s_q}{2}$.

Aggregation [8] achieves up-to-date estimates by periodically restarting the protocol, i.e. local values are re-initialized and aggregation starts again. This is done after a predefined number of cycles γ, called an epoch.

The main disadvantage of Aggregation is that a failure of a single node early in an epoch can severely effect the estimate. For example, if the node with local state 1 crashes after executing a single exchange, 50% of the value will disappear, giving $2 \cdot n$ as the final size estimate. This issue is further elaborated in section 4.3. Another disadvantage, as we discuss in section 4.1, is predefining the epoch length γ.

3 The Network Size Estimation Algorithm

A naive approach to estimate the network size in a ring-based overlay would be pass a token around the ring, starting from, say node i and containing a variable v initialized to 1. Each node increments v and forwards the token to its successor i.e. the next node on the ring. When the token reaches back at i, v will contain the network size. While this solution seems simple and efficient, it suffers from multiple problems. First, it is not fault-tolerant as the node with the token may fail. This will require complicated modifications for regenerating the token with the current value of v. Second, the naive approach will be quite slow, as it will take $O(n)$ time to complete. Since peer-to-peer systems are highly dynamic, the actual size may have changed completed by the time the algorithm finishes. Lastly, at the end of the naive approach, the estimate will be known only to node i which will have to broadcast it to all nodes in the system. Our solution aims at solving all these problems at the expense of a higher message complexity than the naive approach.

Our goal is to make an algorithm where each node tries to estimate the average inter-node distance, $\varDelta$, on the identifier space, i.e. the average distance between two consecutive nodes on the ring. Given a correct value of $\varDelta$, the number of nodes in the system can be estimated as $\frac{N}{\varDelta}$, N being the size of the identifier space.

Every node p in the system keeps a local estimate of the average inter-node distance in a local variable d_p. Our goal is to compute $\frac{\sum_{i \in \mathcal{P}} d_i}{|\mathcal{P}|}$. The philosophy underlying our algorithm is the observation that at any time the following invariant should always be satisfied: $N = \sum_{i \in \mathcal{P}} d_i$.

We achieve this by letting each node p initialize its estimate d_p to the distance to its successor on the identifier space. In other words, $d_p = succ(p) \ominus p$, where $\ominus$ represents subtraction modulo N. Note that if the system only contains one node, then $d_p = N$. Clearly, a correctly initialized network satisfies the mentioned invariant as the sum of the estimates is equal to N.

To estimate $\varDelta$, we employ a modified aggregation algorithm. Since we do not have access to random nodes, we implement the GetNeighbour method in Alg. 1 by returning a node reached by making a random walk of length h. For instance, to perform an exchange, p sends an exchange request to one of its neighbours, selected randomly, with a hop value h. Upon receiving such a request, a node r decrements h and forwards the request to one of its own neighbours, again selected randomly. This is repeated until h reaches 0, after which the exchange takes place between p and the last node getting the request.

Given that GetNeighbour returns random nodes, after a number of exchanges (logarithmic number of steps, to the network size, as show in [8]), every node will have $d_p = \frac{\sum_{i \in \mathcal{P}} d_i}{|\mathcal{P}|}$. On average in each cycle, each node initiates an exchange once, which takes h hops, and replies to one exchange. Consequently, the number of messages for the aggregation algorithm are roughly $hops \times n + n$ per cycle.

3.1 Handling Churn

The protocol described so far does not take into account the dynamicity of large scale peer-to-peer systems. In this section, we present our solution as an extension of the basic algorithm described in section 3 to handle dynamism in the network.

The basic idea of our solution is that each node keeps different levels of estimates, each with a different accuracy. The lowest level estimate is the same as d_n in the basic algorithm. As the value in the lowest level converges, it is moved to the next level. While this helps by having high accuracy in upper levels, it also gives a continuous access to a correct estimated value while the lowest level is re-initialized. Furthermore, we restart the protocol adaptively, instead at a predefined interval.

Our solution is shown in Algorithm 2. Each node n keeps track of the current epoch in *nEpoch* and stores the estimate in a local variable *ndvalue* instead of d_n in the basic algorithm. *ndvalue* is a tuple of size l, i.e.

$$ndvalue = (ndvalue_{l-1}, ndvalue_{l-2}, \cdots, ndvalue_0)$$

The tuple values are called levels. The value at level 0 is the same as d_n in the basic algorithm and has the most recent updated estimate but with high error, while level $l-1$ has the most accurate estimate but incorporates updates slowly.

A node n initializes its estimate, method InitializeEstimate in Alg. 2, by setting level 0 to $succ_{\mathcal{P}}(n) \ominus n$. The method LeftShiftLevels moves the estimate of each level one level up, e.g. left shifting a tuple $e = (e_{l-1}, e_{l-2}, \cdots, e_0)$ gives $(e_{l-2}, e_{l-3}, \cdots, e_0, nil)$. The method Update(a, b) returns an average of each level, i.e. $(\frac{a_{l-1}+b_{l-1}}{2}, \frac{a_{l-2}+b_{l-2}}{2}, \cdots, \frac{a_0+b_0}{2})$.

To incorporate changes in the network size due to churn, we also restart the algorithm, though not after a predefined number of cycles, but adaptively by analyzing the variance. We let the lowest level converge and then restart. This duration may be larger than a predefined γ or less, depending on the system-wide variance in the system of the value being estimated. We achieve adaptivity by using a sliding window at each node. Each node stores values of the lowest level estimate for each cycle in a sliding window W of length w. If the coefficient of variance of the sliding window is less than a desired accuracy e.g. 10^{-2}, the value is considered converged, denoted by the method Converged in Alg. 2.

Once the value is considered to have converged based on the sliding window, there are different methods of deciding which node will restart the protocol, denoted by the method IAmStarter in Alg. 2. One way is as used in [8], each node restarts the protocol with probability $1/\hat{n}$, where $\hat{n}$ is the estimated network size. Given a reasonable estimate in the previous epoch, this will lead to one node restarting the protocol with high probability. It does not matter if more than one node restarts the protocol in our solution. On the contrary, multiple nodes restarting an epoch in [8] is problematic since only one node should set its local estimate to 1 in an epoch. Consequently, an epoch has to be marked with a unique identifier in [8]. Another method is that a node n restarts the protocol which has $0 \in [n, n.succ)$. For our simulations, we use the first method.

Algorithm 2. Network size estimation

```
1: every $\delta$ time units at $n$
2:     if $converged()$ and $iamstarter()$ then
3:         simpleBroadcast($nEpoch$)
4:     end if
5:     sendto $randomNeighbour()$ : ReqExchange($hops, nEpoch, ndvalue$)
6: end event

7: receipt of ReqExchange($hops, mEpoch, mdvalue$) from $m$ at $n$
8:     if $hops > 1$ then
9:         $hops := hops - 1$
10:        sendto $randomNeighbour()$ : ReqExchange($hops, mEpoch, mdvalue$)
11:    else
12:        if $nEpoch > mEpoch$ then
13:            sendto m : ResExchange($false, nEpoch, ndvalue$)
14:        else
15:            trigger ⟨ MoveToNewEpoch | $mEpoch$ ⟩
16:            $ndvalue$ := update($ndvalue, mdvalue$)
17:            updateSlidingWindow($ndvalue$)
18:            sendto m : ResExchange($true, nEpoch, ndvalue$)
19:        end if
20:    end if
21: end event

22: receipt of ResExchange($updated, mEpoch, mdvalues$) from $m$ at $n$
23:    if $updated = false$ then
24:        trigger ⟨ MoveToNewEpoch | $mEpoch$ ⟩
25:    else
26:        if $nEpoch = mEpoch$ then
27:            $dvalue := mdvalues$
28:        end if
29:    end if
30: end event

31: receipt of DeliverSimpleBroadcast($mEpoch$) from $m$ at $n$
32:    trigger ⟨ MoveToNewEpoch | $mEpoch$ ⟩
33: end event

34: upon event ⟨ MoveToNewEpoch | $epoch$ ⟩ at n
35:    if $nEpoch < mEpoch$ then
36:        leftShiftLevels()
37:        initializeEstimate()
38:        $nEpoch := epoch$
39:    end if
40: end event
```

Once a new epoch starts, all nodes should join it quickly. Aggregation [8] achieves this by the logarithmic epidemic spreading property of random networks. Since we do not have access to random nodes, we use a simple broadcast scheme [3] for this purpose, which is both inexpensive ($O(n)$ messages) and fast ($O(\log n)$ steps). The broadcast is best-effort, as even if it fails, the new epoch number is spread through exchanges.

When a new node joins the system, it starts participating in the size estimation protocol when the next epoch starts. This happens either when it receives the broadcast, or its predecessor initializes its estimate. Until then, it keeps forwarding any requests for exchange to a randomly selected neighbour.

Handling churn in our protocol is much simpler and less expensive on bandwidth than other aggregation algorithms. Instead of running multiple epochs as in [8], we rely on the fact that a crash in our system does not effect the end estimate as much as in [8]. This is explored in detail in section 4.3.

4 Evaluation

To evaluate our solution, we implemented the Chord [20] overlay in an event-based simulator [19]. For the first set of experiments, the results are for a network size of 5000 nodes using the King dataset [5] for message latencies. Since we do not have the King dataset for a 5000 node topology, we derive the 5000 node pair-wise latencies from the distance between two random points in a Euclidean space. The mean RTT remains the same as in the King data. This technique is the same as used in [11]. For larger network sizes, the results are for 10^5 nodes using exponentially distributed message latencies with mean 5 simulation time units. For the figures, $\delta = 8 * \textit{mean-com}$ means the cycle length is 8×5.

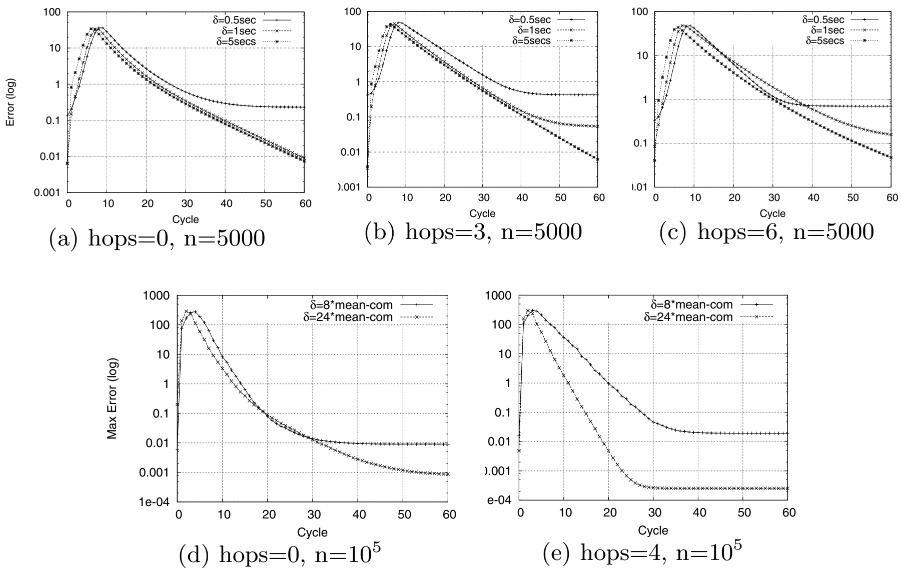
4.1 Epoch Length γ

We investigated the effect of δ on convergence of the algorithm. The results are shown in Figure 1, where error$= \frac{1}{n}\sum_{i=1}^{n} |d_i - \frac{N}{n}|$. It shows that when the ratio between communication delay and δ is significant, e.g. $\delta = 0.5$ *secs* or $8 * \textit{mean-com}$, the aggregate converges slowly and to a value with higher error. For cases where the ratio is insignificant, e.g. $\delta = 5$ *secs* or $24 * \textit{mean-com}$, the convergence is faster and the converged value has lower error. The reason for this behaviour is that when δ is small, the expected number of exchanges per cycle do not occur.

Since δ and γ effect convergence rate and accuracy, our solution aims at having adaptive epoch lengths. Another benefit of using an adaptive approach as ours is that the protocol may converge much before a predefined γ, thus sending messages in vain. If the protocol was restarted, these extra cycles could have been used to get updated aggregate value or include churn effects faster.

4.2 Effect of the Number of Hops

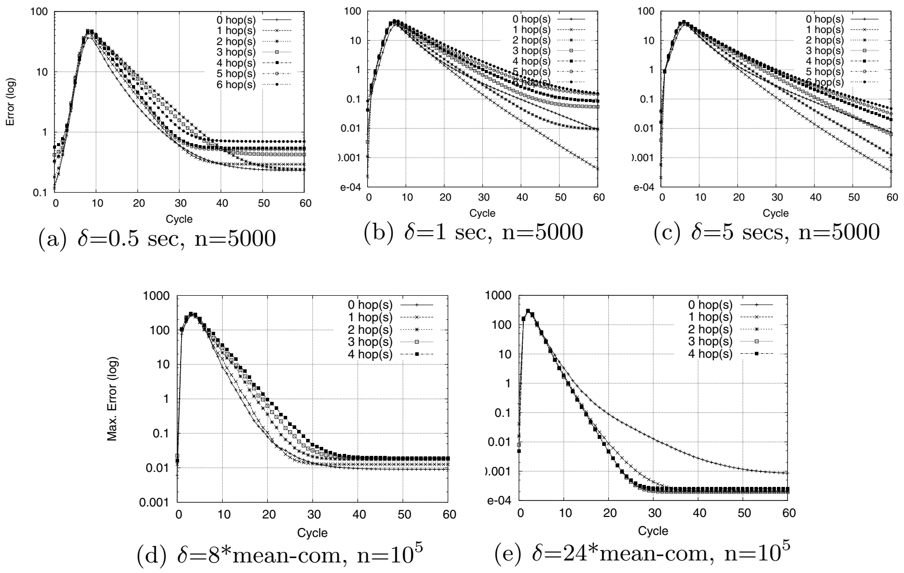
Figure 2 shows convergence of the algorithm for different values of δ and number of hops h to get a random node. For small values of δ, e.g. 0.5 secs and $8 *$



(a) hops=0, n=5000 (b) hops=3, n=5000 (c) hops=6, n=5000

(d) hops=0, n=10^5 (e) hops=4, n=10^5

Fig. 1. Error for the estimate of inter-node distance d in the system

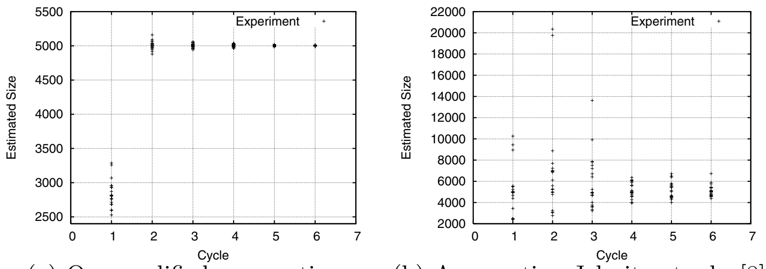


(a) δ=0.5 sec, n=5000 (b) δ=1 sec, n=5000 (c) δ=5 secs, n=5000

(d) δ=8*mean-com, n=10^5 (e) δ=24*mean-com, n=10^5

Fig. 2. Error for the estimate of inter-node distance d in the system

mean-com, $h = 0$ gives best convergence. The reason for this behaviour is that since δ is very small (thus, is comparative to communication delays), having multiple hops will not have enough exchanges in a cycle. Thus, convergence



(a) Our modified aggregation (b) Aggregation Jelasity et. al. [8]

Fig. 3. Estimated size when 50% of the nodes out of 5000 fail suddenly. X-axis gives the cycle at which the sudden death occured in the epoch.

takes longer time and the error is larger for larger values of h. On the contrary, as we increase δ, higher values of h give convergence in lesser time and lower error. These results also advocate to have an adaptive epoch length.

4.3 Churn

Flash Crowds. Next, we evaluated a scenario where a massive node failure occurs. Contrary to [8] where failure of nodes with higher local estimate effects the end estimate more than with lower local estimate, failure of any node is equal in our protocol. The results for a scenario where 50% of the nodes fail at different cycles of an epoch is shown in Figure 3. Our modified aggregation solution, Fig. 3(a), is not as severly affected by the sudden death as the original Aggregation algorithm, fig. 3(b). Infact, in some experiments with Aggregation, the estimate became infinite (not shown in the figure). This happens when all the nodes with non-zero local estimates fail. For our solution, the effect of a sudden death is already negligible if the nodes crash after the third cycle.

Continuous Churn. We ran simulations for a scenario with churn, where nodes join and fail. The results are shown in figure 4, 5 and 7. The results are for extreme churn cases, i.e. 50% nodes fail within a few cycles and afterwards, 50% nodes join within a few cycles. The graphs show how the estimation follows the actual network size. The standard deviation of level 2 is shown as vertical bars, which depicts that all nodes estimate the same size. The standard deviation is high only when a new epoch starts, because while evaluating the mean and standard deviation, some nodes have moved to the new epoch, while others are still in the older epoch. The estimate at level 1 converges to the actual size faster than level 2, but the estimates has higher variance as the standard deviation for level 1 (not shown) is higher than for level 2. Figure 5 also shows that compared to $h = 0$, higher values of h follow the trend of the actual size faster.

Next, we simulated a network of size 4500 and evaluated our algorithm under continous churn. In each cycle, we failed some random nodes and joined new nodes. As explained in section 3, new nodes do not participate in the algorithm

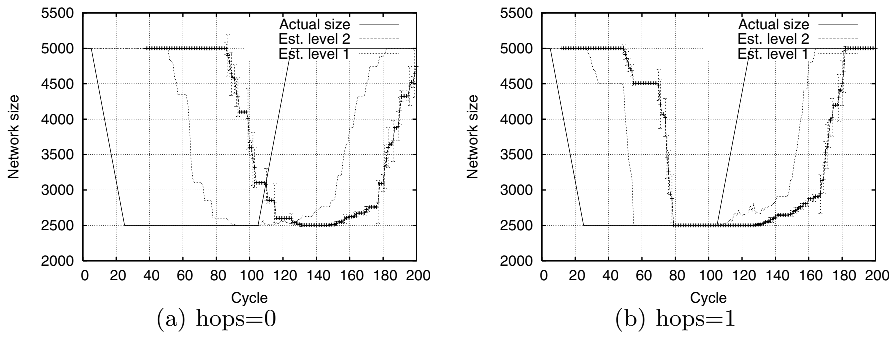


Fig. 4. Mean estimated size by each node with standard deviation

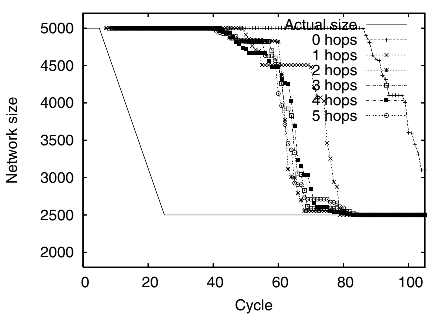


Fig. 5. Mean estimated size for different values of h for level 2. Gives a comparison of how fast the nodes estimation follows the real network size.

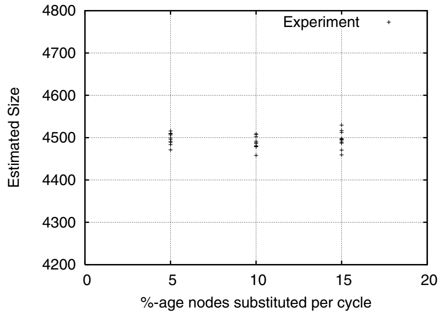


Fig. 6. Estimated network size for 4500 nodes under continuous churn. X-axis gives the percentage of churn events (joins+failures) that occur in each cycle.

till the next epoch starts, yet they can forward requests. Figure 6 shows the results. The plotted dots correspond to the converged mean estimate after 15 cycles for each experiment. The x-axis gives the percentage of churn events, including both failures and joins, that occur in each cycle. Thus, 10% means that $4500 \times \frac{10}{100} \times 15$ churn events occured before the plotted converged value. Figure 6 shows that the algorithm handles continous churn reasonably well.

5 Related Work

Network size estimation in the context of peer-to-peer systems is challenging as these systems are completely decentralized, nodes may fail anytime, the network size varies dynamically over time, and the estimation algorithm needs to continuously update its estimation to reflect the current number of nodes.

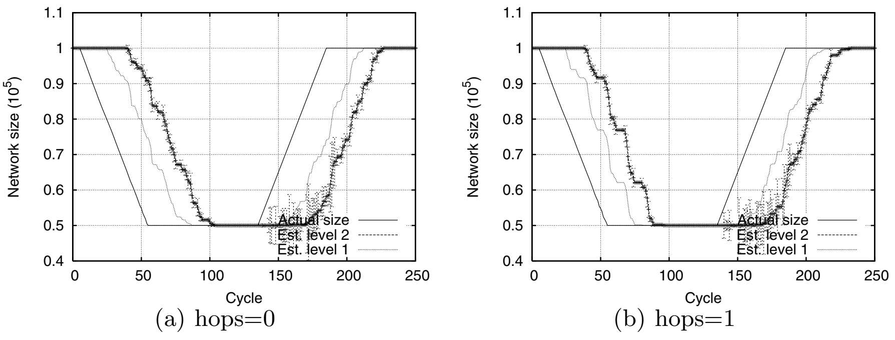


Fig. 7. Mean estimated size by each node with standard deviation

Merrer et. al. [17] compare three existing size estimation algorithms, Sample & Collide [15], Hops Sampling [10] and Aggregation [8], which are representative of three main classes of network size estimation algorithms. Their study yields that although Aggregation is expensive, it produces the most accurate results. Aggregation also has the additional benefit that the estimamte is available on all nodes compared to only at the initiator in the case of Sample & Collide and Hops Sampling. Our work can be seen as an extension of Aggregation, to handle its shortcomings and extend it to non-random topologies, such as structured overlay networks.

The work by Horowitz et. al. [6] is similar to ours in the sense that they also utilize the structure of the system. They use a localized probabilistic technique to estimate the network size by maintaining a structure: a logical ring. Each node estimates the network size locally based on the estimates of its neighbours on the ring. While their technique has less overhead, the estimates are not accurate, the expected accuracy being in the range $n/2 \cdots n$. Their work has been extended by Andreas et. al. [2] specifically for Chord, yet the extended work also suffers similar inaccuracy range for the estimated size. Mahajan et. al. [13] also estimate the network size through the density of node identifiers in Pastry's leafset, yet they neither prove any accuracy range, nor provide any simulation results to show the effectiveness of their technique.

Kempe et. al. [9] have also suggested a gossip-based aggregation scheme, yet their solution focuses only on push-based gossiping. Using push-based gossiping complicates the update and exchange process as a normalization factor needs to be kept track of. On the same, as noted by Jelasity et. al. [8], push-based gossiping suffers from problems when the underlying directed graph used is not strongly connected. Thus, we build our work on push-pull gossip-based aggregation [8]. Similarly, to estimate the network size, Kempe et. al. also propose that one node should initialize its weight to 1, while the other nodes initialize to weight 0, making it highly sensitive to failures early in the algorithm.

The authors of Viceroy [14] and Mercury [1] mention that a nodes distance to its successor can be used to calculate the number of nodes in the system, but provide no reasoning that the value always converges exactly to the correct value, and thus that their estimate is unbiased.

6 Conclusion

Knowledge of the current network size of a structured p2p system is a prime requirement for many systems, which prompted to finding solutions for size estimation. Previous studies have shown that gossip-based aggregation algorithms, though being expensive, produce accurate estimates of the network size. We have demonstrated the shortcomings in existing aggregation approaches to network size estimation and have presented a solution that overcomes the deficiencies. In this paper, we have argued for an adaptive approach to convergence in gossip-based aggregation algorithms. Our solution is resilient to massive node failures and is aimed to work on non-random topologies such as structured overlay networks.

References

1. Bharambe, A.R., Agrawal, M., Seshan, S.: Mercury: Supporting Scalable Multi-Attribute Range Queries. In: Proceedings of the ACM SIGCOMM 2004 Symposium on Communication, Architecture, and Protocols, OR, USA. ACM Press, New York (2004)
2. Binzenhöfer, A., Staehle, D., Henjes, R.: On the fly estimation of the peer population in a chord-based p2p system. In: 19th International Teletraffic Congress (ITC19), Beijing, China (September 2005)
3. Ghodsi, A.: Distributed k-ary System: Algorithms for Distributed Hash Tables. PhD dissertation, KTH—Royal Institute of Technology, Stockholm, Sweden (December 2006)
4. Godfrey, P.B., Stoica, I.: Heterogeneity and Load Balance in Distributed Hash Tables. In: Proc. of the 24th Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM 2005), FL, USA. IEEE Comp. Society, Los Alamitos (2005)
5. Gummadi, K.P., Saroiu, S., Gribble, S.D.: King: estimating latency between arbitrary internet end hosts. In: IMW 2002: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment, pp. 5–18. ACM, New York (2002)
6. Horowitz, K., Malkhi, D.: Estimating network size from local information. Information Processing Letters 88(5), 237–243 (2003)
7. Jelasity, M., Kowalczyk, W., van Steen, M.: Newscast Computing. Technical Report IR–CS–006, Vrije Universiteit (November 2003)
8. Jelasity, M., Montresor, A., Babaoglu, Ö.: Gossip-based Aggregation in Large Dynamic Networks. ACM Trans. on Computer Systems (TOCS) 23(3) (August 2005)
9. Kempe, D., Dobra, A., Gehrke, J.: Gossip-based computation of aggregate information. In: 44th Symp. on Foundations of Computer Science, FOCS (2003)
10. Kostoulas, D., Psaltoulis, D., Gupta, I., Birman, K., Demers, A.J.: Decentralized schemes for size estimation in large and dynamic groups. In: 4th IEEE International Symp. on Network Computing and Applications (NCA 2005), pp. 41–48 (2005)

11. Li, J., Stribling, J., Morris, R., Kaashoek, M.F.: Bandwidth-efficient management of DHT routing tables. In: Proc. of the 2nd USENIX Symp. on Networked Systems Design and Implementation (NSDI 2005), MA, USA, May 2005, USENIX (2005)
12. Liben-Nowell, D., Balakrishnan, H., Karger, D.R.: Analysis of the Evolution of Peer-to-Peer Systems. In: Proceedings of the 21st Annual ACM Symposium on Principles of Distributed Computing (PODC 2002), pp. 233–242. ACM Press, New York (2002)
13. Mahajan, R., Castro, M., Rowstron, A.: Controlling the Cost of Reliability in Peer-to-Peer Overlays. In: Kaashoek, M.F., Stoica, I. (eds.) IPTPS 2003. LNCS, vol. 2735, pp. 21–32. Springer, Heidelberg (2003)
14. Malkhi, D., Naor, M., Ratajczak, D.: Viceroy: A scalable and dynamic emulation of the butterfly. In: Proceedings of the 21st Annual ACM Symposium on Principles of Distributed Computing (PODC 2002). ACM Press, New York (2002)
15. Massoulié, L., Merrer, E.L., Kermarrec, A., Ganesh, A.J.: Peer counting and sampling in overlay networks: random walk methods. In: Proc. of the 25th Annual ACM Symp. on Principles of Distributed Computing (PODC), pp. 123–132 (2006)
16. Maymounkov, P., Mazieres, D.: Kademlia: A Peer-to-Peer Information System Based on the XOR metric. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 53–65. Springer, Heidelberg (2002)
17. Merrer, E.L., Kermarrec, A.-M., Massoulie, L.: Peer to peer size estimation in large and dynamic networks: A comparative study. In: Proc. of the 15th IEEE Symposium on High Performance Distributed Computing, pp. 7–17. IEEE, Los Alamitos (2006)
18. Rowstron, A., Druschel, P.: Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In: Guerraoui, R. (ed.) Middleware 2001. LNCS, vol. 2218, pp. 329–350. Springer, Heidelberg (2001)
19. SicsSim (2008), `http://dks.sics.se/iwsos08sizeest/`
20. Stoica, I., Morris, R., Karger, D.R., Kaashoek, M.F., Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In: Proceedings of the ACM SIGCOMM 2001 Symposium on Communication, Architecture, and Protocols, San Deigo, CA, August 2001, pp. 149–160. ACM Press, New York (2001)

A Framework of Economic Traffic Management Employing Self-Organization Overlay Mechanisms

Simon Oechsner[1], Sergios Soursos[2], Ioanna Papafili[2], Tobias Hossfeld[1], George D. Stamoulis[2], Burkhard Stiller[3], Maria Angeles Callejo[4], and Dirk Staehle[1]

[1] University of Würzburg, Institute of Computer Science, Department of Distributed Systems, Am Hubland, 97074 Würzburg, Germany
{oechsner,hossfeld,dstaehle}@informatik.uni-wuerzburg.de
[2] Athens University of Economics and Business, Department of Informatics, 76 Patission Str., Athens 10434, Greece
{sns,papafili06,gstamoul}@aueb.gr
[3] University of Zürich, CSG@IFI, Binzmühlestrasse 14, CH - 8050 Zürich, Switzerland
stiller@ifi.uzh.ch
[4] Telefonica Investigacion y Desarrollo, Madrid, Spain
macr@tid.es

Abstract. Applications based on overlays have become very popular, due to the separation they provide and the improvement of perceived QoS by the end-user. Recent studies show that overlays have a significant impact on the traffic management and the expenditures of the underlying network operators. In this paper, we define a framework for Economic Traffic Management (ETM) mechanisms that optimize the traffic impact of overlay applications on ISP and telecommunication operator networks based on the interaction of network operators, overlay providers and users. We first provide a definition and an overview of Self-Organization Mechanisms (SOMs) and ETM for overlays. We then describe a basic framework for the interaction of components of SOMs and ETM, in terms of information and metrics provided, decisions made etc. Finally, we describe in detail how SOMs can be used to support ETM and we illustrate our approach and its implications by means of a specific example.

1 Introduction

Today's largest contributor to internet traffic is Peer-to-Peer (P2P) in its different forms. While up to now, P2P file-sharing caused the bulk of the traffic, recently Video-on-Demand applications that are also partly based on P2P are on the rise. What makes P2P traffic disadvantageous for Internet Service Providers (ISPs) is the fact that traffic is forwarded via logical overlay connections. These do not normally take into account the structure or load in the physical network, leading to the usage of many cross-ISP links, which incurs a high cost.

As a result, some providers have started to influence this traffic by throttling or even terminating P2P connections that leave their ISP network. As a short-term solution, it has the desired effect of lowering the provider's cost for forwarding traffic to a remote network. In the long run, however, the service quality for the ISP's customers is diminished, possibly leading to dissatisfaction and less income.

To overcome this dilemma, a solution needs to be found that does not negatively affect the service of the end users, but still lowers the cost for the providers. This is termed Economic Traffic Management (ETM). Currently, several research efforts are aiming at the shaping of P2P traffic according to Economic Traffic Management principles. One of these efforts is the ICT project SmoothIT[1], the main objective of which is to employ ETM to traffic generated by overlay applications in a way that is beneficial to all players involved.

In this work, we want to describe the general interaction possibilities between the physical network and ISP on one hand and the overlay on the other hand. To this end, we will first provide an overview of the related work and we continue by giving definitions and descriptions of the main two concepts used in this paper, Self-Organization (SO) and ETM. The notion of economic incentives in this context is also discussed. Then, we describe how overlays can be influenced and present an example for an architecture that includes these concepts. Finally, we provide some concluding remarks.

2 Background and Related Work

Since P2P systems organize nodes in an overlay network, a significant research effort is currently devoted to the design of scalable, fault-tolerant and communication-efficient overlay topologies. The third objective has become the most important since several studies, such as [2] and [3], show that especially peer-to-peer traffic has the largest share of the Internet traffic. Also, other ones indicate that emerging overlay services have rendered current traffic engineering techniques used by ISPs inadequate to keep the networks performance at a desired level [4].

In initial overlay construction algorithms, especially deterministic ones like CHORD [5], neighbors were chosen at random without any knowledge about the underlying physical topology [6]. This approach offered relatively simple overlay maintenance, but caused significant problems [7]; a packet along an overlay path may visit the same physical node several times [8], a low-bandwidth physical link may be shared by many overlays leading to high congestion and performance degradation, and traditional traffic management techniques may collide with overlay reorganization causing traffic oscillations [4].

In order to avoid such undesirable effects certain researchers proposed overlay construction and maintenance algorithms trying to bridge the gap between the overlay topology and the physical (underlay) network. Most approaches, like [9], [10], [11], probe candidate neighbors in order to select those that are relatively close. But having a large number of independent overlays performing network-layer measurements creates significant traffic. The authors of [12], trying to mitigate this problem, proposed a routing-underlay for overlays and an architecture for inter-overlay cooperation respectively, aiming to reuse underlay information by many different overlay applications. The performance of these approaches is evaluated to a very limited extent. Moreover, there is very little analysis of the issue of players incentives that would make such a collaboration and sharing of information achievable.

The well known capability of overlay networks to launch a new service globally on the Internet with a minimum of network resources being involved from the service provider's point of view has to be emphasized as a strong design approach for commercially successful applications. Therefore, the efficiency of support by caches in network nodes and in terminal equipment has to be considered a key factor with regard to all overlay structures, providing a benefit for ISPs and end-users. An economical evaluation of this practice can be found in [13].

3 Main Concepts

In this section, we will give definitions of the main concepts used in our framework, and we will also briefly discuss the importance of the correct incentives for the stakeholders.

3.1 Self-Organization

The term self-organization (SO) is, in the following, defined on the level of overlay networks, i.e. logical networks above the physical network. In the context of overlays, SO means that a overlay network structure evolves by local decisions made by the peers participating in the network, without any higher authority directly intervening.

A SOM is a concrete algorithm implemented at each peer forming the overlay. It makes the local decisions that, in interplay with the decisions made at other peers, achieve self-organization. In order to influence the overlay network, the SOMs presented here make some kind of choice, e.g. between peers used as overlay neighbors. This choice is based on locally available data that is the input to the algorithm.

Examples for SOMs are the peer and chunk selection processes in file-sharing networks, such as tit-for-tat or Least-Shared-First [14]. Proximity Neighbor Selection and Geographic Layout in DHTs [15] are examples where underlay information is taken into account to form a structured overlay.

The data used as input for a SOM may be provided by other peers or by the underlay. The choice is made by applying a metric to this input. This metric provides semantics to the choice process by defining what makes one alternative 'better' than the other. To give a short example, using RTT as a metric in a SOM would structure an overlay completely differently than using the similarity in shared content as a criterion for selecting an overlay neighbor.

If the input for the SOM can not be provided by the peers themselves, the results of the SO depend on the quality of information available to the mechanism. A RTT measurement done by a peer, for example, may not be as reliable as similar information provided by the underlay itself.

3.2 Incentives for Stakeholders

Stakeholders in the content of this paper are the end-users, the overlay providers and the network operators. As already mentioned, there are various implications of overlay traffic to the cost structure of an ISP that lead to a tussle between ISPs and overlay networks.

There are many types of incentives per stakeholder that favor (or not) the existence of overlay networks. The two most important ones, common for all the stakeholders, are the *monetary benefits* and the *performance improvements*. A common phenomenon is that incentives provided to one stakeholder may introduce negative effects to another one. For example, the performance improvements that an overlay provider may want to introduce may come in direct conflict with the economic incentives for the operator (ISP), since such improvements may change the traffic patterns, affecting the interconnection agreements and charges for the specific ISP. Furthermore, an action taken on the overlay may provide both monetary benefits and performance improvements for the same stakeholder, i.e., reduction of inter-domain traffic causes interconnection costs to decrease while the performance of the network might increase.

At this point, it is necessary to make an important observation: although the stakeholders in this environment are three, conflicts may appear only between the underlay (network operator) and the overlay (end-users and overlay provider) entities. Indeed, conflicts between the end-user and the overlay provider can only occur in the (improbable) case that the provider makes some drastic changes to the overlay application that alter the nature of the service provided, rendering it not beneficial for the end users.

Below, we provide a list of principals that may hold, so that we can reach a situation where all stakeholders are better off with the existence of an overlay network, i.e., such that a win-win-win situation occurs.

- Monetary benefits can offer the desired outcome, if there is a possibility of transferring/recovering costs through charging schemes.
- Monetary benefits can also apply when combined with performance improvements or with service differentiation in general.
- Performance improvements should not be considered as substitutes but as complementary to monetary benefits.
- Performance improvements for one stakeholder can provide monetary benefits for another one or vice versa. In other words, the type of incentives provided may not be the same for all stakeholders.

3.3 Economic Traffic Management

Economic Traffic Management (ETM) is one of the key concepts of this work and was already described in [16]. Its main objective is to achieve the co-operation between the overlay and the underlay, resulting in traffic patterns that optimize the use of network resources according to some given criteria. This is attained by means of ETM mechanisms that are beneficial for all players involved. That is, such mechanisms promote mutual compatibility of the incentives mentioned in the previous subsection.

In particular, ETM employs mechanisms that are related to economic incentives of the users in the overlay. That is, they affect (overlay and underlay) decisions, leading to a) a reduction of the economic cost incurred by the user and/or b) an improvement of the performance as perceived by the user.

At the same time, the way these incentive mechanisms operate and the state to which they lead the overlay is affected by information generated by the underlay and/or by policies employed therein. This way, the outcome of ETM is influenced by the underlay and its objectives. The objective of the ISP is to render this influence beneficial for himself as well.

The main toolkit of ETM is based on incentive mechanisms. The objective of such a mechanism is to shape the behavior of a participating agent by offering choices to him. The agent responds selfishly to the existence of the incentive mechanism and to choices he is offered. In particular, he performs such a selection so as to optimize his own objective function, in this case the performance of the service he is using. That is, he adopts among the valid alternatives the one that optimizes this index. Usually, each choice represents a trade-off between: a) the utility for the agent by the outcome given his choice and b) the relevant cost for him.

To illustrate this, we consider an ISP offering certain ADSL packages, namely different download rates at different charges. Some cases of the customers' objectives include the selection of: a) the highest rate that does not exceed a certain budget threshold, b) the lowest-cost package that exceeds a certain rate threshold, c) the package that represents the best value-for-money, i.e. the best trade-off between download rate and charge where both are considered as flexible.

A widely accepted objective function that quantifies such a trade-off is the net benefit, which equals the difference between the utility and the charge. In general, it is assumed that in order for a choice to be acceptable by an agent the resulting net benefit has to be non-negative.

To summarize what applies to our case: the provider (or in general the entity setting the mechanism) imposes a mechanism, to which the participating agents respond selfishly and there arises an overall outcome; the mechanism should be such that the objective function of the provider is optimized; in this case, his costs are minimized. Of course, the provider should also take into account the fact that certain agents may not participate due to the mechanism, which would, e.g., be the case for the bandwidth throttling mentioned before.

In the context of an overlay offering a specific service, e.g., file download or VoD, an increase in the service quality may always be considered as a benefit to the end user. Therefore, it should be the aim of ETM to encourage peer behaviour that incurs traffic where a provider prefers it, while simultaneously improving the quality of the service it is offering to the user. If it is not technically possible to do this, the user should be compensated in a different way. Still, a win-win situation is sought by ETM.

4 Interaction Possibilities between SOMs and ETM

What makes SOMs useful to ETM is the fact that they run in the overlay; in fact most of the SOMs are already in place with the existing protocols. Of course, not all SOMs are appropriate for ETM. To see why this applies, we first classify the SOMs according to the selections they provide to the users in the overlay. In particular, the following possibilities apply:

1. SOMs offering no selections; e.g. DHT-based content location in Chord.
2. SOMs offering to the user selections that are not based on immediate incentives: e.g. the list of Kademlia-based "neighboring" peers, which, apart from preferring peers with a longer uptime, does not relate directly to any actual distance or other performance-related improvement.

3. Selections based on immediate application-layer incentives: e.g. which chunk to download first in BitTorrent ("rarest first replication"); this does relate to a performance-related improvement , which however is not related to the underlay conditions.
4. Selections based on immediate incentives that are related to the underlay: e.g. bandwidth-based selection of peer to download from in BitTorrent.

It is mainly the last category of SOMs that can serve as enablers of ETM. In particular, the underlay provides agents (i.e. users) participating in such SOMs with the information employed in order to make the selections that the SOM actually prescribes; e.g. RTT or other physical proximity metrics, for SOMs that prescribe that the selection of preferred peers is based on such metrics. This might, for example, be done by a) including such information to the tracker in BitTorrent, which will in turn provide it to the peers b) introducing a separate tracker in a BitTorrent like system that provides additional information about the network location of peers in a swarm, or c) having the underlay provide an interface for more reliable RTT measurements.

Furthermore, the selections made in the context of SOMs influence the traffic actually arising in the underlay, since they influence both the demand for traffic as well as the way it is routed. These interactions are depicted in Figure 1. Of course, the cases and the methods how the underlay can influence such SOMs for the benefit of all players involved are matters for further study, falling out of the scope of the present paper.

Since most of the applications considered are already implemented and working without the application of ETM, the latter should indeed provide extra benefits to the overlay layer, in order for these applications to adopt changes. This is to be achieved by offering incentives to all parties involved, e.g. better QoS/QoE for end users and less

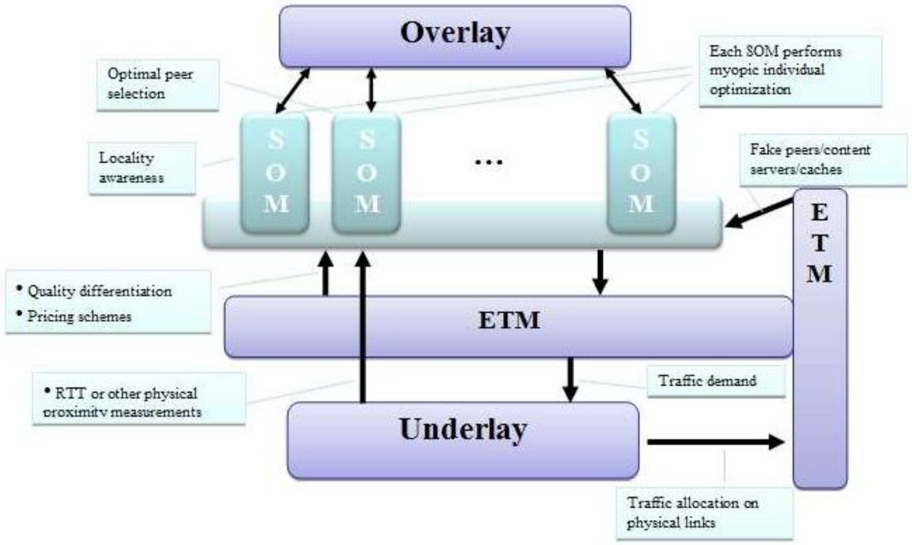

Overlay
Optimal peer selection
Each SOM performs myopic individual optimization
SOM
SOM
...
SOM
Locality awareness
Fake peers/content servers/caches
ETM
• Quality differentiation
• Pricing schemes
ETM
• RTT or other physical proximity measurements
Traffic demand
Underlay
Traffic allocation on physical links

Fig. 1. Interactions among underlay, ETM, SOMs and overlay

cost for ISPs, as discussed in Section 3.2. It should also be noted that SOMs already run under the existing protocols. On the one hand, this imposes constraints to the way ETM can be realistically applied, but on the other hand does provide an opportunity and a challenge for their implementation.

4.1 Realization of Interworking

The purpose of the interworking between ETM and SOMs is the improvement of all players' payoff. That is, the end-users, the overlay provider and the ISP should all benefit from such an interworking, which should thus lead to a win-win-win situation.

Realization of ETM can be divided into two major categories of approaches based on the ”transparency” of those mechanisms as it is seen from the point of view of the overlay. In particular, the first category comprises non- transparent ETM approaches where ETM is not enforced, but there are incentives given to the overlay provider in order to adopt ETM with them, e.g. by means of price differentiation, QoE, etc. The second category comprises transparent (for the overlay provider) ETM approaches where the overlay provider is not involved; on the contrary either the user is given incentives to alter his behaviour according to the information provider by ETM, or ETM is performed ”directly” by introduction of hardware components or other network entities, etc. Three major approaches that reflect the above discrimination are described below.

1. Interworking of SOM and ETM Performed by the Overlay Provider - Non-transparent ETM. In this case, the overlay provider can be given incentives in order to modify his overlay protocol, although these incentives are rather indirect. Here, the interaction between the overlay provider and the operator leads to a win-win situation. Indirectly, 'win' for the overlay provider implies also 'win' for the end-users (see Section 3.2). In order to perform SOMs, it is necessary that the overlay provider is aware of underlay information, e.g. proximity measurements, RTT, link congestion, link costs, etc. This information is provided to the overlay provider by the ETM mechanism. The changes to the overlay protocol that are required have to be implemented, perhaps by means of plugins to the existing software, but have to be carefully done in order to be compatible with older versions of the protocol. In fact, provision of two versions of the software may lead to an even higher benefit for the overlay provider besides the extra satisfaction of the users. For example, he may introduce some charge to the improved version, while keeping the standard version for free. Another possible gain for the overlay provider can be imagined in a scenario where the content is originally offered by the overlay provider, e.g., software patches distributed via BitTorrent. In this case, a fast and efficient download directly influences the popularity of the content provider (which is here the overlay/tracker provider) with his customers.

2. Interworking of SOM and ETM Performed by the End-Users - Transparent ETM. In this case, we assume that the overlay provider is reluctant to modify the application protocol, although this does not necessarily imply that such a modification would not be beneficial to him. So, the interworking between ETM and SOM leads to a win-win situation for the end-users and the operator. Here, the end-users can be given incentives, e.g. QoE, price differentiation, in order to make different choices based on new criteria that would complement or substitute the existing ones. Again, all the information that is necessary is provided to the end-users' clients by the ETM, e.g. by an ISP-provided information service.

Other example of this approach could be the QoS differentiation according to the end users' requests that is being defined in the ITU-T NGN or in ETSI/TISPAN. In this scenario, the end users could request enhanced network performance for overlay based applications in order to optimise the different traffic profiles. In particular, according to the Y.1541[17] the classes of services *HighThroughputData* and *MMStreaming* are specified, the provisioning of network performance guarantees could benefit BitTorrent and/or Joost users.

Effectively, in this scenario, the user could request enhanced capabilities for its overlay application. In this case the ETM must be able to dynamically configure the network resources according to the end users' demands. This request could be also associated with a locality manager. In order to meet this requirement, the ETM could take advantage of the control planes of the next generation networks that allows the dynamic configuration of network resources. Therefore, the ETM must also cover the interaction with the management modules in charge of reserving and configuring network resources, such as the ETSI/TISPAN RACS [18].

This could be the way to build carrier class services based on overlay networks: the ETM mechanisms will receive the request to improve the performance for the overlay application and the ETM could apply different algorithms, such as the combination of locality with QoS guarantees. This could be the basis to implement differentiated pricing schemes: for all those end users that will like to enjoy improved quality, differentiated services will be available, not only for specific operators services but also for overlay applications.

3. Intervention of ISP to SOMs -Transparent ETM. In this case, the operator interferes in the overlay protocol and plays an active role in the re-organization of the overlay network. This can be implemented by introducing extra equipment to its premises. For instance the operator inserts new entities in the overlay network, e.g. caches, ISP-owned peers, etc., that affect the overlay formation. While the mechanisms described there are somewhat artificial with respect to the SO aspect of the overlay, they may provide the possibility for a more direct influence for an ISP.

In this case, the underlay operator participates in the SOMs through these entities in order to achieve its own performance and cost optimization, e.g. reduction of resources consumption, reduction of inter-domain traffic, reduction of monetary cost paid to other operators (transit agreements). There are examples where an ISP just considers its own advantage, e.g., by throttling P2P traffic, regardless of the wishes of its customers. However, the operator must also ensure that the end-users' performance does not degrade; otherwise he will end up losing customers. Here, the interaction of ETM and SOMs results in a win- non-lose situation, because it is enough if the end-users' performance remains the same. In addition, when intervention to SOMs is performed by the operator, all necessary underlay information is directly available by the operator himself. ETM, also by the operator, is necessary in order to concentrate and organize this information.

4.2 Other Issues

There are several other issues concerning the aforementioned interactions. The information exchange between the overlay and the underlay plays a large role, making the timescale of updates and the reliability of information important considerations. Also, it is unclear how the provision of information about the underlay structure may be exploited by malicious peers or overlays. Due to space limitations we will not provide a detailed discussion about these topics here.

5 Application of Interworking Approaches to BitTorrent

In this section, we will provide an example to illustrate how our framework applies to a concrete P2P system. We consider the application file-download, supported by the BitTorrent overlay. The main objective to achieve by implementing changes to this P2P network is to lower the traffic in the inter-ISP links between providers, while improving (or at least without diminishing) the QoS/QoE of the user.

To achieve a traffic reduction on transit links purely by overlay means, a SOM is utilized here. The aim is that peers should prefer download connections to other peers in the same AS instead of exchanging data with remote peers. The SOM used to achieve this goal is the neighbor selection done at the tracker. The tracker assembles a list of peers and returns this list to the querying peer. The decision on which peers to include in that list is made so that the ETM aim is supported. Since in BitTorrent a new peer, who wants to download a specific piece of content, must contact the tracker for that file, this affects every peer in the network.

To this end, information describing the underlay situation must be made available to the overlay. In this case, this is less complicated than in a completely distributed scenario, since only the tracker has to be informed. A possible implementation of this information exchange follows the pull model, where the tracker contacts a separate, ISP-provided information service with the IP of the peer requesting neighbor data. Thus, this method is a non-transparent ETM as described in the previous section. An example of such a service is presented in [19]. The information service is also informed about all peers currently participating, and in case of peers connected via the local ISP also their location and other characteristics. It selects the peers it deems beneficial to both the network and the peer in question using a metric. Then it returns this list to the tracker, which may forward or modify it. To this end, a mapping of peers to ISPs has to take place, in order to allow the tracker to contact the information service of the correct ISP. Alternatively, one central information service might be created, with different implications for the distribution of provider-dependent information. The interface between the tracker and the ISP-provided information service thus serves as an information exchange between the over- and the underlay.

A metric reflecting the ETM purpose described is, e.g., the cost generated by a connection between the local peer and the peer that is evaluated by the metric. This need not be the actual money that has to be spent by the ISP to maintain that connection, but may be normalized. It also could reflect link utilization, i.e., less congested links are treated as less costly. In general, this metric should return better values for peers close to the local peer, i.e., that have short physical links in the same ISP network. The metric

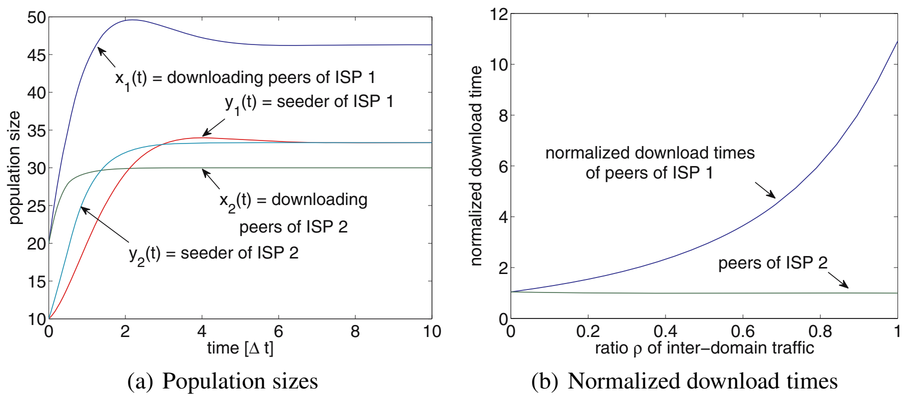


Fig. 2. Qualitative performance results when ISP 1 blocks incoming traffic

allows for an ordering of the peers the tracker knows with respect to the peer requesting a neighbor list. The n best peers are put on the list, perhaps along with a small selection of random peers in order to enhance the stability of the overlay.

As a result in the above example, the response times of the chosen peers are expected to be shorter, and the available bandwidth in a link between the local peer and these peers is expected to be higher, since physical links with low utilization are preferred. Data transfer connections between the local peer and his neighbors therefore may expect higher throughput. This, however, has to be weighted against the different selection of download sources for peers. If extreme precedence is given to peers in the same ISP network as neighbors, the danger of a network separation is created. In this case, peers would experience a much worse download performance, since the resources of all peers sharing the same file are no longer pooled. In general, the neighbor selection process has to avoid compromising the overlay network stability and performance. If it can not be avoided that peers experience a performance degradation, alternatively, other methods of compensation have to be provided by the ISP in order to secure the participation of the users.

Another example for a negative influence of an ETM mechanisms on overlay performance is the naive implementation of peer selection together with traffic blocking. In this scenario, a modified tracker responds to a client's request only with a share ρ of clients in remote ISPs' networks. Additionally, one ISP in this scenario blocks all incoming P2P traffic. We assume that outgoing traffic is not blocked.

To illustrate the impact of ETM on the performance of the peers in that ISP's network, we use the simple fluid model for BitTorrent in [20] and modify it appropriately to take several ISP's networks into account. This allows for a first, rough estimation of the effects of ETM in some abstract scenarios. In the model $x_i(t)$ and $y_i(t)$ is the number of downloaders and seeds within ISP i at time t, respectively. File requests follow a Poisson process with rate λ_i within ISP i. The seeds within ISP i leave the system at rate γ_i. The uploading and downloading bandwidth of a peer is μ and c, respectively. The parameter η describes the effectiveness of the file sharing system and is derived in [20]. The parameter ρ describes the ratio of inter-domain traffic between different ISPs. Then, the system can be described by the following equations:

$$r_1 = \min\{cx_1(t), \mu(1-\rho)\,(\eta x_1(t) + y_1(t))\} \tag{1}$$

$$r_2 = \min\{cx_2(t), \mu\rho\,(\eta x_1(t) + y_1(t)) + \mu\,(\eta x_2(t) + y_2(t))\} \tag{2}$$

$$\frac{dx_i}{dt} = \lambda_i - r_i\ ,\ \frac{dy_i}{dt} = r_i - \gamma_i y_i(t) \tag{3}$$

Fig. 2 shows that the peers in question experience longer download times due to a lower number of eligible sources. Of course, this very coarse-grained model can only give qualitative results hinting at the real system behavior. As a consequence, we plan to investigate such dependencies in-depth in our future work. In general, the incentive for the ISP to provide cost information is the fact that if the overlay prefers to establish low-cost connections, the cost for the ISP to handle the traffic now flowing over these connections is lowered. On the other hand, the end user should be able to observe shorter download times, i.e., a better service quality. Since both parties involved gain an advantage by using the described system instead of the original implementation, it is likely to be accepted.

A transparent alternative to reduce inter-ISP traffic without diminishing the application performance is to attract traffic away from remote networks by simply offering better conditions for downloads in the local network. This can be achieved by placing caches or provider-owned peers that have a high amount of resources to offer.

Due to peer selection mechanisms like tit-for-tat, these entities can bind traffic to them by offering higher upload rates and shorter answer times than remote peers. The client 'chooses' to download from the local caches instead of from remote peers because it experiences a higher throughput by doing so. Therefore, the incentive for the user can be assumed, even if there is no conscious choice, in contrast to the first example, where the overlay/tracker provider or the client version can be changed. Additionally, no information about the underlay is explicitly disclosed to the overlay, which might be an important issue with an ISP.

If popular files are cached, a large portion of the data traffic from the overlay can be affected. No changes in the protocol are necessary for the original peers, making this method transparent to the overlay. The provider alone can implement it without being dependent on a cooperation with the users or overlay providers. Additionally, the provider is able to gather more information about the traffic characteristics of that application and feed it into its network management process.

However, this mechanism has several disadvantages as well. Since much more traffic is created by provider controlled peers than by the information service described above, the resulting cost is also higher. Also, not all peers may be attracted to the provider peers, leading to a comparably lower reduction of traffic on the inter-ISP links. Apart from this, legal issues might prevent a provider from offering storage space for data exchanged in a file-sharing network.

6 Conclusions

In this paper we have investigated the interaction possibilities between the overlay and underlay network. More specifically, due to the tussle between overlay providers and network operators, there exists the need to provide common incentives to all stakeholders to achieve an efficient co-existence of overlay and underlay networks. Initially, we propose to use SOMs as a mean to deploy economic-aware traffic management techniques, leading to minimization of expenditures for the network operators, while offering performance improvements for the end-users. The main contribution of this work is the investigation of the cooperation possibilities that exist between SOMs and ETM. We examine all different approaches to achieve this and we further illustrate our vision by means of a realistic example, specific to the BitTorrent application.

Acknowledgments

This work has been performed in the framework of the EU ICT Project SmoothIT (FP7-2007-ICT-216259). The authors would like to thank all SmoothIT partners for useful discussions on the subject of the paper.

References

1. SmoothIT - Simple Economic Management Approaches of Overlay Traffic in Heterogeneous Internet Topologies (2008), `http://www.smoothit.org/`
2. Saroiu, S., Gummadi, K.P., Dunn, R.J., Gribble, S.D., Levy, H.M.: An analysis of internet content delivery systems. SIGOPS Oper. Syst. Rev. 36 (2002)
3. Sen, S., Wang, J.: Analyzing peer-to-peer traffic across large networks. In: Second Annual ACM Internet Measurement Workshop (2002)
4. Keralapura, R., Taft, N., Chuah, C.-N., Davis, U.C., Iannaccone, G.: Can isps take the heat from overlay networks? In: ACM SIGCOMM HotNets (2004)
5. Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H.: Chord: A scalable Peer-To-Peer lookup service for internet applications. In: Proc. of SIGCOMM 2001 (2001)
6. Castro, M., Druschel, P., Hu, Y.C., Rowstron, A.: Topology-aware routing in structured peer-to-peer overlay networks. In: Schiper, A., Shvartsman, M.M.A.A., Weatherspoon, H., Zhao, B.Y. (eds.) Future Directions in Distributed Computing. LNCS, vol. 2584, pp. 103–107. Springer, Heidelberg (2003)
7. Li, Z., Mohapaira, P.: The impact of topology on overlay routing service. In: Proc. of INFOCOM 2004 (2004)
8. Chun, B.G., Fonseca, R., Stoica, I., Kubiatowicz, J.: Characterizing selfishly constructed overlay routing networks. In: Proc. of INFOCOM 2004 (2004)
9. Ratnasamy, S., Handley, M., Karp, R., Shenker, S.: Topologically-aware overlay construction and server selection. In: Proc. of INFOCOM 2002 (2002)
10. Liu, Y., Liu, X., Xiao, L., Ni, L., Zhang, X.: Location-aware topology matching in p2p systems. In: Proc. of INFOCOM 2004 (2004)
11. Ganesan, P., Sun, Q., Garcia-Molina, H.: Apocrypha: Making P2P overlays network-aware. Technical report, Stanford University (2003)
12. Nakao, A., Peterson, L., Bavier, A.: A routing underlay for overlay networks. In: Proc. of SIGCOMM 2003 (2003)
13. Karagiannis, T., Rodriguez, P., Papagiannaki, K.: Should internet service providers fear peer-assisted content distribution? In: Proc. of IMC 2005 (2005)
14. Cohen, B.: Bittorrent protocol specification (2005), `http://www.bitconjurer.org`
15. Gummadi, R., Gribble, S., Ratnasamy, S., Shenker, S., Stoica, I.: The impact of dht routing geometry on resilience and proximity. In: Proc. of SIGCOMM 2003 (2003)

16. Gimenez, J.P.F.P., Rodriguez, M.A.C., Hasan, H., Hoßfeld, T., Staehle, D., Despotovic, Z., Kellerer, W., Pussep, K., Papafili, I., Stamoulis, G.D., Stiller, B.: A new approach for managing traffic of overlay applications of the smoothIT project. In: Hausheer, D., Schönwälder, J. (eds.) AIMS 2008. LNCS, vol. 5127. Springer, Heidelberg (2008)
17. ITU-T Rec. Y.1541: Network Performance objectives for IP based services (2006)
18. ETSI ES 282 003: Resource and Admission Control Sub-System (RACS): Functional Architecture (2008)
19. Xie, H., Yang, Y., Liu, Y., Krishnamurthy, A., Silberschatz, A.: P4P: Provider portal for applications. In: Proc. of SIGOMM 2008 (2008)
20. Qiu, D., Srikant, R.: Modeling and performance analysis of bittorrent-like peer-to-peer networks. SIGCOMM Comput. Commun. Rev. 34 (2004)

A Fine-Grained Model for Adaptive On-Demand Provisioning of CPU Shares in Data Centers

Emerson Loureiro[*,**], Paddy Nixon, and Simon Dobson

Systems Research Group
School of Computer Science and Informatics
University College Dublin, Dublin, Ireland
{emerson.loureiro,paddy.nixon,simon.dobson}@ucd.ie

Abstract. Data Centers usually host different third party applications, each of them possibly having different requirements in terms of QoS. To achieve them, sufficient resources, like CPU and memory, must be allocated to each application. However, workload fluctuations might arise, and so, resource demands will vary. Allocations based on worst/average case scenarios can lead to non-desirable results. A better approach is then to assign resources on demand. Also, due to the complexity and size of current and future systems, self-adaptive solutions are essential. In this paper, we then present *Grains*, a self-adaptive approach for resource management in Data Centers under varying workload.

1 Introduction

Data Centers usually host different third party applications (e.g., web sites), each of them possibly having different requirements in terms of Quality of Service (QoS) [1,2]. To achieve that goal, however, sufficient resources, like CPU, memory, and storage, must be allocated to each application. One way of achieving this is through a *dedicated model* [3], where sets of servers are exclusively assigned to different application classes. The issue here is to decide the number of servers to be allocated to each class. Another way is through a *shared server model* [3], where a server can host different application classes at the same time. In this case, each class will be assigned a different fraction of the server resources.

The problem is that, in both cases, workload fluctuations might arise [3,4,5], and as a consequence, resource demands across the different application classes will vary [6]. Defining the resources available to a particular application class is thus not an easy task. On the one hand, allocations based on worst-case scenarios could lead to a waste of resources, as most of the times the application class would use less resources than what it currently holds. Average-case scenarios, on the other hand, might cause an application class to be overloaded and thus not able to keep the QoS requirements within reasonable values. A more interesting

[*] This work is supported by UCD Ad Astra Scholarships.

[**] This work is partially supported by Science Foundation Ireland under grant number 04/RPI/1544, "Secure and Predictable Pervasive Computing."

approach is then to assign resources on demand [1], in this case, based on current requirements and availability of resources.

As systems become larger and larger, however, it is unlikely that system administrators will be able to respond to changes in resource demands on time. Also, the frequency upon which that occurs is an issue that further complicates this matter. With this view in mind, it is clear that self-adaptive techniques for dynamically deciding the resources available to application classes are required [4]. In this case, that would mean techniques that calculate and assign resource partitions to each class, over time, based on the current state of the system and its goals.

Based on the above, in this paper we present *Grains*, an approach for adaptively managing the resources, more specifically CPU, assigned to the different application classes in a Data Center under varying workload. The rest of this paper is then organized as follows: in Section 2 we present related works in this area and from that we draw the contributions of our work. Next, in Section 3, we present how our approach contemplates the contributions we are aiming. In Section 4 we then illustrate how our work could be used for managing a system's resources through a simulated case study. Finally, in Section 5, the conclusions and future directions of this work are presented.

2 Background

Resource management has received recent attention from the research community, and consequently, a number of solutions have been proposed. Many of them, e.g., [7,8,9,6,3,10], are focused on the shared server model. More precisely, the different application classes running on each server will receive different resource shares over time. In this case, usually, a controller running on each server will decide at runtime the resource shares each class will hold at a particular time t.

Solutions focused on the dedicated model have also been developed. In [1], for example, the notion of Application Environments (AEs) is defined. In this case, AEs hold one or more Virtual Server (VS), which are deployed in different physical machines. The goal then consists in finding the optimal number of VS each AE will hold and in which machines they will be hosted. When more resources are necessary, new VSs will then have to be created and initialized. The amount of resources assigned to the new VS, however, has to be the same as the amount held by the other VSs hosted in a physical machine, thus leading to allocations which are potentially too large.

In [4] and [11], similar approaches are presented. The main difference is that each AE has a fixed number of VSs, matching the number of tiers of each AE. Therefore, the goal of this solution is to determine the resource capacity allocated to each VS of each AE. Problems will happen, though, if the resources held by a particular AE are not enough for the current demand, as it has no further sources of resources, even if there are idle resources available elsewhere.

Another approach in this context is presented in [12]. In such a work, a Cluster Reserve holds a number of Resource Containers. The goal is then find, for each Cluster Reserve, the allocation each of its Resource Containers will hold in the physical machines available. In this case, a minimum allocation has to be assigned to each Cluster Reserve in each server, even if they don't need it. This means that resources can be wasted if there are many idle service classes.

A similar approach has been proposed in [13], where the servers of the system are partitioned into as many subsets as the number of classes of users. A server can then be shared by different classes of users. However, only up to two classes can be sharing the same server. The works presented in [14] and [15] propose similar ideas. The major difference is that each server in the system is allowed to serve only one application class/tier at a particular moment. Allocating an entire server to an application class/tier, however, can cause over-provisioning.

The assignment of processors to applications is investigated in [5]. More precisely, a CPU Manager is in charge of deciding how many processors to assign to an application class so that its Service Level Agreements can be met. As some of the approaches discussed so far, over-provisioning might happen, since entire processors are assigned to an application.

We then summarize the limitations in current resource management solutions for the dedicated model in the following way:

– **Static allocations:** fixed number of resources available to an application class [4,11]. If an application class is idle, its resources can not be used by one that is under heavy load.
– **Coarse allocations:** entire servers [1,14,15] or processors [5] are assigned to application classes. Allowing servers/processors to be split between different application classes is essential (i.e., fine-grained allocation [16]), since one processor/server dedicated to an application might be too much.
– **Limitations on resource shares:** even when allocations can be fine-grained, either a minimum [12], and possibly idle, or pre-defined [1] allocation is required, or a limited number application shares are allowed to coexist on the server/processor [13].

Based on the above, this work will then extend current resource allocation methods for the dedicated model, by providing *Grains*, **GR**anular **A**llocation **I**n **N**etworked **S**ystems, a solution that aggregates the following characteristics:

– **Dynamic allocations:** resource allocations across the system will change over time to match current needs. Also, consumers (i.e., anything that can use resources) will be using resources from a time-varying set of servers, instead of having a static number of them.
– **Fine-grained allocations:** resources provided by a server can be used by any number of consumers, not being restricted to one or a few application classes.
– **On demand allocations:** allocations across the system will be created and defined on-demand, instead of having a pre-defined or minimum size.
– **Hybrid:** a fine-grained feature will imply in a model that combines characteristics from both the dedicated and shared server models. It will be dedicated in the sense that a specific, but variable, number of resource shares will

be serving an application class at each moment. But it will also be shared, in the sense that a server will be hosting requests from different application classes.

3 Grains

In this section, we describe *Grains*, the solution we are proposing for the resource management problem. We start by describing the high level details of *Grains*, i.e., definitions and architecture, and from that, we present the model that defines in more details how the different parts of the architecture interact and also the optimization problem whose solution will give us the best resource partitioning.

3.1 Concepts and Architecture

In our architecture, a system is viewed as a set of *Servers*. A Server, in this case, is anything that can offer and/or use CPU shares. The CPU resources of a Server are split into two partitions. One of these partitions, called *Dedicated Partition*[1], is to be used exclusively by the Server itself. The other partition, *Donation Partition*, is to be used not only by its own Server but also by any other Server in the system. The size of both partitions are defined in terms of percentage (e.g., the size of a Dedicated Partition might be 65% of the Server's CPU with the remaining 35% being allocated to the Donation Partition). In a multi-core architecture, for example, one of the cores could be defined as the Dedicated Partition and the remaining ones as different shares of the Donation Partition.

A Server can then act as: 1) an application class, thus using its own or another server's CPU shares to fulfill its QoS requirements; 2) a resource provider (like Virtual Servers and Resource Containers described in Section 2), thus providing resources to other Servers; or 3) both. Note that we are not making any assumption about the nature of the Server in terms of how it is deployed. It can be an actual physical server or even a virtual one running on a host operating system in a physical machine.

The CPU shares in a Donation Partition can be allocated to more than one Server at the same time. Shares are allocated in *Blocks* of a specific size, which are the atomic unit (i.e., grains) of allocation and determines the percentage of the Donation Partition that will be allocated. For example, if the size of the Block is 10%, shares in the Donation Partition will have a size which is a multiple of 10 (e.g., 20%, 30%). This way, two different Servers A and B could then hold different shares in another Server's Donation Partition, say, 20% for A and 30% for B. Besides, these shares could have their size changed at any time.

Note that, by having a Donation Partition that uses 100% of a Server's CPU resources, and blocks with sizes of 100%, we then reduce our solution to the case where Servers are allocated exclusively for an application class. This is the case

[1] The Dedicated Partition is defined only for modelling purposes, as it does not play any role in the rest of our solution.

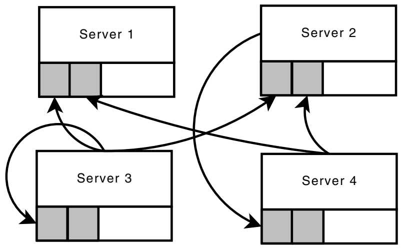


Fig. 1. Possible configuration of a system with four Servers

with some related works in this area. As one can notice, then, we enable this same kind of allocation in a single model, with the advantage of still being fine-grained. It just depends on how the size of the block and the Donation Partitions are set.

In Figure 1 we illustrate a sample configuration of a system, focusing on how Servers, Donation Partitions, and shares relate to each other. In the figure, the gray rectangles on each Server indicate different shares in the Server's Donation Partition. The arrows, on the other hand, indicate which share is assigned to which Server, by pointing from the Server that is holding the share to the one that is providing it.

The extra CPU shares a Server will receive, coming either from its own Donation Partition or from a remote Server's one, will be based on the Server's *Utility Function*. In short, the Utility Function of a Server will indicate, at any point in time, how useful it is for that Server to receive extra CPU shares.

By taking the Utility Functions of all Servers together, a *Global Controller* will then take care of deciding the shares that each Server will receive, and where they will come from. Therefore, the Utility Function of the Servers should be designed in a way that, by having the Global Controller to decide the best CPU shares for each Server over time, the system as whole is driven towards a common goal (i.e., load balancing, improving overall response time).

3.2 Model

From the concepts presented so far, a formal model of the system has been defined. In such a model, let S be the set of Servers in the system, then:

$$S = \{s_i : i \in \mathbb{N} \wedge i \in [1, n]\}$$

where n is the number of Servers in the system. As we've mentioned, the Donation Partition is a share of a Server's CPU that is available to be used by the Server itself as well as by other Servers in the system. Let p^i be the current size of Server's i Donation Partition, then:

$$\forall s_i \in S \left(p^i = n.R\right),$$

where R is the size of the allocation block, $0 \leq R \leq 1$, and n is the number of blocks available in the Donation Partition to be assigned. Consequently,

$$(n \in \mathbb{N}) \wedge \left(0 \leq n \leq \left\lfloor \frac{1}{R} \right\rfloor \right). \tag{1}$$

With that, we then model the fact that Donation Partitions have to have a size defined in terms of a block, and that such a size will be within 0% and 100% of the total server CPU capacity.

Since in each Donation Partition shares can be allocated to one or more Servers, we then have to keep track of the current shares being held in each Server's Donation Partition. Based on that, let d^i be the set of the current shares allocated in the donation partition of server i, then:

$$d^i = \{\{s_k, r^k\}^*\} \tag{2}$$

where s_k is a server having a share in s_i's Donation Partition, $s_k, s_i \in S$, and r^k is the share held by s_k. As mentioned before, shares are allocated in blocks, just like in the definition of the Donation Partition's size. Consequently, each allocation r^k in d^i is of the form $r^k = n.R$ and 1 holds. Given the total size of a Server i's Donation Partition (p^i) and the current allocations on it (d^i), we then denote the amount of free shares, f^i, in such a Donation Partition, by:

$$f^i = p^i - \sum_{j=1}^{|d^i|} d^i_{j2}$$

Since the size of the Donation Partition can be changed at runtime, we need to specify which properties we would like to hold once that happens. For example, we do not want a Donation Partition to be resized to less than the amount of shares currently in use on it, as that would imply reducing each of the shares accordingly. We then denote by $z^i(r)$ the function that will cause the Donation Partition of a Server i to be resized by the amount r, $z^i(r) = p^i + r$. Clearly, if $r > 0$, the Partition is having its size increased. If $r < 0$, on the other hand, its size is being decreased. For that, however, the following has to hold:

$$\left(p^i \geq p^i - f^i\right) \wedge \left(\forall j \in \left[0, |d^i|\right] \left(d^i_{j2} = d^i_{j2}\right)\right) \tag{3}$$

Basically, we are asserting that the size of a Donation Partition can never be less than what is currently being used on it and also that all shares from the different Servers on it will remain the same.

As we are talking about shares being allocated in a Server's Donation Partition to other Servers, we need to formalize that in terms of the system properties that have to hold after an allocation is defined. For that, then, $a^{s_i s_k}(r)$ denotes a function that allocates a share r in the Donation Partition of Server i to Server k. Clearly, if $i = k$, then Server i is receiving extra CPU shares from its own Donation Partition. For that to happen, the following has to hold:

$$\left(\exists j \in \left[0, |d^i|\right] \left(d^i_j = \{s_k, r\}\right)\right) \wedge \left(\forall w \in \left[0, |d^i|\right] \left(w \neq j \rightarrow d^i_{j1} \neq s_k\right)\right),$$

which states that one, and only one, pair <*server*, *allocation*>, where *server* is s_k, should exist in the current set of allocations of Server's i Donation Partition. In this case, all properties of 2 must hold.

Finally, we denote by $u^i(l, r)$ the utility function of Server i, where l is the new share that Server i will receive from its own Donation Partition and r is a vector of the new shares that Server i will receive from remote Servers. More formally:

$$u^i : \left[-l^i, f^i\right] \times \left[-r^{ik}, f^k\right]^{|S|} \rightarrow [0, 1] \tag{4}$$

where l^i is the current share Server i holds in its own Donation Partition and r^{ik} is the current share Server i holds in the Donation Partition of Server k, $\forall k, i \in [0, |S|]\,(k \neq i)$.

From the constructs defined so far, we can then model the optimization problem the Global Controller is supposed to solve. We assume that the controller will execute continuously over time, over a fixed or varying frequency. Every time it runs, it thus has to find the best way of distributing shares in the Donated Partition of all Servers among them. That will consist of solving the following optimization problem:

$$\max_{R^l, R^m} \sum_{i=1}^{|S|} u^i(l^i, r^i) \tag{5}$$

where $R^l = \{l^{1*}, l^{2*}, ..., l^{n*}\}$, $R^m = \{r^{1*}, r^{2*}, ..., r^{n*}\}$, l^{i*} is Server i's optimal CPU share coming from its own Donation Partition, and r^{i*} is Server i's optimal CPU share coming from remote Servers' Donation Partition, given that the following constraint hold:

$$N \leq f^i,$$

where N is the sum of the new shares allocated on Server i, $N = R^l_i + \sum R^m_{k2}$, $\forall k \in [0, |R^m|]\,(R^m_{k1} = s_i)$. This constraint simply states that the sum of new shares on each Server has to be less than the current amount of free shares on it.

4 Case Study

In this section, we present a case study to illustrate the use of *Grains* in the resource management problem. In our case study, we consider a scenario with two physical machines, each of them running a Server, as defined in our model. Each server is associated with an application class (A and B). The global goal of the system is to keep the average response time of the requests in each Server as bellow as possible from a target value (e.g., 3s).

To achieve that, the CPU shares each Server holds will be reallocated on demand, based on the aggregate utility of their utility functions. We consider that the Global Controller executes continuously, at specific intervals, which we will call *iteration*. At each iteration, the Global Controller decides the best CPU shares for each Server, and performs the allocation, which will then hold until the next iteration. In this case, the best CPU shares for the Servers will be those that will cause their response time to be as less as possible from the target value.

The utility function u of a Server is then defined in the following way:

$$u^i(l,r) = \begin{cases} t^i(l,r) & \text{, if } r^i = \emptyset \\ \frac{h^i(l)+y^i(r)}{2} & \text{, if } r^i \neq \emptyset \end{cases} \tag{6}$$

where 1) r^i is the set of the current shares held by Server i that come from remote Servers' Donation Partition, and, since we have only two servers, $|r^i| = 1$ (i.e., a Server i can only have shares from up to one remote Server); 2) $t^i(l,r)$ is the utility of Server i receiving shares l and r, respectively from its own Donation Partition and from a remote Server's one, given that it does not hold any remote share yet ($r^i = \emptyset$); 3) $h^i(l)$ is the utility of Server i receiving the new share l from its own Donation Partition considering that it already holds a remote share; and finally 4) $y^i(r)$ is the utility of the remote share held by Server i receiving the new share r.

Function $t^i(l,r)$ in Equation 6 is then defined in the following way:

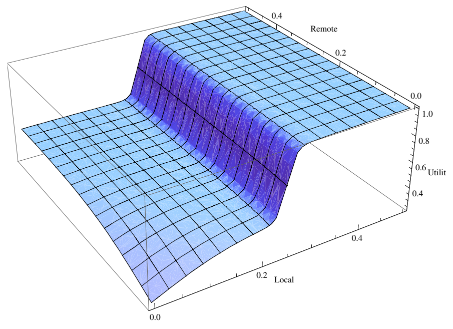
$$t^i(l,r) = \frac{\left(\frac{1}{\left(1+\lambda * e^{\left(\frac{-(1/b(l))}{\eta}\right)}\right)} + \frac{1}{\left(1+\theta * e^{\left(\frac{-(1/a(l,r))}{\zeta}\right)}\right)}\right)}{2},$$

where $b(l)$ is the expected response time in Server i until the next iteration, given the extra local share l, $a(l,r)$ is the expected remaining response time in Server i given that it will receive the extra share l from its own Donation Partition and also the extra share r from a remote Server's Donation Partition, and η and ζ are specific constant values for defining the growth of $t^i(l,r)$ over the x and y axis.

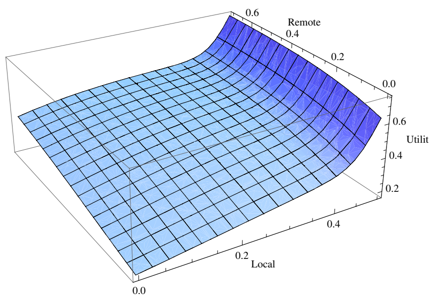
The general idea of this utility function is to make it grow fast to 1 along the x axis (i.e., local share received), once the target response time is reached, indicating the fact that achieving a response time that is less than the target one with a particular local share is of high utility (requests won't have to be redirected from the Server to a remote share). This is the case illustrated in the graph of Figure 2a. The value on the x axis where this growth occurs is controlled by λ, which is calculated by finding λ such that $\frac{\partial^2 t}{\partial l^2} = 0$, with $\theta = 0$ and l being the result of solving $b(l) = 2$, with 2 being the target response time to be reached.

However, when the local share received is not enough to cause the response time to be less than the target one, the utility still increases considerably as the remote share received increases. This is the case illustrated in the graph of Figure 2b. The growth of the function in this direction is controlled by θ, calculated by finding θ such that $\frac{\partial^2 t}{\partial r^2} = 0$, with $\lambda = 0$, $l = 0$, and r being the result of solving $a(l,r) = 2$. Notice that, in this case, there is no point along the x axis to cause the utility to grow fast to 1, which would be the case if the target response time is reached. But still, the utility continues to grow as more remote shares are provided.

As for the utility functions $h^i(l)$ and $y^i(r)$ of a Server i, they behave similarly to $t^i(l,r)$. The difference is that each of these functions only consider either the



(a)



(b)

Fig. 2. Graph of t (a) with enough local shares (b) without enough local shares

share received from i's Donation Partition, for $h^i(l)$, or the share received from a remote Server's Donation Partition, for $y^i(r)$. The definitions for both of these functions have thus been omitted from this paper.

Now we'll illustrate a scenario where the extra CPU shares of Server 1 will be dynamically adjusted to meet current demand. In this scenario, we will be looking at the following aspects: 1) the current extra CPU share held by the Server coming from its own Donation Partition, 2) the current extra CPU share held by the Server coming from Server 2's Donation Partition, and 3) the response time in Server 1. The idea with this scenario is illustrate not only the shares of Server 1 being dynamically adjusted but also the fact that it will receive remote shares on demand, when the local shares it holds is no longer enough to keep the response time bellow or equal to the target value. Initially, Server 1's Donation Partition uses 50% of its CPU, which means that the other 50% are to be used exclusively by it. The total local share held by Server 1 is then 50%, as illustrated by the solid line in the graph of Figure 3a (in the graph, values are in between 0 and 1).

1. Server 1 initiates with a response time of 1.5s, with 2s being the target value, as illustrated in Figure 3b.
2. At iteration 5, the expected response time increases to 2.65s. The Global Controller then decides that the best allocation is to give an extra 20% of local share to Server 1, resulting in a response time around 1.8s. For that, the Global Controller then calls $a^{s_1 s_1}(0.2)$, as specified in the model.
3. At iteration 10, the expected response time increases again, now to 4s. In this case, even by providing the remaining local share (i.e., 30%) to Server 1, the response time on it would not be less than the target value. In this case, the Global Controller then decides to allocate 40% of the Donation Partition of Server 2 to Server 1 (dashed line in the graph of Figure 3a), in a way that by redirecting requests to Server 2, a better response time can be achieved. The result of this decision is then calling $a^{s_1 s_1}(0.3)$ and $a^{s_1 s_2}(0.4)$, to apply the changes in Server 1's local and remote shares. By doing that,

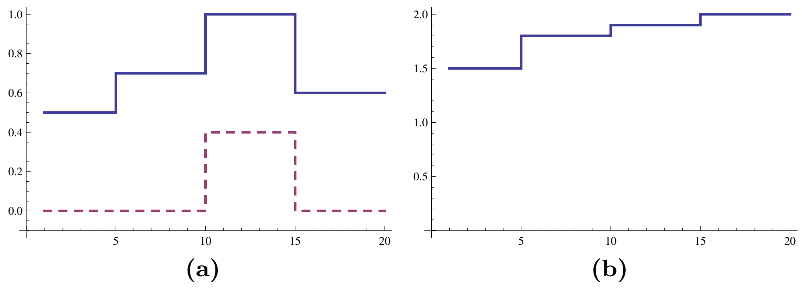


Fig. 3. (a) Variation of the local shares (solid line) and remote shares (dashed line) held by Server 1. (b) Variation of the response time.

the resulting response time of Server 1 is around 1.9, thus bellow the target value.

4. At iteration 15 then, all requests occupying the remote share used by Server 1 are finished, so such a share is revoked by the Donation Partition of Server 2, causing the remote share of Server 1 to go to 0 again. Also, the expected response time in Server 1 drops to 1s, causing the Global Controller to decrease the extra local share held by Server 1 to 10%, since the current local share it holds is too much for that response time. Server 1 then ends up with a total local CPU share of 60%, a setting that causes the response time to be exactly 2, still within the specified target value (in this case, we have chosen to keep the response time bellow the target value but without overusing the CPU resources).

Note that, in item 3, Server 1 had *no* pre-defined or minimum share in Server 2, as is the case with some approaches. Instead, its remote share was allocated at runtime and only when necessary. Also, from then on, Server 1 remote share's utility, $y^i(r)$ in Equation 6, would affect the optimization process, by specifying which size would be the most useful for it in a particular iteration. This means that if it needs more resources, its size would simply be increased, instead of creating more and more shares on different Servers, as some of the current solutions do.

Even though we have not illustrated changes in the size of the Donation Partition, it is worth stressing the fact that, as we have showed, this is supported by the model, using the function $z^i(r)$, as explained in Section 3.2. If at some point, it is decided that the Donation Partition of Server 2, for instance, should be resized to, lets say, 70% of its CPU capacity, all it would need to be done is a call to $z^i(0.2)$ (since it currently has a size of 0.5). This way, more resources would be available on this Donation Partition to be used by Server 1, with all the important properties regarding the allocated shares still holding.

5 Conclusions

In this paper we presented *Grains*, an approach for dynamically managing CPU resources in Data Centers under varying workload. A formal model for that has been proposed, thus clearly defining relationships within the system as well as properties that have to hold over its execution. As we have showed, a number of solutions within this context can be found. However, limitations in these solutions have been pointed out, such as 1) being fixed to a specific number of servers, 2) creating/initializing new Virtual Machines when remote shares are necessary, instead of just resizing their shares, or 3) creating idle shares to satisfy a minimum allocation constraint. In our case, we were able to cope with the resource management problem with a solution that overcomes such limitations (e.g., resources from a Server can be assigned to a number of other Servers and shares are dynamically allocated with no minimum/pre-defined values). Even though we have focused on CPU assignment, the *Grains* model could well be used for other kinds of resources, as long as they can be divided and shared (e.g., storage).

Whereas the approach we propose in this paper is interesting, there are issues to consider in the future. Dynamically deciding the proper size for a Server's Donation Partition, for example, is an issue to be further investigated. By that we mean finding what is the best size for the Donation Partition of a Server, considering parameters like how often it uses extra shares and how idle its Dedicated Partition is. That would allow us to make a better usage of the CPU resources, by setting the size of each Server's Donation Partition to the minimal CPU requirements it would need, and allocate the rest on demand. Another approach would be to set the size to zero, so that all resources a Server holds would be decided based on its current needs. As one can note, our approach is flexible enough for both cases, an aspect that is not found in other solutions.

Another area of future work is to decentralize the control over the system. In this case, the idea is to design algorithms that will converge to the optimal allocation by having different Servers in the system taking decisions using as much local information as possible. The best allocation, in this case, would thus emerge as the result of local interactions between the Servers of the system.

References

1. Wang, X.Y., Lan, D.J., Wang, G., Fang, X., Ye, M., Chen, Y., Wang, Q.: Appliance-based autonomic provisioning framework for virtualized outsourcing data center. In: ICAC 2007: Proceedings of the Fourth International Conference on Autonomic Computing, Washington, DC, USA, p. 29. IEEE Computer Society, Los Alamitos (2007)
2. Harada, F., Ushio, T., Nakamoto, Y.: Adaptive resource allocation control for fair qos management. IEEE Transactions on Computers 56(3), 344–357 (2007)
3. Chandra, A., Gong, W., Shenoy, P.: Dynamic resource allocation for shared data centers using online measurements. In: Proceedings of the 2003 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, New York, NY, USA, pp. 300–301. ACM, New York (2003)

4. Wang, X., Du, Z., Chen, Y., Li, S.: Virtualization-based autonomic resource management for multi-tier web applications in shared data center. Journal of Systems and Software (in press, 2008)
5. Guitart, J., Carrera, D., Beltran, V., Torres, J., Ayguadé, E.: Dynamic cpu provisioning for self-managed secure web applications in smp hosting platforms. Computer Networks 52(7), 1390–1409 (2008)
6. Liu, X., Zhu, Z., Singhal, S., Arlitt, M.: Adaptive entitlement control of resource containers on shared servers. In: 9th IFIP/IEEE International Symposium on Integrated Network Management, pp. 163–176. IEEE Computer Society, Los Alamitos (2005)
7. Menasce, D.A., Bennani, M.N.: Autonomic virtualized environments. In: Proceedings of the 2006 International Conference on Autonomic and Autonomous Systems, Washington, DC, USA, p. 28. IEEE Computer Society, Los Alamitos (2006)
8. Ionescu, D., Solomon, B., Litoiu, M., Mihaescu, M.: A robust autonomic computing architecture for server virtualization. In: Proceedings of the 2008 International Conference on Intelligent Engineering Systems, Washington, DC, USA, pp. 173–180. IEEE Computer Society, Los Alamitos (2008)
9. Garbraick, P., Naik, V.K.: Efficient resource virtualization and sharing strategies for heterogeneous grid environments. In: 9th IFIP/IEEE International Symposium on Integrated Network Management, pp. 40–49. IEEE Computer Society, Los Alamitos (2007)
10. Tesauro, G., Walsh, W.E., Kephart, J.O.: Utility-function-driven resource allocation in autonomic systems. In: Proceedings of the Second International Conference on Autonomic Computing, Washington, DC, USA, pp. 342–343. IEEE Computer Society, Los Alamitos (2005)
11. Padala, P., Shin, K.G., Zhu, X., Uysal, M., Wang, Z., Singhal, S., Merchant, A., Salem, K.: Adaptive control of virtualized resources in utility computing environments. In: Proceedings of the 2nd ACM SIGOPS/EuroSys. European Conference on Computer Systems 2007, pp. 289–302. ACM, New York (2007)
12. Aron, M., Druschel, P., Zwaenepoel, W.: Cluster reserves: a mechanism for resource management in cluster-based network servers. SIGMETRICS Perform. Eval. Rev. 28(1), 90–101 (2000)
13. Zhang, J., Hämäläinen, T., Joutsensalo, J.: A new mechanism for supporting differentiated services in cluster-based network servers. In: Proceedings of the 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS 2002), Washington, DC, USA, p. 427. IEEE Computer Society, Los Alamitos (2002)
14. Urgaonkar, B., Chandra, A.: Dynamic provisioning of multi-tier internet applications. In: Proceedings of the Second International Conference on Automatic Computing, Washington, DC, USA, pp. 217–228. IEEE Computer Society Press, Los Alamitos (2005)
15. Sivasubramanian, S., Pierre, G., van Steen, M.: Towards autonomic hosting of multi-tier internet applications. In: Proceedings of USENIX Hot Topics in Autonomic Computing, Washington, DC, USA. IEEE Computer Society Press, Los Alamitos (2006)
16. Steinder, M., Whalley, I., Carrera, D., Gaweda, I., Chess, D.: Server virtualization in autonomic management of heterogeneous workloads. In: Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management, Washington, DC, USA, pp. 139–148. IEEE Computer Society Press, Los Alamitos (2007)

DySSCo - A Protocol for Dynamic Self-Organizing Service Coverage

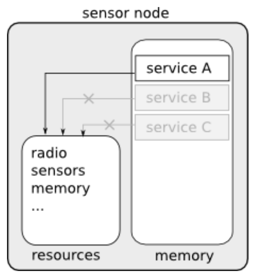
Martin Lipphardt[1], Jana Neumann[2], Sven Groppe[2], and Christian Werner[1]

[1] University of Luebeck, Institute of Telematics, Germany
`lastname@itm.uni-luebeck.de`
[2] University of Luebeck, Institute of Information Systems, Germany
`lastname@ifis.uni-luebeck.de`

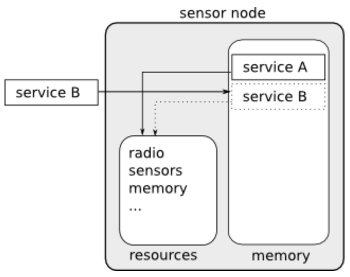
Abstract. Service oriented middleware draws a lot of attention in current research on sensor networks. The automatic distribution of services within a network and the preservation of this distribution is a fundamental aspect of network applications with self-x properties. The network gains the ability to react on mobility, network fragmentation, node failures and new user demands. This paper proposes a distributed self-organizing algorithm for service distribution and preservation of this distribution using demanded coverages for the services. After discussing the theory of the convergence of the algorithm, this paper presents a real-world deployment of a sensor network scenario and evaluates the performance of the algorithm.

1 Introduction

Wireless sensor networks consist of hundreds or thousands of tiny sensor devices with limited resources. These sensor devices collaborate in order to perform complex tasks like observing their environment. For this purpose the sensor nodes must be programmed with application software. In current deployments, all nodes are programmed with an application before deployment, leaving the network inflexible to react on changes in environmental conditions or user demands. In order to increase flexibility and performance of sensor networks, current research examines heterogeneous networks, where nodes perform different tasks and therefore can vary in software setup as well as in hardware. In such networks nodes adopt different roles, e. g. recording sensor data or replicating recorded data, based on which they execute specific functions or program parts. In the following we will refer to these functions as *services*. Every service uses resources on the node such as dynamically allocated memory, radio, sensors and ultimately energy. The ability to activate and deactivate services as shown in Figure 1(a) allows the nodes to perform their designated task without wasting resources for unneeded services. Alternatively, services can migrate onto the nodes as depicted in Figure 1(b), if the hardware is suitable for the migration of program code. The migrated service is then executed on the node. The migration of services ensures that only the functionality is present on the node, which is required for its designated task. Both schemes allow an adaptation of



(a) Activated services (white) and deactivated services (gray)



(b) Migration of a service onto a node

Fig. 1. Scenarios for heterogeneous software setup on sensor nodes

software during the runtime and are important concepts for self-x properties in sensor networks, like self-organization, self-configuration, self-optimization and self-healing. This enables the network to optimize and adapt to changed conditions single-handed without any user interactions.

In this paper, we focus on distributing functionality in sensor networks. For a lot of applications it is sufficient to have a certain density or *coverage* of a service within the network. This distribution can not be done in a static manner, since it would require additional administration and would be inflexible to new user requirements, mobility, node failures and network fragmentation. We propose a decentralized, self-organizing algorithm that achieves and preserves a uniform service coverage in the network during the deployment. The algorithm is based on local knowledge of the nodes about available services and the global coverage demanded by the application for each service. By adapting to the coverage of a service locally by activation or migration the algorithm achieves a global coverage of the service in the whole network. The service information (or in case of migration the service itself) is propagated through the network without being dependent on any routing schemes. Our algorithm can be used either for activating sensor node program parts or for replicating and migrating code in sensor networks. For the rest of the paper we will use the term *activation of a service* as a synonym for both, migration and activation of functionality.

Because of their ability of self-organization, the sensor nodes are able to react on node failures and newly included nodes by adding and removing services during runtime without intervention of the user. Additionally the sensor network can adapt to changed user demands, like a demand for higher granularity of the monitored data or higher reliability.

The rest of the paper is organized as follows: First we outline related work in the field of code propagation and service placement in Section 2. We describe the functionality of the algorithm in Section 3. Section 4 presents theoretical considerations of the convergence of the algorithm. In Section 5 we present a deployment of sensor nodes demonstrating the usability of the algorithm under real-world conditions and evaluate the algorithm on basis of suitable metrics. Finally, Section 6 presents the conclusion and future work.

2 Related Work

In this section, we outline related research which concentrates on code distribution and service placement in sensor networks and mobile ad-hoc networks. Injecting and distributing code throughout the network is an important aspect in the field of (re)programming sensor networks in order to achieve self-organizing behavior. Different code distribution approaches have been developed [9,4]. The code is mostly distributed per gossiping or flooding to every node in the sensor network [4]. However, in [9] nodes have different predefined roles depending on their functionalities. Using this, the code will be propagated only to nodes of the suitable role. Other approaches focus on mobile agents in order to distribute functionalities in sensor networks [5,3,2]. In contrast to services, which offer specific functionalities that are used by the application, mobile agents are dynamic, autonomous, intelligent programs, which migrate and replicate themselves depending on their tasks. Levis et al. [5] proposed a virtual machine on each node to install code. In order to program the nodes of the network, the code replicates itself and is flooded throughout the network. However, these approaches do not concentrate on distributing functionality on basis of suitable metrics, but on how migration is technically realized.

Similar to code propagation, service placement is used to add selective functionalities to the network in the field of service centric architectures. By now, there are many approaches which concentrate on discovering and using services in sensor networks and mobile ad-hoc networks [8,7,1]. Though, they either do not focus on how services are getting into the network or presume that services are installed on the nodes before deployment. [10] summarizes service placement strategies only for mobile ad-hoc networks. The goal of these strategies is to place single services within a network at an optimal location determined by a given metric. By contrast, we aim to achieve a global service coverage within the whole network by replicating different services in order to distribute the functionality among the nodes. To our knowledge, distributing code or functionality with demanded coverage throughout the network and preserving this coverage in a distributed manner has not been examined yet.

3 The Protocol

How can we achieve an uniformly distributed coverage of a service in a sensor network? One way is to forward the information of the service coverage and the service itself through the network. Additionally, we must determine the nodes that activate the service. An algorithm must adapt the distribution of the service to network changes and perform correctly even under radio transmission characteristics that cause e. g. undirectional links. At the same time the algorithm must be lightweight in terms of code size and complexity and cause only little traffic.

3.1 Overview

The *DySSCo* algorithm is a distributed algorithm that achieves a demanded coverage for a service within a network without being dependent on any routing schemes. We define the percentage of nodes in the network that at least must run the service s as demanded coverage $c_{demanded}(s)$. The demanded coverage for each service s is previously specified by the user. By trying to fulfill the demanded coverage locally within the one-hop neighborhood, the algorithm affects the coverage globally in the whole network. In order to achieve this demanded coverage, every node sends within a given time interval t a beacon with the globally unique service identifiers s_id of its currently active services s and the demanded coverage $c_{demanded}(s)$ in percent for each of them. Table 3.1 shows an example of such a beacon. Note that a node might not provide any active services. Hence, the beacon is empty, but has to be sent anyway in order to provide information about the number of direct neighbors, their activated services and their coverage within his direct neighborhood for each node.

Table 1. Content of a *DySSCo* beacon

s_id	$c_{demanded}(s)$
5	50
12	33
17	75
...	...

Based on this information, each node can determine whether the coverage of a service is fullfilled within its direct neighborhood and whether it should activate or deactivate a service.

3.2 Decision Process

In contrast to the demanded coverage, we define the currently achieved percentage of nodes runnig the service s in the direct neighborhood of a node n as current coverage $c_{current}(s)$. In order to decide whether to activate or deactivate a specific service s, each node n has to calculate its current coverage $c_{current}(s)$ among the direct neighbors and compare it to the demanded coverage given by $c_{demanded}$ in the beacon. Let $N(n)$ be the number of direct neighbors of the node n and $S_s(n)$ the number of direct neighbors of n that have activated the service s. $S_s(n)$ includes the node n itself, if n has activated the service s. The current coverage $c_{current}(s)$ of the service s can be easily calculated in percent by:

$$c_{current}(s) = \frac{S_s(n) * 100}{N(n) + 1} \tag{1}$$

The decision process based on the current coverage $c_{current}(s)$ is shown in Algorithm 1. Whenever a *DySSCo* beacon is received, the receiving node chooses a random backoff time *backoff* for a callback (lines 6 - 8). This backoff prevents the nodes from reacting simultaniously, what would lead to an alternating process of activation and deactivation. If the backoff expires (line 9), each node checks the current coverage $c_{current}(s)$ for each service s (line 11 - 23). If the current coverage $c_{current}(s)$ is lower than the demanded coverage $c_{demanded}(s)$ for s and the node n has not already activated the service s, n activates the service s (line 13). If $c_{current}(s)$ is higher than the demanded coverage $c_{demanded}(s)$ for the service s and n has activated the service s, the node n calculates the reachable coverage $c_{reachable}(s)$. The reachable coverage $c_{reachable}(s)$ is the current coverage which results if the node deactivates its service s.

Only if the $c_{reachable}(s)$ is still higher than the demanded coverage $c_{demanded}(s)$, the node n deactivates the service s (line 16 - 21). If a nodes changes the status of one of the services, it immediately sends a *DySSCo* beacon, to keep the direct neighborhood informed about the current coverage of the services (line 24 - 25). Services that do not appear in the beacon are no longer active on the node.

Changes in current service coverage in a neighborhood are influencing indirectly surrounding neighborhoods and thus global network service coverage. In that way, the network is able to balance the global coverage of services.

3.3 Adjustments and Parameters

When looking at the algorithm, some parameters must be chosen like the interval t for the *DySSCo* beacon, the maximum length of the random *backoff* interval and the demanded coverage $c_{demanded}(s)$ for a service.

Based on the *DySSCo* beacon that is sent every t intervals, the nodes update the size of their neighborhood as well as the current coverage of the services. Therefore the choice of t is dependent on the network characteristics, the mobility, the traffic, and the needed flexibility as well as on the presumption, whether to migrate code or just activate program parts. It becomes clear that the smaller the size of t, the faster the network reacts on topology changes. The main drawback is, that this leads to higher traffic, especially when whole code segments must be migrated. Additionally, some changes in the topology might be just some fluctuations on the radio channel. So there is no general advice for the choice of t.

As mentioned above, the backoff interval prevents nodes from reacting on a change in service coverage simultaniously. As an example, let us consider three nodes, that are directly connected. If one node activates a service with a demanded coverage of 50%, both remaining nodes without backoff will activate this service simultaniously. Afterwards, all nodes will deactivate the service because the coverage is too high. This can lead to an infinite alternating process of activation and deactivation. Therefore, a random backoff is needed. In order to preserve a quick reaction of the nodes, we make the maximum value dependent on the size of the neighborhood. The more neighbors, the higher the maximal random value in order to minimize the chances of concurrent reactions of the nodes.

Algorithm 1. The *DySSCo* Algorithm

```
1: while TRUE do                                        // runs periodically
2:     on timeout callback                              // time $t$ expired
3:     $sendBeacon()$                                   // sending DySSCo beacon
4:     $updateNeighborList()$                           // updates the number of direct neighbors
5:     register timeout callback                        // register callback in $t$ for next beacon
6:     on receive DySSCo Beacon                         // receiving a DySSCo beacon
7:     $updateServiceList()$                            // update list $known\_services$
8:     register/reset backoff callback                  // register/reset callback in a random backoff time
9:     on backoff callback                              // backoff time expired
10:    $changed = false$
11:    for each $s$ in $known\_services$ do             // loop over all known services
12:        if $c_{current}(s) < c_{demanded}(s) \wedge s$ inactive then
13:            $activate(s)$                            // activate service $s$
14:            $changed = true$
15:        else
16:            if $c_{current}(s) > c_{demanded}(s) \wedge s$ active then
17:                if $c_{reachable}(s) > c_{demanded}(s)$ then
18:                    $deactivate(s)$                  // deactivate service $s$
19:                    $changed = true$
20:                end if
21:            end if
22:        end if
23:    end for
24:    if $changed == true$ then
25:        $sendBeacon()$                               // sending DySSCo beacon
26:    end if
27: end while
```

The demanded coverage $c_{demanded}(s)$ for a service s is dependent on the applications requirements for this service. However, to achieve the global effect of the *DySSCo* algorithm, the demanded coverage in a network with a neighborhood size of k must be larger than $\frac{1}{k}$. Otherwise no direct neighbor would activate the service since the coverage is already fulfilled. By this, the information of the service would not be forwarded to further nodes throughout the network.

4 Convergence Considerations

When looking at the simplicity of the distributed algorithm, one question becomes obvious: Is the algorithm able to find a stable distribution of services in any network? In a stable distribution of services, the algorithm does not perform any changes, if the network topology and the demanded coverage remain constant. It is easy to see that, e.g. in a network with an odd number of nodes, the algorithm cannot achieve a global coverage of 50%. Having an even number of direct neighbours, the coverage c calculated by a node n can also not be exactly a coverage of 50%. Due to this reason, the nodes precedingly calculate whether a deactivation of its service is still above the demanded coverage, which thereby is a lower bound for the coverage.

We analyzed all possible graphs with a size of 3, 4 and 5 nodes. Note that the graphs with more nodes inherently include the graphs with less nodes by including cases with isolated subgraphs. The number of graphs for each nodecount is shown in Table 2. We calculated all possible distributions of a service with different given demanded coverages. The results in Table 3 show that for different demanded coverages only about 5% of the graphs do not have a possible stable distribution of the services.

Table 2. Graphs for different nodecounts

nodecount	3	4	5
valid graphs	64	4096	1048576

Table 3. Percentage of graphs with no solution for different nodecounts coverages

coverage	*nodecount*		
(in %)	3	4	5
25	2 (3%)	214 (5%)	56392 (5%)
33	2 (3%)	178 (4%)	53532 (5%)
50	2 (3%)	178 (4%)	53532 (5%)
66	0 (0%)	30 (<1%)	39988 (4%)
75	0 (0%)	0 (0%)	2324 (<1%)

The analysis of the graphs with no stable solution shows that most of these graphs have unidirectional cycles. Figure 2 shows the graphs with node count 3, where the algorithm does not find a stable state for a demanded coverage of 50%. When a node in these graphs activates the service s, it causes the downstream neighbour to deactivate the service. Having an odd number of nodes in this unidirectional cycle causes an infinitive loop of successive activation and deactivation. In order to find out the practical relevance of these results, we conducted an experimental deployment described in the next section.

5 Deployment and Evaluation

In order to show the functionality of the *DySSCo* algorithm, we implemented the algorithm on the *pacemate* [6] sensor node platform. We placed 20 *pacemates* in the corridors of our institute as shown in Figure 3. The average neighborhood size was 7. The beacon interval t was set to $t = 3$ seconds. Neighbors were deleted

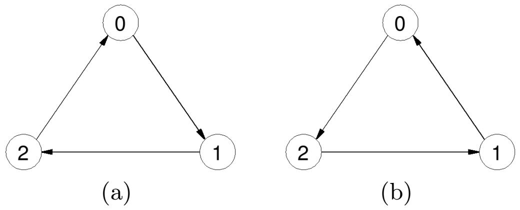
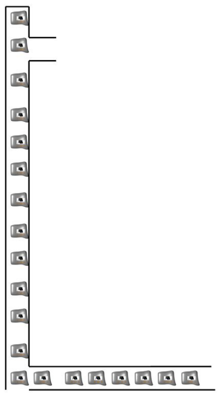


Fig. 2. Graphs with no stable solution for a coverage of 50%

(a) *pacemates* on the corridor



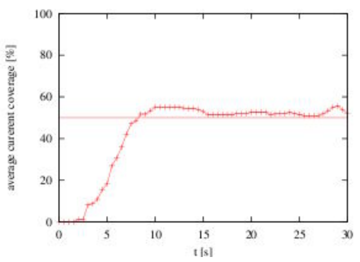
(b) Schematical placement of the nodes

Fig. 3. Deployment of sensor nodes in the corridor

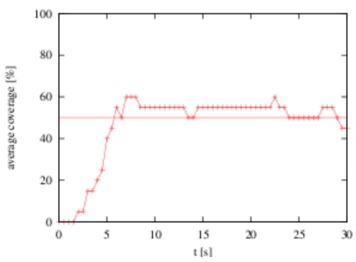
from the neighborhood list of a node after not hearing a beacon for 12 seconds. The maximal backoff time was set to *backoff* $= 200\text{ms} * (N(n) + 1)$. With the push of a button, we activated a service with a demanded coverage $c_{demanded}$ of 50 % on one node. This initializes the hop-by-hop activation according to the *DySSCo* algorithm. To get a visual feedback of its function, this service turns on the LED on the *pacemate*. The *pacemates* keep log about the size of their neighborhood, the observed coverage, their service status and the transmitted messages.

We evaluate our service distribution algorithm on basis of the recorded data including coverage, service status and transmitted messages. Besides average current local coverage of nodes and the global coverage of the service within the network, we analyze the average neighborhood size and the average amount of transmitted messages. We choose to measure the first 30 s since activation of the service.

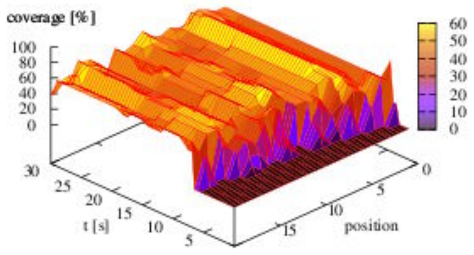
Figure 4(a) shows the average current coverage $c_{current}(s)$ of service from node's point of view. The average coverage rises up to around 55 % in the



(a) Average current coverage of service



(b) Global coverage of service



(c) Coverage over time and position of nodes

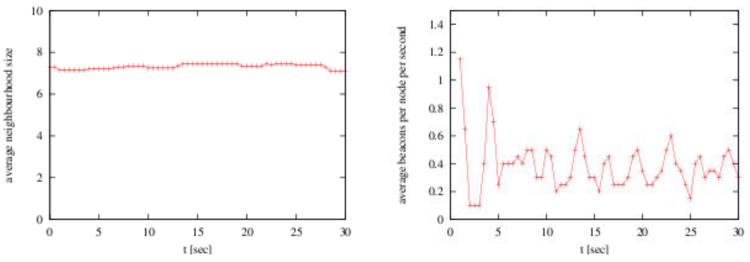
Fig. 4. Results for deployment with 20 *pacemates*

beginning when the service is distributed on the nodes along the corridor. The network finds a stable state and reaches a value slightly over the globally demanded service coverage of 50 % afterwards.

The global coverage represents the actual percentage of nodes running a service s in the whole network. If we compare the current coverage seen by the nodes with the global coverage presented in Figure 4(b), we observe correlations between both. The global coverage also rises up in the first few seconds, underlies variations afterwards and stabilizes in the forthcoming seconds by reaching a value slightly above the demanded coverage of 50%. Due to the relatively small number of nodes (20), changes in global coverage appear cascaded rising by 5% if a node activates the service and falling by 5% if the service is deactivated. Accordingly, 12 activated nodes make up 60% global coverage at a total node count of 20.

Figure 4(c) presents the current service coverage over time ordered by position of the nodes. Here, we can identify the dissemination of the service throughout the network from node 1 to node 20 in the first 10 seconds, observing the global effect of the local injection at node 1.

The variation in the coverage over the time can be explained by changes in the neighborhood size of the nodes as depicted in Figure 5(a). A reduction of neighborhood size increases the current coverage especially in small neighborhood sizes given that one node has greater influence on the coverage. Due to following



(a) Average neighborhood size (b) Average messages per node

Fig. 5. Results for deployment with 20 *pacemates*

reactions of nodes by deactivating services the global coverage decreases. These changes in neighborhood result from disturbances in the radio channel caused by environmental influences such as movement of persons on the corridor. The average neighborhood size stays almost constant between 6 and 8 nodes.

Figure 5(b) shows a peak in the average number of sent messages as a result of the initial activation of the service on the first node. Subsequently the average amount of messages per node commutes around 0.4 messages per second per node almost corresponding to the chosen interval t of the beacon sent every 3 seconds. Additionally, Figure 5(b) shows periodical fluctuations due to synchronisation which was necessary for our measurement process.

Our evaluation shows that the *DySSCo* algorithm reaches the demanded coverage of approximately 50% in the network scenario. Despite temporally variations in neighborhood size the algorithm preserves the demanded coverage. Additionally, the number of transmitted messages does not increase.

6 Conclusion and Future Work

The controlled replication of functionalities in a sensor network is an important prerequisite for implementing self-organizing applications: For a lot of applications it is sufficient that a certain functionality is available somewhere in the network. If first instances of this service fail it is important to have other instances that can replace the first ones. Additionally, it might be valuable if there is an instance near the caller in order to keep the communication costs low. Hence, it is important to investigate reliable strategies for distributing services in a sensor network.

A service might be already present on the node in form of a piece of software waiting for being activated, or might physically migrate onto this node during runtime. In both cases it is crucial to control the distribution process in terms of the desired service coverage. If the coverage is higher than required, resources in the sensor network are wasted, and if it is too low, the service load gets too high for the remaining instances.

Unfortunately, it is not sufficient to distribute the services in a static manner: Due to hardware failures, new user requirements, network fragmentation or mobility it might be necessary to activate new service instances while the sensor network is already in operation. Hence, activating services in a dynamic fashion during runtime requires self-organization itself.

In this paper, we presented a novel distributed algorithm called *DySSCo* . It is based on simple local rules, that achieve a uniform distribution of services in the sensor network. Our algorithm realizes a dynamic adaptation and optimization of the service distribution during runtime.

A theoretical analysis of all existing graphs with 3, 4, and 5 node showed, that the algorithm finds a stable solution in more than 94 % of all existing graphs with these sizes. Note that the remaining, unsolvable graphs include cases that are very unlikely in reality (high amount of unidirectional links).

Furthermore, we conducted a real-world experiment with 20 nodes. The results show the correct operation and practical fitness of our algorithm. We could show that our algorithms finds a stable solution after a reasonable amount of time. The remaining fluctuations date back to changing node neighborhoods during runtime caused by non-ideal radio characteristics.

We are currently investigating improvements of our protocol and the algorithm itself in order to achieve stable solutions for all graph topologies. Future work will include further enhancements in order to achieve a fair distribution among all service types and all nodes in the network. Additionally, we will improve the implementation of our algorithm by adding a feature that allows for changing the demanded coverage of a certain service type during runtime.

Further theoretical work will include a formal proof of convergence and stability for the service distributions provided by our algorithm. Additionally, we will try to prove that the desired global coverage is reached in all cases.

Another ongoing research work focuses on dynamic linkage of machine code in order to replicate and distribute parts of a programm during runtime. The *DySSCo* algorithm will be used for keeping control of the replicated numbers of a certain program part (i.e. service).

Although the correct operation of our algorithms has not been proven for all possible network topologies yet, we are very confident that it is already useful for a lot of practical applications. In order to cope with "exotic" network topologies, the algorithm must be extended. All in all we are of the opinion that the *DySSCo* algorithms is a very promising solution enabling self-organizing service distribution in sensor networks or similar systems. The current results show that it is already working and suitable for practical usage.

References

1. Blumenthal, J., Handy, M., Golatowski, F., Haase, M., Timmermann, D.: Wireless sensor networks - new challenges in software engineering. In: Emerging Technologies and Factory Automation, Proceedings. ETFA 2003. IEEE Conference, vol. 1 (2003)
2. Boulis, A., Han, C.-C., Srivastava, M.B.: Design and implementation of a framework for efficient and programmable sensor networks. In: MobiSys 2003: Proceedings of the 1st international conference on Mobile systems, applications and services, pp. 187–200. ACM, New York (2003)

3. Fok, C.-L., Roman, G.-C., Lu, C.: Rapid development and flexible deployment of adaptive wireless sensor network applications. In: International Conference on Distributed Computing Systems, vol. 00, pp. 653–662 (2005)
4. Levis, P., Patel, N., Culler, D., Shenker, S.: Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In: First Symposium on Network Systems Design and Implementation, NSDI (2004)
5. Levis, P., Culler, D.: Maté: a tiny virtual machine for sensor networks. SIGOPS Oper. Syst. Rev. 36(5), 85–95 (2002)
6. Lipphardt, M., Hellbrueck, H., Pfisterer, D., Ransom, S., Fischer, S.: Practical experiences on mobile inter-body-area-networking. In: Proceedings of the Second International Conference on Body Area Networks, BodyNets 2007 (2007)
7. Koutsoukos, X., Neema, S., Kushwaha, M., Amundson, I., Sztipanovits, J.: Oasis: A programming framework for service-oriented sensor networks. In: IEEE/Create-Net COMSWARE 2007 (January 2007)
8. Marin-Perianu, R., Scholten, H., Havinga, P.: Prototyping service discovery and usage in wireless sensor networks. In: Conference on Local Computer Networks (LCN), vol. 0, pp. 841–850 (2007)
9. Marron, P.J., Lachenmann, A., Minder, D., Hahner, J., Sauter, R., Rothermel, K.: Tinycubus: A flexible and adaptive framework for sensor networks. In: Proceeedings of the Second European Workshop on Wireless Sensor Networks, pp. 278–289 (2005)
10. Wittenburg, G., Schiller, J.: A survey of current directions in service placement in mobile ad-hoc networks. In: IEEE International Conference on Pervasive Computing and Communications, vol. 0, pp. 548–553 (2008)

An Approach to Autonomic Deployment Decision Making

Rico Kusber, Sandra Haseloff, and Klaus David

University of Kassel, Chair for Communication Technology (ComTec),
Wilhelmshöher Allee 73, 34121 Kassel, Germany
{rico.kusber,sandra.haseloff,
klaus.david}@comtec.eecs.uni-kassel.de

Abstract. Adding autonomicity to computing systems seems to be a promising way to deal with the problem of increasing system complexity. One step along the way to self-managing computing systems – especially with regard to distributed, modularized, service based environments – is to solve the problem of how to autonomically decide in a most useful and resource efficient way which alternative to choose in order to deploy a service. Deploying a service means, to either copy or move it from a source to a destination device or to use it remotely. In this paper we motivate the domain of autonomic service deployment and present an approach for deployment decision making (DDM). We explain all steps of the deployment decision making process and assemble them into an algorithm accordingly. Furthermore, we define all necessary components of DDM and enumerate a set of research questions which we address in order to fully explore the concerned domain. An experiment illustrates the potential of the presented approach.

Keywords: Autonomic computing, autonomic communication, service deployment, software deployment, deployment decision making.

1 Introduction or What Is Deployment Decision Making

Self-management abilities of computing and communication systems, as described within the IBM autonomic computing initiative (cf. [4]), have considerably advanced. Among other things, one of these self-management abilities is having the capability of obtaining services which a system requires to fulfill its tasks. Such services can either be directly dedicated to the system's user (e.g. a recording of a concert in form of music files) or to the system itself (e.g. software updates or a formal XML specification of additional data types). To be able to use those services they need to be deployed which means either being copied or moved from a source to the destination device or being accessed remotely. Deploying such additional – possibly only temporarily needed – services, requires that an autonomic computing system discovers suitable alternatives where to get a desired service from. Once this is done, it has to select one out of all discovered alternatives in a self-dependent, most useful manner. The process of selecting an appropriate and most useful alternative is what we call *deployment decision making (DDM).*

A number of steps have to be performed to come to a deployment decision. After discovering potential alternatives, a set of parameters used to describe available services needs to be specified. Based on that, an evaluation of each alternative's usefulness has to take place. The result of this evaluation then determines one alternative as the winner of the DDM process. Experiences made during the deployment of the winning service help to adapt the DDM system and improve deployment decision making in general. In this paper we present our approach to making deployment decisions and identify all necessary steps. We propose how to describe services for DDM, and we explain how to tackle the calculation of an alternative's usefulness. The remainder of this paper is structured as follows: Section two gives an overview of groundwork and related research fields. Section three presents our approach for DDM. In section four we describe a scenario, our DDM implementation, and evaluations we have done on that. Conclusions and an outlook are given in section five.

2 Groundwork

Since IBM's autonomic computing initiative (cf. [5], [4]) was launched in order to research self-* abilities of computing systems (cf. [6]), various advances in that field have been made. Scientists have exploited a variety of related research fields. Fusion with different scientific areas, like e.g. artificial intelligence, leads to interdisciplinary attention to autonomic computing and communication.

What is addressed when talking about autonomic computing is not completely different from, but rather more specialized than what is envisaged within the scope of Artificial Intelligence (AI) in general. Paradigms, methods, and results of AI research are applied to the domain of autonomic computing and are utilized to achieve self-* behavior. Examples therefore are agent technologies as a representation of software, defining policies in order to make decisions, or including learning algorithms for optimization purposes (cf. [7]). Concerning the deployment of services, which can be seen as the domain for deployment decision making, a number of ideas and approaches has been developed until now (cf. [1], [3]). All of them are advantageous in dedicated environments and under defined circumstances. Nevertheless, points that have not sufficiently been addressed so far are how to autonomically and efficiently come to deployment decisions and how to optimize the decision making process. A set of service discovery protocols and mechanisms are also available (cf. [8]). Each of them has special properties depending on the kind of the concerned services and the environment in which these services are embedded. Quality constraints of utilized services can be determined with the help of service level agreements in order to ensure stable marginal conditions (cf. [9]). Frameworks to describe services, their properties, and their demands are being under development and are already standardized to a certain extend (cf. [10]).

One open issue is the facility to optimize the deployment of services over time and depending on user preferences. The selection of a suitable set of service level agreements is not enough. Also the possibility to incorporate formerly made experiences concerning e.g. the reliability of uncertain service descriptions is necessary to come to satisfying deployment decisions and to optimize the resource consumption during the deployment process and the operation of the deployed services. By defining and investigating a process for deployment decision making we hope to step towards the direction of equipping computer systems with the ability to autonomically obtain services in a most useful and economical way.

3 The DDM Approach

Above, we described what deployment decision making means and what it is needed for. We now outline the approach we have developed to tackle the DDM problem. Before describing step by step how we envisage performing deployment decision making, a set of terms has to be clarified.

In the context of DDM, a *service* is a piece of software that fulfils a certain task. This includes not only executable programs but e-books, sound files, and other things as well. Such a service resides on a *device* which can be a server, a cell phone, an embedded micro controller or anything similar. *Deploying* a service to a target device d_t means copying or moving it from any source d_s to d_t, or using the service remotely. If a desired service can be obtained from either multiple devices or from one device under different conditions we call each of these possibilities to obtain the service a *deployment alternative*. Semantically speaking, it is of maximal benefit to select the alternative for deployment which is most useful for the selecting system or a human user at the end. Following that, we call the benefit of deploying an alternative the alternative's *usefulness*. To decide upon an alternative's usefulness each alternative needs to be described by a set of *service parameters*. The system that performs deployment decision making utilizes a set of *decision parameters* which are compared to the service parameters of each alternative, and selects the one which seems to be most useful. The set of utilized decision parameters is not fixed but rather can change for each decision process.

3.1 Scenario

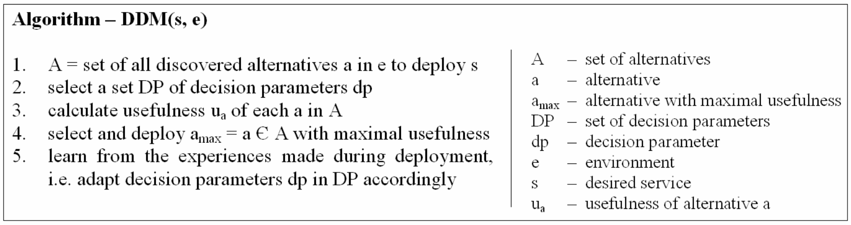
To motivate our deployment decision making approach we have designed the following scenario. Imagine a music management system that has the task of autonomously searching for music files. The system, thereby, acts according to a set of preferences defined by its user, e.g. a monthly money budget, preferred file formats, quality demands and so on. Once the system has determined a song that probably fits the users taste in music, possible alternatives from where to get the file from will be discovered and evaluated, and then a decision which alternative to deploy will be made. At last, the music file will be obtained from the selected source and stored on the target device.

We implemented an environment which is capable of simulating this scenario. What we have so far is a possibility to define a scenario in form of an XML description, to load this scenario accordingly, and to perform DDM on it. Thereafter, the scenario as well as the simulation results can be translated back to XML. A scenario consists of a number of rounds where each round is a repetition of the DDM simulation. In turn, one round comprises all source as well as target devices. Devices hold a set of services which are associated with a set of service parameters. Beside these entities, a set of decision parameters is defined in each round which is utilized to assess all available deployment alternatives. What needs to be defined as well is the desired service, i.e. the service that should be deployed on the target device.

Based on this scenario design, experiments can be performed in two ways. First, in each round all variables can have exactly the same values. This procedure enables to gain statistical information about the simulated scenario. Second, variables can be modified from round to round which allows the investigation of a variable's influence on an experiment. Evaluations we have performed with the help of the described scenario are presented in section four.

3.2 Algorithm

In order to address the process of service deployment decision making in an efficient manner, we have developed an algorithm that includes all steps needed for deployment decision making. This algorithm is a straight forward implementation of the DDM process described above. We do, at this point in time, not claim the algorithm to be optimal in terms of computational complexity, but rather use it to elaborate on what is necessary to perform deployment decision making. Fig. 1 describes the DDM algorithm.



Algorithm – DDM(s, e)

```
1. A = set of all discovered alternatives a in e to deploy s
2. select a set DP of decision parameters dp
3. calculate usefulness $u_a$ of each a in A
4. select and deploy $a_{max}$ = a ∈ A with maximal usefulness
5. learn from the experiences made during deployment,
   i.e. adapt decision parameters dp in DP accordingly
```

A	– set of alternatives
a	– alternative
a_{max}	– alternative with maximal usefulness
DP	– set of decision parameters
dp	– decision parameter
e	– environment
s	– desired service
u_a	– usefulness of alternative a

Fig. 1. Algorithm for deployment decision making (DDM)

Below we describe how we have realized the single steps of the DDM algorithm.

- *Step 1*, service discovery, is the point from where we start deployment decision making. At this time we have available a set of alternatives to deploy a desired service. These alternatives have been determined by a service discovery mechanism which we see as a prerequisite, but not as part, of DDM itself. At the moment, a variety of service discovery protocols exist where each of them fulfils a set of needs in a dedicated environment (cf. [8]). Which of these protocols is chosen, or which of them can be combined, is a question that we do not address in detail when investigating deployment decision making. To be as general as possible, we assume instead that a sufficient service discovery method exists. That means we assume that a list of discovered deployment alternatives is available as the result after executing step 1 of the DDM algorithm.
- *Step 2*, selecting a set of decision parameters, is one major issue for deployment decision making. For that reason, we set a focus here for our research work. We currently include all available decision parameters into the decision process.

Available parameters are those which we have designed to evaluate the scenario described in section 3.1. In our approach, we have defined a decision parameter as a 4-tuple of its name, value, interpreted value, and weight.

$$dp = (n, v, iv, w)\ . \tag{1}$$

The name is the parameter's identifier. The value can be any, symbolic or sub symbolic, information. Arbitrary information with unspecified sense cannot be processed in a meaningful way, at least not by the DDM algorithm. To cope with this problem we have introduced interpreters, one for each decision parameter. These Interpreters express how useful a value of a parameter is for an individual user of the DDM system. They are exchangeable and can be defined by any user according to any personal preferences. Interpreters map the decision parameter value to a usefulness scale which is defined as follows:

The usefulness scale S_N is a measure for assessing the value v of a parameter p in a deployment decision making system.

- $S_N = [-1,1]$, $iv_p(v_p) \in S_N$ with
- $0 < iv_p <= 1$, the parameter p has the usefulness iv_p for the associated deployment alternative
- $iv_p = 0$, the parameter p has no relevance for assessing the usefulness of the associated deployment alternative and will not be regarded in the deployment decision making process
- $iv_p < 0$, the associated deployment alternative is not suitable for deployment because of the value v_p of the parameter p

The weight w of a decision parameter dp is a factor that expresses the parameter's importance for the user of a DDM system. In our current implementation these weights are used as factors when calculating the overall usefulness of a deployment alternative. How w can be modified and if it is necessary at all, needs to be further investigated. This depends to a large extent on the method used to calculate the usefulness of a deployment alternative. Moreover, when we think about a non-toy application of DDM, there needs to be a significantly larger number of decision parameters than we have regarded so far. Involving them all in the decision process may cause the efficiency of the algorithm to diminish. In general, we have to investigate how parameters can look like in detail and what effects are implied by their design. Based on that, an efficient selection method for decision parameters can be developed.

- *Step 3*, calculating the usefulness u_a of each alternative $a \in e$, is an operation that combines all interpreted values of each parameter of a deployment alternative. In our approach we have applied a simple summation of each interpreted value iv_p and multiplication with the weight w_p accordingly:

$$\forall a \in e : u_a = \sum_{p \in DP} iv_p \cdot w_p\ . \tag{2}$$

In general, the calculation of the usefulness of an alternative can be any other operation as well. By using a weighted summation, we have applied a well-known method that has already been shown to be advantageous for similar tasks in other fields like in integration functions of artificial neural networks (cf. [11]) or in affinity functions of artificial immune systems (cf. [2]). We now have to evaluate to what extent it is applicable to the domain of deployment decision making as well, or if any other method is more suitable for the calculation of the usefulness of deployment alternatives.

- *Step 4*, select and deploy the alternative with maximal usefulness, is the last action that needs to be performed before a desired service is available on a target device. Selecting the most suitable alternative is the consequence of the usefulness calculation that took place in step 3 of the DDM algorithm. In our approach we select the alternative with maximal usefulness as the winner of the deployment decision making process. Making a deployment decision without deploying a service would of course not be satisfying in the end. Anyway, we do not address deployment itself. How to install, configure and integrate a service into a computing system is not within the scope of our research. Depending on the kind of service and the environment into which it has to be deployed, there are approaches already concerned with this problem (cf. [1], [3]). Instead, we take deployment as the final step after DDM for granted and assume that, once an alternative is selected, it can be deployed to the target system.
- *Step 5*, learning from the experiences made during deployment, is a major issue when talking about autonomic computing and communication systems. It enables the adaptation to changes in the environment, and so, avoids the need for user intervention in various situations. Here we see extensive research potential which we exploit in our further work. A number of questions arise when we are talking about learning from experiences, e.g.:

  - What experiences can be used to improve the DDM process? For example, information about the success of a deployment process can indicate the reliability of a service source.
  - How can we cope with unreliable information concerning service parameters? For example, we can measure the discrepancy between the denoted and the experienced parameter values. A low discrepancy increases the reputation of the service source, a high one decreases it.
  - Which algorithms or paradigms can be applied from the field of machine learning? For example, artificial immune systems can turn out to be suitable for detecting deceptions in terms of denoted parameter values.

In order to better understand the domain of deployment decision making, we address all these points with the intent to improve the performance of DDM. Fig. 2 gives an overview of issues that we tackled within that scope. In this paper we concentrate especially on describing how we put into practice the parameter definition, the usefulness calculation, and the environment which we have developed for simulation and evaluation.

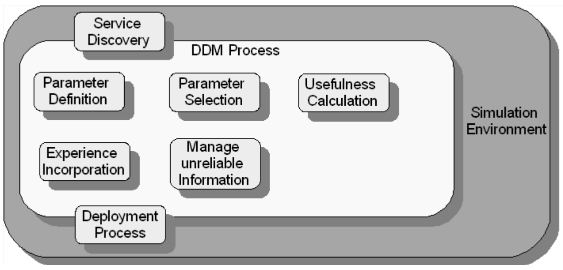


Fig. 2. Relevant issues in deployment decision making

4 Implementation and Evaluation

For evaluation purposes, we developed a simulation environment that enables us to investigate the DDM algorithm. This environment is a practical tool to load and store scenarios, perform deployment decision making on these scenarios, design and choose service and decision parameters, as well as to repeat simulations for an arbitrary number of rounds. For our future work, we will continuously enhance this simulation environment in order to adapt it to requirements that emerge during our research.

To justify the relevance of a deployment decision making mechanism, we make two simple assertions.

Assertion 1

Applying DDM can save resources and can therefore optimize the process of deploying services.

Assertion 2

DDM selects an alternative with maximal usefulness according to a set of user preferences.

Fig. 3 represents the XML structure of the scenario described in section 3.1. We have utilized this scenario in order to investigate assertions 1 and 2 and to analyze the behavior of our DDM algorithm. In particular, we simulated an environment with the following settings.

- Desired service is John Lennon's "Imagine" as a music file
- One target device and three source devices providing the desired service
- Decision parameters are

  1. the price to obtain the music file with usefulness

$$u_p = \left\{ \begin{matrix} 1 \text{ for price } p = 0 \text{ Euro} \\ \max\left(0, \min\left(1, \frac{1}{p}\right)\right) \text{Euro for } p \neq 0 \end{matrix} \right\} . \tag{3}$$

2. the bit rate b used to digitalize the music, with 192 kBit/s being the optimal bit rate, usefulness

$$u_b = \min\left(1, \max\left(0, \sin\left(\frac{\pi \cdot (b - 64)}{320 - 64}\right)\right)\right) \cdot \tag{4}$$

As described in section 3.2, the usefulness of decision parameters can and should individually be adapted to any user's preferences. Equations (3) and (4) express our personal preferences we assumed within our evaluations.

- Ten rounds, where each of them has random values for all parameters of all services on all devices, all other settings are equal in each round

The above mentioned decision parameters and deployment alternatives are exemplarily chosen for this experiment. When considering real world scenarios, there needs to be a significantly larger number of parameters to be regarded in order to properly describe different deployment alternatives. Furthermore, it is important to

```
- <scenario>
    <scenarioName>DDM02 - Scenario003 - Music Download</scenarioName>
  - <round number="1">
    - <devices>
      + <device name="myDevice" target="true"></device>
      - <device name="www.funnymusic.example">
        - <services>
          - <service name="John Lennon - Imagine">
            + <parameter name="decisionparameters.MusicFilePrice"></parameter>
            - <parameter name="decisionparameters.MusicFileBitrate">
                <parameterValue>256</parameterValue>
                <parameterInterpretedValue/>
              </parameter>
              <usability/>
            </service>
          </services>
        </device>
      + <device name="www.musicmusicmusic.example"></device>
      + <device name="www.soundland.example"></device>
      </devices>
    - <decisionParameters>
      + <parameter name="decisionparameters.MusicFilePrice"></parameter>
      + <parameter name="decisionparameters.MusicFileBitrate"></parameter>
      </decisionParameters>
      <desiredService>John Lennon - Imagine</desiredService>
    </round>
  + <round number="2"></round>
  + <round number="3"></round>
  + <round number="4"></round>
  + <round number="5"></round>
  + <round number="6"></round>
  + <round number="7"></round>
  + <round number="8"></round>
  + <round number="9"></round>
  + <round number="10"></round>
  </scenario>
```

Fig. 3. XML structure of DDM simulation scenarios

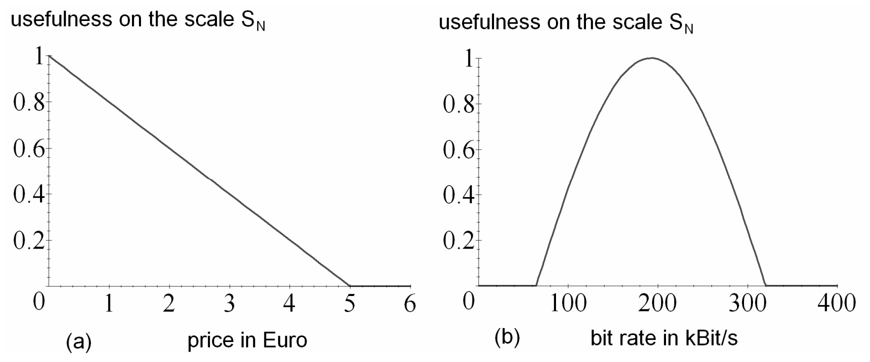


Fig. 4. Usefulness of the decision parameter (a) *price* and (b) *bit rate*

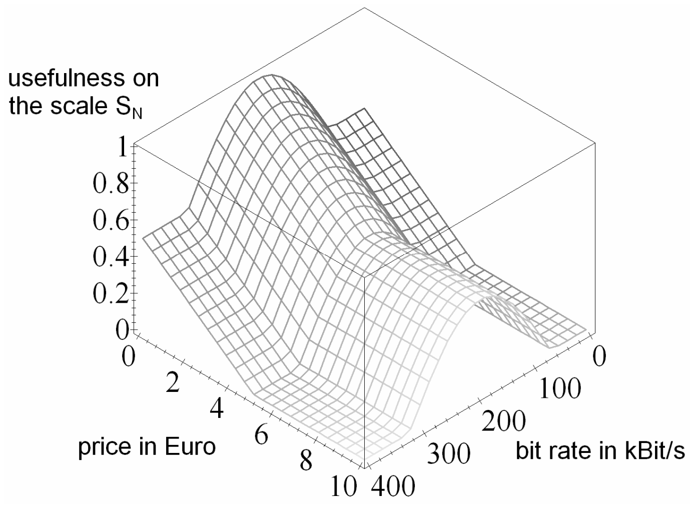


Fig. 5. Overall usefulness of the decision parameters *price* and *bit rate*

recall that service deployment is not restricted to downloading a piece of software. Beside the three alternatives to obtain the desired service we regarded in our example scenario, an additional alternative can e.g. be utilizing the service remotely by accessing a streaming server. In that case, experienced reliability to the source server in terms of packet loss, steadiness of connection speed, or the service access method itself (i.e. download or stream) can be other interesting decision parameters.

Fig. 4(a) shows how the value of the parameter price is interpreted in terms of usefulness u_p. An alternative with a lower price is more useful than an alternative with a higher price. How the bit rate is interpreted is shown in Fig. 4(b). The highest usefulness u_b for the parameter bit rate is reached at 192 kBit/s. Higher or lower coded music files are less useful. A combination of both decision parameters, price and bit rate, is presented in Fig. 5. The overall usefulness u of a deployment alternative is calculated as a mean value by applying the equation

$$u = \frac{u_p + u_b}{2} \ . \tag{5}$$

The simulation of the scenario described above has revealed no secret but rather confirmed the expectations stated in Assertion 1 and Assertion 2. Applying the DDM mechanism leads to always deciding for the deployment alternative with the highest overall usefulness according to the set of utilized decision parameters.

Fig. 6 displays all discovered deployment alternatives according to their usefulness over a simulation period of ten rounds. The alternatives represented by black bars are the ones which were selected by the DDM algorithm. Those are always the ones with maximal usefulness during the corresponding round. Grey and white bars represent all other, not selected alternatives. We repeated the experiment using 20 different deployment alternatives over a period of 50 rounds with random values for all service parameters. All other parameters remained unchanged. Fig. 7 shows that the usefulness of the DDM selected alternative is always greater than the minimal and mean usefulness of all alternatives per round. For this experiment we have calculated a mean usefulness of all alternatives over all 50 rounds u_{mean} = 0,57 on the scale S_N and a mean usefulness of all alternatives selected by DDM over all 50 rounds $u_{selected}$ = 0,92 on the scale S_N.

These results are not surprising because the algorithm is designed accordingly. What is as interesting as simple, is the consequence: Applying the deployment decision making mechanism in the described scenario can, firstly, save money and, secondly, save bandwidth, i.e. computational power each time the deployed music file is played on the target device (cf. Assertion 1), always being in line with the user's preferences represented by the defined decision parameters (cf. Assertion 2). Following that, it is clear that the DDM process proposed above has the potential to optimize the resources consumption of autonomic computing systems and satisfy user's demands.

Based on the results of the experiments described above, we will continue to investigate the DDM process in line with the research questions we have elaborated. The developed simulation environment will therefore be further enhanced, adapted, and used. Incorporating for example, a mechanism to group, manipulate, and select a set of decision parameters per scenario and round will enable finding out how the parameter selection influences the deployment decision. An advanced interface will ease the control and adjustment of all variables that affect the DDM process. Developing a

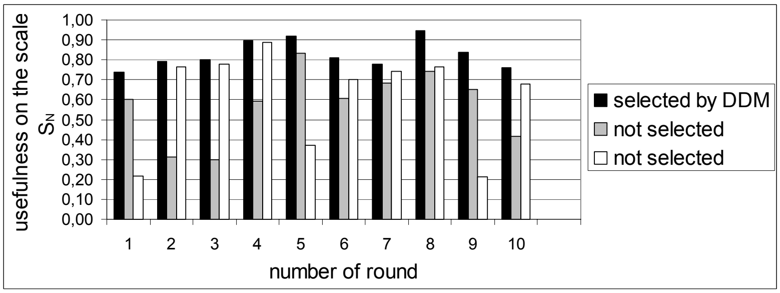


Fig. 6. All deployment alternatives of ten rounds

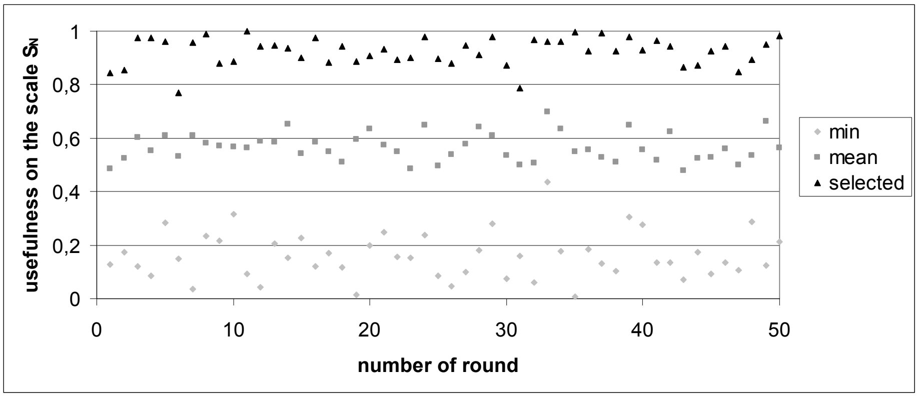


Fig. 7. Minimum and mean usefulness as well as usefulness of the selected alternative per round over a simulation period of 50 rounds

feedback mechanism will allow us to investigate how step 5 of the DDM algorithm, learning from experience, can be addressed and explored. With the described simulation environment we have a practical tool which we adapt and extend to the needs that arise during the investigation of deployment decision making.

5 Conclusions and Outlook

In this paper we have presented an approach to equip autonomic computing systems with the ability to decide upon a number of different alternatives when a service needs to be deployed. We have presented an algorithm that includes the steps of the deployment decision making process and we have defined the elements we need to apply this algorithm. In order to test and validate the described mechanism, the assertion that DDM can save resources, and the assertion that DDM selects an alternative that fits users' needs, we have developed a music download scenario were one out of several alternatives has to be selected in order to obtain a music file. Beside the algorithm and the scenario, we have realized a simulation environment which we have utilized to evaluate the DDM approach. It was shown that applying DDM always led to equal or higher usefulness, compared to simply selecting the first discovered deployment alternative.

We have analyzed the steps of the DDM process and identified a set of research questions which we address in our further work. We investigate how to define and select decision parameters, how the usefulness of deployment alternatives properly can be expressed, how we can improve the DDM process with the help of collected experiences, and how to deal with unreliable information concerning service parameters.

The insight into the DDM domain we have gained so far and the questions that arose are motivating to continue our research. Investigating thoroughly and obtaining a deep understanding of the mechanism for making deployment decisions is a necessary step into the direction of putting autonomy into computing systems.

Acknowledgements. The authors would like to acknowledge the European Commission for funding the Integrated Project CASCADAS "Component-ware for Autonomic, Situation-aware Communications, And Dynamically Adaptable Services" (FET Proactive Initiative, IST-2004-2.3.4 Situated and Autonomic Communications) within the 6th IST Framework Program. The authors also wish to thank other project Partners for the fruitful cooperation within the project.

References

1. Carzaniga, A., et al.: A Characterization Framework for Software Deployment Technologies. Technical report, Department of Computer Science, University of Colorado (1998)
2. De Castro, L.N., Timmis, J.: Artificial Immune Systems - A New Computational Intelligence Approach. Springer, London (2002)
3. Hillenbrand, M., Müller, P., Mihajloski, K.: A Software Deployment Service for Autonomous Computing Environments. In: Proceedings of the International Conference on Intelligent Agents, Web Technology and Internet Commerce, Gold Coast (2004)
4. Horn, P.: Autonomic Computing – IBM's Perspective on the State of Information Technology. International Business Machines Corporation, Armonk (2001)
5. Kephart, J.O.: Research Challenges of Autonomic Computing. In: Proceedings of the 27th International Conference on Software Engineering, St. Louis, pp. 15–22 (2005)
6. Kephart, J.O., Chess, D.M.: The Vision of Autonomic Computing. IEEE Computer 36(1), 41–50 (2003)
7. Kephart, J.O., Walsh, W.E.: An Artificial Intelligence Perspective on Autonomic Computing Policies. In: Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks, Yorktown Heights, pp. 3–12 (2004)
8. Lee, C., Helal, S.: Protocols for service discovery in dynamic and mobile networks. Journal of Computer Research 11(1), 1–12 (2002)
9. Massam, P.: Managing Service Level Quality – Across Wireless and Fixed Networks. Wiley Europe Ldt., Chichester (2003)
10. Oaks, P., ter Hofstede, A.H.M., Edmond, D.: Capabilities – Describing What Services Can Do. In: Proceedings of the First International Conference on Service Oriented Computing, Trento, pp. 1–16 (2003)
11. Rojas, R.: Neural Networks: A Systematic Introduction. Springer, Berlin (1996)

Self-Organizing Multirobot Exploration through Counter-Ant Algorithm

Ilhem Kallel[1,2], Abdelhak Chatty[1,2], and Adel M. Alimi[1]

[1] REGIM, Research Group on Intelligent Machines,
National School of Engineers,
University of Sfax, BP W 3038, Tunisia
[2] High Institute of Computer Science and Management,
University of Kairouan, Tunisia
{ilhem.kallel,abdelhak_chatty,adel.alimi}@ieee.org

Abstract. This paper presents an evolving method for a self-organizing multirobot exploration of an unknown environment. In such problem, a big consideration is given to the coordination behavior of robots in order to achieve the common tasks in an optimal way. Ant algorithms are proved to be very useful in solving such distributed control problems. We present here a modified version of the known ant algorithm, called Counter-Ant Algorithm (CAA). Indeed, the robots'collective behavior is based on repulsion instead of attraction to pheromone, which is a chemical matter open to evaporation and representing the core of ants' cooperation. A series of experimentations with MINDSTORMS LEGO robots, and simulations under Madkit platform, in laboratory conditions similar to real ones, show the usefulness of our algorithm for self-organizing and cooperative exploration.

Keywords: Counter-Ant algorithm, self-organizing multirobot, cooperative exploration, pheromone, stagnation recovery, Lego robots, Madkit.

1 Introduction

Multirobot systems affects our society in a fundamental way; to promote development in this field, several approaches in robotics'world have emerged and they focus on different aspects such as models of cooperation/collaboration, detection, location and exploration. However, these methods suffer from lack of robustness related to coordination between robots, resulting negative impacts on the convergence time of the operating multirobot system. To overcome these challenges, the attention of researchers has been focused on the wild world which unveiled a self-organization even more complex and harder which proves the rate of intelligence among other species. The operating results led to a new field called swarm intelligence, covering ants' algorithms. These algorithms provide powerful methods for the design of algorithms and optimization of distributed problems involving a collaborative swarm behavior [2] [4] [5] [19]. The intrusion of these algorithms in the world of robotic improve the communication quality between robots but the convergence time remains problematic [1] [6] [7] [11] [12] [13] [18] [20][21]. Can we go by ants algorithms in order to achieve a more efficient multi-robot collaboration, while minimizing the system convergence time and realizing a best trade-off between optimization and goal to reach? We present in this paper a modified version of the known ant algorithm, called Counter-Ant Algorithm (CAA). Indeed, the robots' collaborative behavior is based on repulsion instead of attraction to pheromone, which is a chemical matter open to evaporation and representing the core of ants' cooperation. The robots' reaction consists henceforth in avoiding paths more covered by this chemical substance. In order to test the performance of our CAA, we had to look firstly for a suitable assembly of Mindstorms Lego robots to be cooperative explorers and pickers [14] [15] and an appropriate representation of pheromone [16]; secondly, for a suitable simulation platform that can represent both swarm behavior and cooperative robot behavior [3] [8] [10].

The paper is organized as follows: section 2 overviews some collaborative swarm robots based on ant algorithms. Sections 3 and 4 present our proposal for a CAA in multirobot collaboration while providing solution to stagnation situation. Section 5 and 6 present and discusse some experimentation and similation results. Finally, last section concludes the paper.

2 Cooperative Swarm Robots Based on Ant-Algorithm

Among swarm intelligence techniques, some are mature. In fact, the observation of ants' colony led to ants algorithms [5] [6]. These revolve around a key concept called stigmergy which is an indirect activities' coordination in an unknown environment: Ants are attracted to each other thanks to chemical material called pheromone. It transpires that in an unknown environment, one of the most important problem related to multirobot systems, is to decide how to coordinate actions in order to achieve tasks in an optimal way [11]. Also, it is very important to know what needs to be accomplished and what is the number of robots required for the task. Also, among the multirobot systems, we note the emergence of self-organizing cooperative behavior. To give solution to self-organization, researchers resort to stigmergy. For example, when a robot concludes that it can not conclude a mission, it marks the unfinished task with a quantity of pheromone. More difficult the task is, the greater the amount of pheromone increases. Thus, robots will be attracted to try to accomplish this task [20]. It is clear that box pushing is more natural if it is accomplished with ants' multirobot system: In fact, the ant tries to move the food alone. it spent moments to test the food resistance by varying the orientation of its body. Changing the direction of the applied strength can be enough to really move the cargo. Should the realignment would not be sufficient, the ant frees prey and find another position to seize the cargo. If multiple replacement attempts are not successful, the ant recruits other ants [11]. Many works such as cited by Kube and Zhang [12][13] are based on ant algorithm to solve transportation problem. Other research consists to change the ant algorithm using a new technique for multirobot pheromone placement. This technique enables robots to place more pheromones in the tasks that are about to be fulfilled: A very large amount of pheromone is placed in the balls that are almost being moved or are nearly to their destinations, while a small amount of pheromone is placed in the balls, which require a great force to make them move. This helps to attract other robots to complete the task [20]. Except that the amount of pheromone is constant, whereas in [1] [21] is variable and depends on task difficulty and the robots strength. The problem of task allocation in the field of cooperative robotics was led by several studies especially in unknown environment. Each robot has to adapt to its environment without any training stage. Accordingly, swarm intelligence allows self-organization into an unfamiliar environment and adapting behaviors through simple individuals' interactions.

3 Multirobot Exploration Based on Counter-Ant Behavior

We present here another view of swarm intelligence inspired from ant colony optimization; it is based on a modified version of ant algorithm called Counter-Ant-Algorithm (CAA). Figure 1 expresses the pseudo-code of our CAA: the modules task_Achievement encapsulates the objective operation (i.e; cleaning, collecting, picking, hunting, etc.) pheromone_Update formulates the evaporation phenomenon of the pheromone and daemon_Counter_Reactions expresses the reaction of the robots, which is repulsion instead of attraction to pheromone; it includes also the solution of stagnation recovery. To feign well our algorithm, we propose a nominal scenario describing the behavior of self-organized ant-robots explorers (see figure 2).

Given the evaporation property of the pheromone, the paths eventually disappear which enables ants expand their scanning areas. In other words, as old paths marked by the pheromone no longer exist, then the ants can reach, when they move randomly, new places, which were not detected by other ants.

3.1 Pheromone Update

Ants move and marking paths with a constant amount of pheromone, after a fixed time, the paths of pheromone begin to disappear. The amount of pheromone in each zone of the cleaning environment is updated according the equation 1.

```
Procedure Counter-Ant
  While (not_termination)
    taskAchievement()
    pheromoneUpdate()
    daemonCounterReactions()
  End While
End Procedure
```

Fig. 1. Counter-Ant behavior pseudo code

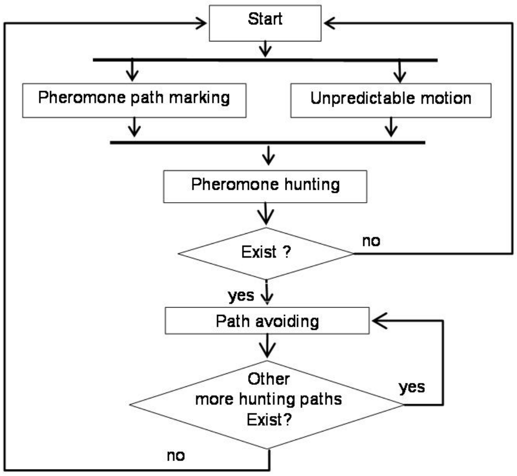


Fig. 2. Collaborative exploration behavior based on CAA

$$\tau_{x,y} = \rho\tau_{x,y} + \Delta\tau_{x,y} \tag{1}$$

where, τ is the amount of pheromone in the zone of coordinates x, y; ρ is the rate of pheromone evaporation and $\delta\tau$ is the amount of pheromone deposited when an ant go through the zone x, y.

4 Resolution of Stagnation with the Pheromone

A problematic case appears: the stagnation of robots motion due to dead ends. This means that robots lost totally the possibility of moving in their unknown environment. We propose to include in our CAA a stagnation recovery which allows the robots to overtake dead ends and to resolve stagnation problems. The unpredictable motion of ants' robots is accompanied by a secretion of pheromone so that the other ants do not borrow the already marked paths. After a while, the ant-hill (the robot environment) will be blocked by pheromone, this can create what we call stagnation situations (SS). In our system, the SS are similar to dead ends [1][11][20]: they refer to situations where ants are surrounded by the pheromone; this causes the blocking of their actions because they have to avoid the perceived pheromone.

4.1 Stagnation Recovery Using Evaporative Pheromone

The preservation of the pheromone evaporation property has allowed the ants to enlarge their sweeping areas one hand, and on the other hand, reduced the stagnation situations since the paths end up disappearing, which allows the ants to continue exploring the dynamic environment. Evaporation of the pheromone is not instantaneous; it requires some time to be completed. This helps to inform, in an interval of time, the maximum number of ants already passed. However, the lifetime of the pheromone is itself a factor of stagnation situations: ants can be found surrounded by tracks of pheromones that have not yet evaporated. The first idea for resolving these problematic situations is to take advantage from evaporation: ants stuck and have to wait till evaporation of one condemning issue, but this causes a lack of exploring workforce! If all ants are imprisoned, our algorithm will enter a waiting phase until their release. This issue will negatively affect the convergence time of our system because it will be affected by pending active ants that are in stagnation situations.

4.2 Stagnation Recovery Using Positions' Pheromone

The solution is then to give ant the ability to become localized according to pheromone position: if its location is parallel or confused to the pheromone, it has to change direction in order to find another way, elsewhere it may be in the presence of other ants. On the other hand, if its location is not parallel, it can move to overcome this stagnation. In this case, the ant doesn't avoid the pheromone since both don't have the same direction and therefore we no longer refer to congestion areas. This idea is encapsulated in daemonCounter-Reactions() function

5 Experimentation Results with Lego Robots

The inter-robot cooperation takes place in a totally unknown environment. It consists of chemical-sensitive navigating platform containing some objects to pick up and two Lego Mindstorms ant-robots. In order to test the performance of our CAA, we had to look for a suitable assembly of Lego Mindstorms robots[1] [14] [15] to be cooperative explorers and cleaners [3][8] and an appropriate materialization of pheromone [16]. We utilize magic ink (see figure 3), which is a chemical substance made up of ethanol, thymolphthalein, demineralized water and sodium hydroxide capable of producing chemical traces of disappearing blue until their total fading.

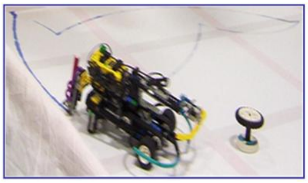


Fig. 3. Path traces and their evaporation

[1] http://mindstorms.lego.com/eng/products/ris/index.asp

We construct our ant-robots from Lego Mindstorms of Robotics Invention System series. One box contains one RCX 2.0 (Robotics Command System) which presents the intelligent programmable part. Using these boxes, we were able to build a twin Lego robots. In addition to the various Lego parts, each one is built by one RCX box, carpet of loading and unloading objects, light, contact sensors and the magical ink. This construction may well realize the basic functions of our algorithm which can be summarized in the motion, loading/unloading objects, and the detection of pheromone, obstacles and borders. Figure 4 presents some experimental scenarios photographed in the occasion of Sfax University fair (Tunisia).

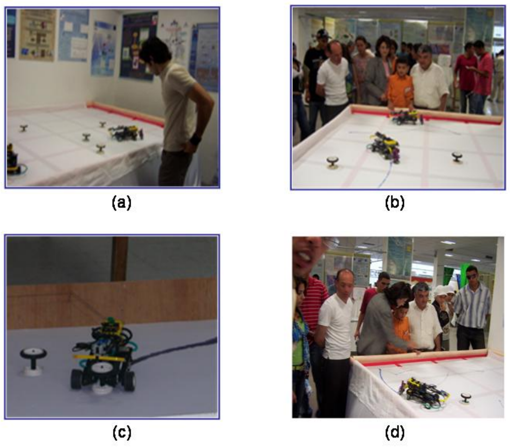


Fig. 4. Some experimental scenarios (a) Robots starting, (b) Motion with ink trace, (c) Object cargo, and (d) Evaporation of some paths

A series of experimentations show the usefulness of our algorithm for self-organizing and cooperative exploration and cleanup. We tried to clean the experimental environment with only one robot and then with two robots. We noticed, in all cases, a temporal gain around 40% and a spectacular self partitioning of the explored area. In order to analyse the behaviour of robots during the execution of the CAA, it is judicious to highlight the individual behaviour of a robot. The robot moves randomly when running the treadmill. It should be mentioned that the robots are blind due to the absence of a mechanism that can detect remote object (figure 5a). Figure 5b shows the object cargo found on the robot path. This operation is provided via the treadmill. Figure 5c and 5d show that the treadmill charges object till the touch sensor. The robot carries the object outside, stops, removes the object and performs a reverse to change direction (figure 5e and 5f). It continues moving randomly for cleaning, There are two cases that arise showing the reaction of the robot according to the magic-ink (pheromone): if the trace is perpendicular to the robot path (figure 5g), the robot continues crossing (figure 5h). But if the trace coincides with the robot trajectory (figure 5i), the robot moves away (figure 5j). The robot is equipped with a front bumper, which allows it to detect static and moving obstacles such as another robot (figure 5k); in this case, both have to change direction (figure 5l). The conservation of the property of evaporation generates disappearance of the pheromone traces, which makes it possible the robots to increase their sweeping zones.

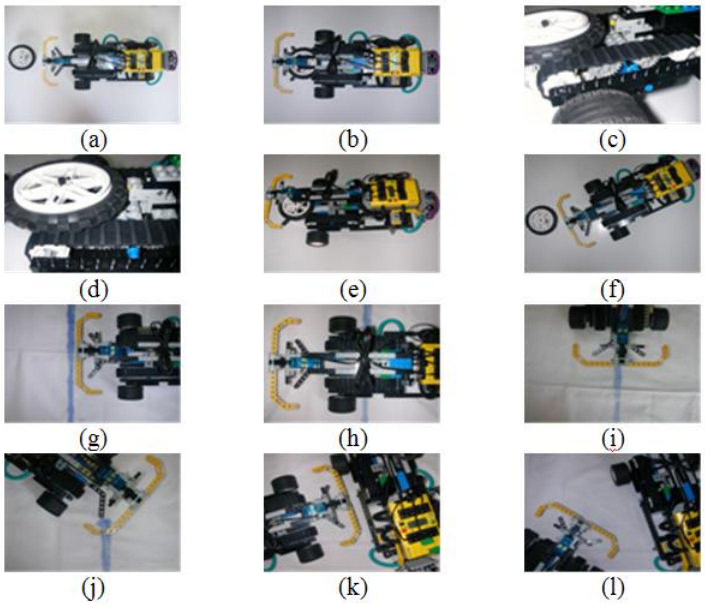


Fig. 5. Individual behavior of our ant-robot

The experimental phase has shown an acceptable assembly of reactive and cooperative cleaning robots and their adaptability to functions requested in our counter-Ant algorithm: both Lego Mindstorms robots are designed to move randomly for objects hunting, marking paths by an evaporating Magic ink. To identify ink plots, robots use light sensors having however some failures due to the non-precision values that depend on the ambient light in the room. A series of experimentations allowed us to improve our program. Indeed, we have been able to improve speed motion and load; this precludes our construction holds some constraints related to the adaptability degree of objects to the treadmill.

6 Simulation Results under Madkit Platform

We carried our interest in the cleaning field and we noticed that cleaning an unknown environment by a multirobot system, apparently simple, conceals some complexity in terms of inter-robots coordination and efficiency. In fact, we noticed a congestion in the cleaning zones which influences negatively the convergence time. To better discuss this function, we implemented our CAA in the simulation platform Madkit [2]. Collaboration between robots is done in a completely unknown environment. It is made up mainly by objects and cleaning robots which are implemented and generated by the simulation platform. These cleaning robots are provided by a skeletal sensor enabling them to detect the pheromone trace. The robots are completely blind and their motion is completely random. For the cleaning operation, it is question to remove meted. This enables us to focus on our Counter-Ants Algorithm to emphasize our multirobot collaboration method. Moreover, we can consider also that our robots are provided by tanks able to contain a collection of garbage. Figure 6a presents the simulation environment at the starting time t=0 containing five cleaning robots and 80 objects. The time of pheromone evaporation is firstly fixed to 45s. Figure 6b is a screen caption after a running time t =18,63s; according to pheromone traces, we can see that the robots move away or cross other marked paths, since the robots behavior is based on repulsion instead of attraction to pheromone. The cleaning robot motion is accompanied by the activation of pheromone detectors in order to locate the recent robots trajectories in the environment. The figure 6c shows the end of the cleaning operation after t=135,20s. As the paths end up disappearing, it is what explains the existing number of pheromone traces in the environment, and after an additional 45s, all first pheromone traces will disappear (see Figure 6d).

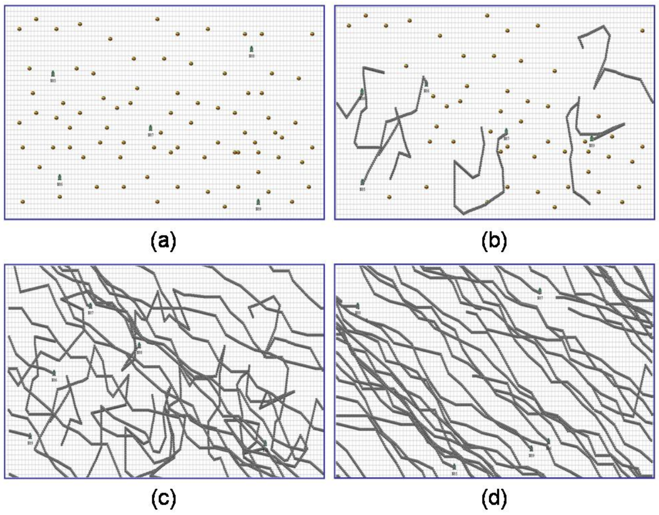


Fig. 6. Some simulation screen captions

[2] http://www.madkit.org

6.1 Influence of Parameters' Variation

Figure 7 shows (from 1 to 2 robots) that in spite of the fixed time of evaporation to 8s, the convergence time is improved, from 1816,09s with only one robot, to 1426,17s with two robots. This emphasizes the effectiveness of swarm robotics in the realization of cleaning operation. This figure also shows (from 2 to 5 robots) that the long life of pheromone paths guarantees the reduction of obstruction zones, since their presence allow robots to recognize previously explored zones. This is ensured thanks to the reaction of robots with respect to the pheromone which consists of repulsion and not attraction. The increase in the number of robots with the persistence of the traces does nothing but improve time of convergence since the robots any more will not seek objects in open zones but rather in closed spaces.

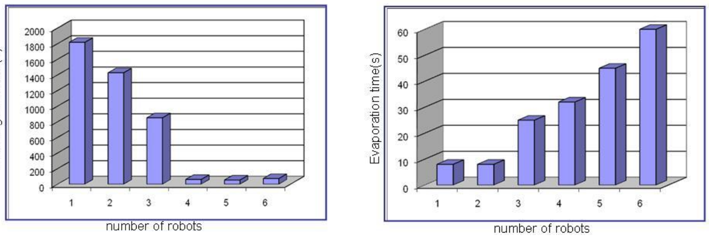


Fig. 7. Evaluation of the convergence time following the variation of the number of robots and the evaporation time

In terms of stagnation recovery, we observe that in spite of the long life of the pheromone evaporation, the robots are able to progress while accelerating the convergence time. However, (from 5 to 6 robots) the persistence of the pheromone trace for a long life (45s to 60s) increase the convergence time. In fact, the environment will be encumbered by pheromone traces, thing that limits the robot motion and causes slowness of convergence time.

6.2 Exploration Behavior vs. Random Behavior

To better point out the effectiveness of counter-ant behavior and stagnation recovery, we propose to compare it with a random behavior. Both simulation environments contain 80 objects and 5 robots. 45s is the time of pheromone evaporation for counter-ant behavior. The figure 4a shows the initial state of the random system. The figures 8b and 4c show the random operation of cleaning where lines represent robotsḿotion traces and cluttering lines are due to pure rondom motion. The figure 8d shows a screen caption after t=843,56s. Whereas, the conversion time with our CAA is t=135,20s according to the figure 6d.

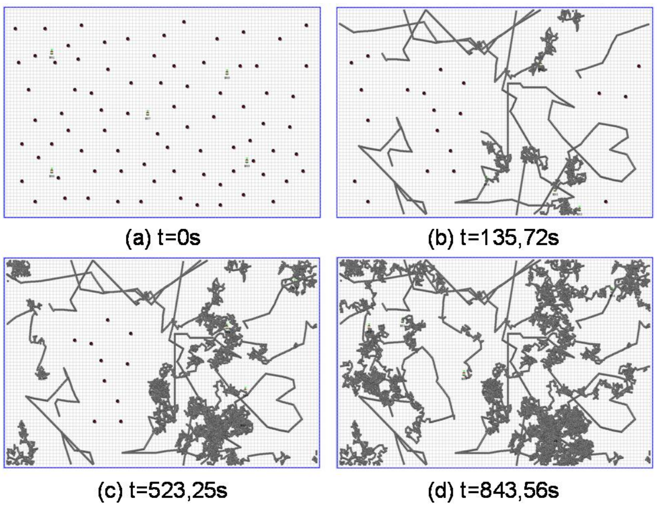


Fig. 8. Simulation of hazardous behaviors

6.3 Counter-Ants vs. Alliance

The Alliance architecture suggested by Parker [17] based on faults tolerance is a supervised architecture which consists on collaborating robots to clean two sources of rubbish. The communication between robots is done through the diffusion of messages about sources localization via a supervision system. Alliance is certainly based on the faults tolerance however, the supervision of the robots present in itself a limit of the solution. It is important to mention that the quality of the exploration of the environment was not approached. Figure 9 summarizes this assessment

Approaches	Supervision structure	Method of communication	faults Tolerance	Based	Environment exploration	Optimization	Application	Convergence Time
Counter-Ants	No	Stigmergy : pheromone	Yes	pheromone	assured	Yes	Exploration and cleanup	Medium
Alliance	Yes	sensors+ diffusion of Messages	Yes	Behavior	-	No	Cleanup	Long

Fig. 9. Counter-Ants vs. Alliance

Alliance presents a specific solution for rubbish cleaning gathered and not scattered. However, our CAA can be well applied to scattered objects cleaning in unknown environment and to any exploration problem. It is an unsupervised solution with stigmergy as the base of communication and collaboration. In fact, it possible to solve the exploration problem and the dilemma convergence time/goal to reach.

7 Concluding Remarks

This work consists to present and apply a modified version of the known ant algorithm, called Counter-Ant Algorithm (CAA). It consists on a new collaborative behaviour in relation to the pheromone in a multirobot system. This technique enables robots to place the pheromone while moving randomly. This will no longer attract other robots, but rather to avoid motion paths (already explored and cleaned areas) by other robots. As a result, this method allows improving system efficiency and minimizes the creation of congested areas. As the old paths marked by the pheromone will disappear, new places become open to explore, which were not detected by other ant-robots. This leads us to say that the subdivision of the area to explore is self organized [9]. We present also a solution for the stagnation recovery which allows the robots to overtake dead ends. We simulate, implemented and experiment our algorithm in laboratory conditions similar to real ones. A series of simulations and experimentations show the usefulness of our algorithm for adaptive and cooperative exploration (including cleaning, picking, etc). As perspectives, we propose firstly to look for a more natural and less glaring representation of pheromone, Secondly, we intend to hybridize our algorithm with a soft computing technique, such as fuzzy rule base system, in order to surround more fluently the stagnation situations.

Acknowledgements

The authors thank the Tunisian General Direction of Scientific Research and Technological Renovation (DGRSRT), under the ARUB program 01/UR/11-02, Tunisia.

References

1. Borzello, E., Merkle, L.D.: Multi-Robot Cooperation Using the Ant Algorithm with Variable Pheromone Placement. Proceedings of the IEEE (2005)
2. Casillas, J., Cordón, O., Fernández de Viana, I., Herrera, F.: Learning cooperative linguistic fuzzy rules using the best-worst ant system algorithm. International Journal of Intelligent Systems 20(4) (2005)
3. Cao, Y.U., Fukunaga, A.S., Khang, A.B.: Cooperative mobile robotics: antecedents and directions. Autonomous Robots 4(1), 7–27 (1997)
4. Dorigo, M., DiCaro, G.: The Ant Colony Optimization Meta-Heuristic, New Ideas in Optimization 1999. McGraw-Hill, New York (1999)

5. Dorigo, M., Stützle, T.: Ant colony optimization. The MIT Press, Cambridge (2004)
6. Dudek, G., Jenkin, M., Milios, E., Wilkes, D.: A taxonomy for swarm robots, Intelligent Robots and Systems. In: IROS 1993. Proc. of the 1993 IEEE/RSJ, July 1993, vol. 1, pp. 441–447 (1993)
7. Holland, O., Melhuish, C.: Stigmergy, self-organization, and sorting in collective robotics. Artificial Life 5(2), 173–202 (1999)
8. Johnson, J., Sugisaka, M.: Complexity science for the design of swarm robot control systems, Industrial Electronics Society. In: IECON 2000. 26th Annual Conference of the IEEE, vol. 1, pp. 695–700 (October 2000)
9. Kallel, I., Mezghani, S., Alimi, A.M.: Towards a Fuzzy Evaluation of the Adaptivity Degree of an Evolving Agent. In: Proc. of the 3rd International Workshop IEEE GEFS, March 2008, pp. 29–34 (2008)
10. Kasabov, N., Filev, D.: Evolving intelligent systems: methods, learning, applications. In: Proc. Int. Symposium on Evolving Fuzzy Systems EFS, pp. 8–18 (2006)
11. Kube, C.R., Bonabeau, E.: Cooperative transport by ants and robots. Robotics and Autonomous Systems 30, 85–101 (2000)
12. Kube, C.R., Zhang, H.: Collective robotics: From social insects to robots. Adaptive Behavior 2, 189–218 (1994)
13. Kube, C.R., Zhang, H.: Stagnation recovery behaviors for collective robotics. In: Proceedings of the 1994 IEEE/RSJ/GI International Conference on Intelligent Robots and Systems, pp. 1883–1890. IEEE Computer Society Press, Los Alamitos (1995)
14. Lund, H.H.: Adaptive robotics in entertainment. Applied Soft. Computing (2001)
15. Lund, H.H., Pagliarini, L.: RoboCup with LEGO MINDSTORMS. In: Proceedings of the International Conference on Robotics and Automation (ICRA 2000). IEEE Press, Los Alamitos (2000)
16. Mamei, M., Zambonelli, F.: Pervasive Pheromone-Based Interaction with RFID Tags. ACM Transactions on Autonomous and Adaptive Systems 2(2), 1–28 (2007)
17. Parker, L.: ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation. IEEE Transactions on Robotics and Automation 14(2), 220–240 (1998)
18. Purnamadjaja, A.H., Russell, R.A.: Pheromone communication: implementation of necrophoric bee behaviour in a robot swarm. In: Robotics, Automation and Mechatronics, vol. 2, pp. 638–643. IEEE, Los Alamitos (2004)
19. Szu, H., Chanyagorn, P., Hwang, W., Paulin, M., Yamakawa, T.: Collective and distributive swarm intelligence: evolutional biological survey. International Congress Series, August 2004, vol. 1269, pp. 46–49 (2004)
20. Yingying, D., Yan, H., Jingping, J.: Multi-Robot Cooperation Method Based On The Ant Algorithm. Proc. of the IEEE (2003)
21. Zhang, D., Xie, G., Yu, J., Wang, L.: Adaptive task assignment for multiple mobile robots via swarm intelligence approach. Robotics and Autonomous Systems 55(7), 572–588 (2007)

Self-Organization for Fault-Tolerance

Elena Dubrova

Department of Electronics, Computers, and Software
Royal Institute of Technology, Electrum 229, 164 46 Kista, Sweden
dubrova@kth.se

Abstract. In the last decade, there has been a considerable increase of interest in fault-tolerant computing due to dependability problems related to process scaling, embedded systems, and ubiquitous computing. In this paper, we present an approach to fault-tolerance inspired by gene regulatory networks of living cells. Living cells are capable of maintaining their functionality under a variety of genetic changes and external perturbations. They have natural self-healing, self-maintaining, self-replicating, and self-assembling mechanisms. The fault-tolerance of living cells is due to the ability of their gene regulatory network to self-organize and produce a stable attractors' landscape. We introduce a computational scheme which exploits the intrinsic stability of attractors to achieve fault-tolerant computation. We also test fault-tolerance of the presented scheme on the example of a gene regulatory network model of *Arabidopsis thaliana* and show that it can tolerate 68% single-point mutations in the outputs of the defining tables of gene functions.

1 Introduction

The concept of self-organization is inherent to biological systems, from the genetic to the ecosystem level. Contrary to chemistry, where self-organization is often taken as a synonym for self-assembly, in biology, self-organization is usually associated with emergence of patterns at the global level solely from interactions of the lower-level components of a system [11]. Spontaneous folding of proteins, the development of an embryo, or the origin of life itself, are all examples of the phenomena of self-organization in biology.

In this paper, we investigate how self-organization of biological systems can exploited for achieving a fault-tolerant computation. Fault-tolerance is becoming a critical issue as fail-safe technologies migrate to embedded system applications well away from the traditionally security- and safety-conscious military-aerospace and automotive sectors. The International Technology Roadmap for Semiconductors 2007 (ITRS)[1] identifies fault-tolerance as one of the five "cross cutting" design challenges, along with productivity, power, manufacturing integration, and interference. On one hand, smaller process sizes lead to an increased likelihood of noise-related faults caused by crosstalk.

[1] The ITRS, officially updated every two years, is a road map for the semiconductor industry that encompasses such topics as devices and structures, interconnect, lithography, metrology, test, assembly and packaging, and yield (http://www.itrs.net/).

On the other hand, lower supply voltage levels, which are used to decrease power dissipation, result in even higher susceptibility to noise [28]. Moreover, denser feature sizes considerably increase the probability of *soft errors* (also called *transient faults*, or *glitches* [1]) caused by cosmic rays and alpha particles [9].

The interest in fault tolerance is further boosted by the ongoing shift from the traditional desktop information processing paradigm, in which a single user engages a single device for a specialized purpose, to *ubiquitous computing*, in which many small, inexpensive networked processing devices are engaged simultaneously and distributed at all scales throughout everyday life [35]. The demand for low-cost, low-area, low-power devices which satisfy high safety and security requirements of ubiquitous computing brings a need for unconventional approaches to fault-tolerance.

The design of a fault-tolerant electronic system capable of operating under changing environments and noisy input and exhibiting the desired behavior under constraints such as power consumption, area, and performance is a fundamental challenge. Yet biological systems are able to handle this challenge with an elegance and efficiency. Living cells can maintain their performance under a broad range of random perturbations, varying from temporary chemical or physical changes in the environment to permanent genetic mutations. Our goal is to find new ways to achieve fault tolerance by learning more about how gene regulatory networks of living cells perform fault-tolerant computation and to attempt to mimic it.

Previous works have shown that the fault-tolerance of gene regulatory networks is due to the intrinsic stability of their attractors' landscape [5]. In this paper, we propose a computational scheme which exploits the stability of attractors. In this scheme, the states of a network represent variables of the computed function, and attractors represent function's values. We test the fault-tolerance of the presented computational scheme on the example of a gene regulatory network model of *Arabidopsis thaliana* [12] and show that it can tolerate 68% single-point mutations in the outputs of the defining tables of gene functions.

The paper is organized as follows. In Section 2, we give a brief introduction to gene regulatory networks and, in Section 3, to Boolean networks. In Section 4, we present the new computation scheme and, in Section 5, evaluate its fault-tolerance. Section 6 concludes the paper and discusses open problems.

2 Gene Regulatory Networks

The *Gene Regulatory Network* (GRN) is one of the most important signaling networks in living cells [4]. It is composed of the interactions of proteins with the genome. The major discovery related to GRNs was made in 1961 by French biologists François Jacob and Jacques Monod [23]. They found that a small fraction of the thousands of genes in the DNA molecule acts as tiny "switches". By exposing a cell to a certain hormone, these switches can be turned "on" or "off". The activated genes send chemical signals to other genes which, in turn, get either activated or repressed. The signals propagate along the cell until it settles down into a stable pattern.

Jacob and Monod's discovery showed that the DNA is not just a blueprint for the cell, but rather an automaton which allows for the creation of different types of cells. It answered the long open question of how one fertilized egg cell can differentiate itself into brain cells, lung cells, muscle cells, and other types of cells that form a newborn baby. Each kind of cells corresponds to a different pattern of activated genes in the automaton.

Jacob and Monod introduced the first model of the GRN described by a system of differential equations for the activation and deactivation of the set of genes controlling the transport and metabolism of lactose in *E. coli* bacteria. Since then, this little genetic circuit, known as the *lac operon*, has been a prototype for the modeling of the GRN. It is a point of view of many biologists that the only valid approach for the modeling of the GRN is through differential equations.

In 1969 Stuart Kauffman proposed an alternative model of the GRN, called Boolean network [24]. In this model, one is interested in the state of expression of the genes rather than in the concentration of their products. The genome is represented by a set of Boolean variables which are related to each other through some logical rules. For many years, the Kauffman model was considered as an over-simplification of the GRN. Most did not believe that the Boolean approach can yield accurate descriptions of real biological systems.

However, recently it turned out that Kauffman's model is much more powerful than it was originally thought. Experimental and numerical evidence have shown that gene expression profiles of real organisms can be recovered by using the Boolean approach [29, 2, 17]. Kauffman's hypothesis stating that dynamical attractors of the GRN correspond to cell types have been investigated and confirmed experimentally [22, 21]. These results show that the Boolean network model indeed captures the essential aspects of the interactions between the genes.

Many other network models of the GRN have been proposed (see [31] for an overview). Although the details of the network's dynamics might change from continuous parameters in one model to discrete ones in the other, the general properties of the dynamics appear to be model independent. It has been demonstrated that continuous and discrete descriptions of the GRN exhibit similar dynamical properties under very general conditions [32, 13]. For example, a Boolean network of the common fruit fly *Drosophila melanogaster* [3] has been shown to recover the same patterns for segment polarity genes as those recovered by a continuous model [33]. Discrete models are further justified because recent experimental evidence suggests that, at the individual cell level, gene expression is digital and stochastic rather than continuous [12].

Apart of the simplifying assumption regarding the discrete states of a gene, the Boolean network model assumes the synchronous updating. It has been demonstrated that synchronous and asynchronous updating schemes yield very similar critical stability values, meaning that the transition between the dynamical phases does not depend on the updating scheme [19].

Lastly, the restriction to only two binary states has also been shown justifiable. In [12], the original ternary-state network model of a small flowering plant *Arabidopsis thaliana* [17] has been translated to a binary-state one. It has been shown that the Boolean model reaches the same number and type of attractors as the ones reached by the original model. It also responds to perturbations in a qualitatively identical manner to the ternary one. These results suggest that binary states are sufficient to capture the dynamical features of the GRN.

3 Boolean Networks

In the *synchronous Boolean network model* of the GRN [24], every gene is represented by A vertex in a directed graph with an associated Boolean variable x_i that can take two values: $x_i = 1$ if the gene is expressed and $x_i = 0$ otherwise. The genome is represented by a set of n variables, $x_1, x_2, ..., x_n$. An edge from one vertex to another indicates that the former gene regulates the latter. Time is viewed as proceeding in discrete steps. At each step, the expression of the gene x_i changes according to the equation

$$x_i(t+1) = f_i(x_{i_1}(t), x_{i_2}(t), \ldots, x_{i_{k_i}}(t)),$$

where $x_{i_1}, x_{i_2}, \ldots, x_{i_{k_i}}$ are *regulators* of x_i and f_i is a Boolean function which is assigned according to the inhibitory or activatory nature of the regulators. The k_i-tuples of values of the regulators for which $f_i = 1$ are called *activatory* assignments and those for which $f_i = 0$ are called *inhibitory* assignments.

The *state of the network* is defined as an ordered n-tuple of values of variables $x_1, x_2, ..., x_n$ describing which genes in the network are expressed or not at a particular moment. Since the network is deterministic and finite, any sequence of consecutive states eventually converges to either a single state, or a cycle of states, called *attractor*. The *basin of attraction* of an attractor A, denoted by $B(A)$, is the set of all states from which A can be reached. Kauffman hypothesized that attractors correspond to a combination of gene expressions which specifies a particular cell type or cell fate of an organism [25]. The number of cell types predicted by the Boolean network model agrees well with our current knowledge [22, 26].

An example of a Boolean network is shown in Figure 1. Arrows indicate activatory regulation and blunt-ends represent inhibitory regulation. The following Boolean functions are associated to the vertices:

$$f_a = x_c', \quad f_b = x_b \cdot (x_a' + x_c'), \quad f_c = x_b,$$

where "$\cdot$", "$+$", and "$'$" stand for the Boolean AND, OR and NOT, respectively. The state transition graph describing the dynamics of this network is shown on the right-hand side of Figure 1. There are two point (i.e. single-state) attractors: (100) and (011).

The fraction p of activatory assignments of regulators in the entire network, called the *gene expression probability*, is an important parameter which controls the dynamical phase in which the network operates. Dynamics of randomly generated Boolean networks has been extensively studied (see [6] for an overview). It has been shown that, for a given gene expression probability p, the critical value k_c of the mean input degree of vertices at which the transition from ordered to chaotic phase occurs is given by the equation:

$$k_c = \frac{1}{2p(1-p)}. \quad (1)$$

In the infinite size limit, if the mean input degree k of vertices in the network is smaller than k_c, then the network is in the *ordered phase* [18]. If $k > k_c$, then the network

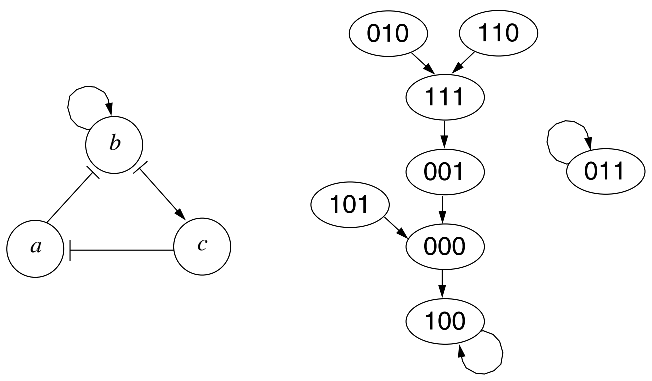


Fig. 1. A Boolean network and its state transition graph. Each state is a triple (x_a,x_b,x_c).

is in the *chaotic phase* [27]. If $k = k_c$, then the network exhibits *self-organized critical* behavior [8].

Kauffman hypothesized that self-organized critical Boolean networks are good candidates for the modeling of real GRNs [25]. On one hand, networks in the ordered phase exhibit "frozen" dynamics in which small variations in the initial state of the network typically die out over time. On the other hand, networks operating in the chaotic phase are extremely sensitive to small changes in the initial state and therefore unstable [20, 27]. A compromise between frozen and chaotic behavior is achieved close to the critical line between the phases, where the network exhibits maximum robustness and evolvability simultaneously [5]. If a system is both robust and evolvable, then, on one hand, it is able to sustain a broad range of external perturbations, and, one the other hand, these perturbations can eventually make the system to acquire novel functions which suit the new conditions better. Recent works have shown evidence that GRNs of living cells exhibit self-organized critical behavior [7, 30].

4 Boolean Network-Based Computational Scheme

Suppose that we have a Boolean network G with n vertices $v_1,\dots,v_n$ and m point attractors $A_1,A_2,\dots,A_m$ such that the attractor A_i is given by the state $(a_{i1},a_{i2},\dots,a_{in}) \in \{0,1\}^n$, $i \in \{1,2,\dots,m\}$. The basins of attractions $B_i(A)$ partition the Boolean space $\{0,1\}^n$ into m connected components via a dynamic process. We assume that the set of all points of the Boolean space corresponding to the states in the basin of attraction of A_i is mapped into the state of the attractor A_i. Then, G defines a set of n Boolean functions of type $g_j : \{0,1\}^n \to \{0,1\}$ of variables $x_1,\dots,x_n$, where the variable x_i corresponds to the variable associated to the vertex i. More formally:

Definition 1. *A Boolean network with n vertices and m point attractors $A_1,A_2,\dots,A_m$ such that the attractor A_i is given by the state $(a_{i1},a_{i2},\dots,a_{in}) \in \{0,1\}^n$, $i \in \{1,2,\dots,$ $m\}$, represents a set of n Boolean functions of type $g_j : \{0,1\}^n \to \{0,1\}$ which are defined as follows:*

$$g_j(s_1,\ldots,s_n) = a_{ij} \textit{ if and only if } (s_1,\ldots,s_n) \in B(A_i),$$

for all $(s_1,\ldots,s_n) \in \{0,1\}^n$ *and all* $i, j \in \{0,1,\ldots,m-1\}$.

The size of the resulting Boolean network representation is linear in the number of variables of the represented function. The maximal number of steps required to compute the value of the function for a given assignment of variables is equal to the longest path to an attractor. Although such a path can potentially be exponential in the number of variables, it is known that self-organized critical Boolean networks reach an attractor in a relatively small number of steps [6]. For example, in the Boolean network model of *Arabidopsis thaliana* considered in the next section, an attractor is reached in at most 8 steps [12], while the mean number of steps varies"

As an example, consider the Boolean network in Figure 1. According to Definition 1, it represents the Boolean functions g_1, g_2, g_3 specified by the following table:

x_a	x_b	x_c	g_1	g_2	g_3
0	0	0	1	0	0
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	0
1	1	0	1	0	0
1	1	1	1	0	0

The computational scheme presented above is an improved version of our previous idea [16], in which a Boolean network with n vertices and m attractors $A_1, A_2, \ldots, A_m$ was defined to represent a function of type $g : \{0,1\}^n \rightarrow \{0,1,\ldots,m-1\}$ given by:

$$g(s_1,\ldots,s_n) = i \text{ if and only if } (s_1,\ldots,s_n) \in B(A_i),$$

for all $(s_1,\ldots,s_n) \in \{0,1\}^n$ and all $i \in \{0,1,\ldots,m-1\}$.

The previous scheme assumed that it is possible to store a logical value with every attractor. This assumption might be difficult to implement. In the new scheme, the values of the states of attractors represent the computed value of the function, eliminating the need for any additional storage. Other advantages of the modified scheme are:

1. The represented functions are always Boolean, independently of the number of attractors m.
2. All n functions are computed in parallel by an n-vertex Boolean network;
3. The stopping criteria for the computation is easier since a point attractor is simple to detect by checking whether the current state is equal to the previous state.

5 Evaluation of Fault-Tolerance

Living organisms can sustain a wide variety of genetic changes. Gene regulatory networks and metabolic pathways self-organize and re-accommodate to make the organism

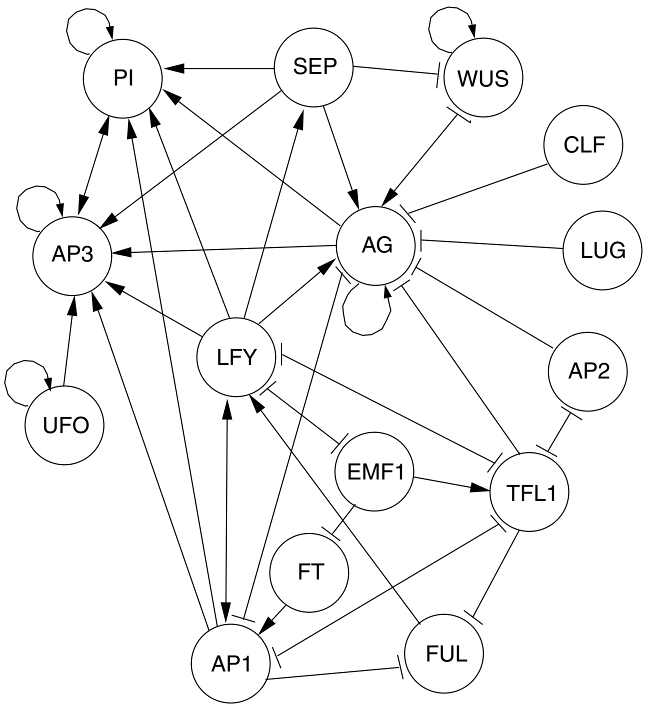


Fig. 2. A gene regulatory network model of the wild-type *Arabidopsis thaliana*

continue performing under many point mutations, gene duplications and gene deletions [34]. In this section, we demonstrate that the proposed computational scheme inherits the intrinsic robustness of living organisms. As a case study, we take gene regulatory network model of *Arabidopsis thaliana* floral organ cell fate determination presented in [12] (see Figure 2). The description of Boolean functions associated to vertices is given in the Appendix. These functions were derived by the authors of [12] from molecular genetic experimental data. The state transition graph of the Boolean network in Figure 2 has 10 point attractors, shown in Table 1. These attractors coincide with the gene expression profiles that have been documented experimentally in the cells of wild-type *Arabidopsis*. The number of states in each basin of attraction is shown in Table 2. Note that the sizes of the basins of attraction may indicate which genes are critical to attain each cell type. A larger basin usually implies a more stable attractor, suggesting that the cells of the reproductive organs (stamens and carpels) are more stable than the ones of the perianth organs (sepals and petals) [12].

We have tested fault-tolerance of the computational scheme presented in the previous section by single random point alterations of the outputs in the defining tables of gene functions of the Boolean network in Figure 2. At each run, one alteration was done in one of the outputs of the associated function of a selected network's vertex. After each alteration, the basins of attraction were recomputed and the functions represented by the network in accordance with Definition 1 compared to the original ones.

Table 1. Point attractors of the Boolean network in Figure 2; Infl = inflorescence meristematic cells; Pe = petal primordial cells; Sep = sepal primordial cells; St = stamen primordial cells; Car = carpel primordial cells

AP3	UFO	FUL	FT	AP1	EMF1	LFY	AP2	WUS	AG	LUG	CLF	TFL1	PI	SEP	Cell type
0	0	0	0	0	1	0	0	0	0	1	1	1	0	0	Infl1
0	1	0	0	0	1	0	0	0	0	1	1	1	0	0	Infl2
0	1	0	0	0	1	0	0	1	0	1	1	1	0	0	Infl3
0	0	0	0	0	1	0	0	1	0	1	1	1	0	0	Infl4
1	1	0	1	1	0	1	1	0	0	1	1	0	1	1	Pe1
1	0	0	1	1	0	1	1	0	0	1	1	0	1	1	Pe2
0	0	0	1	1	0	1	1	0	0	1	1	0	0	1	Sep
1	1	1	1	0	0	1	1	0	1	1	1	0	1	1	St1
1	0	1	1	0	0	1	1	0	1	1	1	0	1	1	St2
0	0	1	1	0	0	1	1	0	1	1	1	0	1	1	Car

Table 2. Size of the basins of attraction of the Boolean network in Figure 2

Attractor	Infl1	Infl2	Infl3	Infl4	Sep	Pe1	Pe2	St1	St2	Car
Basin size	512	512	256	256	448	8	440	15168	568	14600

Table 3. Results of injecting single faults in each output of all Boolean functions associated to vertices of the Boolean network in Figure 2; N_{tol} = the total number of tolerated faults; N_{all} = the total number of possible faults

Vertex	AP3	UFO	FUL	FT	AP1	EMF1	LFY	AP2	WUS	AG	LUG	CLF	TFL1	PI	SEP	total
N_{tol}	82	0	1	0	2	0	0	0	0	381	0	0	6	56	0	528
N_{all}	128	2	4	2	16	2	16	2	8	512	1	1	16	64	2	776

This was repeated for each output of the Boolean functions associated to all vertices of the Boolean network in Figure 2 (776 times in total). The results are summarized in Table 3. As we can see, only 248 of the 776 altered networks (31.96%) yielded a different function from the original one. So, 68.04% of single-point mutations in the outputs of the defining tables of gene functions were tolerated.

In our experiments, we checked the equivalence of functions before and after the alternation using the software package for computing attractors in Boolean networks which we have developed [15, 14, 16]. The algorithm starts from a randomly selected state and applies forward reachability analysis to find a state in some attractor A_i. Then, using this state as a final state, backward reachability analysis is performed to find the remaining states in the basin of attraction $B(A_i)$. The process is repeated starting from a state not previously visited until the complete state space is covered. We used this algorithm for checking the equivalence of functions represented by Boolean networks by running it in parallel on two networks from the same initial state. For each computed attractor A_i, the basins of attractions of the two networks were compared for equivalence of states. We used Binary Decision Diagrams (BDDs) [10] for representing the set of states in the basin of attraction, which made the equivalence checking particularly efficient (a constant-time operation).

6 Summary

The paper present a novel approach to design robust systems by mimicking the behavior of gene regulatory networks. Traditionally, fault-tolerance of a system is achieved by adding redundant components. We show that there are other alternatives. In the computational scheme which we introduce in this paper, fault-tolerance is due to the non-uniqueness of paths leading to an attractor. A fault may change a path, but the destination remains the same with a high probability. Therefore, if attractors are stable, the network is able of sustain the majority of faults.

In our future work, we plan to test the pros and cons of the presented approach with respect to complexity and accuracy on realistic problems characterized by large networks.

References

1. Terrestrial cosmic rays and soft errors. IBM Journal of Research and Development, 40(1) (1996)
2. Albert, R., Othmer, H.G.: The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in drosophila melanogaster. J. Theor. Biol. 223, 1–18 (2002)
3. Albert, R., Othmer, H.G.: The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in drosophila melanogaster. J. Theor. Biol. 223, 1–18 (2003)
4. Alberts, B., Bray, D., Lewis, J., Ra, M., Roberts, K., Watson, J.D.: Molecular Biology of the Cell. Garland Publishing, New York (1994)
5. Aldana, M., Balleza, E., Kauffman, S., Resendis, O.: Robustness and evolvability in gene regulatory networks. J. Theor. Biol. 245, 433–448 (2007)
6. Aldana, M., Coopersmith, S., Kadanoff, L.P.: Boolean dynamics with random couplings, `http://arXiv.org/~abs/adap-org/9305001`
7. Balleze, E., Alvarez-Buylla, E., Chaos, A., Kauffman, S., Shmulevich, I., Aldana, M.: Critical dynamics in genetic regulatory networks: Examples from four kingdoms, `http://www.fis.unam.mx/~max/English/paperplos-one-1.pdf`
8. Bastola, U., Parisi, G.: The critical line of Kauffman networks. J. Theor. Biol. 187, 117 (1997)
9. Baumann, R.: Soft errors in advanced comuter systems. IEEE Design and Test of Comuters 22(3), 258–266 (2005)
10. Bryant, R.: Graph-based algorithms for Boolean function manipulation. Transactions on Computer-Aided Design of Integrated Circuits and Systems 35, 677–691 (1986)
11. Camazine, S., Deneubourg, J.-L., Franks, N.R., Sneyd, J., Theraulaz, G., Bonabeau, E.: Self-Organization in Biological Systems. Princeton University Press, Princeton (2003)
12. Chaos, A., Aldana, M., Espinosa-Soto, C., de Leon, B.G.P., Arroyo, A.G., Alvarez-Buylla, E.R.: From genes to flower patterns and evolution: Dynamic models of gene regulatory networks. Journal of Plant Growth Regulation 25(4), 278–289 (2006)
13. Chaves, M., Sontag, E.D., Albert, R.: Methods of robustness analysis for Boolean models of gene control networks. IEE Proceedings of Systems Biology 153, 154–167 (2006)
14. Dubrova, E., Teslenko, M.: Compositional properties of Random Boolean Networks. Physical Review E 71, 056116 (May 2005)

15. Dubrova, E., Teslenko, M., Martinelli, A.: Kauffman networks: Analysis and applications. In: Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, pp. 479–484 (November 2005)
16. Dubrova, E., Teslenko, M., Tenhunen, H.: A computational model based on random Boolean networks. In: BIONETICS 2007 (2007)
17. Espinosa-Soto, C., Padilla-Longoria, P., Alvarez-Buylla, E.R.: A gene regulatory network model for cell-fate determination during arabidopsis thaliana flower development that is robust and recovers experimental gene expression profiles. The Plant Cell 16, 2923–2939 (2004)
18. Flyvbjerg, H., Kjaer, N.J.: Exact solution of Kauffman model with connectivity one. J. Phys. A: Math. Gen. 21, 1695 (1988)
19. Gershenson, C.: Updating schemes in random Boolean networks: Do they really matter? In: Proceedings of the Ninth International Conference on the Simulation and Syuntheis of Living Systems, pp. 238–243 (2004)
20. Glass, L., Hill, C.: Ordered and disordered dynamics in random networks. Europhysics Letter 12, 599–604 (1998)
21. Huang, S., Eichler, G., Ber-Yam, Y., Ingber, D.E.: Cell fates as high-dimensional sttractor states of a complex gene regulatory network. Physical Rev. Let. 94, 128701–128704 (2005)
22. Huang, S., Ingber, D.E.: Shape-dependent control of cell growth, differentiation, and apoptosis: Switching between attractors in cell regulatory networks. Experimental Cell Research 261, 91–103 (2000)
23. Jacob, F., Monod, J.: Genetic regulatory mechanisms in the synthesis of proteins. Journal of Molecular Biology 3, 318–356 (1961)
24. Kauffman, S.A.: Metabolic stability and epigenesis in randomly constructed nets. Journal of Theoretical Biology 22, 437–467 (1969)
25. Kauffman, S.A.: The Origins of Order: Self-Organization and Selection of Evolution. Oxford University Press, Oxford (1993)
26. Kauffman, S.A.: Investigations. Oxford University Press, Oxford (2002)
27. Luque, B., Sole, R.V.: Stable core and chaos control in random Boolean networks. Journal of Physics A: Mathematical and General 31, 1533–1537 (1998)
28. Meindl, J.D., Chen, Q., Davis, J.A.: Limits on silicon nanoelectronics for terascale integration. Science 293, 2044–2049 (2001)
29. Mendoza, L., Alvarez-Buylla, E.R.: Genetic regulation of root hair development in Arabidopsis thaliana: A network model. J. Theor. Biol. 204, 311–326 (2000)
30. Nykter, M., Price, N.D., Aldana, M., Ramsey, S.A., Kauffman, S.A., Hood, L.E., Yli-Harja, O., Shmulevich, I.: Gene expression dynamics in the macrophage exhibit criticality. Proc. Natl. Acad. Sci. U.S.A. 105(6), 1897–1900 (2008)
31. Schlitt, T., Brazma, A.: Current approaches to gene regulatory network modelling. BMC Bioinformatics 6, S9 (2007)
32. Thomas, R., Thieffry, D., Kaufman, M.: Dynamical behaviour of biological regulatory networks I. Biological role of feedback loops and practical use of the concept of the loop-characteristic state. Bull. Math. Biol. 57, 247–276 (1995)
33. von Dassow, G., Meir, E., Munro, E.M., Odell, G.M.: The segment polarity network is a robust developmental module. Nature 406, 188–193 (2000)
34. Wagner, A.: Robustness and evolvability in living systems. Oxford University Press, Oxford (2007)
35. Weiser, M.: Some computer science problems in ubiquitous computing. Communications of the ACM 36(7), 74–83 (1993)

Appendix: This appendix presents defining tables for Boolean functions associated to vertices of the Boolean network in Figure 2. The sign "-" stands for any value, 0 or 1. In each table, only the assignments of variables for which the function evaluates to 1 are shown; for all remaining assignments the function evaluates to 0. The function of LUG and CLF is the constant 1.

UFO	UFO
1	1

AP3	FT
0	1

TFL1	AP2
0	1

LFY	EMF1
0	1

LFY	SEP
1	1

AP1	TFL1	FUL
0	0	1

AP1	EMF1	LFY	AP2	TFL1
0	1	0	-	1

WUS	AG	SEP	WUS
1	0	-	1
1	-	0	1

AP1	LFY	AG	PI	SEP	AP3	UFO	AP3
1	-	-	1	1	1	-	1
-	-	1	1	1	1	-	1
-	1	-	-	-	-	1	1

FT	LFY	AG	TFL1	AP1
1	-	0	-	1
-	1	0	-	1
-	-	0	0	1

FUL	AP1	EMF1	TFL1	LFY
1	-	-	0	1
-	1	-	0	1
-	-	0	-	1

AP1	LFY	AG	PI	SEP	AP3	PI
-	0	1	1	1	1	1
-	1	0	-	-	1	1
-	1	1	-	-	-	1
1	0	-	1	1	1	1

AP1	LFY	AP2	WUS	AG	LUG	CLF	TFL1	SEP	AG
-	-	0	-	-	-	-	0	-	1
-	1	-	-	1	-	-	-	1	1
-	1	-	-	-	-	0	-	-	1
-	1	-	-	-	0	-	-	-	1
0	1	-	-	-	-	-	-	-	1
-	1	-	1	-	-	-	-	-	1
-	1	0	-	-	-	-	-	-	1

On Autonomy and Emergence in Self-Organizing Systems[⋆]

Richard Holzer[1], Hermann de Meer[1], and Christian Bettstetter[2]

[1] Faculty of Informatics and Mathematics, University of Passau, Innstrasse 43, 94032 Passau, Germany
{holzer,demeer}@fim.uni-passau.de,
[2] Institute of Networked and Embedded Systems, Mobile Systems Group, University of Klagenfurt, Lakeside B02, 9020 Klagenfurt, Austria
christian.bettstetter@uni-klu.ac.at

Abstract. For analyzing properties of complex systems, a mathematical model for these systems is useful. In this paper we describe how discrete complex systems can be modeled mathematically and we give a framework for the analysis of the system with respect to the properties autonomy and emergence, which are two of the most important properties of self-organizing systems. The modeling is done by using a multigraph to describe the connections between objects and stochastic automatons for the behavior of the objects.

Keywords: Self-Organziation, Autonomy, Emergence, Mathematical modeling, Systems.

1 Introduction

A design goal for future computing and networking systems is to limit administrative requirements for both users and operators. Technical systems should be easy to use and configure themselves as much as possible. They should auto-detect failures and auto-correct them if possible. Another trend in networked systems is to become more and more independent of centralized servers and control instances, but to be able to distribute the network control and data among the entities of a network.

Having these goals in mind, computing researchers have become very much interested in the interdisciplinary topic of "self-organization" during the past few years. Much research has been done, for instance, on the design of fixed peer-to-peer networks, mobile ad hoc networks, and principles for autonomic computing.

Despite these technological advances, a systematic and profound research on the fundamental concepts and principles as how to design and exploit self-organization in networked computing systems did not take place so far. In particular, there is no common understanding on the building blocks and the design

⋆ The presented work has been supported by the EU projects Euro-NF (NoE, FP7, ICT-2007-1-216366) and AutoI (STREP, FP7, ICT-2007-1-216404).

process of self-organizing networked systems, and it is unclear how frequently mentioned properties of self-organizing systems, such as *autonomous behavior* and *emergent behavior* are formally defined.

The goal of this paper is to contribute to this issue. Using the notion of entropy known from information theory, we give a formal definition of the level of autonomy and emergence in a networked system. We also apply this model to a practical example, namely the self-organized slot-synchronization in wireless networks.

In this paper, Section 2 gives an overview of the related work and Section 3 gives a mathematical model for complex systems. Section 4 defines the concept of autonomy in the model and Section 5 defines the concept of emergence in the model. In Section 6, we apply our definitions to model slot synchronization in wireless networks. Section 7 contains some discussions, applications and advanced conclusions of the formal model described in this paper.

2 Related Work

In the recent years, much work has been done in the field of self-organizing systems, but self-organization has no generally accepted meaning. According to [1] and [2] typical features (among others) of self-organization are autonomy, emergence, adaptivity, decentralization, self-maintenance and optimization.

For a non-technical overview of self-organization see [2]. Other definitions and properties of self-organizing systems can be found in thermodynamics [3], synergetics [4], information theory [5] and cybernetics [6], [7], [8]. A comprehensive description about the design of self-organizing systems is in [9]. A good overview about modeling complex systems can be found in [10]. For modeling continuous self-organizing systems and a comparison between discrete and continuous modeling see [11]. [12] gives a survey about practical applications of self-organization.

One important concept of the information theory for measuring self-organization is the *statistical entropy* (see [2], [13], [5], [9]). Self-organization implies the decrease of the statistical entropy of the system. The entropy is also used in this paper for the definition of autonomy and emergence.

For the modeling of a system in Section 3, we follow the basic ideas of [2]: The model has a state space and the evolution of the system is the transition between the different states. The main difference between the definitions in Section 3 and [2] is the granularity of the view on the system: The concept *state* in [2] is seen as a global state of the system and also the transition matrix of the Markov chain contains the probabilities in this global view of the system. In Section 3 of this paper, each node of the system has its own stochastic automaton with its own probability distribution. This local view is important to be able to formally define properties like the emergence (see Section 5), because in the global view it would not be possible to define the dependencies in different components of the system in the current state. Also the differentiation between the information inside the nodes and the information between different nodes is important for the analysis of properties like autonomy. While in [2] the information between different nodes is just a part of the global state, we seperate this information from the local states of the objects to be able to measure the information flow between the nodes. For the formal definition of the concept *autonomy* we also need to distinguish user data (data from the environment that is processed by the system) and control data (data from the environment to change the behavior of the system). Another difference to [2] is the clock of each node: Here we do not need a global clock for the whole system, but each node has its own clock.

3 Discrete Systems

In this section we give a mathematical definition for modeling discrete systems. The purpose of the definitions of this section is to be able to describe mathematically a wide variety of complex systems of the real world. One major demand for the right definition is that the behavior of each object is affected by the interaction with other objects. Another demand is that not only deterministic systems should be considered, but also uncertainty and nondeterminism should be possible in the model. In the real world, not all properties are known in all detail (e.g. it would be very difficult to describe a deterministic behavior of an animal), but there are many things, that can better be described by probabilities. In this section, we use directed multigraphs to model interaction between objects: Each vertex in the multigraph corresponds to an object and each edge of the multigraph is used to model the interaction (e.g. transfer of data) between the objects. To incorporate the external influence of the environment we use special vertices in the multigraph, where the edges from these vertices represent the channels for the input into the system, and the edges to these vertices represent the output of the system. To measure the autonomy of the system, we distinguish between user data (data from the environment that is processed by the system) and control data (data from the environment to change the behavior of the system). The behavior of the objects is modeled by finite stochastic automatons.

Definition 1. *A* discrete system $\mathcal{S} = (G, E, C, A, a)$ *consists of*

- *a finite directed multigraph* $G = (V, K, \tau)$, *where* V *is the set of vertices,* K *is the set of directed edges (loops are also allowed), and* $\tau : K \to V^2$ *assigns to each edge* $k \in K$ *the corresponding vertices* $\tau(k)$, *where the starting vertex is also denoted by* $k-$ *and the ending vertex is denoted by* $k+$. *Therefore we have* $\tau(k) = (k-, k+)$ *for each* $k \in K$. *For a vertex* $v \in V$ *the set of edges ending in* v *is denoted by* $v- := \{k \in K \mid k+ = v\}$ *and the set of edges starting at* v *is denoted by* $v+ := \{k \in K \mid k- = v\}$. *Analogously for* $T \subseteq V$ *the sets* $T-$ *and* $T+$ *are defined by* $T- := \bigcup_{v \in T} v- = \{k \in K \mid k+ \in T\}$ *and* $T+ := \bigcup_{v \in T} v+ = \{k \in K \mid k- \in T\}$.
- *a subset of vertices* $E \subseteq V$, *which elements are called* external nodes. *The other vertices are called* internal nodes. *The* input edges *are denoted by* $I = E+ = \{k \in K \mid k- \in E\}$. *The* output edges *are denoted by* $O = E- = \{k \in K \mid k+ \in E\}$. *All other edges are called* internal edges.

– *a subset of the input edges $C \subseteq I$. The elements of C are called* control edges*. The other input edges $U := I \setminus C$ are called* user edges*.*
– *a finite set A, which is used as alphabet for communication between the nodes.*
– *a family $a = (a_v)_{v \in V}$ of stochastic automatons $a_v = (A^{v-}, A^{v+}, S_v, P_v, d_v)$, where*
  - $A^{v-} = \{(x_k)_{k \in v-} \mid x_k \in A, k \in v-\}$ *are the* local input values*,*
  - $A^{v+} = \{(x_k)_{k \in v+} \mid x_k \in A, k \in v+\}$ *are the* local output values*,*
  - *S_v is the set of states,*
  - *$P_v : S_v \times A^{v-} \times S_v \times A^{v+} \to [0,1]$ is a function, such that $P(q, x, \cdot, \cdot) : S_v \times A^{v+} \to [0,1]$ is a probability distribution on $S_v \times A^{v+}$ for each $q \in S_v$ and $x \in A^{v-}$. The value $P(q, x, q', y)$ is the probability, that the automaton moves from state $q \in S_v$ into the new state $q' \in S_v$ and gives the local output $y \in A^{v+}$ when it receives the local input $x \in A^{v-}$.*
  - *$d_v : S_v \to \mathbb{R}^+$ is a map, where $d_v(s)$ describes the delay between two pulses of the clock when the automaton is in state $s \in S_v$ (there is no global clock, but each automaton has its own clock).*

Let us consider the example in Figure 1: A computer network is used by some people for collaboration and to build a shared work space. An administrator controls the access to the network and the options for the groupware. In the corresponding model of Definition 1, the users and the administrator are represented by external nodes and the computers in the network are represented by internal nodes. Since the administrator can control the behavior of the system, we have one control edge. The other external edges are user edges. The internal nodes represent computers, so the automatons in the model of Definition 1 are

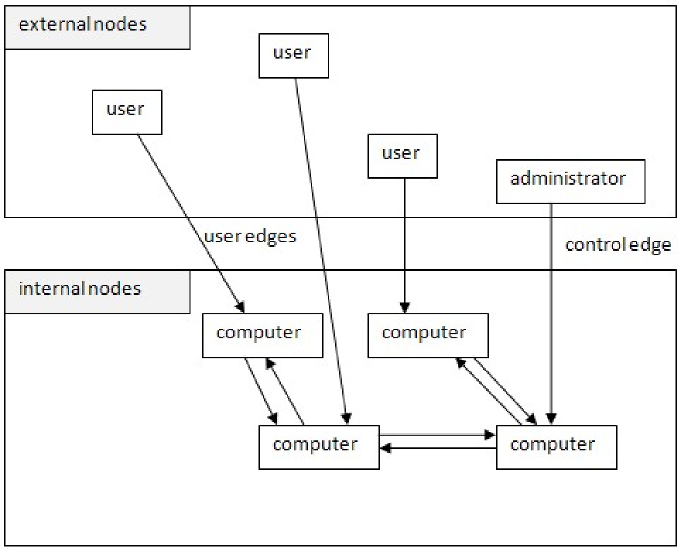


Fig. 1. Example

deterministic for these nodes. Nondeterminism can be used to model the behavior of the users and of the administrator. For the clocks in the automatons we can choose $d_v(s) = 1$ for each vertex $v \in V$ and each state $s \in S_v$, so the points of time, where events occur, are natural numbers. At each point of time $n \in \mathbb{N}$, the internal nodes receive new data from the external nodes, change their internal state and send data to other nodes.

For analyzing changes in the system states, we now define the concept of *configurations*, which can be seen as a snapshot of the system at a given point of time.

Definition 2. *Let $\mathcal{S}$ be a system. A* configuration *$c = (r, c_V, c_K, D)$ consists of*

- *a point of time $r \in \mathbb{R}_0^+$,*
- *a tuple $c_V \in \prod_{v \in V} S_v$ of states, which defines the current states of the automatons,*
- *a map $c_K : K \to A$, which defines the current symbols on the communication channels,*
- *a map $D : V \to \mathbb{R}^+$, where $D(v) \le d_v(c_V(v))$ describes the length of the time interval between r and the next pulse of the clock in the automaton a_v.*

For a configuration $c = (r, c_V, c_K, D)$ and a set $T \subseteq K$ of edges the assignment $c_K|_T : T \to A$ of the edges in T is also denoted by $c|_T$. The configuration $c = (r, c_V, c_K, D)$ is called start configuration, *if $r = 0$ and $D(v) = d_v(c_V(v))$ for $v \in V$. An* initialization *of $\mathcal{S}$ is a pair (Γ, P_Γ), where Γ is a set of start configurations and $P_\Gamma : \Gamma \to [0,1]$ is a probability distribution on Γ, which describes, with which probability the system starts in a certain configuration $c \in \Gamma$. For a configuration $c = (r, c_V, c_K, D)$ the* duration of c *is defined by $d_c = \min\{D(v) \mid v \in V\}$. Let $N_c = \{v \in V \mid D(v) = d_c\}$ be the set of nodes with the soonest clock pulse after the point of time r. A configuration $c' = (r', c'_V, c'_K, D')$ is a* successor configuration *of $c = (r, c_V, c_K, D)$ with probability $p \in [0,1]$ (notation: $c \to_p c'$ or $P(c \to c') = p$) if*

- *$r' = r + d_c$,*
- *$c'_V(v) = c_V(v)$ for $v \in V \setminus N_c$,*
- *$c'_K(k) = c_K(k)$ for $k \in K$ with $k- \in V \setminus N_c$,*
- *$D'(v) = D(v) - d_c$ for $v \in V \setminus N_c$,*
- *$D'(v) = d_v(c'_V(v))$ for $v \in N_c$,*
- *$p = \prod_{v \in N_c} P_v(c_V(v), (c_K(k))_{k \in v-}, c'_V(v), (c'_K(k))_{k \in v+})$.*

A (finite or infinite) tuple $t = (c_0, c_1, c_2, \ldots)$ of configurations is called configuration sequence *if for each $j \ge 0$ there exists $p > 0$ with $c_j \to_p c_{j+1}$. If there exists a configuration sequence $(c_0, c_1, c_2, \ldots, c_j)$ from $c = c_0$ to $c' = c_j$, we also write $c \to^* c'$ to indicate that c' is reachable from c. For an initialization (Γ, P_Γ) we also write $\Gamma \to^* c'$ to indicate that c' is reachable from Γ, i.e. there exists a configuration $c \in \Gamma$ with $P_\Gamma(c) > 0$. For a configuration c let $succ(c)$ be a random variable with the probability distribution $P(succ(c) = c') = P(c \to c')$ for each successor configuration c' of c. For an initialization (Γ, P_Γ) of the system and a* *configuration c let $P(\Gamma \to^* c)$ be the probability that c is reached from Γ. For $t \geq 0$ define*

$$\Gamma_t = \{c \mid \Gamma \to^* c = (r, c_V, c_K, D) \text{ with } r \leq t < r + D(v) \text{ for } v \in V\},$$

i.e. Γ_t is the set of all configurations c that may be active at time t. For $t \geq 0$ let Conf_t *be the random variable taking values in Γ_t with the probability distribution $P(\mathrm{Conf}_t = c) = P(\Gamma \to^* c)$ for $c \in \Gamma_t$.*

For the example of Figure 1, a configuration (r, c_V, c_K, D) is a snapshot of the whole network. All information (point of time, state of each computer, state of each user, state of the administrator, values on the edges, duration until next clock pulse) is contained in the current configuration. The successor configuration $c' = succ(c)$ contains all these information after the next step. If we consider the behavior of the users as nondeterministic, the successor configuration is not unique.

In the following sections, we use the concept of configuration to get a formal definition of autonomy and emergence.

4 Autonomy

To compute the level of autonomy, we compare the information contained in the control data with the information of the whole system. The information of data can be quantified by the entropy (see [14], [15]). The entropy can be seen as a measure of uncertainty: a high entropy for the system means, that we have nearly no information in advance about a configuration, while a low entropy means, that nearly all information of the configuration is known in advance. For the entropy we use the logarithm for the base 2, so it is the average number of bits, which are needed for an optimal encoding of the information.

Definition 3. *For a discrete random variable X taking values from a set W the* entropy $H(X)$ of X *is defined by [14]*

$$H(X) = -\sum_{w \in W} P(X = w) \log_2 P(X = w).$$

For $t \geq 0$, the value $H(\mathrm{Conf}_t)$ measures the *system entropy* at time t, i.e. $H(\mathrm{Conf}_t)$ is the average number of bits that are needed to encode the information of the configuration at time t in an optimal way. By restricting the configuration to a set of edges, we can analogously measure the information of the values on these edges. For example the *control entropy* $H(\mathrm{Conf}_t\,|_C)$ is the average number of bits that are needed to encode the control information.

Definition 4. *Let $\mathcal{S}$ be a system and (Γ, P_Γ) be an initialization. For a configuration c the* level of autonomy of c *is defined by*[1]

$$\alpha(c) = 1 - \frac{H(succ(c)|_C)}{H(succ(c)|_K)}.$$

[1] If $H(succ(c)|_K) = 0$ then we define $\frac{0}{0} := 0$ and $\alpha(c) = 1$.

For a point of time $t \geq 0$, *the* level of autonomy at time t *is defined by the weighted mean value of all these autonomy levels of configurations, which may be active at time* t, *i.e.*[2]

$$\alpha_t(\mathcal{S}, \Gamma) = \sum \{P(\Gamma \to^* c)\alpha(c) \mid c \in \Gamma_t\}.$$

For points of time $s > r \geq 0$ *the* level of autonomy of the interval $[r, s]$ *is defined by the mean value of all these autonomy levels in the time interval* $[r, s]$, *i.e.* $\alpha_{[r,s]}(\mathcal{S}, \Gamma) = \frac{1}{s-r} \int\limits_r^s \alpha_t(\mathcal{S}, \Gamma)dt$. *The* level of autonomy of the system $\mathcal{S}$ *is defined by* $\alpha(\mathcal{S}, \Gamma) = \liminf\limits_{t \to \infty} \alpha_{[0,t]}(\mathcal{S}, \Gamma)$.

In this definition the value $\frac{H(succ(c)|_C)}{H(succ(c)|_K)}$ describes the relation between the entropy on the control edges and the entropy on all edges: A high value for this ratio means, that much control data is needed in the configuration c, and a low value for this ratio means, that the system behaves nearly autonomously in the configuration c. Therefore $\alpha(\mathcal{S}, \Gamma)$ measures how much control data is needed relative to the data on all edges during the whole run of the system. Since $\alpha(\mathcal{S}, \Gamma) \in [0, 1]$, a level of autonomy with $\alpha(\mathcal{S}, \Gamma) \approx 1$ means, that the information contained in the control data will be very low, i.e. the system will behaves autonomously if we wait long enough. A level with $\alpha(\mathcal{S}, \Gamma) \approx 0$ means, that it needs much control data to keep the system running, i.e. the system will not be autonomous even if we wait a very long time. Therefore we use the following definition of autonomous systems:

Definition 5. *A system* $\mathcal{S}$ *is called autonomous with respect to an initialization* (Γ, P_Γ), *if* $\alpha(\mathcal{S}, \Gamma) = 1$.

Note that the usage of lim inf in Definition 4 does not imply the convergence of information of a sample path, it is just the smallest accumulation point of the mean value $\alpha_{[0,t]}(\mathcal{S}, \Gamma)$ when t goes to infinity. For example if the level of autonomy $\alpha_t(\mathcal{S}, \Gamma)$ fluctuates uniformly between 0 and 1, then the mean value $\alpha_{[0,t]}(\mathcal{S}, \Gamma)$ will approximate $\frac{1}{2}$, so $\alpha(\mathcal{S}, \Gamma) = \frac{1}{2}$.

Concerning the example in Figure 1, the autonomy depends on the behavior of the administrator. If the administrator only initializes the software and creates the user accounts, then the users can work with the groupware without further input of the administrator, so in this case the system is autonomous. If the administrator also has to solve some problems of the users, change some options of the software or create a new user account when a new user joins the working group, then we have $\alpha(\mathcal{S}, \Gamma) < 1$, but since the entropy of all edges of the network is usually much larger than the entropy for the input of the administrator, the system is nearly autonomous.

[2] When we write a set after the sum symbol $\sum$, it should be considered as a multiset, i.e. in the following formula a value is added twice if it is contained twice in the multiset.

5 Emergence

In some systems, it may happen, that some patterns or properties appear in the system as a whole, but do not appear in the single components. Such an appearance is called emergence. This leads to the following definition:

Definition 6. *Let $\mathcal{S}$ be a system and (Γ, P_Γ) be an initialization. For a point of time $t \geq 0$ the* level of emergence at time t *is defined by*

$$\varepsilon_t(\mathcal{S}, \Gamma) = 1 - \frac{H(\mathrm{Conf}_t\,|_K)}{\sum\limits_{k \in K} H(\mathrm{Conf}_t\,|_{\{k\}})}.$$

For points of time $s > r \geq 0$ the level of emergence of the interval $[r, s]$ *is defined by the mean value of all these emergence levels in the time interval $[r, s]$, i.e. $\varepsilon_{[r,s]}(\mathcal{S}, \Gamma) = \frac{1}{s-r} \int\limits_r^s \varepsilon_t(\mathcal{S}, \Gamma) dt$. The* level of emergence of the system $\mathcal{S}$ *is defined by $\varepsilon(\mathcal{S}, \Gamma) = \liminf\limits_{t \to \infty} \varepsilon_{[0,t]}(\mathcal{S}, \Gamma)$.*

For the level of emergence, the information of all edges is compared to the information contained in each single edge. Analogously to the level of autonomy, also the level of emergence is a value in the interval [0, 1]. If at the current point of time $t \geq 0$ there are large dependencies between the values on the single edges (which can be seen as patterns), the level of emergence is high: $\varepsilon_t(\mathcal{S}, \Gamma) \approx 1$. If the values of nearly all edges are independent, there will be no pattern, so the level of emergence is low: $\varepsilon_t(\mathcal{S}, \Gamma) \approx 0$. Therefore $\varepsilon(\mathcal{S}, \Gamma)$ measures the dependencies occurring during the whole run of the system.

Note that in literature the definition of emergence is not unique. There also exist some work (see e.g. [16]), where emergence is defined as an unexpected decrease in relative algorithmic complexity. Here we consider also expected dependency as an emergence.

Concerning the example in Figure 1, the entropy depends on the collaboration: A high collaboration between the users imply more dependencies in the system than without collaboration, so the level of emergence is in this example an indicator for the collaboration.

6 Example: Synchronization in Wireless Networks

In this section we apply the definitions for autonomy and emergence to the process of self-organized slot-synchronization in wireless networks [17]. We consider a set of nodes, each node being able to communicate with some of the other nodes. The access on the shared medium is organized in time slots. Since there is no central clock, which defines when a slot begins, the nodes perform a slot synchronization in a completely distributed manner. An algorithm for this purpose is proposed by Tyrrell, Auer, and Bettstetter in [17]. It is based on the model of pulse-coupled oscillators by Mirollo and Strogatz [18].

In the latter synchronization model, the clock is described by a phase function ϕ which starts at time instant 0 and increases over time until it reaches a threshold value $\phi_{th} = 1$. The node then sends a "firing pulse" to its neighbors for synchronization. Each time a node receives such a pulse from a neighbor, it adjusts its own phase function by adding $\Delta\phi := (\alpha - 1)\phi + \beta$ to ϕ, where $\alpha > 1$ and $\beta > 0$ are constants.

The system can be formally described as follows. In the system $\mathcal{S}_{MS} = (G, E, C, A, a)$, the graph G describes the connections between the nodes, i.e. a node $v \in V$ is linked by an edge to another node $w \in V$ if a direct communication from v to w is possible. We have no external nodes, so $E = \emptyset = C$. For the alphabet A we only need the values $A = \{0, 1\}$, where 0 means "not firing" and 1 means "firing". Each automaton $a_v = (A^{v-}, A^{v+}, S_v, P_v, d_v)$ for $v \in V$ describes the behavior of the node v.

Let us first assume the duration $d_v(s)$ of one step to be constant for all $v \in V$ and $s \in S_v$. The values of the phase function ϕ can be stored in the current state of the automaton a_v. Let T be the duration of an uncoupled period (i.e. if no firing of neighbors are received). In each step the current value of the phase function ϕ is increased by the constant value $\delta = \frac{d_v}{T}$. If some neighbors of v are firing, further increments of ϕ have to be done. We use S_v as a discrete subset of the real interval $[0, 1]$ (note that only finitely many values can be reached by ϕ, because the time is discrete). For each state $\phi \in S_v$ and each local input $x = (x_w)_{w \in v-}$ the automaton a_v calculates the next value of ϕ as follows: Let $\phi' := \phi + \delta + \Delta\phi \cdot \sum_{w \in v-} x_w$ with $\Delta\phi = (\alpha - 1)\phi + \beta$. If $\phi' < 1$ then ϕ' is the new state of a_v and the local output for every neighbor is 0, otherwise the new state is 0 and the local output is 1. So we have a deterministic automaton $a_v = (A^{v-}, A^{v+}, S_v, P_v, d_v)$, where $P_v(\phi, x, \phi', (0)_{w \in v+}) = 1$ for $\phi' < 1$ and $P_v(\phi, x, 0, (1)_{w \in v+}) = 1$ for $\phi' \geq 1$.

In [17] the pulse-coupled oscillator synchronization model is adapted to wireless systems, where also delays (e.g., transmission delay, decoding delay) are considered. The duration of an uncoupled period is now $2T$ with $T > 0$. This period is divided into four states (see Figure 2). Let $\gamma \in [0, 2T]$ be a time instant. Then the node is in a

– waiting state, if $\gamma \in [0, T_{wait}) =: I_{wait}$,
– transmission state, if $\gamma \in [T_{wait}, T_{wait} + T_{Tx}) =: I_{Tx}$,
– refractory state, if $\gamma \in [T_{wait} + T_{Tx}, T_{wait} + T_{Tx} + T_{refr}) =: I_{refr}$,
– listening state, if $\gamma \in [T_{wait} + T_{Tx} + T_{refr}, 2T) =: I_{Rx}$,

where the constants are defined as follows:

– T_{Tx}: Delay for transmitting a value,
– T_{wait}: Waiting delay after the phase function reached the threshold. The transmission of the firing pulse begins after this delay. The waiting delay is calculated by $T_{wait} = T - (T_{Tx} + T_{dec})$, where T_{dec} is the delay for decoding the received value.
– T_{refr}: Refractory delay after the transmission of the firing pulse to avoid an unstable behavior.

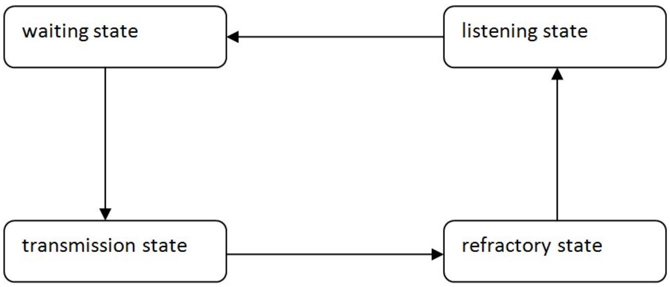


Fig. 2. State diagram

Let $T_{Rx} = 2T - T_{wait} - T_{Tx} - T_{refr}$ be the duration of an uncoupled listening state. We assume that each of these durations $T_{wait}, T_{Tx}, T_{refr}, T_{Rx}$ is less than T. The listening state is the only state, in which firing pulses from the neighbors can be received and decoded, and the phase function is changed only during the listening state.

Now we describe this network as a discrete system $\mathcal{S} = (G, E, C, A, a)$. For the alphabet we use again the values 0 (not fired) and 1 (fired). The states S_v of the automaton $a_v = (A^{v-}, A^{v+}, S_v, P_v, d_v)$ for $v \in V$ can be used to store the current value of the phase function, information about the current position during period $2T$, and information about received firing pulses. We assume $d_v < \min(T_{Tx}, T_{dec}, T_{wait}, T_{refr})$ and that the value $d_v(s)$ is constant for each state $s \in S_v$, but it needs not be constant for each node $v \in V$. A state $s \in S_v$ is a triple $s = (\phi, \gamma, D)$, where $\phi \in [0, 1]$ is the current value of the phase function, $\gamma \in [0, 2T]$ is the time elapsed since the last firing and $D = (D_w)_{w \in v-}$ are the decoding delays for the received firing pulses of predecessor nodes, i.e. during the transmission of the firing pulse from a predecessor $w \in v-$ the value D_w is initialized by T_{dec}, and after the end of the transmission, D_w is decreased in each step until D_w reaches 0. Then the phase function is adjusted by adding $\Delta\phi$ to ϕ like above. If no firing pulse is received, then the phase function is increased by $\delta = \frac{d_v}{T_{Rx}}$ in each step until ϕ reaches the threshold $\phi_{th} = 1$. We get a deterministic automaton a_v with $P_v((\phi, \gamma, D), x, (\phi', \gamma', D'), z) = 1$ for the following cases:

For $\gamma \in I_{wait}$**:**	**For** $\gamma \in I_{Rx}$**:**
$\phi' = 0$	$\phi' = \phi + \delta + \Delta\phi \cdot \lvert\{w \in v- \;:\; 0 < D_w \leq d_v\}\rvert$ with $\delta = \frac{d_v}{T_{Rx}}$
$\gamma' = \gamma + d_v$	$\gamma' = \begin{cases} \gamma + d_v & \text{for } \phi' < 1 \\ 0 & \text{for } \phi' \geq 1 \end{cases}$
$D' = (0)_{w \in v-}$	$D'_w = \begin{cases} T_{dec} & \text{for } x_w = 1 \\ \max(0, D_w - d_v) & \text{for } x_w = 0 \end{cases}$
$z = (0)_{v \in v+}$	$z = (0)_{v \in v+}$

For $\gamma \in I_{Tx}$**:**	**For** $\gamma \in I_{refr}$**:**
$\phi' = 0$	$\phi' = 0$
$\gamma' = \gamma + d_v$	$\gamma' = \gamma + d_v$
$D' = (0)_{w \in v-}$	$D' = (0)_{w \in v-}$
$z = (1)_{w \in v+}$	$z = (0)_{w \in v+}$

Starting at $\gamma = 0$, the automaton a_v goes through the different states: Waiting state, transmission state, refractory state and listening state. Note that the duration of one period usually is less than $2T$, because each firing pulse of a predecessor node, which is received and decoded during the listening state, shortens the period by increasing the phase function ϕ.

Simulation results in [17] show that during the run of the system, groups of synchronizations are built, i.e. inside each group we have good synchronization (each node of the group fires at nearly the same time like the other nodes of the group), and if we wait long enough, then there are only two groups left firing T time units apart from each other. For the system $\mathcal{S}_{MS}$ it can be shown (see [17], [18]), that the system always converges to a state, in which all nodes fire at the same time.

We now apply our definitions of the previous sections. Both synchronization systems $\mathcal{S}_{MS}$ and $\mathcal{S}$ are autonomous because there are no external nodes. According to the emergence, we note that the behavior of the automatons is deterministic. Let Γ be a set of start configurations with a given distribution (e.g. random uniform distribution). In both systems $\mathcal{S}_{MS}$ and $\mathcal{S}$ we get high dependencies between the values on the edges if we wait for a long time, so we have a high emergence: $\varepsilon(\mathcal{S}, \Gamma) \approx 1$ and $\varepsilon(\mathcal{S}_{MS}, \Gamma) \approx 1$.

7 Discussion of the Results

The main result of this paper is, that we get a formalism to analyze complex systems with respect to emergence and autonomy. While the previous work in literature does not define, how to compute the level of emergence and the level of autonomy of a given system, this paper gives formal definitions of these concepts.

These definitions can be applied as follows: Assume that we have a real world system and we would like to analyze this system with respect to self-organizing properties. Then we first create the model with the definitions of Section 3: We use a multigraph to model the objects in the real world system and we use stochastic automatons for modeling the behavior of the objects. In this model we can apply the definitions of Section 4 and 5. With Definition 4 we get the level of autonomy $\alpha(\mathcal{S}, \Gamma)$ of the modeled system. A large value for this level means, that the system is nearly autonomous and a low value for this level means, that much data is needed from the environment to control the system. With Definition 6 we get the level of emergence of the system. A large value for this level means, that global patterns appear in the system in form of dependencies in the communication between the nodes and a low value for this level means, that there are nearly no global patterns. Since autonomy and emergence are two of the main properties of self-organization, these levels are also indicators for the degree of self-organization: A self-organizing system cannot have a low level of autonomy or a low level of emergence.

The examples in this paper show, that even if the system is too large to compute the levels of autonomy and emergence exactly, the definitions can still be useful to get an approximation of these levels.

Also for the design of a new system the definitions might be useful, since they can give hints, how to design better self-organizing systems. For example if we see, that we need some data to control the behavior of some nodes, Definition 4 tells us, that we should try to produce this control data in an internal node of the system, since the level of autonomy would increase in this case. According to the definition of the level of emergence, the situation might be more difficult: When we design a new system, each object of the system is designed locally, but emergence is a global property which usually cannot be seen directly from the local behavior of each object. But Definition 6 still might help us for the design of the system: During the design, we could calculate the level of emergence for different variants of the system (e.g. by changing some system parameters) and from the results we see which variant gives us a high emergence in the system.

The formalism of this paper could be applied to many self-organizing applications [12]:

– Multi-agent systems: A multi-agent system could consist of some robots or some software applications. Each agent has some rules for defining its behavior and different agents can communicate with each other (e.g. for cooperation) and with the environment. In such a multi-agent system, Definition 4 can be used to compute the level of autonomy. Obviously, a high level of autonomy is desirable, because then the agents need nearly no control from the users, so the users are released from work. The level of emergence is an indicator for the cooperation between the agents. This could be synchronization like in Section 6 or any other global pattern, which is induced by the local interactions between the agents.
– Network security: Mobile agents can be used to build intrusion detection systems and intrusion response systems. These agents can sense network conditions and they are able to load dynamically new functionality into other network nodes. They can work autonomously without a centralized control. In case of an intrusion, the intrusion response systems will be loaded into the nodes for providing appropriate response. This can be seen as an emergent behavior of the system: There are many dependencies in the communications between the nodes, so we have a high level of emergence.

In [12] there are also many other applications mentioned, where the definitions of this paper could be applied for the analysis of self-organizing properties: Sensor networks, web communities, robots, business process infrastructures and many more.

8 Conclusion

In this paper we described how discrete complex systems can be modeled by multigraphs with stochastic automatons. For the two main properties of self-organizing systems, a mathematical definition has been given:

– autonomy
– emergence

For each system, the level of autonomy indicates, how much external data is needed to control the system and the level of emergence indicates, how many patterns in form of dependencies between the transmitted data between the nodes appear.

References

1. De Meer, H., Koppen, C.: Characterization of self-organization. In: Steinmetz, R., Wehrle, K. (eds.) Peer-to-Peer Systems and Applications. LNCS, vol. 3485, pp. 227–246. Springer, Heidelberg (2005)
2. Heylighen, F.P.: The science of self-organization and adaptivity. In: Kiel, L.D. (ed.) Knowledge Management, Organizational Intelligence and Learning, and Complexity, The Encyclopedia of Life Support Systems. EOLSS Publishers (2003)
3. Nicolis, G., Prigogine, I.: Self-Organization in Non-Equilibrium Systems: From Dissipative Structures to Order Through Fluctuations. Wiley, Chichester (1977)
4. Haken, H.: Synergetics and the Problem of Selforganization. In: Self-organizing Systems: An Interdisciplinary Approach, pp. 9–13. Campus Verlag (1981)
5. Shalizi, C.R.: Causal Architecture, Complexity and Self-Organization in Time Series and Cellular Automata. PhD thesis, University of Wisconsin-Madison (2001)
6. von Foerster, H.: On Self-Organizing Systems and their Environments. In: Self-Organizing Systems, pp. 31–50. Pergamon, Oxford (1960)
7. Ashby, W.R.: Principles of Self-organization. In: Principles of Self-organization, pp. 255–278. Pergamon, Oxford (1962)
8. Heylighen, F., Joslyn, C.: Cybernetics and second order cybernetics. In: Encyclopaedia of Physical Science & Technology, vol. 4, pp. 155–170 (2001)
9. Gershenson, C.: Design and Control of Self-organizing Systems. PhD thesis, Vrije Universiteit Brussel, Brussels, Belgium (May 2007)
10. Boccara, N.: Modeling Complex Systems. Springer, Heidelberg (2004)
11. Holzer, R., de Meer, H.: On modeling of self-organizing systems. In: Autonomics 2008 (2008)
12. Di Marzo Serugendo, G., Foukia, N., Hassas, S., Karageorgos, A., Mostéfaoui, S.K., Rana, O.F., Ulieru, M., Valckenaers, P., Van Aart, C.: Self-organisation: Paradigms and applications. In: ESOA 2003. LNCS, vol. 2977, pp. 1–19. Springer, Heidelberg (2004)
13. Gerhenson, C., Heylighen, F.: When can we call a system self-organizing? In: Banzhaf, W., Ziegler, J., Christaller, T., Dittrich, P., Kim, J.T. (eds.) ECAL 2003. LNCS (LNAI), vol. 2801, pp. 606–614. Springer, Heidelberg (2003)
14. Cover, T.M., Thomas, J.A.: Elements of Information Theory, 2nd edn. Wiley, Chichester (2006)
15. Shannon, C.E.: A mathematical theory of communication. Bell System Technical Journal 27, 379–423 (1948)
16. Dessalles, J.L., Phan, D.: Emergence in multi-agent systems:cognitive hierarchy, detection, and complexity reduction. In: Computing in Economics and Finance 2005, vol. 257. Society for Computational Economics (November 2005)
17. Tyrrell, A., Auer, G., Bettstetter, C.: Biologically inspired synchronization for wireless networks. In: Dressler, F., Carreras, I. (eds.) Advances in Biologically Inspired Information Systems: Models, Methods, and Tools. Studies in Computational Intelligence, vol. 69, pp. 47–62. Springer, Heidelberg (1979)
18. Mirollo, R., Strogatz, S.: Synchronization of pulse-coupled biological oscillators. SIAM Journal of Applied Mathematics 50, 1645–1662 (1990)

A Method to Derive Local Interaction Strategies for Improving Cooperation in Self-Organizing Systems*

Christopher Auer, Patrick Wüchner, and Hermann de Meer

Faculty of Informatics and Mathematics,
University of Passau, Innstraße 43,
94032 Passau, Germany
{auerc,wuechner,demeer}@fim.uni-passau.de

Abstract. To achieve a preferred global behavior of self-organizing systems, suitable local interaction strategies have to be found. In general, this is a non-trivial task. In this paper, a general method is proposed that allows to systematically derive local interaction strategies by specifying the preferred global behavior. In addition, the resulting strategies can be evaluated using Markovian analysis. Then, by applying the proposed method exemplarily to the iterated prisoner's dilemma, we are able to systematically generate a cooperation-fostering strategy which can be shown to behave similar to the "tit for tat with forgiveness" strategy that, under certain circumstances, outperforms the well-known "tit for tat" strategy used, for instance, in BitTorrent peer-to-peer file-sharing networks.

1 Introduction

In self-organizing systems (SOSs), numerous entities interact with their neighborhood following certain interaction strategies. The system's entities and their strategies define the *micro-level* of the SOS. From the behavior on the micro-level, a global behavior of the SOS emerges at the *macro-level*. In general, the micro-level entities of an SOS rely on local knowledge only, i.e., solely on the information that is provided by the entities within their vicinity. Therefore, we call these entities and their strategy *vicinity-dependent*. For instance, to find the shortest path from the ant colony to a food place, an ant follows pheromone trails laid out by other ants in its vicinity (cf. [1]). Hence, the behavior of the ants is vicinity-dependent and results in an emergent macro-level behavior, i.e., the discovery of shortest paths.

When engineering and designing SOSs, the strategies on the micro-level need to be specified to attain the preferred emergent behavior on the macro-level. Since the system's entities are purely vicinity-dependent, this is in general a

* This research is partially supported by the AutoI project (STREP, FP7 Call1, ICT-2007-1-216404) and by the Networks of Excellence EuroFGI (IST 028022) and EuroNF (IST 216366).

non-trivial task. The main contribution of this paper is a generic method to derive suitable micro-level strategies.

Our solution approach is based on the following observation: The task of specifying suitable strategies would be less challenging if an entity and its strategy could rely on global knowledge, i.e., information of the SOS's global state and/or the other entities' strategies. In the following, we call an entity (and its strategy) that relies on global knowledge *omniscient*.

As an example, in this paper, we investigate a class of systems where cooperation is of particular importance, most prominently for peer-to-peer (P2P) file-sharing systems like BitTorrent (see [2]), where the dissemination of content relies on the collaboration of numerous entities, i.e., the peers. Often, game-theoretic approaches are used to model the peers in such systems, where a common game-theoretic model is the iterated prisoner's dilemma (IPD; cf. [3,4,5,6,7]). The IPD assumes that the peers are rationally thinking players that try to maximize their individual benefit and that a player's interaction strategy is vicinity-dependent, i.e., each player can only observe the behavior of the other players but does not know the other players' strategies. For such systems, a vicinity-dependent interaction strategy is sought that fosters global cooperation on macro-level. Based on the assumption that the strategy does not know its opponents' strategy, this is a challenging task in general. Often, it would be easier to find an omniscient strategy which knows the other players' strategies, and hence, could optimize its behavior to foster the preferred macro-level behavior.

In our method, we use an artificial omniscient entity called the Laplace's Demon (LD): This name stems from a hypothetical demon envisioned by the French mathematician and astronomer Pierre-Simon Laplace (1749–1827). This demon exactly knows the state of the whole universe and, hence, is able to predict future states by Newton's laws of physics.[1] The behavior of our LD is then investigated by a time-series analyzing algorithm to derive a purely vicinity-dependent strategy, which, in contrast to an omniscient strategy, can be effectively implemented into technical SOSs.

The method is generic in the sense that it does not focus on a specific application area or modeling formalism. The most restrictive constraint is the requirement that the system's model can be evaluated using discrete-event simulation. However, this requirement does not impose a hard constraint on a large class of engineerable systems.

Our approach allows to explicitly foster the preferred macro-level behavior and also provides means to analyze to which extent the preferred macro-level behavior is reached.

The remainder of this paper is structured as follows: In Sec. 2, we discuss related work. Then, the IPD as a model for cooperation in SOSs is shown in Sec. 3. Section 4 briefly introduces a time-series analyzing algorithm — the CSSR algorithm developed by Shalizi et al. (cf. [8,9]). In Sec. 5, we present in more detail our generic core method that is based on LDs. We apply this method in

[1] Pierre-Simon Laplace formulated this idea in the foreword to his essay "Essai philosophique sur les probabilités" from 1814.

Sec. 6 to derive vicinity-dependent interaction strategies for IPD-like problems. Finally, Sec. 7 draws conclusive remarks and gives an outlook to future work.

2 Related Work

We distinguish two complementary approaches for deriving local interaction strategies of entities in SOSs: At one extreme, it is assumed that the entities explicitly disseminate and periodically update global state information within the system, i.e., the entities are omniscient. At the other extreme, the system entities remain purely vicinity-dependent.

Omniscient entities are able to optimize their strategies to attain the preferred macro-level behavior based on global state information. Examples for such systems are distance vector and link state based algorithms for routing protocols as presented in [10,11], where the omniscient entities are able to calculate optimal routing paths. However, the periodic dissemination and local storage of global state information clearly suffers from scalability issues in large systems (cf. [12]).

Many application-driven approaches to derive *vicinity-dependent* strategies exist. For example, the AntNet routing protocol to find shortest routing paths in mobile networks was developed by investigating the exchange of pheromones between ants (cf. [1]). In [13], the approach of epidemic information dissemination uses the principle behind the spread of a disease to disseminate information in distributed systems. To form topologies in P2P overlay networks, a vicinity-dependent strategy was developed in [14] by investigating cell adhesion processes. Further examples are intuitive approaches like the random walk search algorithm for P2P networks in [15] and the gossip-based routing algorithm for ad hoc networks in [16], where messages are forwarded to randomly chosen neighbors on a best effort basis. Unfortunately, these application-driven approaches do not provide a generic method that can be used in a wider range of application areas. Hence, finding purely *vicinity-dependent* strategies remains a non-trivial task in general, and, to the best of the authors' knowledge, no general method yet exists.

3 The Iterated Prisoner's Dilemma

We now give a primer to the IPD and introduce the problem to be solved with the proposed method in Sec. 6.

In SOSs, where the collaboration between the entities is of particular importance, micro-level interaction strategies are sought from which global cooperation on the system's macro-level emerges. For instance, to disseminate content, a P2P file-sharing network needs to define strategies that foster global cooperation between all peers, i.e., each peer shares its bandwidth and content to other peers. The IPD is a commonly used modeling formalism to study cooperation in such SOSs (see, e.g., [3,4,5]).

The IPD models the interaction at discrete time steps between two entities called the players. At each time step of the IPD, each of the two players can choose either to cooperate (C) or to defect (D). Both players make their current choice independently from each other but the choice may well depend on the outcome of previous iterations. Both players present their choice and get a payoff, e.g., according to the following utility matrix **U**, which is taken from [17]:

$$\mathbf{U} = \begin{array}{c c} & \begin{array}{cc} \text{D} & \text{C} \end{array} \\ \begin{array}{c} \text{D} \\ \text{C} \end{array} & \begin{bmatrix} (1,1) & (4,0) \\ (0,4) & (3,3) \end{bmatrix} \end{array} . \quad (1)$$

The rows of **U** correspond to the choice of the first player, the columns to the choice of the second player, and the entries of **U** correspond to the payoffs, where the first element of each tuple is the payoff for the first player and the second element is the payoff for the second player.

It can easily be seen that cooperation on both sides maximizes the sum of the payoffs of both players, that is, the global welfare. However, cooperation on both sides may increase the temptation of one player to defect in order to maximize its personal payoff to a value of 4, whereas the payoff of the other player is minimized to a value of 0. This rational thinking process might lead both players to defect, which minimizes the global welfare and results in a lower individual payoff for both players compared to the case of mutual cooperation. In the IPD, the prisoner's dilemma is iterated and, hence, the outcome of past iterations can be incorporated into the decision process of the players.

When modeling the micro-level interactions of an SOS by the IPD, a strategy is sought from which global cooperation emerges on the systems' macro-level. Such a strategy should adjust its behavior to foster cooperation if its opponent is willing to cooperate and it should defect otherwise. The latter implicitly fosters global cooperation, since it makes defection less profitable than mutual cooperation. However, finding such a strategy is a non-trivial task in general since the IPD assumes that the players' strategies are purely vicinity-dependent, i.e., each player can incorporate only previous behavior of its opponent into its decision process but does not know its opponent's strategy.

In this paper, we systematically derive such a cooperation-fostering strategy by investigating an omniscient player, i.e., the LD. The LD knows its opponent's strategy and is, hence, able to easily adjust its behavior to foster cooperation. By investigating the behavior of the LD, a vicinity-dependent strategy for the IPD is derived. To do so, we first propose a generic method to derive vicinity-dependent interaction strategies for SOSs (see Sec. 5). The method is then applied to the IPD (see Sec. 6).

4 The CSSR Algorithm

This section briefly describes the Causal-State Splitting Reconstruction (CSSR) algorithm, which we utilize in our method as described later in Sec. 5. We choose this algorithm because it is sufficiently abstract to be applicable in a wide range of application scenarios. The interested reader is referred to [8,9] for a comprehensive description of the algorithm.

The CSSR algorithm takes a sequence of $N \in \mathbb{N}$ symbols as input. This sequence is drawn from a discrete alphabet $\mathcal{A}$ and is generated by some random process. Based on the given sequence, the algorithm approximates the ideal predictor of the random process in the form of a hidden Markov model: The CSSR algorithm outputs the state space $\mathcal{S}$ of a discrete-time Markov chain (DTMC; see [18] for theoretical background) and its transition probabilities, where at each state transition, i.e., time step, the DTMC outputs a symbol from the alphabet $\mathcal{A}$. In the context of computational mechanics, this Markov chain is called ϵ-machine. For every time step, let S denote the random variable of the current state of the ϵ-machine, S' the random variable of the successor state of S, and A the random variable of the symbol which is output during the transition from state S to S'. For any two states $s, s' \in \mathcal{S}$ and any symbol $a \in \mathcal{A}$, $P[S' = s', A = a|S = s]$ denotes the probability that the DTMC leaves state s and goes to state s' while emitting the symbol a.

The CSSR algorithm additionally requires the parameter $L_{\max} \in \mathbb{N}_0$: To approximate the ideal predictor of the time series, the CSSR algorithm investigates subsequences of length L within the input sequence, starting from $L = 0$. L is subsequently increased by one until $L = L_{\max}$. Thus, $L_{\max}$ corresponds to the maximal amount of time steps of the past that are incorporated by the CSSR algorithm to approximate the probability distribution of the next output symbol. The choice of $L_{\max}$ is based on a trade-off between the required computation time and the accuracy of the resulting ϵ-machine.

5 A Generic Method to Derive Local Interaction Strategies

In the following, we propose our generic method that consists of four steps. The approach is briefly discussed at the end of Sec. 5. The method is carried out as follows:

Step 1: *Modeling the System* Our method assumes that the system is modeled as a group of entities that interact by exchanging information at discrete time steps. Each entity of the system receives input from a countable set $\mathcal{I}$ and generates some output from a countable set $\mathcal{O}$. The entities' strategies map any input $i \in \mathcal{I}$ to some output $o \in \mathcal{O}$. The time-series analysis in Step 3 requires that the model can be evaluated by simulation. Besides these assumptions, our method does not impose any further restrictions on the modeling process. There are standard modeling formalisms that we expect to be particularly suitable to our approach since the local interaction of the entities play a central role, e.g., in game-theoretic models, cellular automatons, and graph-based models.

Step 2: *Specifying the Laplace's Demon* One entity of the system is chosen to be the LD and is modified as follows: The LD becomes an omniscient entity, i.e., it is equipped with global knowledge. Based on global knowledge, the LD's strategy can be optimized to foster the preferred macro-level behavior.

Step 3: *System Simulation and Analysis of the Laplace's Demon's Behavior* Based on the model derived in Steps 1 and 2, a simulation environment is set up to simulate the system including the LD. The simulation is executed and the input/output (I/O) behavior of the LD is analyzed: For each time step $t \in \mathbb{N}$, we denote with $(i_t, o_t) \in \mathcal{I} \times \mathcal{O}$ the I/O behavior of the LD, where i_t is the input which leads to the output o_t of the LD. A simulation of $N \in \mathbb{N}$ time steps generates a sequence $(i_t, o_t)_{t=1,\dots,N}$, which can be analyzed by the CSSR algorithm as a time series with alphabet $\mathcal{I} \times \mathcal{O}$. Then, the output of the CSSR algorithm is an ϵ-machine with state space $\mathcal{S}$ and transition probabilities $P[S' = s', (I, O) = (i, o)|S = s]$ for all $s, s' \in \mathcal{S}$ and $(i, o) \in \mathcal{I} \times \mathcal{O}$, where S is the random variable of the current state of the ϵ-machine, S' the random variable of the successor state of S, and (I, O) the random variable of the I/O behavior the LD exposes at the state transition from S to S'. Thus, in other words, the derived ϵ-machine describes the I/O behavior of the LD.

Step 4: *Imitating the Laplace's Demon* This step derives a vicinity-dependent strategy $\mathcal{L}$ that imitates the omniscient LD. To derive $\mathcal{L}$, the ϵ-machine obtained in Step 3 is used.

$\mathcal{L}$ adopts the state space $\mathcal{S}$ of the ϵ-machine and imitates the LD's behavior as follows: At first, $\mathcal{L}$ randomly chooses its initial state (e.g., according to the stationary probability distribution of the ϵ-machine). If $\mathcal{L}$ is in state $s \in \mathcal{S}$ and receives some input $i \in \mathcal{I}$, then $P[S' = s', O = o|S = s, I = i]$ is the probability that $\mathcal{L}$ changes its state to $s' \in \mathcal{S}$ and outputs $o \in \mathcal{O}$. This probability can be calculated by conditioning on the event $\{I = i\}$ and using the definition of the conditional probability:

$$P[S' = s', O = o|S = s, I = i] = \frac{P[S' = s', (I, O) = (i, o)|S = s]}{P[I = i|S = s]} .$$

The probability $P[I = i|S = s]$ in the denominator can be calculated by marginalization:

$$P[S' = s', O = o|S = s, I = i] = \frac{P[S' = s', (I, O) = (i, o)|S = s]}{\sum\limits_{s' \in \mathcal{S}} \sum\limits_{o \in \mathcal{O}} P[S' = s', (I, O) = (i, o)|S = s]} , \quad (2)$$

where the probabilities on the right side of (2) are the known transition probabilities of the ϵ-machine obtained in Step 3.

Since the ϵ-machine describes the I/O behavior of the omniscient LD, $\mathcal{L}$ effectively reproduces the LD's behavior while being purely vicinity-dependent, i.e., its output behavior is determined only by its current state s, its input i, and its transition probabilities.

Remarks. The resulting vicinity-dependent strategy is tailored towards the chosen model and its parameters, i.e., the strategy might lead to a different behavior when applied in a different environment as specified in Step 1 of the method. Hence, it is crucial to take care that the chosen model and its parameters accurately reflect the behavior of the system under investigation.

Nevertheless, resulting strategies can easily be adapted to a different environment or preferred macro-level behavior by applying the method again with refined omniscient strategies, a modified model, and/or parameter set.

Since the derived strategy is available in form of a DTMC, Markovian analysis can be used to evaluate its behavior.

We are completely aware of the fact that the simulation of systems with numerous interacting entities cannot guarantee that all possible states are investigated since such systems often expose a very complex and sometimes even chaotic behavior. Nevertheless, extensive simulation in Step 3 of the method results in a vicinity-dependent strategy which behaves optimal at least in the simulated cases.

Our method can also be applied iteratively. The derived strategy can be fed back to the environment leading to a refined strategy during the next iteration.

6 Application of the Method to the Iterated Prisoner's Dilemma

The method proposed in Sec. 5 is now applied to find two vicinity-dependent micro-level strategies for the IPD (cf. Sec. 3), which have different macro-level goals: The strategy $\mathcal{L}_{\mathrm{A}}$ imitates the *altruistic* LD, which fosters global cooperation. In contrast, the strategy $\mathcal{L}_{\mathrm{E}}$ imitates the *egoistic* LD, which has the aim of maximizing its own payoff. The method is applied as follows:

Step 1: *Modeling the System* In our model, we investigate the behavior of a single player, which plays the IPD with another player we call the opponent. The opponent randomly chooses one strategy from AC, AD, Tit-for-Tat, p-Tit-for-Tat, and Random: The strategy AC always cooperates and is, therefore, a naïve friendly strategy which gets easily exploited by a defector. AD always defects and is an archetype of an egoistic and distrustful strategy. Tit-for-Tat initially cooperates and then always makes the same choice as the other player during the last iteration. Tit-for-Tat is a smart strategy: It is friendly since it initially wants to cooperate. But in contrast to AC, it is not exploited if its opponent only defects. If its opponent tries to cooperate again, then it immediately forgives. With p-Tit-for-Tat we denote the pessimistic equivalent to Tit-for-Tat, which initially defects. Random is a strategy that at each iteration chooses C or D with a probability of 0.5. From the viewpoint of Random's opponent, Random is an unreliable strategy in the sense that no action leads to mutual cooperation or defection or any other preferred behavior.

In our model, the opponent randomly chooses one of the five mentioned strategies according to a uniform distribution. Then, the IPD starts: At each time step, the player plays the prisoner's dilemma with its opponent. At the end of each iteration, the player is informed of the choice of the opponent, and vice versa. Hence, the input and the output alphabet for both players is $\mathcal{I} = \mathcal{O} = \{\mathrm{C}, \mathrm{D}\}$. At each time step and with a probability $p_c \ll 1$ ($p_c \in (0, 1)$), the IPD is stopped and the opponent randomly chooses a different strategy. Then, the IPD is restarted. Thus, for each chosen strategy, the IPD is carried out in sequence for $1/p_c$ time steps in the mean.

Step 2: *Specifying the Laplace's Demon* We investigate the behavior of two LDs: The altruistic and the egoistic LD. Both are made omniscient by informing them of the strategy of their opponent. Depending on this knowledge, the LDs implement a micro-level strategy which fosters the preferred macro-level behavior.

The altruistic LD follows a smart strategy which fosters cooperation: If the opponent is willing to cooperate, i.e., with AC, TIT-FOR-TAT, or P-TIT-FOR-TAT, then the LD implements the strategy AC. In contrast, if the opponent always defects, then the LD implements AD, which hinders the opponent to exploit the LD and implicitly fosters global cooperation by making defection less profitable than mutual cooperation. In confrontation with RANDOM, the LD implements TIT-FOR-TAT, which in the mean keeps the balance between cooperation and defection.

The egoistic LD has the aim of maximizing its own payoff. To do so, it answers with AD if its opponent implements AC, AD, or RANDOM. In contrast, if its opponent implements a smart strategy, i.e., TIT-FOR-TAT or P-TIT-FOR-TAT, it answers with AC.

Step 3: *System Simulation and Analysis of the Laplace's Demon's Behavior* The IPD iterations, as described in Step 1, are carried out 2^{20} times while p_c is set to 0.01. Thus, in the mean, the investigated LD plays the IPD for 100 times before the IPD gets restarted with another strategy of the opponent.

Each iteration step generates a pair $(i_t, o_t) \in \{\mathrm{C}, \mathrm{D}\}^2$, where i_t is the input that leads to the answer o_t of the LD at time step t. The generated sequence of values from the alphabet $\mathcal{A} = \{\mathrm{C}, \mathrm{D}\}^2$ is then analyzed by the CSSR algorithm. The algorithm's parameter $L_{\max}$ is chosen to be 1 since for all investigated strategies of the opponent, at most one step of the past is relevant for making their choice.

For both LDs, the altruistic and the egoistic, the CSSR algorithm outputs an ϵ-machine with three states and twelve transition probabilities. For all of these twelve transition probabilities, a 99%-confidence interval is calculated based on 100 applications of the method. The radius of the largest confidence interval evaluates to 0.001 which shows that 2^{20} iterations can be assumed to be enough to obtain reasonably reliable results.

Step 4: *Imitating the Laplace's Demon* From the ϵ-machines obtained in Step 3, the vicinity-dependent IPD strategies $\mathcal{L}_\mathrm{A}$ and $\mathcal{L}_\mathrm{E}$ can be derived by adopting the ϵ-machines' state space and calculating the strategies' transition probabilities according to (2).

The strategy $\mathcal{L}_\mathrm{A}$ is shown on the left and $\mathcal{L}_\mathrm{E}$ on the right side of Fig. 1: The states of each strategy are represented by nodes, numbered from 0 to 2. An arc from state s to s' ($s, s' \in \{0, 1, 2\}$) is labeled with $i|p|o$: If the opponent's last choice was $i \in \{\mathrm{C}, \mathrm{D}\}$, then the strategy changes its state from s to s' with

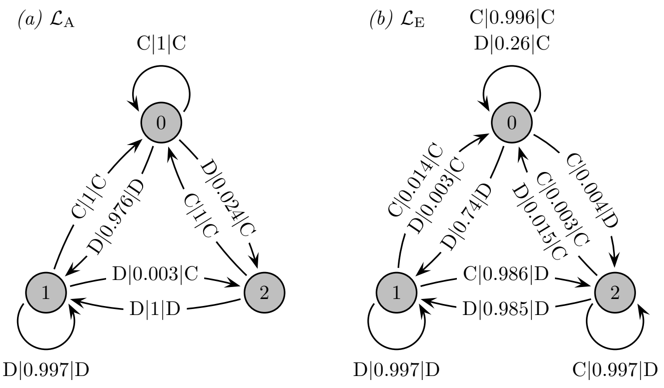


Fig. 1. The systematically derived vicinity-dependent strategies for the IPD: *(a)* shows the altruistic strategy $\mathcal{L}_\mathrm{A}$ and *(b)* the egoistic strategy $\mathcal{L}_\mathrm{E}$. The nodes correspond to the states of the strategies. An arc from node s to s' ($s, s' \in \{0, 1, 2\}$) is labeled with $i|p|o$, where $p \in (0, 1]$ is the probability that the strategy chooses $o \in \{\mathrm{C}, \mathrm{D}\}$ and goes to state s' if its current state is s and the opponent chose $i \in \{\mathrm{C}, \mathrm{D}\}$.

transition probability $p \in (0, 1]$ and answers with $o \in \{\mathrm{C}, \mathrm{D}\}$ in the current iteration of the IPD.

6.1 Discussion of the Results

In Fig. 1, it can be seen that $\mathcal{L}_\mathrm{A}$ exposes a similar behavior to Tit-for-Tat: If the opponent cooperates, it answers with cooperation and goes to State 0. If the opponent defects, it also defects with a high probability and goes to State 1. Moreover, $\mathcal{L}_\mathrm{A}$ implements a forgiveness mechanism, just like the "tit for tat with forgiveness" strategy proposed by Anatol Rapoport (cf. [19]): With a relatively low probability, $\mathcal{L}_\mathrm{A}$ cooperates even if the opponent defects and goes to State 2. In State 2, $\mathcal{L}_\mathrm{A}$ waits for the reaction of the opponent: Cooperation leads to State 0 and defection to State 1. By this mechanism, $\mathcal{L}_\mathrm{A}$ is able to find out whether a defecting opponent is willing to cooperate in principle. This is especially helpful for establishing cooperation if the opponent implements the initially defecting p-Tit-for-Tat strategy.

The egoistic strategy $\mathcal{L}_\mathrm{E}$ in Fig. 1 can be described similarly: In State 0, $\mathcal{L}_\mathrm{E}$ cooperates with a high probability if its opponent cooperates. However, from time to time $\mathcal{L}_\mathrm{E}$ answers with defection to find out if the opponent implements AC and can, thus, be exploited. This exploitation is done in State 2, where cooperation is answered with defection with a high probability. If the opponent answers with defection, $\mathcal{L}_\mathrm{E}$ goes to State 1 from any state and defects with a high probability. Hence, State 1 corresponds to mutual defection. However, if $\mathcal{L}_\mathrm{E}$'s opponent implements Tit-for-Tat or p-Tit-for-Tat, then mutual cooperation leads to a higher personal payoff than mutual defection. Hence, $\mathcal{L}_\mathrm{E}$ implements a forgiveness mechanism and cooperates on defection with a low probability.

Most interestingly, both derived strategies are able to reconstruct the same knowledge that was granted to the omniscient LDs explicitly while being purely vicinity-dependent. $\mathcal{L}_\mathrm{A}$ is able to find out if the opponent is willing to cooperate. $\mathcal{L}_\mathrm{E}$ finds out if the opponent can either be exploited, implements a smart strategy (i.e., Tit-for-Tat or p-Tit-for-Tat), or always defects. Based on this information, the derived strategies adjust their behavior to either establish cooperation or to maximize the personal payoff.

It is worthwhile to note that the transition probabilities of both strategies reflect the chosen model assumptions and parameters. This can, for example, be seen in $\mathcal{L}_\mathrm{E}$'s transition between states 2 and 0 labeled with C|0.003|C, which does not lead to maximal payoff if the opponent implements AC. In this case, cooperation should be answered with defection with a probability of 1. This is because $\mathcal{L}_\mathrm{E}$ assumes that the opponent's strategy changes with a probability of $p_c = 0.01$ according to a uniform probability distribution on the set of the other strategies. Consequently, changing this probability distribution or the value of p_c leads to other transition probabilities of the strategy $\mathcal{L}_\mathrm{E}$ or, respectively, $\mathcal{L}_\mathrm{A}$.

6.2 Steady-State Analysis

Since the derived vicinity-dependent strategies are obtained in form of a DTMC, results obtained by steady-state Markovian analysis provide measures to compare the strategies $\mathcal{L}_\mathrm{A}$, Tit-for-Tat, and $\mathcal{L}_\mathrm{E}$. Each of the three strategies play the IPD against the strategies AC, AD, Tit-for-Tat, p-Tit-for-Tat, and Random. For all encounters, steady-state analysis is applied to obtain the expected payoff per iteration of the IPD. To obtain concrete values, the payoff matrix $\mathbf{U}$ given in (1) is considered. The numerical results are summarized in Tab. 1. The table is subdivided into three columns: The first column lists the five opponents, the second column consists of the expected payoffs for the strategies $\mathcal{L}_\mathrm{A}$, Tit-for-Tat, and $\mathcal{L}_\mathrm{E}$, and the third column lists the expected payoffs for the respective opponent.

It can be seen that $\mathcal{L}_\mathrm{A}$ behaves similar to Tit-for-Tat and clearly outperforms Tit-for-Tat if confronted with p-Tit-for-Tat since $\mathcal{L}_\mathrm{A}$ is able to establish cooperation. If Tit-for-Tat plays against p-Tit-for-Tat, mutual cooperation cannot be established, since the two strategies alternately

Table 1. Results from the steady-state analysis of $\mathcal{L}_\mathrm{A}$, Tit-for-Tat, and $\mathcal{L}_\mathrm{E}$

	Payoff			Payoff (Opp.)		
	$\mathcal{L}_\mathrm{A}$	TfT	$\mathcal{L}_\mathrm{E}$	$\mathcal{L}_\mathrm{A}$	TfT	$\mathcal{L}_\mathrm{E}$
AC	3.0	3.0	3.571	3.0	3.0	1.286
AD	0.997	1.0	0.996	1.009	1.0	1.012
Tit-for-Tat	3.0	3.0	1.336	3.0	3.0	1.336
p-Tit-for-Tat	3.0	2.0	1.336	3.0	2.0	1.336
Random	1.993	2.0	2.477	2.020	2.0	0.569

cooperate and defect. When playing against AD, $\mathcal{L}_\mathrm{A}$ receives a lower expected payoff than Tit-for-Tat. This is due to the fact that $\mathcal{L}_\mathrm{A}$ once in a while tries to establish cooperation. The numerical results for $\mathcal{L}_\mathrm{A}$ show that it indeed fulfills the requirements of a strategy that fosters global cooperation: If its opponent is willing to cooperate, $\mathcal{L}_\mathrm{A}$ is always able to establish cooperation. If its opponent always defects, then $\mathcal{L}_\mathrm{A}$ answers with defection, which implicitly fosters global cooperation by making AD a less profitable strategy than a cooperating strategy like Tit-for-Tat.

$\mathcal{L}_\mathrm{E}$ is able to exploit AC and Random. If confronted with AD, $\mathcal{L}_\mathrm{E}$'s expected payoff is slightly lower than that of Tit-for-Tat because it cooperates with a low probability to check if the opponent's strategy has changed. Tit-for-Tat and p-Tit-for-Tat cannot be exploited by $\mathcal{L}_\mathrm{E}$. However, $\mathcal{L}_\mathrm{E}$ still tries to exploit the opponent from time to time. This again leads to a lower expected payoff compared to Tit-for-Tat.

7 Conclusion and Future Work

In this paper, we have shown the advantages and disadvantages of implementing purely vicinity-dependent or omniscient entities within the micro-level of self-organizing systems. Based on this investigation, we proposed a generic top-down approach to systematically derive vicinity-dependent strategies based on the specification of the preferred macro-level behavior.

Our approach simplifies the development of self-organizing systems in the sense that suitable interaction strategies can first be defined based on global knowledge, i.e., information of the self-organizing system's global state and/or the other entities' strategies. From these omniscient strategies, our method then systematically generates vicinity-dependent strategies. The resulting strategies can then be evaluated using standard Markov chain-based analysis techniques and can also be implemented easily into the entities of the self-organizing system.

The proposed method was proven to be applicable in a game-theoretic setting, where an altruistic and egoistic strategy for players of the iterated prisoner's dilemma could be obtained.

However, our method is presented independently of the applied modeling formalism. We expect the method to be applicable also to other modeling formalisms that focus on the local interaction of the micro-level entities, e.g., cellular automatons or graph-based models, and more complex scenarios. This has to be investigated in future work along with scalability issues that arise when engineering large-scale SOSs. We also foresee an extension to our method which allows to specify and evaluate scenarios comprising more than one LD with possibly different goals. The outcome of these investigations will also lead to further application examples and to more concrete advises on how to build suitable models to be used with the proposed method.

References

1. Caro, G.D., Dorigo, M.: AntNet: a mobile agents approach to adaptive routing. Technical Report IRIDIA/97-12, Université Libre de Bruxelles, Belgium
2. Cohen, B.: Incentives build robustness in BitTorrent (2003)
3. Jiang, J., Bai, H., Wang, W.: Trust and cooperation in peer-to-peer systems. In: Li, M., Sun, X.-H., Deng, Q.-n., Ni, J. (eds.) GCC 2003. LNCS, vol. 3032, pp. 371–378. Springer, Heidelberg (2004)
4. Lai, K., Feldman, M., Stoica, Chuang, J.: Incentives for cooperation in peer-to-peer networks. In: Workshop on Economics of Peer-to-Peer Systems (2003)
5. Jun, S., Ahamad, M.: Incentives in BitTorrent induce free riding. In: P2PECON 2005: Proceeding of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems, pp. 116–121. ACM Press, New York (2005)
6. Axelrod, R.: The Evolution of Cooperation. Basic Books, Inc., Publishers, New York (1984)
7. Yew Chong, S., Humble, J., Kendall, G., Li, M., Yao, X.: The Iterated Prisoner's Dilemma: 20 Years On. In: The Iterated Prisoner's Dilemma: 20 Years On. World Scientific, Singapore (2006)
8. Shalizi, C.R.: Causal Architecture, Complexity and Self-Organization in Time Series and Cellular Automata. PhD thesis, University of Wisconsin, Supervisor: Martin Olsson (2001)
9. Shalizi, C.R., Shalizi, K.L.: Blind construction of optimal nonlinear recursive predictors for discrete sequences. In: AUAI 2004: Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence, Arlington, Virginia, United States, pp. 504–511. AUAI Press (2004)
10. Perkins, C., Bhagwat, P.: Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In: ACM SIGCOMM 1994 Conference on Communications Architectures, Protocols and Applications, pp. 234–244 (1994)
11. Clausen, T., Jacquet, P.: Optimized link state routing protocol, OLSR (2003)
12. Diamantopoulos, F., Economides, A.A.: A performance study of DSDV-based CLUSTERPOW and DSDV routing algorithms for sensor network applications. In: ISWPC 2006: Proceedings of the 1st Internation Symposium on Wireless Pervasive Computing, pp. 1–6. IEEE Press, Los Alamitos (2006)
13. Eugster, P.T., Guerraoui, R., Kermarrec, A.M., Massoulieacute, L.: Epidemic information dissemination in distributed systems. Computer 37(5), 60–67 (2004)
14. Jelasity, M., Babaoglu, O.: T-Man: Gossip-based overlay topology management. In: Brueckner, S.A., Di Marzo Serugendo, G., Hales, D., Zambonelli, F. (eds.) ESOA 2005. LNCS, vol. 3910. Springer, Heidelberg (2006)
15. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and replication in unstructured peer-to-peer networks. In: ICS 2002: Proceedings of the 16th International Conference on Supercomputing, pp. 84–95. ACM, New York (2002)
16. Haas, Z.J., Halpern, J.Y., Li, L.: Gossip-based ad hoc routing. IEEE/ACM Trans. Netw. 14(3), 479–491 (2006)
17. Osborne, M.J., Rubinstein, A.: A Course in Game Theory. The MIT Press, Cambridge (1994)
18. Bolch, G., Greiner, S., de Meer, H., Trivedi, K.S.: Queueing Networks and Markov Chains, 2nd edn. John Wiley & Sons, Inc., Hoboken (2006)
19. Rapoport, A., Cammah, A.M.: Prisoner's Dilemma: A Study in Conflict and Cooperation. University of Michigan Press (1965)

A Self-powered Module with Localization and Tracking System for Paintball

Andrey Somov[1], Vinay Sachidananda[2], and Roberto Passerone[1]

[1] University of Trento, Trento, Italy
{somov,roby}@disi.unitn.it
[2] Darmstadt University of Technology, Darmstadt, Germany
vinay@cs.tu-darmstadt.de

Abstract. In spite of the popularity of wireless sensor networks (WSN), their application scenarios are still scanty. In this paper we apply the WSN paradigm to the entertainment area, and in particular to the domain of *Paintball*. This niche scenario poses challenges in terms of player localization and wireless sensor node lifetime. The main goal of localization in this context is to locate and track the player in order to facilitate his/her orientation, and to increase the level of safety. Long term operation could be achieved by adopting appropriate hardware components, such as storage elements, harvesting component, and a novel circuit solution. In this work we present a decentralized localization and tracking system for *Paintball* and describe the current status of the development of a self-powered module to be used between a wireless node and an energy harvesting component.

1 Introduction

A Wireless Sensor Network (WSN) is a distributed collection of nodes which are resource constrained and capable of operating with minimal user attendance. Wireless sensor nodes operate in a cooperative and distributed manner. However, there are application areas, such as the entertainment domain, where WSNs are still seldom applicable. In this paper we apply the concept of WSN in a *Paintball* application. *Paintball* is a sport in which players eliminate opponents from play by hitting them with paint. For a successful *Paintball* application we have to solve two important problems: player localization and long term operation of the wireless sensor node.

At present, players which are involved in *Paintball* use the Global Positioning System (GPS) [6] for orienting, walkie-talkie to communicate with each other or do not use these devices at all in case of short-term and bounded space scenario. However, providing each of e.g. hundred players with an individual GPS receiver is not always the preferred solution for cost reasons. Moreover, GPS itself is a power hungry component and needs at least four known satellites to be visible in order to find the coordinates of the receiver.

Over the years, many protocols [1] have been devised to enable the location discovery process in WSNs to be autonomous and able to function independently of GPS and other manual techniques. In all cases, the focal point of location discovery has been a set of special nodes known as *beacons*, which have been referred to by some researchers as anchor, locator, or seed nodes.

As for long term operation, it has always been the key issue in power supply design for wireless sensor nodes. Energy harvesting components, such as solar cells or piezoelectric converters, have to some extent solved the lifetime problem. However, the availability of heterogeneous WSN platforms has resulted in a problem of inflexibility: *it is difficult to exploit a wireless sensor node and a harvesting component together*. Basically, WSN platforms are being developed for particular cases [3, 5] or have already been integrated with a harvesting component [2].

In this paper we propose a localization and tracking system solution that is an extension of the work based on the existing systems RADAR [26] and MoteTrack [25]. The proposed system does not rely upon any back-end server or network infrastructure. The location of each mobile node is computed using a received radio signal strength signature from numerous beacon nodes to a database of signatures that is replicated across the beacon nodes themselves.

To provide "perpetual" support of the hardware, we designed a mediator board between a wireless sensor node and a harvesting component. This board has a two-level energy storage buffer including two supercapacitors wired in series as a primary buffer and a li-ion battery with high capacitance as a secondary buffer. A novel circuit solution coupled with advanced electronic components is used to charge the supercapacitors and the li-ion battery safely to prolong their lifetime. Moreover, the board supports AC or DC based ambient power source that will make the entire WSN more flexible.

The paper is organized as follows: Section 2 will introduce a review of (a) recent localization systems, and (b) existing wireless sensor platforms with energy scavenging technology; Section 3 will present the main goals of the paper; Section 4 will describe the architecture of the self-powered module and the implemented self-powered board, and finally, Section 5 will present the designed localization and tracking system with the experimental results.

2 Background

In this section we briefly review some of the existing and widely used location tracking systems, as well as wireless platforms with energy scavenging technology.

2.1 Localization Systems Overview

In Global Positioning System (GPS) [6], [13], triangulation [26], [16] uses ranges to at least four known satellites to find the coordinates of the receiver, and the clock bias of the receiver. The triangulation procedure starts with an a priori estimated location that is later corrected towards the true location. Due to high power consumption, relative inaccuracy and indoor operation instability it is undesirable to use GPS in WSN.

RADAR [26] is the first indoor localization system. It is based on empirical Received Signal Strength Indication (RSSI) measurements as well as a simple yet effective signal propagation model. RADAR uses a standard 802.11 network adapter to measure signal strength values with respect to the base stations that are within range of the network adapter. A drawback of this location approach is that the signal pattern is recorded statically in the database and may greatly differ from the values measured in the dynamic environment.

MoteTrack [25] is a decentralized localization system which is more sucessful in indoor localization and it is largely based on the RADAR [26] approach. The location of each mobile node is computed using RSSI signature from numerous beacon nodes to a database of signatures that is replicated across the beacon nodes themselves. MoteTrack can tolerate the failure of up to 60% of the beacon nodes without severely degrading accuracy, making the system suitable for deployment in highly volatile conditions.

Cricket [14] was designed and implemented as an indoor mobile decentralized navigation system called CricketNav [14]. It is based on Ultrasound and RF, and uses independent, randomized transmission schedules for its beacons and a receiver decoding algorithm that uses the minimum of modes from different beacons to compute a maximum likelihood estimate of the location. However, the CricketNav system requires large amount of extra hardware to be installed.

A distributed and leaderless algorithm [28] performs the detection of the logical location in specks. This method employs binary connection information for range estimation, as opposed to RSSI method applied in our work.

There are plenty of RSSI based localization systems, e.g. RADAR [26], MoteTrack [25], and commercial applications, for example Ekahau [27]. However, we have focused our attention on developing more robust and efficient RSSI based approach.

2.2 Wireless Sensor Platforms with Energy Scavenging Technology

In this section we will review recent wireless sensor platforms with energy scavenging technology.

VIBES [2] and PMG Perpetuum [21] are microsystems powered from ambient vibrations. VIBES is an energy aware system and has a possibility to adjust the duty cycle according to the available energy. PMG platforms have a primary energy buffer for improved flexibility and do not require any maintenance.

The next three wireless sensor platforms with energy scavenging technology, namely Heliomote [3], Prometheus [4] and Trio [5], contain off-the-shelf modules Mica2 [9] for the first one, and Telos [7] for Prometheus and Trio. In contrast to VIBES, Heliomote has solely NiMH rechargeable batteries as energy buffer. Prometheus is a wireless sensor platform which includes the Prometheus power board [4] and Telos. The entire system is being powered by the Prometheus power board. This power board is implemented with a two-stage storage system containing two supercapacitors as primary buffer and a lithium rechargeable battery as secondary buffer. Prometheus, Telos and XSM [8], in turn, are combined into the Trio node. Telos is necessary for low power operation, Prometheus is responsible for power supply, and finally XSM is a set of indispensable sensors.

However, all the listed platforms support either DC (solar radiation) or AC (vibration) ambient power source that leads to inflexibility of the entire WSN.

3 Key Principles of the System

The most well-known and popular Paintball types are speedball, woodsball and scenarioball. The scenarioball type appears to be most interesting for our research.

The playing field in scenario paintball (the storyline can be anything like civil war events, gangster wars of 1930s, the Oklahoma D-Day, storming a building or rescuing hostages) is normally large and in most cases unknown for the players. Moreover, the game may last several hours, therefore it would be reasonable to provide the players with a wireless sensor node to track their position by beacon nodes deployed throughout the playing field. It could be done for safety reasons in the interests of a paintball player, and with the aim of assisting him/her in orientation by showing his/her location on a Personal Digital Assistant (PDA).

However, we would like to pay your attention to some important requirements and constraints of the localization system. First of all, the beacon nodes should be distributed over the playing field in such a way that the players are within the radio range to be tracked. The player location computation should be performed quickly to guarantee his relevant position. Since the playing field is in most cases a natural setting, the energy proves to be a limiting factor. The system accuracy depends on the signal power and frequency band, which, again, leads to energy restrictions. Thus, the main goal is to design an accurate and failure-tolerant long-term localization system for Paintball players tracking which is applicable to the indoor and outdoor operation.

The system requirements and our contribution are described in more detail in Section 4.1 and Section 5.

4 Self-powered Module Design

In this section we present the general block diagram of the self-powered module (see Figure 1), describe the implemented self-powered board (see Figure 2), and finally discuss the hardware selection.

4.1 Self-powered Module Architecture

In our design we apply a modular uniform technology for WSN with a power management technique. The entire system is to contain three main parts: the ambient power source → the power management board → the wireless sensor node. The ambient power source consists of a harvesting component converting ambient energy into electric energy. In other words, the self-powered board designed is an enhancement for wireless sensor nodes which provides interoperability with different types of harvesting components and long term operation for localization system.

The general block diagram of the self-powered board is depicted in Figure 1. The parts shadowed in grey refer to the self-powered module.

There are three power supplies in the entire self-powered system: an Ambient Power Source, a Supercap (Primary Buffer), and a Rechargeable Battery (Secondary Buffer). Since the Rechargeable Battery has the lowest lifetime, which mostly depends on the quantity of charge-discharge cycles, its energy is the most valuable. The Ambient Power Source is inexhaustible, but unstable. Thus, it is more reasonable to exploit the Ambient Power Source energy by charging Supercap using Voltage Stabilization. This means that the entire system must operate at first from the Primary Buffer. When the Supercap is discharged, the system is to use the Rechargeable Battery as power supply. After being discharged, the Secondary Buffer must be replenished with Supercap energy. The wireless sensor node microcontroller must monitor the energy level of the power supplies via the Voltage Monitor of the Supercap and the Charge/monitor controller of the Li-ion battery and manage the Analog Switch in accordance with the energy availability of power supplies. This solution will increase the lifetime of the energy storage devices.

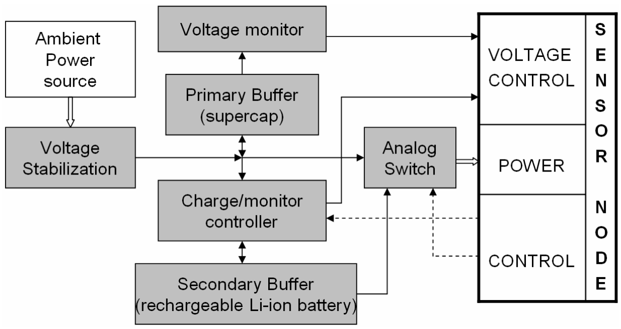


Fig. 1. The general block diagram of the self-powered module

The current flexible solution will allow one to connect a wireless sensor node with a harvesting component which operates using an environment energy source like solar rays, vibration or temperature difference. A user just needs to choose an appropriate wireless sensor node which suits according to cost and adequate ambient power source for the application. Moreover, the schematics proposed enables two storage buffers to support power hungry applications and to increase the lifetime of the sensor node. Voltage monitors allow the MCU to see how storage buffers are charged.

The major distinguishing feature of the self-powered board with respect to traditional energy scavenging technology is the possibility of charging its energy buffers either from DC or AC ambient power sources by switching a slide switch for the appropriate source. The self-powered platforms observed in Section 2 are able to operate with a preset energy scavenging component.

4.2 Hardware Selection

The self-powered board is shown in Figure 2. Its size (40 x 70 mm) is similar to usual wireless sensor nodes. We have marked out at the top and bottom layers of the board the pads of the main parts according to the general schematics sketched out in Figure 1. Since our flexible design makes it possible to apply different kinds of ambient power sources, the first component is a slide switch which performs switching between AC or DC options. The self-powered module supports a 1.3W ambient power source. AC voltage is converted into DC voltage by a diode rectifier DB102S and an electrolytic capacitor which are standard components for this case. Then the zener diode BZV85C4V7 stabilizes the DC voltage up to 4.7V.

We used two supercapacitors wired in series to reduce leakage current as a primary energy buffer. The supercapacitors by Nesscap EDLC [24] with 2.3V maximum voltage rating are an excellent fit for our design. Besides, we applied a resistor in series with the 4.7V stabilized DC voltage source to charge the primary buffer with an appropriate voltage. In line with [4], we have chosen 25F capacitance for each of the supercapacitors.

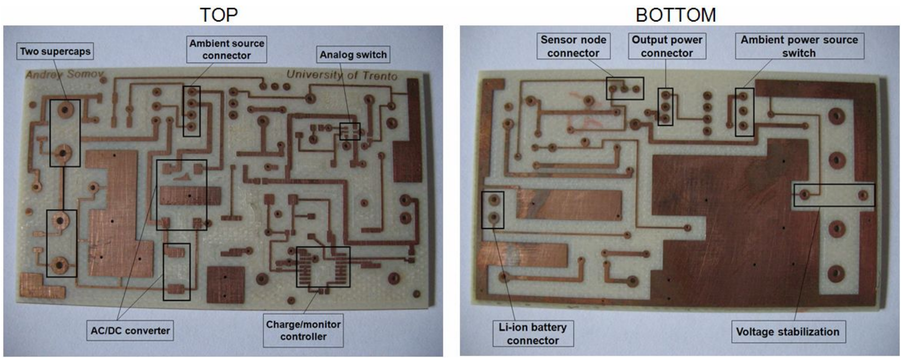


Fig. 2. The 40x70 mm self-powered board

Panasonic [11] CGR18650E Li-ion rechargeable battery, with 3.7 nominal voltage and 2550mAh typical capacity, acts as a secondary energy buffer. Lithium rechargeable batteries have no memory effect while they provide high charge-discharge cycle, highest density and lowest leakage [4]. However, they require a more complex charging circuit like, for example, a special charge controller.

To charge the secondary energy buffer properly we applied a DS2770 battery monitor and a charge controller [29]. This integrated circuit performs several functions needed for thorough battery maintenance. The secondary buffer is charged by the primary buffer, but when the primary buffer is being charged and the ambient power source provides a stable voltage, then it is possible to charge the secondary buffer directly.

We used the Single Pole Double Throw (SPDT) switch ADG849 [18] to choose either the primary or secondary power buffer to supply the wireless sensor node. ADG849 has high current carrying capability and low power consumption.

The self-powering module enables 3.3V (1W, 276mA maximum current) output voltage by applying the zener diode 1N4728A which stabilizes the voltage up to this magnitude.

For interconnection between the self-powered module and the wireless sensor node we applied a 3-pin connector for charge/monitor battery control, supercapacitor voltage monitoring, and analog switch control. Another 3-pin connector carries the power supply for wireless sensor node. A 4-pin ambient source power connector makes it possible to attach an external energy harvesting component.

5 Decentralized Localization Algorithm

This section describes a new accurate and robust decentralized localization algorithm which supports database distribution. To obtain these results we made the following contributions. At first, the computation is completely done in a decentralized fashion to get a better localization system which works on both CC1000 [19] and CC2420 radios [20]. Moreover, there is no back-end server, so all the computation is performed in the deployed sensor motes. Secondly, the location signature database is replicated across the beacon nodes, similarly to previous systems [25], [26], but here the replication is done while dividing the database with the basic *stable marriage algorithm*. This minimizes per-node storage overhead and provides high robustness to failure and ensures more accuracy. Thirdly, there is a new memory management technique and queuing technique which is needed for the distribution of data and the location estimation process. Finally, we employ a dynamic radio signature distance metric which is quite similar to RADAR [26] and an idea from MoteTrack [25] that adapts to loss of information and partial failures of the beacon infrastructure.

5.1 Distance Metric and Location Estimation

The signature-based localization scheme used requires a set of base stations, i.e. beacon nodes, generally at fixed locations, to either transmit periodic beacon messages or receive signals from mobile nodes. Beacon nodes broadcast beacon messages at a range of transmission power and frequency levels. Using multiple transmission power levels and frequencies will cause a signal to propagate at various levels in its medium and therefore exhibit different characteristics at the receiver. The mobile nodes collect the signature, where signature is collection of tuples, and then use a distance metric to compare multiple locations and pick the one that best matches the observed signal strength.

Let s be the received signature and let R be the set of reference signatures. The location of a mobile node can be estimated as follows. The first step is to compute the signature distance, D, from s to each reference signature r which belongs to R. As for RADAR and MoteTrack the user can choose between two distance metrics for the performance and environmental behavior. The Euclidean distance is computed as:

$$D = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2} \qquad (1)$$

where we take the three dimensional way of the moving target using $X = (x_1, x_2, x_3)$ and $Y = (y_1, y_2, y_3)$. Where x and y are the distance metrics.

We use the Manhattan distance [25] metric as well:

$$D(r,s) = \sum_{t \in T} | \, meanRSSI(t)_r - meanRSSI(t)_s \, | \qquad (2)$$

where T is the set of signature tuples represented in both signatures, and $meanRSSI\ (t)_r$ is the mean $RSSI$ value in the signature tuple t appearing in signature r.

The mobile node first acquires its signature s by listening to beacon messages, and then broadcasts s, requesting for the estimation of its location. Then one or more of the beacon nodes computes the signature distance between s and their slice of the reference signature database. They report either a set of reference signatures to the mobile node, or directly compute the location of the mobile node. Usually the request is given to the beacon with the strongest RSSI, which is explained below.

The design we used here is strongest RSSI. In this design, we assume that the most relevant (closest in signal space) reference signatures are stored on the beacon node with the strongest signal. The beacon node sending the StrongestRSSI must contain the database of the particular area where the mobile node is requesting for the estimation of the location and it sends a request to the beacon node from which it received the strongest RSSI, and only that beacon node estimates the mobile node's location. As long as this beacon node stores an appropriate slice of the reference signature database, this should produce very accurate results. The main advantages are low communication overhead, accurate location estimates and communication cost is very low because only one reply is sent to the mobile node containing its location coordinates.

Therefore it is crucial that the reference signatures are distributed in an "optimal" fashion. In centralized fashion all the signatures are stored in the mobile mote and we wish to ensure that each reference signature is replicated across several beacon nodes in the event of beacon node failures. Here we use two algorithms for database distribution, which we refer to as *stable marriage algorithm* and the one referred from the MoteTrack which is *greedy distribution algorithm.*

The decentralized approach to RF-based localization is based on a network of battery-operated wireless nodes based on the 802.15.4 ZigBee standard [22].

5.2 Decentralized Localization

After we deal with the distance metric we need to concentrate on how the location of the moving target is estimated. This is done with the given set of signature distances D from the above calculation algorithm of the metric distances. When the mobile node wants to track its location it sends a request to estimate the location. The beacon node, which contains the database, calculates the location of the mobile mote as a centroid, in which centroid is of set of signatures. Here we use the same approach as in MoteTrack where we consider the centroid of the set of signatures within some ratio of the nearest reference signature. Given a signature s, a set of reference signatures R, and the nearest signature r, we select all reference signatures r belonging to R that satisfy a mathematical function which is less than constant c. The given geographic centroid of the locations of this subset of reference signatures is then taken as the position of mobile node. Then this request from the mobile node is processed and the reply is sent to the mobile mote. Here the memory management, the fragmentation of the signature which is sent as the request for location estimation, the reassembly of this signature in the beacon mote and the queue system during the request for location and reply of the estimated location are the main ideas which are developed and expanded via our approach.

5.3 Implementation and Data Collection Mode

The developed decentralized localization and tracking system is programmed on the Arslogica-3Tec mote [15] platform using the TinyOS [10] operating system which are designed for low-power operation. It is relatively small and can be deployed unobtrusively in an indoor environment and, moreover, the developed system works on both CC1000 radio and CC2420 radio, which provides both programmable transmission power levels and direct sampling of received signal strength. We do not require any central server, which is one of the advantages of the system. A laptop connected to a mote is used to build the reference signature database, but thereafter the system is self-contained.

The developed system has a total code size for the beacon and mobile node software of about 2,500 lines of NesC [23] code and currently there is a front end GUI with normal display in JAVA which is still in the development phase. Since we need that each beacon node store a different set of reference signatures depending on the distribution mechanism used, we are concentrating primarily on the distribution of the reference signature.

Each reference signature consists of a tuple of the form {SourceID, powerLevel, frequency, meanRSSI}.

We have deployed the decentralized localization and tracking system over one floor of ArsLogica IT Laboratory. Our current installation consists of 20 beacon motes.

We collected a total of 250 reference signatures overall from throughout the laboratory. Each signature was collected for 1 minute, during which time every beacon node transmitted at a rate of 4 Hz, each cycling through 4 frequency and 2 transmission power levels (from –20 to 10 dBm in steps of 5 dBm).

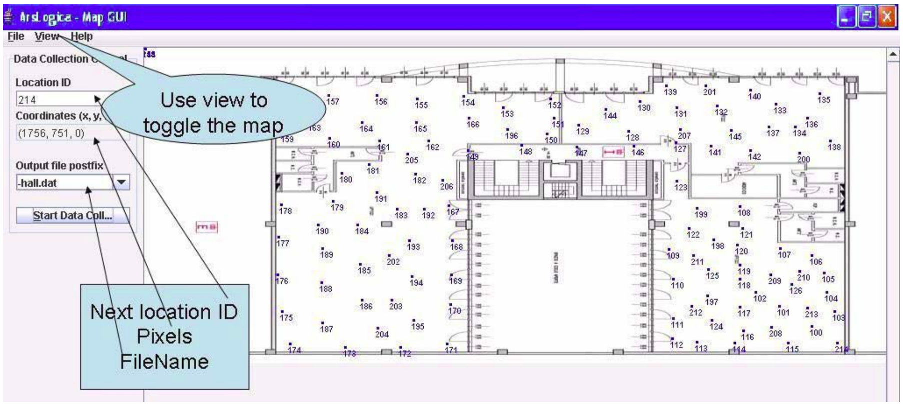
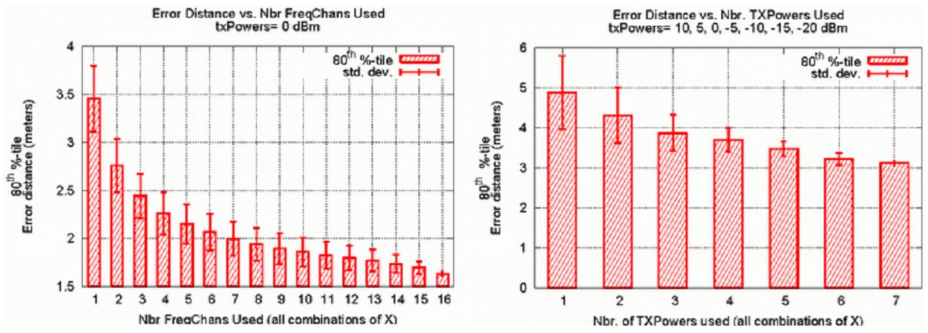


Fig. 3. The map of the IT laboratory with 250 Reference Signatures collected

A beacon message consists of a four-byte payload: 2 bytes for the source node ID, one byte representing the transmission power level and one byte representing the frequency level. We divided the collected signatures into two groups: the training data set (used to construct the reference signature database) and the testing data (used only for testing the accuracy of location tracking). Our analysis investigates the effects of a wide range of parameters including whether signatures are collected in a hallway or in a room, whether the room's door is open or closed, the time of the day (to account for solar radiation and building occupancy) and the use of different mobile nodes (to account for manufacturing differences).

5.4 System Accuracy Evaluation

The system operation and accuracy can be improved by broadening the frequency band and increasing power, as it is shown in Figure 4 (a, b) respectively. Figure 4a demonstrates the relationship between the error distance given in the *y* axis and the frequency band in the *x* axis. For predefined frequency 1, the accuracy decreases and the error rate increases up to 3.5m. When the frequency is gradually increased the accuracy improves and the error rate decreases. In other words, the system demonstrates the highest accuracy (less than 1m) at the frequency rate of 16.



a) b)

Fig. 4. The graph showing the relationship between the system accuracy and the frequency (MHz) variation (a), and TX power (μW) variation (b)

The main advantage of frequency increase is improved accuracy. However, there are some disadvantages, e.g., the longer period of signature collection and the larger size of signatures.

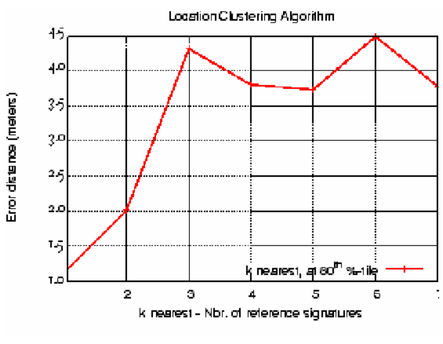
The graph presented in Figure 4b depicts the accuracy achieved when the power varies from 1 to 7. As this graph is derived from experiments, we can see that when the TX power decreases to its minimum, the error rate can reach its maximum. For example, when we had set power 2 and frequency 4, we got the average result from 2m to 3m, which can be good enough for the system in which the RF technology is applied.

The main advantage of the TX power increase is improved accuracy. However, it has the same drawbacks (and larger power consumption) as we have mentioned above for the frequency increase.

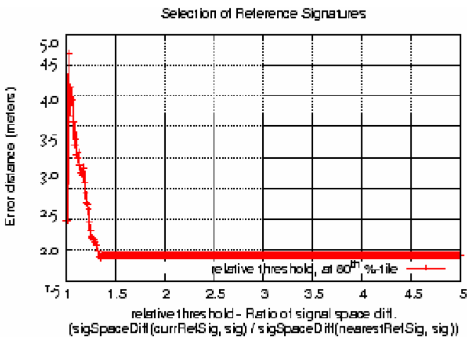
5.5 Selection of Reference Signature and Reference Algorithm

The graphs presented in Figure 5 (a, b) show the error rate according to the type of algorithm applied for reference signature selection.

Our experiment was based mainly on the k-nearest algorithm. For the k-nearest algorithm (Figure 5a), the small values of *k* are applicable for the location centroid computation, but the use of greater values results in significant errors. The relative thresholding graph (Figure 5b) is more accurate as it limits the set of locations considered in accordance with the signature distance metric.



(a)



(b)

Fig. 5. The graphs of reference signature selection with k-nearest algorithm (a) and threshold-nearest algorithm (b)

6 Conclusions and Future Work

This paper describes the architecture of the self-powered module using environmental energy and the current status of its development. Besides, we have presented a decentralized localization and tracking system with the possibility of reference signature database distrbution. The experimental results have demonstrated its accuracy and tolerance to failure.

The robustness of the localization and tracking system can be improved by dividing its architecture into three modes, namely beacon, estimator and mobile motes. This solution is to reduce the beacon mote´s overhead and improve the system performance.

In the very near future we intend to complete the self-powered module and test it together with the localization system for the potential use in scenario *Paintball*. However, the results achieved can be adapted for a variety of power-intensive applications where tracking and localization of mobile object is required.

Acknowledgements

This work was supported in part by a grant from ArsLogica, S.p.A. We would like to thank Fabrizio Stefani, Engineer (ArsLogica), and Prof. Alberto Montresor (University of Trento) for their valuable comments.

References

1. Ohta, Y., Sugano, M., Murata, M.: Autonomous localization method in wireless sensor networks. In: Pervasive Computing and Communication Workshops, March 8-12 (2005)
2. Torah, R.N., Tudor, M.J., Patel, K., Garcia, I.N., Beeby, S.P.: Autonomous low power microsystem powered by vibration energy harvesting. In: 6th annual IEEE Conference on Sensors, Atlanta, USA, October 28-31 (2007)
3. Raghunathan, V., Kansal, A., Hsu, J., Friedman, J., Srivastava, M.: Design considerations for solar energy harvesting wireless embedded systems. IEEE SPOTS (2005)

4. Jiang, X., Polastre, J., Culler, D.: Perpetual environmentally powered sensor networks. IEEE SPOTS (April 2005)
5. Dutta, P., Hui, J., Jeong, J., Kim, S., Sharp, C., Taneja, J., Tolle, G., Whitehouse, K., Culler, D.: Trio: Enabling sustainable and scalable outdoor wireless sensor network deployments. IEEE SPOTS (2006)
6. Localization with GPS. From GPS Theory and Practice, 5th edn. (2005)
7. Polastre, Szewczyk, R., Culler, D.: Enabling ultra-low power wireless research. IEEE SPOTS (2005)
8. Dutta, P., Grimmer, M., Arora, A., Bibyk, S., Culler, D.: Design of a wireless sensor network platform for detecting rare, random and ephemeral events. IEEE IPSN (2005)
9. Mica2 wireless measurement system (November 2007), `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf`
10. TinyOS Programming, `http://csl.stanford.edu/~pal/pubs/tinyos-programming.pdf`
11. Panasonic Industrial (June 2008), `http://www.panasonic-industrial.com`
12. Lorincz, K., Welsh, M.: A robust, decentralized approach to RF-based location tracking. Technical Report TR-19-04, Harvard University (2004)
13. Capkun, S., Hamdi, M., Hubaux: GPS-free positioning in mobile ad-hoc networks. System Sciences. In: Proceedings of the 34th Annual Hawaii International Conference, Fed. de Lausanne, Switzerland (2001)
14. Priyantha, N.B., Chakraborty, A., Balakrishnan, H.: The Cricket location-support system. In: Proceedings of the 6th annual International Conference on Mobile Computing and Networking (2000)
15. Using TREMATE. TRETEC S.r.l., Trento, Italy
16. van Greunen, J.: Services in sensor networks. Master thesis, University of California, Berkley (2003)
17. Harter, Hopper, A., Steggles, P., Ward, A., Webster, P.: The Anatomy of a Context-Aware Application. In: 5th ACM MOBICOM, Seattle, WA (August 1999)
18. Analog Devices (June 2008), `http://www.analog.com`
19. CC1000 Description (June 2008), `http://focus.ti.com/lit/ds/symlink/cc1000.pdf`
20. CC2420 Description (June 2008), `http://enaweb.eng.yale.edu/drupal/system/files/CC2420_Data_Sheet_1_4.pdf`
21. Perpetuum (December 2007), `http://www.perpetuum.co.uk`
22. Ergen, S.C.: ZigBee/IEEE 802.15.4 Summary (2004)
23. David, G., Levis, P., Culler, D., Brewer, E.: nesC 1.1 Language Reference Manual (May 2003)
24. Nesscap, `http://www.nesscap.com/`
25. Lorincz, K., Welsh, M.: MoteTrack: A Robust, Decentralized Approach to RF-Based Location Tracking. In: Strang, T., Linnhoff-Popien, C. (eds.) LoCA 2005. LNCS, vol. 3479. Springer, Heidelberg (2005)
26. Bahl, P., Padmanabhan, V.N.: RADAR: An In-Building RF-based User Location and Tracking System. In: Infocom 2000 (2000)
27. Ekahau positioning engine (September 2008), `http://www.ekahau.com`
28. McNally, R., Arvind, D.: A Distributed Leaderless Algorithm for Location Discovery in Specknets. In: Kermarrec, A.-M., Bougé, L., Priol, T. (eds.) Euro-Par 2007. LNCS, vol. 4641. Springer, Heidelberg (2007)
29. MAXIM (June 2008), `http://www.maxim-ic.com`

Autonomous Deployment of Self-Organizing Mobile Sensors for a Complete Coverage

Novella Bartolini, Tiziana Calamoneri, Emanuele Guido Fusco, Annalisa Massini, and Simone Silvestri

Department of Computer Science, Sapienza University of Rome, Italy
{bartolini,calamo,fusco,massini,simone.silvestri}@di.uniroma1.it

Abstract. In this paper we propose an algorithm for the autonomous deployment of mobile sensors over critical target areas where sensors cannot be deployed manually. The application of our approach does not require prior knowledge of the working scenario nor any manual tuning of key parameters. Our algorithm is completely distributed and sensors make movement decisions on the basis of locally available information. We prove that our approach guarantees a complete coverage, provided that a sufficient number of sensors are available. Furthermore, we demonstrate that the algorithm execution always terminates preventing movement oscillations. We compare our proposal with one of the most acknowledged algorithms by means of extensive simulations, showing that our algorithm reaches a complete and more uniform coverage under a wide range of operating conditions.

1 Introduction

Research in the field of mobile wireless sensor networks is motivated by the need to monitor critical scenarios such as wild fires, disaster areas, toxic regions or battlefields, where static sensor deployment cannot be performed manually. In these working situations, sensors may be dropped from an aircraft or sent from a safe location.

We address the problem of coordinating sensor movements to improve the coverage of an Area of Interest (AoI) with respect to the initial deployment, which typically is neither complete nor uniform. Centralized solutions are inefficient because they require either a prior assignment of sensors to positions or a starting topology that ensures the connectivity of all sensors (for global coordination purposes). Therefore, feasible and scalable solutions should employ a distributed scheme according to which sensors make local decisions to meet global objectives. Due to the limited power availability at each sensor, energy consumption is a primary issue in the design of any self-deployment scheme for mobile sensors. Since sensor movements and, to a minor extent, message exchanges, are energy consuming activities, a deployment algorithm should minimize movements and message exchanges during deployment.

Among the previous proposals, the virtual force approach (VFA) [1,2,3,4] models the interactions among sensors as a combination of attractive and repulsive forces. Other approaches are inspired by the physics as well: in [5] the sensors are modelled as particles of a compressible fluid, in [6] the theory of gas is used to model sensor movements in the presence of obstacles. A similar approach is used in [7] to give a unified solution to the problem of deployment and dynamic relocation of mobile sensors in an open environment.

Other proposals are inspired by computational geometry. Among these, the Voronoi approach (VOR) [8] provides general rules for sensor movements on the basis of a local calculation of the Voronoi diagrams determined by the sensor deployment. A variant of this approach [9] analyzes the problem of sensor deployment in a hybrid scenario, with both mobile and fixed sensors in the same environment. The authors of [10] propose the use of Delaunay triangulation techniques to obtain a regular tessellation of the AoI. In [11] a novel approach to sensor deployment is designed with particular emphasis on operative settings where coverage does not imply connectivity.

Differently from our work, the cited approaches require the manual tuning of constants and thresholds whose proper values are closely dependent on the particular operative setting.

We propose an original algorithm for mobile sensor deployment, PUSH & PULL. It does not require any prior knowledge of the operative scenario nor any manual tuning of key parameters. It constitutes a wide extension of our previous proposal, Snap & Spread [12] to which we added two basic activities to guarantee the coverage completeness and uniformity and to improve the network fault tolerance. PUSH & PULL is based on the interleaved execution of four activities designed to produce a hexagonal tiling by moving sensors out of high density regions and attracting them towards coverage holes. Decisions regarding the behavior of each sensor are based on locally available information.

Our algorithm has the basic self-* properties of autonomic computing, i.e. self-configuration and self-adaptation. Its design follows the grassroots approach [13] to autonomic computing. This way self-organization emerges without the need of external coordination or explicit control, giving rise to a fully decentralized algorithm, according to which the sensor behavior is democratic and peer structured. We formally prove that our algorithm reaches a final stable deployment and a complete coverage of the AoI in a finite time. We ran extensive simulations to evaluate the performance of our algorithm and compare it to existing solutions. Experimental results show that PUSH & PULL performs better than one of the most acknowledged and cited algorithms [8].

2 The Push and Pull Algorithm

Let V be a set of equally equipped sensors with isotropic communication and sensing capabilities. The transmission radius is $R_{\mathtt{tx}}$ and the sensing radius is $R_{\mathtt{s}}$. We propose a distributed algorithm according to which sensors aim at forming a hexagonal tiling over the AoI, with hexagon side length equal to $R_{\mathtt{s}}$. This setting guarantees both coverage and connectivity when $R_{\mathtt{tx}} \geq \sqrt{3}R_{\mathtt{s}}$ and is not restrictive as most wireless devices can adjust their transmission range by properly setting their transmission power. Finally, we assume that sensors are endowed with location capabilities. It should be noted that this assumption is only necessary if the algorithm has to be applied over a specific AoI, with given coordinates. As in [7] this assumption can be removed when dealing with an open environment.

A sensor being positioned in the center of a hexagon is referred to as a *snapped sensor*. Given a sensor x, snapped to the center of a hexagon, we define *slaves of x* all the other sensors lying in its hexagon. We denote by $S(x)$ the set of slaves of x and by $Hex(x)$ the hexagonal region whose center is covered by x. All sensors that are neither snapped nor slaves are called *free*. We define $L(x)$, the set composed by the free sensors located in radio proximity to x and by its slaves $S(x)$.

The algorithm Push & Pull is based on the interleaved execution of four fundamental activities: *Snap*, *Push*, *Pull* and *Merge*. The coordination of these activities requires the definition of a communication protocol that we do not detail due to space limitations. The interested reader may find more information in [14].

Snap activity
At the beginning, each sensor may act as starter of a snap activity from its initial location at an instant randomly chosen over a given time interval $T_{\mathtt{start}}$. Initially, several sensors may likely create different tiling portions. A starter sensor elects its position as the center of the first hexagon of its portion. It selects at most six sensors among those located within its transmission radius and makes them snap to the center of adjacent hexagons. Such snapped sensors, in their turn, give start to analogous activities, thus expanding the boundary of their tiling portion.

More precisely, a snapped sensor x performs a *neighbor discovery*, that allows x to gather information regarding $S(x)$ and all the free and snapped sensors located in radio proximity. After the neighbor discovery, x determines whether some adjacent snapping positions are still to be covered and leads the corresponding snap activity. The sensor x snaps the closest sensor in $L(x)$ to each uncovered position. A snapped sensor leads the snapping of as many adjacent hexagons as possible. If all the hexagons adjacent to $Hex(x)$ have been covered, x stops any further snapping and gives start to the push activity. Otherwise, if some hexagons are left uncovered because no more sensors in $L(x)$ are available, x starts the pull activity.

Push activity
After the completion of the snap activity, a snapped sensor x, may still have some slave sensors inside its hexagon, so $S(x) \neq \emptyset$. In this case, it can proactively push such slave sensors towards lower density areas.

Given two snapped sensors x and y located in radio proximity from each other, x may offer one of its slaves to y and push it inside the hexagon of y if $|S(x)| \geq |S(y)|+1$. Notice that, when $|S(x)| = |S(y)|+1$ the flow of a sensor from $Hex(x)$ to $Hex(y)$ leads to a symmetric situation in which $|S(x)| + 1 = |S(y)|$,

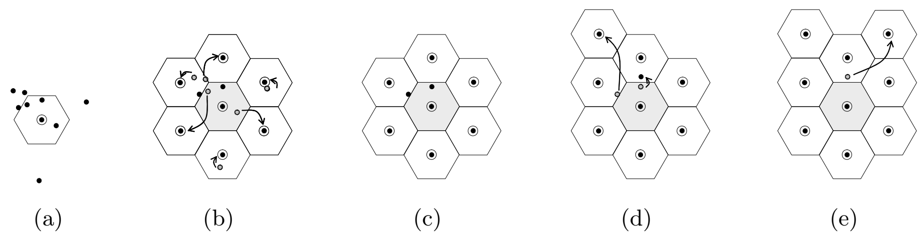


Fig. 1. Snap and Push: an example

leading to possible endless cycles. In such cases we restrict the push activity to only one direction: x pushes its slave to y only if $id(y) < id(x)$, where $id(\cdot)$ is any function such that $id : V \rightarrow \mathbb{N}$.

We formalize these observations by defining the following condition, that enables the movement of a sensor from $Hex(x)$ to $Hex(y)$:
Moving Condition: $\{|S(x)| > |S(y)| + 1\} \ \vee \ \{|S(x)| = |S(y)| + 1 \ \wedge \ id(x) > id(y)\}$.
The snapped sensor x pushes one of its redundant sensors towards the hexagon of the snapped sensor y which has the lowest number of slaves among those in radio connectivity with x. If more than one hexagon contains the minimum number of sensors, the closest to x is preferred. Among its slaves, x selects the sensor to push according to the criterion of minimum traversed distance to $Hex(y)$.

Figure 1 shows an example of the execution of the first two activities. Figure 1(a) depicts the initial configuration, with nine randomly placed sensors and highlights the starter sensor $s_{\texttt{init}}$ creating the first hexagon of the tiling. In Figure 1(b) the starter sensor $s_{\texttt{init}}$ selects six sensors to snap in the adjacent hexagons, according to the minimum distance criterion. Figure 1(c) shows the configuration after the snap activity of $s_{\texttt{init}}$. In Figure 1(d), a deployed sensor starts a new snap activity while $s_{\texttt{init}}$ starts the push activity sending a slave sensor to a lower density hexagon. In Figure 1(e) the deployed sensor snaps the sensor just received from the starter, reaching the final configuration.

As a consequence of the push activity, slave sensors generally consume more energy than snapped sensors, because they are involved in a larger number of message exchanges and movements. Thus we let slave and snapped sensors occasionally exchange their role in order to balance the energy consumption. Any time a slave s has to make a movement across a hexagon occupied by the snapped sensor z, the two sensors perform a role exchange if the residual energy of s is lower than the one of z.

Pull activity
The sole snap and push activities are not sufficient to ensure the maximum expansion of the tiling, and may likely leave coverage holes. Even when the number of available sensors is sufficient to completely cover the AoI, a snapped sensor x could not have any sensor in $L(x)$) to cover the adjacent vacant snapping positions. This may happen due to the Moving Condition introduced to avoid moving cycles. For this reason, we introduce the pull activity: snapped sensors

detecting a coverage hole adjacent to their hexagons, and not having available sensors to snap, send hole trigger messages in order to attract slave sensors and make them fill the hole.

If a snapped sensor x, with $L(x) = \emptyset$, detects a hole, and the Moving Condition is not verified for any of its snapped neighbors, then the following trigger mechanism is activated. The sensor x temporarily alters the value of its id function to 0 and notifies its neighbors of this change. Then x waits until either a new slave comes into its hexagon or a timeout occurs. If a new slave enters in $Hex(x)$, x sets back its id value and snaps the new sensor, filling the hole. If otherwise the timeout expires and the hole is still present, the trigger mechanism is extended to the adjacent hexagons of x, whose snapped sensors set their id value to 1 and notify their neighbors.

Each snapped sensor involved in the trigger extension mechanism sets its id to a value that is proportional to the distance from x. All the timeouts related to each new extension are set in proportion to the maximum distance reached by the trigger mechanism. This mechanism is iterated by x over snapped sensors at larger and larger distance in the tiling until the hole is covered. At this point, as a consequence of timeouts, each involved node sets back its id to the original value.

Observe that, more snapped sensors adjacent to the same hole may independently activate the trigger mechanism. In this case, the only message with lowest id is honored. The detection of several holes may cause the same node to receive more a number of trigger messages that are stored in a pre-emptive priority queue, giving precedence to the messages related to the closest hole.

Tiling merge activity

The fact that many sensors act as starters implies the possible creation of several tiling portions with different orientations. The tiling merge activity starts when two tiling portions come in radio proximity. We propose a merge mechanism according to which as soon as a sensor x receives a neighbor discovery message from another tiling portion, it chooses to belong to the oldest one. The sensor x discriminates this situation by evaluating the time-stamp of the starter action that is propagated at each snap action.

Figure 2 shows an example of the execution of the tiling merge activity. Figure 2(a) shows two tiling portions meeting each other. The portion on the left has the

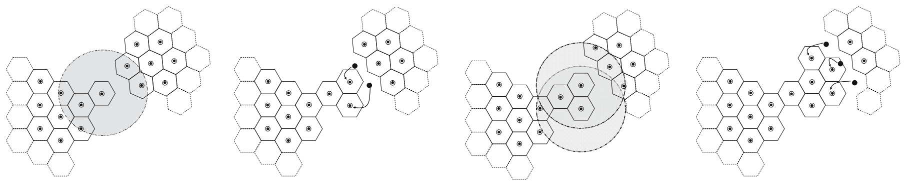


Fig. 2. Tiling merge activity: an example

oldest time-stamp, hence it absorbs the other one. Two nodes of the right portion detect the presence of an older tiling and abandon their original tiling to honor snap commands coming from a sensor of the left portion (Figure 2(b)). These just snapped nodes, now belonging to the older portion, detect the presence of three nodes belonging to the right one (Figure 2(c)) and snap them as soon as they leave their portion (Figure 2(d)).

3 Algorithm Properties

In this section we discuss some key properties of the Push & Pull algorithm: coverage, connectivity and termination.

3.1 Coverage and Connectivity

We denote by $N_{\texttt{tight}}$ the *tight number of sensors*, that is the maximum number of hexagons necessary to cover the AoI for each possible initial position of the sensor set and each possible tiling orientation. Notice that an upper bound on this number can be calculated by increasing the AoI with a border whose width is the maximal diameter of the tiling hexagon and dividing this increased area AoI' by the area of the hexagon. Formally, $N_{\texttt{tight}} \leq \lceil \frac{\text{Area(AoI')}}{\text{Area(Hex)}} \rceil$. This upper bound is valid regardless of the number of tiling portions generated by different starter sensors.

Theorem 1. *Algorithm* Push & Pull *guarantees a complete coverage, provided that at least the tight number of sensors are available.*

Proof. Let us assume that a coverage hole exists and let x be the sensor which detects it. The hypothesis on the number of sensors implies that it certainly exists a hexagon with at least one redundant slave. Let us call C_x the connected component containing the sensor x. Two different cases may occur depending on the position of redundant slaves with respect to C_x:

- A redundant slave exists in C_x.
  The snapped sensor x starts the trigger mechanism that eventually reaches a hexagon with a redundant slave. Such a slave is then moved towards x and finally fills the hole.
- All redundant slaves belong to connected components separated from C_x.
  The area surrounding each connected component is in fact a coverage hole that will eventually be detected by a snapped node located at the boundary. According to the previous item, all the separated connected components containing redundant slaves will expand themselves to fill as many coverage holes as possible. Since by the hypothesis the number of sensors is at least $N_{\texttt{tight}}$, it certainly exists a component containing redundant slaves that will eventually merge with C_x, leading to the situation described in the first item, thus proving the theorem.

We assume that sensors operate with $R_{\mathtt{tx}} \geq \sqrt{3}R_{\mathtt{s}}$. Simple geometrical considerations allow us to conclude that under this assumption, a hexagonal tiling with side $R_{\mathtt{s}}$, i.e. where the distance between any two grid neighbors is $\sqrt{3}R_{\mathtt{s}}$, guarantees minimal node density (as argued in [15]) and connectivity.

3.2 Termination of Push and Pull

Let $L = \{\ell_1, \ell_2, \ldots, \ell_{|L|}\}$ be the set of snapped sensors.

Definition 1. *A* network state *is a vector* $\mathbf{s} =< s_1, s_2, \ldots, s_{|L|} >$*, where* $s_i = |S(i)| + 1$*, is the number of sensors deployed inside the hexagon* $Hex(i)$*,* $\forall i = 1, \ldots, |L|$.

Definition 2. *A state* **s** *is* stable *if the Moving Condition is false for each couple of snapped sensors located in radio proximity to each other.*

Theorem 2. *Algorithm* Push & Pull *terminates in a finite time.*

Proof. Due to Theorem 1, the expansion of the tiling generated by Push & Pull eventually ends either because all sensors have been snapped or the AoI has been completely covered by snapped sensors. In the first case, no algorithm actions are necessary, then the algorithm terminates producing a stable state of the network. Thus, in order to prove the theorem, it suffices to prove that, once the AoI is fully covered by snapped sensors, the algorithm produces a stable network state in a finite time. After the complete coverage of the AoI, the set L of snapped sensors remains fixed. The value of the order function related to each snapped sensor, $id(\ell_i)$ can be modified by the pull activity only a finite number of times and remains steady onward.
Let us define $f : \mathbb{N}^{|L|} \to \mathbb{N}^2$ as follows: $f(\mathbf{s}) = \left(\sum_{i=1}^{|L|} s_i^2, \sum_{i=1}^{|L|} id(\ell_i)s_i\right)$.
We say that $f(\mathbf{s}) \succ f(\mathbf{s}')$ if $f(\mathbf{s})$ and $f(\mathbf{s}')$ are in lexicographic order. Observe that function f is lower bounded by the pair $(|L|, \sum_{i=1}^{|L|} id(\ell_i))$, in fact $1 \leq s_i \leq |V|$. Therefore, if we prove that the value of f decreases at every state change from $\mathbf{s}$ to $\mathbf{s}'$, we also prove that no infinite sequence of state changes is possible.

Let us consider a generic state change which involves the snapped sensors x and y, with x sending a slave sensor to $Hex(y)$. We have that $s_i = s_i' \quad \forall i \neq x, y$, and $s_x' = s_x - 1$ and $s_y' = s_y + 1$. As the transfer of the slave has been done according to the Moving Condition, two cases are possible: either $s_x > s_y + 1$, or $(s_x = s_y + 1) \wedge (id(x) > id(y))$. In the first case, $s_x > s_y + 1$ trivially implies that $\sum_{i=1}^{|L|} s_i^2 > \sum_{i=1}^{|L|} {s'}_i^2$. In the second case, from $s_x = s_y + 1$ and $id(x) > id(y)$, easy calculations imply that $\sum_{i=1}^{|L|} s_i^2 = \sum_{i=1}^{|L|} {s'}_i^2$ and $\sum_{i=1}^{|L|} id(\ell_i)s_i > \sum_{i=1}^{|L|} id(\ell_i){s'}_i$. Therefore in both cases $f(\mathbf{s}) \succ f(\mathbf{s}')$. The function f is lower bounded and always decreasing of discrete quantities (integer values) at any state change. Thus, after a finite time the network will be in a *stable state*, thus the theorem is proved.

4 Simulation Results

In order to evaluate the performance of Push & Pull and to compare it with previous solutions, we developed a simulator on the basis of the wireless module of the OPNET modeler software [16]. We compared our proposal to one of the most acknowledged and cited algorithms [8], which is based on the use of Voronoi diagrams. According to this approach, each sensor adjusts its position on the basis of a local calculation of the Voronoi cell determined by the current sensor deployment. This information is used to detect coverage holes and consequently calculate new target locations according to three possible variants. Among these variants we chose MiniMax, that is the one that gives better guarantees in terms of coverage extension. We also adopted all the mechanisms provided in [8] to preserve connectivity, to guarantee the algorithm termination, to avoid oscillations and to deal with position clustering. In the rest of this section this algorithm will be named VOR.

The experimental activity required the definition of some setup parameters: $R_{\mathtt{tx}} = 11$ m and $R_{\mathtt{s}} = 5$ m. This setting does not significantly affect the qualitative evaluation of Push & Pull but is motivated by the need to satisfy the requirement $R_{\mathtt{tx}} \geq 2R_{\mathtt{s}}$ given in [8]. The sensor speed is 1 m/sec. The energy spent by sensors for communications and movements is expressed in energy units (i.e. the cost of receiving one message): a single transmission costs the same as 7 receptions [17], a 1 meter movement costs the same as 300 transmissions [8] and a starting/braking action costs the same as 1 meter movement [8].

The length of the time interval $T_{\mathtt{start}}$ is set to $R_{\mathtt{tx}}/v$, where v is the sensor movement speed. The setting of this parameter ensures that different grid portions are not created too close to each other. Nevertheless, it does not affect the algorithm performance significantly.

Before showing the performance of our algorithm with respect to VOR, we show some examples of final deployments provided by the two approaches.

Figures 3 and 4 show the sensor deployment over a 80 m $\times$ 80 m squared AoI. The two initial deployments reflect the realistic scenarios in which sensors are dropped from an aircraft (Figure 3(a)) and sent from a safe location at the boundaries of the AoI (Figure 4(a)). For both figures, subfigure (a) represents the initial deployment, while subfigures (b) and (c) show the final deployment obtained by Push & Pull and VOR respectively.

We now show the performance of our algorithm with respect to VOR, starting from the initial deployments described above. We studied the algorithm behavior by varying the number of available sensors. In order to give a reliable performance

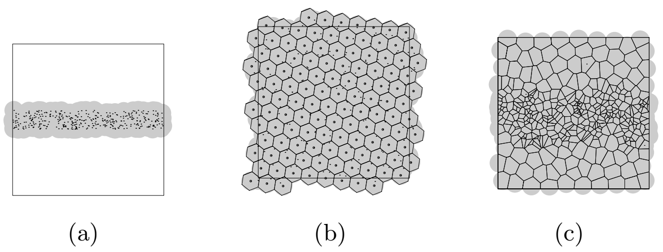


Fig. 3. Comparison between Push & Pull and VOR - Trail initial deployment

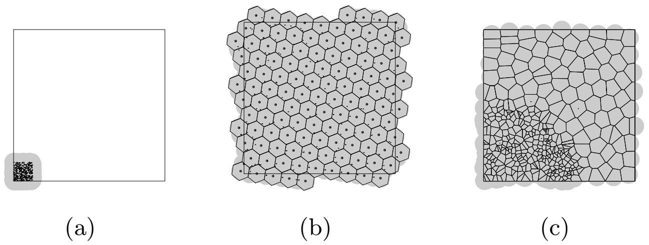


Fig. 4. Comparison between Push & Pull and VOR - Safe-location initial deployment

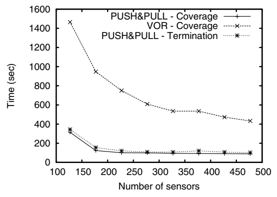


Fig. 5. Term. and coverage time

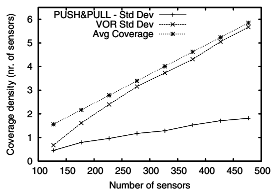


Fig. 6. Coverage density

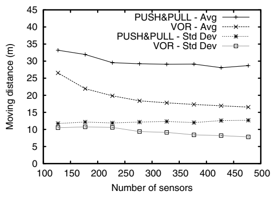


Fig. 7. Traversed distance

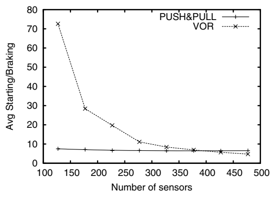


Fig. 8. Starting and braking

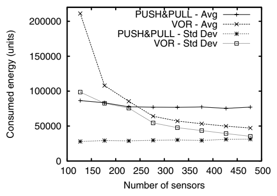


Fig. 9. Energy consumption

comparison, we show the average results of 30 simulation runs conducted by varying the seed for the generation of the initial random deployment of the sensors.

Figures 5 through 9 show the performance results for the first set of experiments regarding the trail initial deployment.

Figure 5 shows the coverage and termination time for both the Push & Pull and the VOR algorithms. Notice that, when the number of sensors is close to the tight value, defined in Section 3.1, VOR requires a very long time to achieve a complete coverage, while Push & Pull terminates much earlier. When the number of sensors increases, both algorithms terminate faster, but VOR always requires more time than Push & Pull to achieve its final coverage. Note also that while for the VOR algorithm the termination and coverage completion times coincide, because it moves sensors with the only objective to increase coverage, for Push & Pull some more movements still occur even after the coverage completion. These movements are performed to keep on uniforming the sensor density.

In order to evaluate the coverage uniformity, we compute the coverage density as the number of sensors covering the points of a squared mesh with side 1 m. Figure 6 shows the standard deviation of the coverage density. The standard deviation of the coverage density obtained by Push & Pull is much smaller pointing out a more uniform sensor placement. This result is particularly important as a uniform sensor redundancy is necessary to guarantee fault tolerance and to prolong the network lifetime with selective sensor activation schemes.

Figure 7 shows the average and standard deviation of the distance traversed by the sensors. Push & Pull let sensors move more than VOR because it aims at making the coverage as uniform as possible. Notice that both the average and the standard deviation of the traversed distance of VOR are decreasing with the number of sensors since more and more sensors maintain their initial positions. It should be noted that the result of this comparison must not be interpreted as a negative aspect of our protocol. Indeed, Push & Pull keeps on moving sensors until a quite uniform coverage is reached while the movements determined by VOR terminate as soon as the AoI is completely covered. Hence the average and standard deviation of the traversed distance are more stable under Push & Pull than under VOR when the number of sensors varies.

Figure 8 highlights that VOR spends much more energy than Push & Pull in starting/braking actions. The average value of such energy cost decreases with a growing number of available sensors as the majority of them do not move at all under VOR.

In Figure 9 we give a global evaluation of the above contributions, and show the average and standard deviation of the total consumed energy (i.e. the sum of the contributions due to movements, starting/braking and communications). This figure shows that when the number of sensors is small (lower than about 250 for this experimental setting), although VOR consumes less energy in movements, the impact of starting/braking actions is not negligible and compensate the higher cost of movements paid by Push & Pull. When the number of sensors grows, VOR consumes less energy with respect to Push & Pull as a large part of the sensors are left unmoved.

In all the simulated situations, the execution of VOR implies that a number of sensors are moved away from overcrowded regions toward uncovered areas. As soon as all the coverage holes are eliminated, VOR stops, leaving some zones with very low density coverage. Such zones represent possible points of future failures and coverage holes. Push & Pull mitigates this problem by sending much more sensors than VOR to the farthest and less dense regions of the AoI.

In the second set of experiments the initial deployment consists in a high density region at the boundaries of the AoI as depicted in Figure 4(a). It is worth noting that this initial deployment constitutes a critical scenario for VOR as this algorithm works at its best for more uniform initial sensor distributions. Indeed, Figure 10 shows that VOR requires much more time than in the previous set of experiments to achieve its final deployment. This figure also shows that

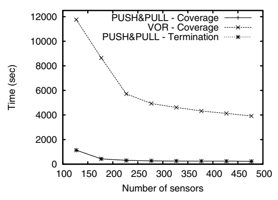


Fig. 10. Term. and coverage time

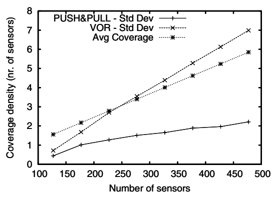


Fig. 11. Coverage density

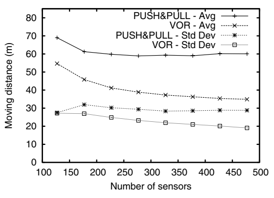


Fig. 12. Traversed distance

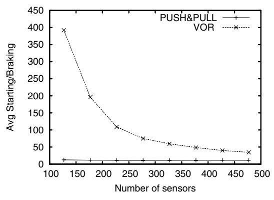


Fig. 13. Starting and braking

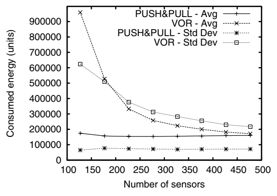


Fig. 14. Energy consumption

VOR is much slower than Push & Pull in completing the coverage of the AoI. For what concerns the density of the distribution, Figure 11 shows that even in this operative setting, VOR terminates as soon as the AoI is completely covered, without uniforming the density of the sensor deployment. This implies that VOR spends less energy in movements (see Figure 12) than Push & Pull but at the expense of the quality of the final coverage in terms of uniformity. Furthermore, VOR shows a much higher number of starting/braking actions (see Figure 13) than Push & Pull, with a much higher value of the total consumed energy (see Figure 14).

5 Conclusions and Future Work

We proposed an original algorithm for mobile sensor self-deployment named Push & Pull. According to our proposal, sensors autonomously coordinate their movements in order to achieve a complete coverage with moderate energy consumption. The execution of Push & Pull does not require any prior knowledge of the operating conditions nor any manual tuning of key parameters as sensors adjust their positions on the basis of locally available information. The proposed algorithm guarantees the achievement of a complete and stable final coverage, provided that there is a sufficient number of sensors. Some improvements are being considered as a future extension of this work. In particular, it seems reasonable that the algorithm can be generalized in order to guarantee a k-coverage. Mechanisms for obstacle detection and avoidance are also being investigated.

References

1. Zou, Y., Chakrabarty, K.: Sensor deployment and target localization based on virtual forces. In: Proc. IEEE INFOCOM 2003 (2003)
2. Heo, N., Varshney, P.: Energy-efficient deployment of intelligent mobile sensor networks. IEEE Transactions on Systems, Man and Cybernetics 35 (2005)
3. Chen, J., Li, S., Sun, Y.: Novel deployment schemes for mobile sensor networks. Sensors 7 (2007)
4. Poduri, S., Sukhatme, G.S.: Constrained coverage for mobile sensor networks. In: Proc. of IEEE Int'l Conf. on Robotics and Automation, ICRA 2004 (2004)
5. Pac, M.R., Erkmen, A.M., Erkmen, I.: Scalable self-deployment of mobile sensor networks; a fluid dynamics approach. In: Proc. of IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems, IROS 2006 (2006)
6. Kerr, W., Spears, D., Spears, W., Thayer, D.: Two formal fluid models for multi-agent sweeping and obstacle avoidance. In: Kudenko, D., Kazakov, D., Alonso, E. (eds.) AAMAS 2004. LNCS, vol. 3394. Springer, Heidelberg (2005)
7. Garetto, M., Gribaudo, M., Chiasserini, C.F., Leonardi, E.: A distributed sensor relocation scheme for environmental control. In: The Fourth ACM/IEEE Int. Conf. on Mobile Ad-hoc and Sensor Systems, MASS (2007)
8. Wang, G., Cao, G., Porta, T.L.: Movement-assisted sensor deployment. IEEE Transaction on Mobile Computing 6 (2006)
9. Wang, G., Cao, G., Porta, T.L.: Proxy-based sensor deployment for mobile sensor networks. In: IEEE Int. Conf. on Mobile Ad-hoc and Sensor Systems, MASS (2004)
10. Ma, M., Yang, Y.: Adaptive triangular deployment algorithm for unattended mobile sensor networks. IEEE Transactions on Computers 56 (2007)
11. Tan, G., Jarvis, S.A., Kermarrec, A.M.: Connectivity-guaranteed and obstacle-adaptive deployment schemes for mobile sensor networks. In: Proc. of ICDCS (2008)
12. Bartolini, N., Calamoneri, T., Fusco, E.G., Massini, A., Silvestri, S.: Snap & spread: a self-deployment algorithm for mobile sensor networks. In: Nikoletseas, S.E., Chlebus, B.S., Johnson, D.B., Krishnamachari, B. (eds.) DCOSS 2008. LNCS, vol. 5067. Springer, Heidelberg (2008)
13. Babaoglu, O., Jelasity, M., Montresor, A.: Grassroots approach to self-management in large-scale distributed systems. In: Banâtre, J.-P., Fradet, P., Giavitto, J.-L., Michel, O. (eds.) UPP 2004. LNCS, vol. 3566. Springer, Heidelberg (2005)
14. Bartolini, N., Massini, A., Silvestri, S.: P&p protocol: local coordination of mobile sensors for self-deployment (2008), `http://arxiv.org/abs/0805.1981`
15. Brass, P.: Bounds on coverage and target detection capabilities for models of networks of mobile sensors. ACM Transactions on Sensor Networks 3 (2007)
16. Opnet technologies inc: `http://www.opnet.com`
17. Smart dust: `http://www-bsac.eecs.berkeley.edu/archive/users/warnekebrett/SmartDust/index.html`

A Self-Organized Head Selection for Hierarchical Routing in Wireless Sensor Networks*

Heesang Lee, Kyuhong Lee, and YounHo Lee

Dept. of Systems Management Eng., Sungkyunkwan University, Korea
leehee@skku.edu, hideto25@nate.com, sksdkrkal@nate.com

Abstract. Efficient energy consumption is critical for deployment and operation of wireless sensor networks (WSNs). There have been several suggestions using a hierarchical approach that organizes sensor nodes into clusters and uses one node as a cluster head to forward message received from its cluster member nodes to the base station of WSN. Efficient head selection is a major issue for the cluster-based hierarchical routing protocols. We propose a self-organized algorithms for cluster head selection and cluster-based routing. To decide cluster heads and organize cluster members, each sensor node uses only local information and simple rules which are inspired by nature. By these self-organized interactions among sensor nodes, the suggested method can form clusters for a WSN and decide routing paths efficiently. We compare our algorithms with two known routing protocols which are based on random head selections and transmitter-based code assignment schemes. In our computational experiments, we show that the energy consumption and the lifetime of our algorithm are better than those of the compared protocols.

Keywords: Wireless Sensor Network, Self-Organization, Cluster Head, Hierarchical Routing.

1 Introduction

The sensors should operate own tasks; sensing or monitoring the environment and transmitting or receiving the data. To achieve reliable and accurate operation of sensors, hundreds or thousands of wireless connected micro-sensor nodes are used in the form of a wireless sensor network (WSN) [1]. Since a sensor node is tiny, a low-power sensing device and used for a long period (without any replacements), energy-efficiency is an important issue for a WSN.

Recently, the studies of self-organized, self-adaptive and self-configuring are advanced in many fields. These autonomic systems (AS) could deliver a considerable decrease of operational expenditure in many application fields [2]: manufacture or production, management and telecommunication. One of them is an operation of a WSN that consists of a plenty of sensor nodes without a centralized control mechanism.

* "This work was supported by the Korea Research Foundation Grant funded by the Korean Government (MOEHRD) (KRF-2005-041-D00885)".

Communication channel for a WSN can be established via multi-hop mesh network. In this case routing is the problem to decide a transmission route from a sensor node to the *base station* that is a final processing node for the WSN. We call this routing *flat routing*. On the other hand, a WSN can be decomposed into several *clusters*. Each cluster has a special sensor node called a *cluster head* and several ordinary sensor nodes called *cluster members*. For this case, data packets that are transmitted from cluster members should be sent to the base station only via the cluster head of the same cluster. We call this routing a *hierarchical routing* [3].

Selecting the cluster head randomly is one of the approaches for self-organization in the cluster-based WSNs. In this approach, the main focus is decentralized clustering that uses the randomness to avoid external control or intervention. Randomness is a good instance for AS and very simple design to implement in a WSN. Another work is called *nature-inspired computing* approach that is motivated by biology and the natural world: for example social animals such as ants and birds.

In this paper, we propose a decentralized algorithm that is motivated by nature for organizing a WSN into clusters. We want to compare our approach with a randomness based approach. In our approach each sensor node broadcasts messages to the neighborhood and uses information from the received messages to decide a cluster head for each cluster of a WSN. The main information is the signal strength that is broadcasted from the neighbor sensor nodes of each sensor node. All sensor nodes collect and compare the signals and obtain information that comes from neighbor sensor nodes. Furthermore, since all the sensor nodes are able to control the power of signal, each sensor node can obtain necessary neighbor information by changing the transmission power if needed.

The rest of the paper is organized as follows. In section 2, we investigate hierarchical routing and cluster head selection mechanism and nature inspired approaches for cluster-based routing in WSNs. In section 3, we present our nature inspired model and related algorithms. In section 4, we try to set several parameters for efficient implementation. In section 5 we perform some computational experiments to compare our approach with two known randomness based cluster head selection mechanism. In section 6 we present some further research topics and conclude the paper.

2 Related Work

Several cluster-based protocols using the randomness have been studied. The most well known protocol of this type is LEACH protocol [4]. In the LEACH protocol, it is basic idea to select sensor nodes randomly as cluster head nodes. In this protocol, the operation of LEACH is repeated at each *round* and the round is divided into two phases, the *set-up* phase and the *steady* phase. In set-up phase, the cluster heads are determined randomly and the network is partitioned into several clusters. In this phase, each node compares random number with threshold $T(n)$ to elect itself to a cluster head and this process is performed independently for each cluster. In steady phase, the sensor nodes being sensing and transmitting data and cluster heads aggregate the data before sending the data to the base station.

XLEACH [5] has been proposed as an extended version of the LEACH protocol. The main idea of XLEACH is that it considers the sensor node's remaining energy level in cluster head selection process. Due to such effort, XLEACH claims to improve the life time of WSN.

The above two protocols are based on the randomness. To make up for the drawbacks of randomness, they added other factors such as cluster head rate, history of the cluster head and energy consumption rates. Though these efforts give each sensor node to a fair possibility becoming a cluster head they however, still cannot try to find good selection of heads.

The most widely known algorithm of nature-inspired computing is ant colony optimization (ACO). The ACO exploits the process behind the ant colony's foraging behavior to obtain many optimization problems. In [6], T-ANT protocol is designed as a hierarchy and scalable data gathering protocol in WSNs. The algorithm uses a swarm of ants to control the cluster head selection and is able to guarantee the uniform distribution of cluster heads.

3 Model and Algorithms

3.1 Assumptions

In this study we assume that the locations of the sensor nodes and the base station are fixed. Each data packet of a sensor node is generated per time unit called *period.* Every data packet has the same size, k bits and each sensor node has finite energy E_i which can not be recharged.

We need not decide on cluster heads for each sensing period, since this frequent change makes the operation of the WSN instable. Hence in our model we use the concept of "*rounds*" which is a set of consecutive periods for the same clustering configuration.

In a clustering model all sensor nodes are partitioned into several clusters. Each cluster consists of one cluster head and several cluster members. Hence exactly one cluster head should be selected for each cluster and each cluster member must be assigned to exactly one cluster head. A cluster member can communicate with the base station only through a cluster head. The life of a sensor network may have several thousand sensing periods.

A sensor node consumes energy to transmit and receive data packets. In wireless data transmission, energy consumption is correlated with the data packet size and the distance between two sensor nodes. We use the following *first order radio model* as presented in [2] as our energy consumption model.

- When transmitting the data packet, a sensor node consumes $\varepsilon_{elec} = 50nJ/bit$ at the transmitter circuitry and $\varepsilon_{amp} = 100pJ/bit/m^2$ at the amplifier.
- When receiving the data packet, a sensor node consumes $\varepsilon_{elec} = 50nJ/bit$ at the receiver circuitry.
- When a k-bit data packet is transmitted from sensor node i to and sensor node j, and d_{ij} is the distance between two sensor nodes i and j, energy consumption of sensor node i is given by,

$$T_{ij} = \varepsilon_{elec} \times k + \varepsilon_{amp} \times d_{ij}^2 \times k \ . \tag{1}$$

– While sensor node *i* receives *k*-bit data packet, energy consumption of sensor node *i* is given by,

$$R_i = \varepsilon_{elec} \times k \ . \tag{2}$$

The transmission of message has a symmetric propagation channel. That is, the same energy is required to transmit the data from sensor node *i* to sensor node *j* and the data from sensor node *j* to sensor node *i*

3.2 SOHS Protocol

In this section we propose a protocol called *Self-Organized Head Selection* (SOHS) protocol. SOHS organizes clusters itself without any help from the base station. SOHS is operated by local information and several simple rules which are inspired by nature. Our main idea is inspired with a semi-aquatic spider *Dolomedes triton.* The D. triton performs consisting of leg-waving, drumming, and jerks that cause bursts of concentric water surface waves which provide information for the female [7]. The semi-aquatic spider determines the direction from a difference of the slope of his legs. These differences are caused by other semi-aquatic spiders or the wavelength of the ripple. Similarly, our model uses the radio frequency signal to determine the distance and identity of the neighboring sensor nodes.

The lifetime of SOHS has many rounds. Each round consists of a *set-up phase*, when cluster head sensor node is decided for this round, and a *steady phase*, when data transmission is performed. In the first round, instead of an ordinary set-up phase there is a special set-up phase, so called *set-up phase for initial cluster head selection*, which needs extra initialization works. The initial cluster head selection phase is operated once when sensor nodes are dissipated in the sensor field.

Every sensor node is at a state of four possible states; *undecided state, head candidate state, cluster head state* and *member state*. If a sensor node is at undecided state, then the sensor node does neither act as a cluster head nor a cluster member. When a sensor node receives more signals than a certain threshold, the sensor node reaches head candidate state. The sensor node only at head candidate state can participate in a head selection process. If a sensor node is selected as a cluster head, then the sensor node reaches cluster head state. If a sensor node is at member state, it means the sensor belongs to one cluster and becomes a cluster member.

3.2.1 Set-Up Phase for Initial Cluster Head Selection

Set-up phase for initial cluster head selection consists of three steps which are *broadcasting step, head selection step* and c*lustering step.*

Broadcasting Step

After sensor nodes are dissipated in the sensor field, no sensor node has the necessary information to start communication. To assign the first cluster head for each cluster, all sensor nodes start to broadcast *first-signal* to the every sensor node within *p* meters. First-signal has information about transmission sensor. Each sensor node counts the number of received first-signals from other sensor nodes. If a sensor node has received first-signals more than a threshold *t*, then the sensor node becomes a head candidate.

Head Selection Step

A head candidate broadcasts *candidate-signal* to the every sensor node within *p* meter. The broadcasting range of the candidate-signal is as the same as that of broadcasting step. Candidate-signal has information about the number of received signals. The head candidates that received candidate-signals from other head candidates compare the number of received candidate-signals. If a head candidate has more number of received candidate-signals, then this candidate node is a better head candidate. For example, let's assume that a head candidate A broadcasts candidate-signal to a head candidate B. If the head candidate A has fewer number of received candidate-signals than that of the head candidate B, then the head candidate B send a *candidate-cancel signal* to the head candidate A. The head candidate A changes its state from a head candidate state to an undecided state. If a head candidate does not receive candidate-signals which have more number of received candidate-signals within a given time interval, then this head candidate is the best one. It becomes a cluster-head.

Clustering Step

To assign undecided sensor nodes to a cluster head, the cluster head broadcast *head-signals* within whole sensor field. Then all undecided sensor nodes received this head-signals find the nearest cluster head and join that cluster. In this step, a cluster head calculates its *head coverage* that is the distance between the cluster head and the farthest cluster member. All sensor nodes start from the undecided state, and end the initial cluster head selection phase as two absorbing states, cluster head state and member state. Hence at the end of this phase SOHS decides initial cluster heads and cluster members. All sensor nodes are now ready for the following steady phase for data transmission.

3.2.2 Steady Phase for Data Transmission

After initial cluster head selection phase completes, cluster members sense surrounding and transmit a data packet to its cluster head for each period. Each cluster head aggregates data packets and transmit to the base station. This procedure repeats during one round.

3.2.3 Set-Up Phase for Cluster Head Change

A WSN needs perform cluster head change phase after steady phase for data transmission. Set-up phase starts from the end of a steady phase. Set-up phase for cluster head change consists of three steps which are *broadcasting step, head selection step* and c*lustering step* as the same as the set-up phase of the initial cluster head selection.

Broadcasting Step

Each cluster heads broadcast *head-change-signal* within *x%* of head coverage to choose next cluster head at the near the current cluster head, then the cluster head node changes its state from a cluster head state to an undecided state. Each cluster member that receives a head-change-signal from its cluster head changes its state from a member state to a head candidate state. A cluster member that does not receive head-change signal from its cluster head within a given time interval changes its state from a member state to an undecided state.

Head-selection Step

Head candidates broadcast *candidate-signals* within *y%* of the head coverage of its cluster head. Each head candidate counts the number of received candidate-signals from other head candidates. Each head candidates broadcast the number of received candidate-signals within the whole head coverage of its cluster head. The maximum value of the number of received candidate-signals is stored at each head candidates of the same cluster. All head candidates calculate fitness values. (We explain how to calculate and use the fitness value in the next section.)

Each head candidates broadcast its fitness value to the every sensor node within the head coverage of its cluster head. The broadcasting range of the candidate-signal is as the same as that of broadcasting step. The head candidates which received fitness values from other head candidates compare them. For example, we assume that a head candidate A broadcasts its fitness value to a head candidate B. If the head candidate A has less fitness value than that of the head candidate B, then the head candidate B sends candidate-cancel signal to the head candidate A. The head candidate A changes its state from a head candidate state to a undecided state. The head candidate A that has more fitness value than the head candidate B, then the head candidate B changes its state from a candidate state to a undecided state. Having a better fitness value means that the head candidate is a better head candidate. A head candidate that does not receive better signals from other candidates during a given time interval, the head candidate is the best head candidate. The head candidate becomes a cluster-head.

Clustering Step

To assign undecided sensor nodes to a cluster head, the cluster heads broadcast head-signal within whole sensor field. Then all undecided sensor nodes received this head-signal find nearest cluster head and join that. The whole procedure of this clustering step is the same as the procedure of the clustering step of the set-up phase for initial cluster head selection.

3.3 Fitness Values

In SOHS, the most important consideration is calculation of fitness values and its use for cluster head selection. According to fitness values, performance of SOHS can be improved a lot. First, we use the number of received signals and the maximum number of received signals among the same cluster at same round. These are very important values since the number of received signals reflects the number of sensor nodes that locate in the neighborhood. A sensor node has more received signals than others, that is better head candidate. Therefore we use the ratio of the number of received signals and the maximum number of received signals of a cluster. We consider this value as a first component of our fitness function.

$$\text{value}\,1 = \frac{\text{the number of received signal}}{\text{maximum number of received signal}} \times 10 \ . \tag{3}$$

Second, we consider the remaining energy of each sensor node. Since cluster head consumes much more energy than that of cluster members, balanced energy consumption by choosing cluster heads in fairness is important.

$$\text{value2} = \frac{\text{current remaining energy}}{\text{initial remaining energy}} \times 10 \ . \tag{4}$$

Using these two fitness factors we finally have the following fitness value:

$$\text{fitness value} = \text{value } 1^{(1-\alpha)} + \text{ value } 2^{\alpha} \ . \tag{5}$$

4 Parameter Settings

In SOHS we have several parameters to be set for good operation of SOHS as the followings. We have simulated SOHS using C programming language. The sensor field is assumed 50m * 50 m of plain area and initial energy is set to 0.5J for any experiment. For the number of sensor nodes in the field varies from 100 to 500.

To obtain a reasonable *p* value, we experiment broadcasting range of sensors from 7 meter to 20 meter. To obtain a threshold *t*, the ratio of threshold to the number of sensors varies from 8% to 12%. To obtain a reasonable *x*, broadcasting range varies from 10% to 100% of the head coverage. To obtain a reasonable *y,* broadcasting range varies from 10% to 100% of the head coverage. To obtain a reasonable *a*, we experiment *a* from 0.05 meter to 0.95. We set these parameters by using preliminary computational experiments and obtain the following parameter setting.

Table 1. Parameter Setting

Notation	Definition	Parameter Setting
p	Broadcasting range of undecided sensor nodes	10 meter
T	Threshold of Number of Received Signals	10% of the number of total sensor nodes
x%	Broadcasting range of head-change-signal in broadcasting step	50% of head coverage
y%	Broadcasting range of candidate-signal in head selection phase	50% of head coverage
a	Parameter of fitness value	0.15

5 Computational Experiments

In this section we want to show that the performance of SOHS is valid and efficient enough for reasonable size WSNs by using computer simulations. To achieve this goal, we have performed several computational experiments that compare our model with other models.

The reference network of our simulations is composed of 100 nodes distributed randomly and the sensor field is 50m * 50 m of plain area which is represented by a coordinate (x, y), where x and y are real numbers from 0 to 50. In this artificial sensor field we locate a given number of sensor nodes randomly as if we scatter micro sensor nodes in a real sensor field.

Initial energy of sensor nodes is 0.5J. We have run the same set of experiments with 5 different sensor node distributions to gain the average performance of SOHS protocols. The base station is located at coordinate (25,150) except experiment 5.4. Every sensor nodes transmit a 1,000-bits message per each period and the signal size is 50-bits. In the experiments, we have measured the number of period called the *lifetime* of a WSN that is the number of periods when the first sensor node is drained of its energy.

5.1 Initial Energy and Periods Per Round

The comparison models are LEACH and XLEACH those are typical randomness-based protocols. They include different improving factors. LEACH performs generic random head selection with a threshold for decision. XLEACH has an additional factor in a threshold that is current energy rate of each sensor node. LEACH and XLEACH have a fixed rate *P*, percentage of cluster head. We have assumed that the parameter *P* is fixed to 0.05 for LEACH and XLEACH as in many previous researches. For operation of SOHS a threshold of the number of received signal is set to (the number of sensor nodes) * 0.1.

To compare our model with LEACH and XLEACH we have simulated our model using C programming language and MATLAB. We have used different number of periods per a round.

Fig.1 shows the lifetime of WSNs when the number of periods per round varies from 5 to 25. The number of sensor nodes in this field is 100. SOHS has always better lifetime than those of LEACH and XLEACH. The gaps between SOHS and LEACH and the gaps between SOHS and XLEACH are larger than 25%.

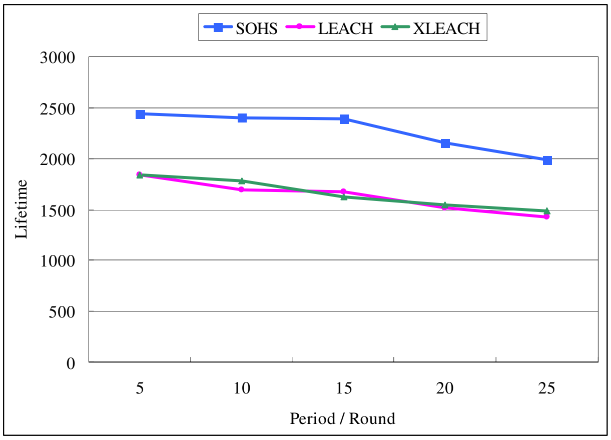


Fig. 1. Lifetimes of SOHS, LEACH, and XLEACH when initial energy is set to 0.5J

5.2 Remaining Energy When the Lifetime Expires

As second sets of experiments, we try to see the remaining energy ratio of WSNs when the lifetime expires for SOHS, LEACH and XLEACH. Fig.2 shows the remaining energy of WSNs when the number of periods per round varies from 5 to 25. The number of sensor nodes in this field is 100.

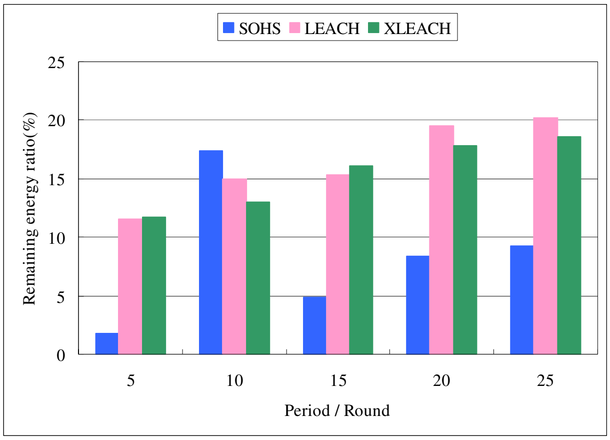


Fig. 2. Remaining energy ratios of SOHS, LEACH, and XLEACH

In Fig.2, the remaining energy of SOHS is less than LEACH and XLEACH, except when the number of round is 10. When the number of round is 10, remaining energy of SOHS is more than others. In that case two instances of the experiment have remaining energy over 30% but average remaining energy of other three instances is 0.03%. This experiment shows that SOHS consumes energies economically until the lifetime of a WSN expires not for every sensor node but for the first drained sensor node specially, which determine the lifetime of the WSN.

5.3 Different Number of Sensor Nodes in the Same Sensor Field

As a third set of experiments, we compare the lifetimes of WSNs when the number of sensor nodes varies from 100 to 500 for SOHS, LEACH at the sensor field of 50m * 50 m area. In Fig.3, we can know that the lifetime of a WSN is good when the sensor nodes are not too many or not too few. We can also see the lifetime of SOHS is still better than those of LEACH and XLEACH for these different numbers of sensor nodes. The gaps between SOHS and LEACH or XLEACH are larger than 26% for various number of sensor nodes in 50m*50m sensor field.

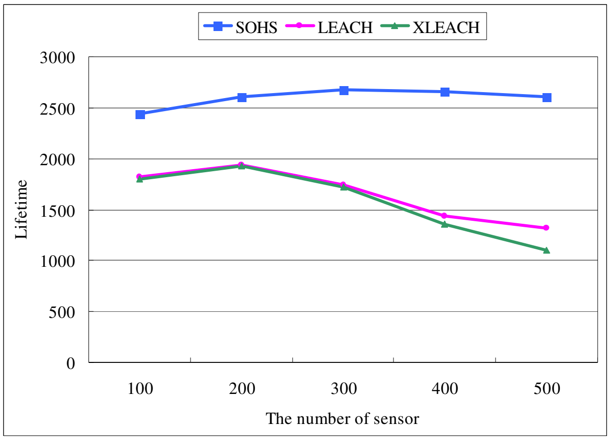


Fig. 3. Lifetimes of SOHS, LEACH, and XLEACH for different size of WSNs

5.4 Scalability of SOHS

So far we performed experiments at the sensor field of 50m * 50m. WSNs of real world can be bigger than this size. Hence we perform experiments that have varies sensor fields from 50m * 50m to 250m * 250m to show how scalable SOHS protocol is. For these larger sensor field, we assume that a similar distribution of sensor nodes in the sensor fields. We start with a sensor field of 50m*50m with 100 sensor nodes. Average density is 1 sensor node per 25m^2 area. We expand it 100m * 100m with 400 sensor nodes. In this case *t*, threshold of number of received signals are increase from 10 for 5m*5m sensor field to 40 for 10m*10m sensor field. For 50m*50m sensor field we use 10m as *p*, an initial broadcasting range. For larger sensor fields, we try to find most effective *p* by preliminary experiments for candidate broadcasting ranges. The following table is summary of our experimental data.

Table 2. Experimental data for scalability of SOHS

Sensor Fields	50m*50m	100m*100m	150m*150m	200m*200m	250m*250m
# of sensors	100	400	900	1,600	2,500
t, threshold of number of received signals	10	40	90	160	250
Candidate broadcasting ranges	7m ~ 20m	15m ~ 25m	25m~35m	35m~45m	45m~55m
p, broadcasting ranges	10m	21m	27m	37m	46m

In Fig.4. we compare SOHS, LEACH and XLEACH when the size of sensor field varies. In this result, the advantage of SOHS is still proved. The gaps between SOHS

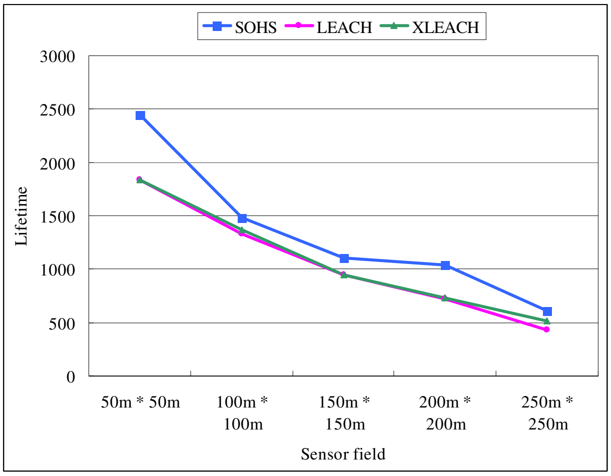


Fig. 4. Performance comparison with different size of sensor fields

and LEACH and the gaps between SOHS and XLEACH are larger than 8%. We can say that the advantage of SOHS is robust when the size of sensor field varies.

6 Conclusions

In this paper we proposed a new nature-inspired algorithm for efficient energy consumption of WSNs. We use a hierarchical approach that organizes sensor nodes into clusters and select one node as a cluster head. We have proposed SOHS algorithm and compared this with LEACH and XLEACH. In experimental result, SOHS is better than LEACH and XLEACH. SOHS has a simple but efficient head selection phase.

The base station of the WSN has a strong computational power and has no battery problem since it is connected via wire lines to the rest of the world. Hence if we can exploit the base station of the WSN for centralized information processing that is minimally used for efficient operation of the WSN. By these self-organized interactions among wireless sensor nodes and the wired base station, we can extend the lifetime of WSN and use it more efficiently. Hence a self adoptable sensor nodes and its coordination with the wired base station will be a promising research topic. We can also have interests to exploit a coordinated scheme for a more robust protocol when the operations of WSN's are uncertain.

References

[1] Akyildiz, I.F., et al.: A survey of Sensor Networks. IEEE Communications, 102 (August 2002)

[2] Peter, D., et al.: A Discussion on Fundamental Approaches for the Engineering of Autonomic Systems. In: Proceedings of the 2nd IEEE International Workshop on Modeling Autonomic Communications Environments (MACE), San José, CA, USA, October 29-30 (2007)

[3] Al-Karaki, J.N., Kamal, A.E.: Routing Techniques in Wireless Sensor Networks: A Survey. IEEE Wireless Communications 11(6), 6–28 (2004)
[4] Heinzelman, W.R., Chandrakasan, A., Balakrishnan, H.: Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In: IEEE Hawaii International Conference on System Science (2000)
[5] Handy, M., Hasse, M., Timmermann, D.: Low Energy Adaptive Clustering Hierarchy with Deterministic ClusterHead Selection. In: IEEE MWCN, Stockholm, Sweden (September 2002)
[6] Selvakennedy, S., Sinnappan, S., Shang, Y.: T-ANT: A Nature-Inspired Data Gathering Protocol for Wireless Sensor Networks. Journal Of Communications 1, 22–29 (2006)
[7] Bleckmann, H., Bender, M.: Water Surface Waves Generated by The Male Pisaurid Spider Dolomedes Triton (Walckenaer) During Courtship Behavior. Journal of American Arachnology 15, 363–369 (1987)

A Cell Outage Detection Algorithm Using Neighbor Cell List Reports

Christian M. Mueller, Matthias Kaschub, Christian Blankenhorn, and Stephan Wanke

Universität Stuttgart
Institute of Communication Networks and Computer Engineering
Pfaffenwaldring 47, 70569 Stuttgart, Germany
{firstname.lastname}@ikr.uni-stuttgart.de, stephan.wanke@web.de

Abstract. Base stations experiencing hardware or software failures have negative impact on network performance and customer satisfaction. The timely detection of such so-called outage or sleeping cells can be a difficult and costly task, depending on the type of the error. As a first step towards self-healing capabilities of mobile communication networks, operators have formulated a need for an automated cell outage detection. This paper presents and evaluates a novel cell outage detection algorithm, which is based on the neighbor cell list reporting of mobile terminals. Using statistical classification techniques as well as a manually designed heuristic, the algorithm is able to detect most of the outage situations in our simulations.

1 Introduction

1.1 Self-Organizing Networks

The configuration, operation and maintenance of mobile communication networks becomes increasingly complex, due to a number of different reasons. First, the continuous demand for bandwidth and coverage, driven by more and more widespread mobile Internet usage and new applications, urges operators to upgrade their backhaul networks and invest in new backhaul and air interface technologies. These need to be integrated into an existing infrastructure, which increases heterogeneity and in-turn complicates the management of these networks. Second, more sophisticated algorithms are implemented at the radio layers to exploit various physical effects of a wireless communications channel in order to increase system capacity. Third, more and more optimizations are conducted at cell-individual level to better exploit the particularities of a site. Given that nowadays, UMTS networks provide around 10^5 operator-configurable parameters [1], manual optimization already is a tedious and costly task.

Facing the challenge of increasingly complex network management, operators are at the same time under a tremendous cost pressure, driven by a strong competition among existing players and new entrants to the mobile communications sector. In this situation, the ability to efficiently manage their internal resources and the ability to cut down capital and operational expenditure becomes a competitive edge.

These constraints currently foster a trend towards a higher degree of automation in mobile communication networks. Under the term *Self-Organizing Networks (SON)*, the Next Generation Mobile Networks (NGMN) alliance has published a set of use cases for the application of self-configuration, self-optimization and self-healing concepts in deployment, planning and maintenance of mobile communication networks [2,3]. The 3GPP has also recognized the need for more automation and cognitive functions in future wireless networks and has therefore started to work on concepts, requirements and solutions [4,5] In addition, research projects are underway to develop solutions to these requirements [6,7].

1.2 SON Use Case: Cell Outage Detection

One of the SON use cases listed in [3] concerns the automated detection of cells in an outage condition, i.e. cells which are not operating properly. The reasons for the erroneous behavior of an outage cell, often also denoted as sleeping cell, are manifold. Possible failure cases include hardware and software failures, external failures such as power supply or network connectivity, or even misconfiguration. Consequently, the corresponding observable failure indications are diverse, too. Indications can be abnormal handover rates, forced call terminations, decreased cell capacity or atypical cell load variations. While some cell outage cases are easy to detect by Operations Support System (OSS) functions of the network management, some may not be detected for hours or even days. Detecting those outage cases today is usually triggered by customer complaints. Subsequently, discovery and identification of the error involves manual analysis and quite often requires unplanned site visits, which makes cell outage detection a costly task.

A rough classification of sleeping cells is provided in [8]. According to this, a *degraded* cell still carries some traffic, but not as it would if it was fully operational. A *crippled* cell in contrast is characterized by a severely decreased capacity due to a significant failure of a base station component. Finally, a *catatonic* cell does not carry any traffic and is inoperable.

In our work, we focus on the detection of a catatonic cell. More specifically, our failure assumption is that the base station experiences a failure in one of its high-frequency components. This might be an error of the HF amplifier, antenna or cabling. The consequence is, that the cell is not visible anymore to users or neighbor base stations, i.e. it does not transmit a pilot signal anymore. However, from a network point of view, the cell appears to be empty, but still operational. As requirements to the detection algorithm, we have decided to only use already available measurement data and to avoid the need for new sensor equipment or the introduction of dedicated measurement procedures. Furthermore, the algorithm shall be able to reduce detection time to seconds or minutes.

While the basic idea and some preliminary results of our outage detection technique have already been presented in [9], here a a detailed description of the algorithm and an in-depth evaluation of its performance is provided. The remainder of this paper is structured as follows: Section 2 presents the outage detection algorithm. The evaluation methodology and system model are described in section 3. Section 4 examines the performance of the detection algorithm and presents simulation results. Section 5 finally summarizes and draws conclusions.

2 Cell Outage Detection Algorithm

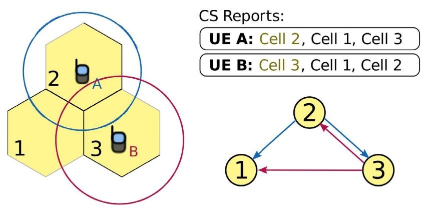
The idea behind our outage detection algorithm is to use Neighbor Cell List (NCL) reports to create a graph of visibility relations, where vertices of the graph represent cells or sectors. Edges of the graph are generated according to the received NCL reports, where an edge weight determines the number of mobile terminals that have reported a certain neighbor relation. More details on how the graph is constructed will be given in section 2.1.

Neighbor Cell List reports are always generated during active connections, when mobile terminals continuously measure the signal strength and quality of the radio channel of the cell it is currently connected to. They also measure the signals of several neighbor cells, which constitute candidate cells for potential handovers. These measurements are transmitted to the current cell to decide whether a handover might have to be performed. In GSM, for example, a terminal in an ongoing connection measures up to 16 base stations within its visibility range and sends NCL reports at 480 ms intervals, containing the best 6 measurement results [10]. The NCL reports of different terminals are retrieved by a subordinated network entity, e.g. a radio network controller in UMTS, respectively a base station controller in GSM. While measurement reporting details differ from GSM to UMTS or LTE, we only assume that lists of neighbor cells are reported at regular intervals, which can be realized in all the different radio access technologies.

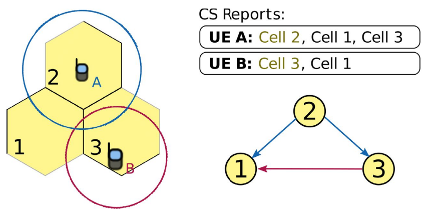
2.1 Monitoring the Visibility Graph

Monitoring changes in the visibility graph is a key element of our detection algorithm. The visibility graph is created at regular intervals and any unusual variations of the graph might indicate a cell outage. Figure 1 depicts examples of such a graph. Comparing two successive graphs $G(t_1)$ and $G(t_1 + T)$, the detection algorithm is sensitive towards any nodes becoming isolated in $G(t_1+T)$, in which case a so-called *change pattern* is created.

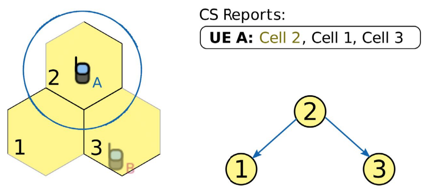
In Fig. 1(a), a mobile terminal UE_A is currently being served by sector 2, while UE_B is in an active connection currently being handled by sector 3. The visibility range of both mobiles in this simplified example is denoted as a circle around their respective positions. Thus, the NCL report of UE_A contains cell 2 as the currently serving cell and cells 1 and 3, which are neighbor cells within its visibility range. The NCL report of UE_B is constructed accordingly. In the resulting visibility graph, the NCL report of UE_A now leads to the two directed edges (blue) originating from node 2 to the nodes 1 and 3. Analogously, the



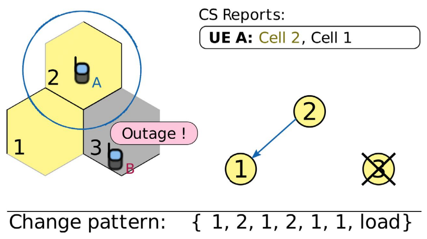
(a) $G(t_1)$: Initial state



(b) $G(t_1 + T)$: NCL change



(c) $G(t_1 + T)$: Call end



(d) $G(t_1 + T)$: Outage case

Fig. 1. Visibility Graph examples with two terminals being served by cells 2, respectively 3

NCL report of UE_B results into the two edges (red) originating from node 3. Every edge in the graph has a certain weight, representing the number of UEs which have reported a neighbor relation between a pair of cells. In the example in Fig. 1, all edge weights equal one and are therefore not shown.

During normal operation of the network, the visibility graph is subject to frequent changes, caused by starting and ending calls, user mobility, changes in radio propagation and changes in the NCL reporting itself. Figures 1(b) and 1(c) show typical examples of graph variations. In Fig. 1(b), UE_B has changed its position such that cell 2 is no longer in its visibility range. The corresponding NCL reports are depicted on the right hand side of Fig. 1(b), together with the resulting topology of the visibility graph. Another variation of the graph is depicted in Fig. 1(c), where UE_B has terminated its connection and does not send NCL reports anymore. Consequently, the visibility graph now only contains the neighbor relations reported by UE_A.

Figure 1(d) finally depicts the resulting visibility graph if cell 3 experiences a failure and becomes catatonic. The NCL reports of UE_B are not received anymore. At the same time, cell 3 disappears from the candidate set of UE_A. The resulting graph now only contains a single edge from node 2 to node 1. Note that every outage situation results into an isolated node in the visibility graph.

Figure 1(d) shows the change pattern of the graph variation compared to the initial state of the graph in Fig. 1(a). A change pattern is a 7-tuple with the following attributes, that is created for every isolated node:

– number of disappeared edges to the isolated node
– number of disappeared edges from the isolated node

- sum of the edge weights to the isolated node
- sum of the edge weights from the isolated node
- number of UEs of which no reports are received anymore
- number of UEs with changed neighbor cell list reports
- load as sum of the number of active calls of the direct neighbor cells

The respective elements of the 7-tuple are the features or input parameters of the classification step described in the following section 2.2.

The change patterns that are created when nodes become isolated are not necessarily unique. In the sample scenario in Fig. 1, the same pattern would have been created if a movement of UE_A and the end of a connection of UE_B occurred within a single monitoring interval. Thus, an isolated node pattern in the visibility graph is a necessary condition for an outage situation, but not a sufficient one. However, our assumption is that cell outages create characteristic change patterns which, in many cases, can be distinguished from normal fluctuations of the visibility graph. The outage detection problem thus translates into a classification problem on change patterns of the visibility graph.

2.2 Outage Detection as a Binary Classification Problem

The classification of a set of items is the assignment of similar items to one out of several distinct categories. The task of separating change patterns into outage and non-outage situations can be regarded as a binary classification problem with a set of predefined classes.

Classification algorithms are widely used in the field of automatic pattern recognition [11]. A classifier can either be knowledge-based (e.g., expert systems, neural networks, genetic algorithms) or make use of mathematical techniques from multivariate statistics (e.g. cluster analysis, classification and regression trees). We applied three different classification techniques, a manually designed expert system and two others using statistical classification techniques. In statistical classification, the n-dimensional tuple space is separated according to statistical measures, which allows to automatically construct a classifier from a given training set of patterns. A large number of statistical methods exist to infer the classification rules, of which an iterative decision-tree algorithm and a linear discriminant analysis have been applied here and will be further detailed in the following:

Expert System. From the class of knowledge-based classifiers, an expert system has been applied to the outage detection problem. A set of rules has been manually constructed, based on the observation of the change patterns of the visibility graph. The rules are expressed as a sequence of *if... then* statements, with threshold values tailored to our evaluation scenario. The expert system has been designed such that it performs a conservative classification, i.e., the number of false alarms shall be minimized.

Decision-Tree (DT). A tree of binary decisions on the attributes of the respective patterns is automatically created from a training set. The decision about when to split into separate branches is thereby determined by the Gini impurity function, which is used by the classification and regression tree (CART) algorithm [12]. Similar to the expert system, the outcome of the DT algorithm can be regarded as a sequence of *if*... *then* statements, with the difference that now these rules are being determined automatically.

Linear-Discriminant Function (LDF). The LDF classifier belongs to the class of linear classification algorithms. The discriminant function is a linear combination of the pattern's attributes, whose coefficients are determined from the training set using a least square estimation [13].

3 Evaluation Methodology

The performance evaluation of the cell outage detection algorithm involves several steps. First, a system-level simulation of a multiple sites scenario is run several times for a duration of 100.000 s without any outage situation, in order to collect change patterns from the normal operation of the network. Second, a Monte Carlo-like simulation is performed to collect change patterns for cases where a cell becomes catatonic. For each of the 3000 drops of this Monte Carlo-like simulation, mobile terminals were positioned randomly and the change patterns are obtained immediately after a cell sector is configured to be in an outage state. Finally, a training set and a test set is constructed from the collected patterns and fed to the classification algorithms. For the statistical classification, the *LSS Classification Toolbox* in its default parametrization is used [14]. The system-level simulations have been conducted with the event-driven IKR Simlib simulation library [15].

The scenario used in the event-driven simulations consists of 19 sites, respectively 57 cells, with a base station distance of 1000 m. The system was modeled as an FDM system with a maximum capacity of 20 simultaneously active calls per cell. A fractional frequency reuse is assumed with half the capacity reserved for the re-use 1 region in the cell center and the remaining capacity distributed over the re-use 3 regions at the cell border. Inter-cell interference is considered as proportional to the actual load in the corresponding re-use areas

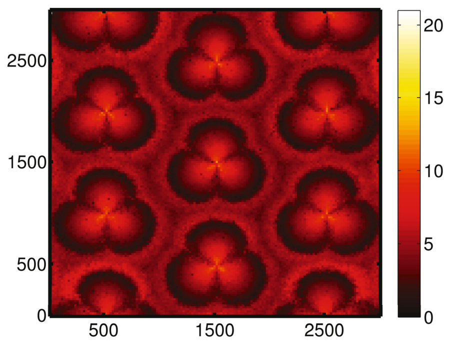


Fig. 2. SINR without outage

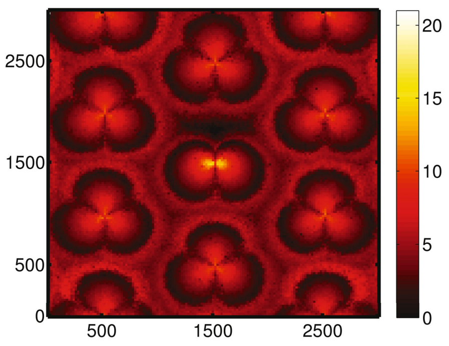


Fig. 3. SINR in outage case

Table 1. Scenario and simulation parameters

Parameter	**Value**
Base station distance	1000 m
Shadowing std. dev. (global)	7.0 dB
Shadowing std. dev. (local)	1.0 dB
Shadowing correlation length	50 m
BS power to individual UE	35 dBm
Signal detection threshold at UE	-96 dBm
Traffic model (Poisson arrivals)	$\lambda = \frac{1}{100\,\mathrm{s}}$
Call holding time (neg.-exp.)	$h = 35\,\mathrm{s}$
User movement	Random Direction
User speed	$v = 1 - 5\frac{\mathrm{m}}{\mathrm{s}}$
Segment length (uniform)	$l = 20 - 800\,\mathrm{m}$
Direction angle (uniform)	$0 - 360$

of neighbor cells. The Walfish-Ikegami model (non-LOS) determines the path loss and spatially correlated shadowing planes similar to [16] are used. A global shadowing plane models the influence of buildings and other common obstructions to radio propagation, while local shadowing planes for each base station account for the different antenna locations. The 3-sector antenna pattern corresponds to [17]. A handover algorithm is implemented which is sensitive towards SINR and RSSI (Received Signal Strength Indicator) values. The measurement reporting of neighbor cell lists is assumed to be GSM-like with a reporting period of 480 ms, which in our setting corresponds to the graph update period. Further parameters are given in Table 1. SINR plots extracted from the simulation for an outage and a non-outage case for an average cell load of 25% are given in Figures 2 and 3, respectively.

4 Analysis and Results

4.1 Outage Observability

The quality of the outage detection is largely determined by the performance of the classification algorithm. However, even with perfect classification, some outage cases might still not be detected. In particular in low load situations when user density is scarce, a cell might not serve any users and might not be measured by any of the UEs of the surrounding cells. A cell outages thus does not result into changes of the visibility graph. In this case, the outage would not be detectable at all.

Figure 4 depicts the upper bound of detectable outages for the evaluation scenario. The black solid line denotes the ratio of detectable outages over the average number of active connections per cell. It can be seen that in low load conditions, a large proportion of the cell outages have no representation in the visibility graph. Starting from an average 1.6 active connections per sector, which in our setup corresponds to a cell load of 10%, already more than 95% of the

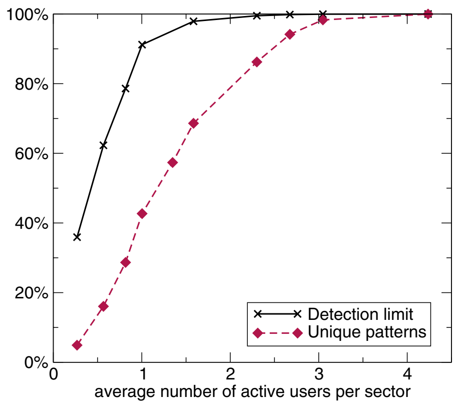


Fig. 4. Upper bound of cell outage observability and ratio of unique patterns

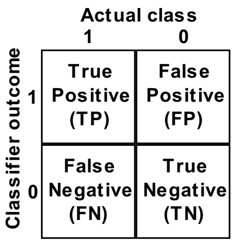


Fig. 5. Confusion Matrix

outages can theoretically be detected. As outlined in section 2.1, outages do not necessarily create unique change patterns because normal fluctuations of the visibility graph might lead to the exact same patterns. The dashed red curve denotes the proportion of patterns uniquely identifying a cell outage. This gives a reference value which constitutes the performance achievable by a hypothetical look-up table based detection algorithm, assuming that all possible outage patterns are known in advance.

4.2 Classification Quality

The quality of a classification algorithm is characterized by the so-called confusion matrix (see Fig. 5).

True Positive. A *TP* denotes the case that an outage (i.e. a cell becomes catatonic) has occurred and it has been successfully detected.

False Negative. A *FN* denotes the case that an outage has occurred, but it has not been detected.

False Positive. A *FP* occurs, when there is no catatonic cell but the detection algorithm nevertheless reports an outage.

True Negative. A *TN* is when there is no outage and the algorithm correctly recognizes the change pattern as normal fluctuation of the visibility graph.

TP, *FN*, *FP*, and *TN* denote the total numbers of occurrences for these four cases in a simulation run. From the confusion matrix, a number of metrics can be derived. On the one hand, the *Sensitivity* or *True-Positive-Rate* R_{TP} denotes whether a classifier is able to correctly detect the outage patterns from the set of all patterns:

$$R_{\mathrm{TP}} = \frac{TP}{TP + FN} \tag{1}$$

On the other hand, the *Fall-out* or *False-Positive-Rate* R_{FP} gives the tendency of a classification algorithm to create false alarms:

$$R_{\mathrm{FP}} = \frac{FP}{FP + TN} \tag{2}$$

Achieving a high sensitivity generally comes with an increase in the rate of false alarms. The two metrics can therefore be regarded as complementary. A so-called *Receiver-Operating-Characteristics* (ROC) plot is a way to visualize the performance of binary classification algorithms regarding these two metrics and is primarily used in signal detection theory [18]. In a ROC plot, the sensitivity is drawn over the false-positive-rate. A perfect classification would thus result in a mark in the upper left corner of the plot. The line through the origin gives the reference values for a random decision, where each of the classes is chosen with probability one half.

Figure 6 shows a ROC plot of the classification algorithms applied here, for the range of cell load values also used in the other figures. After having been trained with a training set, the classifiers had to determine the correct mapping for test sets, where each of the patterns is tested only once. In doing so, Fig. 6 is independent of the frequency of occurrence of the different patterns in the outage detection scenario. It thus solely gives the performance of the classification algorithms, which is not identical to the resulting performance of the outage detection algorithm. It can be observed that the Expert System and the Decision Tree classifiers are able to achieve a high sensitivity, especially for higher load scenarios, which is consistent with the results from Fig. 4. The Expert System has been designed to perform a conservative detection. In other words, the Expert System is more likely to indicate False Negatives than False Positives. Compared to this, the DT classifier is characterized by a much higher risk to create false alarms. Finally, the LDF classifier has high false alarm probability and, in contrast to the other algorithms, does not show any improvement with increasing system load.

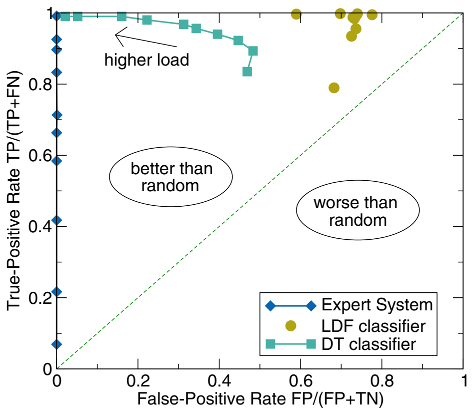


Fig. 6. ROC plot of the classification algorithms

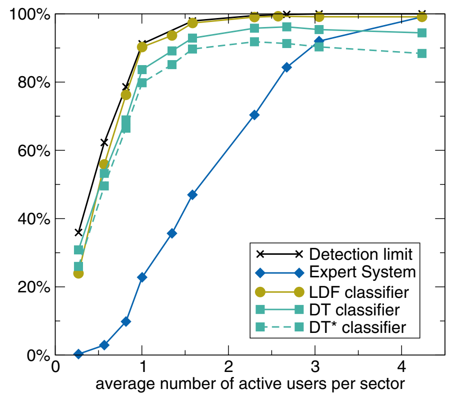


Fig. 7. Sensitivity of the classification algorithms

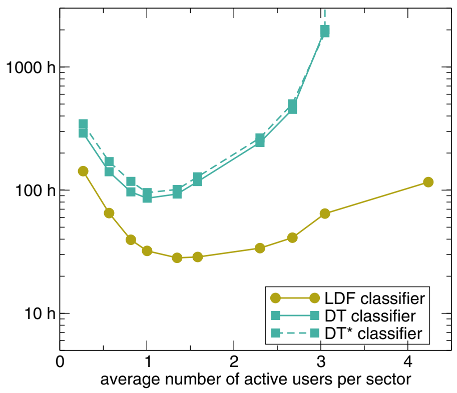


Fig. 8. Mean time between false alarms per cell sector

4.3 Outage Detection Performance

In order to determine the performance of the outage detection algorithm, not only the classification quality, but also the problem-specific frequency of occurrence of the patterns has to be taken into account. A wrong classification of frequently occurring patterns could significantly deteriorate the suitability of the classification algorithms to the outage detection problem, whereas a classification error for rare patterns does not have much influence. Figure 7 depicts the sensitivity of the classification algorithm with respect to the observability bound presented in Fig. 4. It thus gives a measure of the proportion of outages that can be detected with the approach presented here. Using the statistical classifiers, sensitivity is close to the maximum achievable sensitivity. The Expert System only attains moderate sensitivity due to its conservative decision rules.

Finally, Fig. 8 depicts the fall-out risk per cell in terms of the mean time between false alarms. Only the statistical classifiers are shown, given that the Expert System did not produce any false positive classifications over the whole monitoring period. For lower load, overall sensitivity is low and therefore the risk of rising false alarms also is lower than in the middle range of the curves in Fig. 8. However, a mean time between false alarms of only several tens or hundreds of hours would already result into several false alarms per cell being raised within a single week. Even though the applied classification algorithms here provide means to mitigate this problem by giving higher penalty to false positive classification, the results do not differ significantly. An example is shown in Figures 7 and 8 for a modified decision tree classifier (denoted as DT$*$), which has given a ten times higher penalty for false positive classifications.

4.4 Discussion

From the previous subsections, it can be summarized that although the statistical classifiers show an overall good detection sensitivity, tendency towards generation of false alarms is still far too high for practical applications. This is mainly due to the partly ambiguous change patterns and the non-linearity of the problem. The expert system achieves much better performance here, but generally suffers from the need to manually tune its threshold parameters to the scenario under investigation. These observations motivate further work, employing more flexible classification algorithms such as fuzzy classifiers or Support Vector Machines [19].

From a network operation point of view, a detection algorithm as described here can be applied to whole or parts of a network. That is, collection of NCL reports and its interpretation can be done in an RNC or BSC, without the need for additional signaling. For LTE, the scope of a single eNodeB is too small and a master eNodeB or another management device would have to monitor the visibility graph. Although this results in additional signaling, the NCL reports of each terminal in a cell can be aggregated by the eNodeB over one monitoring period, which results in one signaling message per monitoring interval being sent to the respective management device.

5 Conclusion

Automated detection of base station failures or sleeping cells is a prerequisite for future self-healing capabilities in mobile communication networks. The detection of some failure conditions still imposes a challenge to network operators and demands for additional means to observe base station behavior from the outside. An algorithmic detection without the need for additional sensor equipment thereby represents the most cost effective way of improving outage detection capabilities in existing networks.

The presented approach of using neighbor cell measurement reports of mobile terminals to determine outage situation showed good performance, even in moderate cell load conditions. However, the relatively high risk of false alarms still prevents from practical application. Although the algorithm is able to quickly detect a large proportion of the outages in a certain cell load range, it has been shown that there is no possibility to detect outages when no active users are close to the respective cell. In this case, the presented algorithm may be combined with other detection mechanisms operating on a longer time scale.

References

1. Lehser, F.: Self Organising LTE/SAE Network: Operator Requirements and Examples. In: 20th workshop of the ITG group 5.2.4, Ulm, Germany (September 2006), `http://www.ikr.uni-stuttgart.de/Content/itg/fg524/`
2. NGMN Alliance: Use cases related to self organising network: Overall description (April 2007), `http://www.ngmn.org`
3. NGMN Alliance: Annex A (informative) of use cases related to self organising network: Overall description (April 2007), `http://www.ngmn.org`
4. 3GPP: Telecommunication management; Self-Organizing Networks (SON); Concepts and requirements. TS 32.500 (2008)

5. 3GPP: Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Self-configuring and self-optimizing network (SON) use cases and solutions. TR 36.902 (2008)
6. FP7 Socrates: Self-optimisation & self-configuration in wireless networks, `http://www.fp7-socrates.org/`
7. FP7 E3: End-to-end efficiency (e3), `http://ict-e3.eu`
8. Cheung, B., Fishkin, S.G., Kumar, G.N., Rao, S.: Method of monitoring wireless network performance. USPTO Patent Application 20060063521 (March 2006)
9. Mueller, C.M., Kaschub, M., Blankenhorn, C., Wanke, S.: Design and evaluation of detection algorithms for base stations outages. In: 11th COST 290 MCM, TD(08)003, Tampere, Finland (2008)
10. Eberspächer, J., Vögel, H.J., Bettstetter, C.: GSM Global System for Mobile Communication. B.G. Teubner Stuttgart, Leipzig, Wiesbaden (2001)
11. Jain, A., Duin, R., Mao, J.: Statistical pattern recognition: a review. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(1), 4–37 (2000)
12. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.: Classification and Regression Trees, 1st edn. Chapman & Hall/CRC (1984)
13. Duda, R.O., Hartr, P.E.: Pattern Classification and Scene Analysis. John Wiley and Sons Ltd., Chichester (1973)
14. Uhlich, S.: Classification Toolbox V1.2 (March 2008), `http://www.lss.uni-stuttgart.de`
15. IKR: Simulation library v2.6, `http://www.ikr.uni-stuttgart.de`
16. Forkel, I., Schinnenburg, M., Ang, M.: Generation of two-dimensional correlated shadowing for mobile radio network simulation. In: Proceedings of the Wireless Personal Multimedia Communications, WPMC (2004)
17. NGMN Alliance: Radio access performance evaluation methodology (June 2007), `http://www.ngmn.org`
18. Fawcett, T.: ROC Graphs: Notes and Practical Considerations for Researchers. Technical report, HP Laboratories, Palo Alto, USA (2004)
19. Wang, L. (ed.): Support Vector Machines: Theory and Applications. Studies in Fuzziness and Soft Computing. Springer, Berlin (2005)

A Semi-Autonomic Framework for Intrusion Tolerance in Heterogeneous Networks

Salvatore D'Antonio[1], Simon Pietro Romano[2], Steven Simpson[3], Paul Smith[3], and David Hutchison[3]

[1] CINI – ITeM Laboratory
Via Cinthia 80126 Napoli, Italy
[2] University of Napoli "Federico II"
Via Claudio 21 – 80125 Napoli, Italy
{saldanto,spromano}@unina.it
[3] Computing Department, InfoLab21
Lancaster University, Lancaster, UK
{ss,p.smith,dh}@comp.lancs.ac.uk

Abstract. A suitable strategy for network intrusion tolerance—detecting intrusions and remedying them—depends on aspects of the domain being protected, such as the kinds of intrusion faced, the resources available for monitoring and remediation, and the level at which automated remediation can be carried out. The decision to remediate autonomically will have to consider the relative costs of performing a potentially disruptive remedy in the wrong circumstances and leaving it up to a slow, but more accurate, human operator. Autonomic remediation also needs to be withdrawn at some point – a phase of recovery to the normal network state.

In this paper, we present a framework for deploying domain-adaptable intrusion-tolerance strategies in heterogeneous networks. Functionality is divided into that which is fixed by the domain and that which should adapt, in order to cope with heterogeneity. The interactions between detection and remediation are considered in order to make a stable recovery decision. We also present a model for combining diverse sources of monitoring to improve accurate decision making, an important pre-requisite to automated remediation.

1 Introduction

Network intrusion tolerance—detecting intrusions and remedying them—can be carried out manually or automatically, with various trade-offs of time and reliability. The choice is influenced by the resources available to the organization responsible for protecting a network. The kinds of attacks faced and the actual detection and remediation mechanisms may also vary depending the kind of network to be protected.

Automating intrusion tolerance could be problematic if it is applied in the wrong circumstances (e.g., a node is isolated because it is incorrectly supposed to be compromised). Also, when a temporary remedy is applied automatically, it may affect the original detection of the intrusion, and one must consider whether to use that original detection mechanism to also detect the end of the intrusion and withdraw the remedy automatically.

As part of the INTERSECTION project [1], we have devised a framework for deploying security and resilience strategies against intrusions in networks. Here, we discuss its design relating to the problems of heterogeneity, and the automation of both deployment and withdrawal of mitigation mechanisms.

This section continues by discussing the problems of automating remediation to achieve intrusion tolerance, and how to achieve it in heterogeneous environments. Section 2 surveys intrusion-detection systems and touches on some systems that enable automatic remediation. Section 3 presents a framework for deploying intrusion-tolerant systems. Section 4 describes how a part of the framework is to be implemented to improve confidence of the initial detection.

1.1 To Automate Intrusion Tolerance

Automation of network intrusion tolerance is desirable because of the unpredictability of attacks and the time taken by human operators to respond manually, which could otherwise lead to significant loss of service and financial cost. However, an automated intrusion-tolerance system that overreacts to an anomalous but innocent event can also be costly, so a balance must be struck between fast automation and more accurate but slow manual control.

The decision to perform different forms of remediation automatically will involve a trade-off between the potential impact on the threatened service if detection is mistaken and the cost of human intervention. An intensive attack will require an immediate response, even if it's not an ideal solution (involving some cost of its own), as it may take several hours for an operator to come up with a better solution. The cost of doing nothing in the meantime outweighs the cost of doing the automatic response.

This balance may vary across domains. Network-based malicious behaviour is becoming increasingly profit-driven, as attacks have shifted from being mostly targeted at large governmental and commercial organisations towards more profit-yielding small-to-medium enterprises (SMEs), and individuals. Larger organisations can afford 24-hour staffing, thereby reducing the need for automation and the risks of false positives. This is more difficult for SMEs, who may prefer to risk downtime for false positives if they can automatically recover quickly too.

1.2 Intrusion Tolerance in Heterogeneous Environments

The hardware and software resources available for intrusion tolerance may vary significantly. For example, on a wireless mesh network (WMN) [17], there may be little in the way of resources, and those available may be highly constrained mesh devices; whereas in an enterprise setting, dedicated hardware may be available. Related to this point are the probable attacks a domain may face. For example, an end-system on a community WMN is unlikely to be the victim of a TCP

SYN attack (as the most likely victims, servers, are better placed on a wired network), which is distinct from the servers of a large financial institution.

This situation suggests that the mechanisms available for monitoring network traffic with the aim of detecting attacks, and the strategies for remedying them are very much specific to a domain. While the attacks that can occur within a domain are likely to be domain-specific, they will be drawn from a set of known attacks or attack types. This suggests that re-usable approaches to detecting attacks and general strategies for dealing with them (that can use locally-relevant monitoring and remediation mechanisms) can be developed.

2 Related Work

Intrusion Detection Systems (IDSs) can be classified as belonging to two main groups, depending on the detection technique employed: *anomaly detection* and *misuse detection*, also known as *signature detection* [5]. Both techniques depend on the existence of a reliable characterization of what is *normal* and what is not, in a particular networking scenario. Anomaly detection techniques base their evaluations on a model of what is normal, and classify as anomalous all the events that fall outside such a model. Indeed, if anomalous behaviour is recognized, this does not necessarily imply that an attack activity has occurred. Thus, a serious problem exists with anomaly detection techniques which generate a great amount of false alarms. Conversely, the primary advantage of anomaly detection is its ability to discover novel attacks.

An example anomaly detection system is presented in [10], where the authors propose a methodology to detect and classify network anomalies by means of analysis of traffic feature distributions; they adopt *entropy* as a metric to capture the degree of dispersal or concentration of the computed distributions. NETAD [13] detects anomalies based on analysis of packet structure: the system flags suspicious packets based on unusual byte values in network traffic.

The most known open-source signature-based intrusion detection systems are SNORT [6] and BRO [14]. These systems allow the user to define a customized set of rules in order to codify specific types of attacks. P-BEST [12] is a signature-based intrusion detection system able to detect computer and network misuse by means of a rule translator, a library of run-time routines, and a set of garbage collection routines.

There are approaches that aim to ensure network security by exploiting traffic monitoring information. In [2] and [16], the authors describe how to correlate netflow system and network views for intrusion detection. Their approach is human-driven, since they propose to use visualization tools in order to obtain useful information for security purposes. This approach demonstrates how data collected by flow monitoring systems can be used in the context of intrusion detection. In [3] and [11], data coming from both network monitoring and system logs are correlated in order to detect potential attacks. The authors prove that using data from more sources increases IDS performance. However, system logs are not always available, as in the case of servers owned by Internet providers.

Automatic remediation can cause problems if not applied to genuine intrusions. In [7], the authors highlight how automated intrusion response is often disabled due to the cost of responding to too many false positives. Their solution is to balance the cost of restarting more components of a system against the cost of not restarting enough, to produce an optimum response strategy even with uncertainty about the intrusion. In [4], an automatic approach to mitigating the effects of large volumes of ARP traffic caused by the scanning behaviour of Internet worms on switched networks is presented. ARP requests are dropped probabilistically in proportion to rate, and this is effective against supposedly compromised systems which generate the higher rates. However, it is shown the network's gateway router is unfairly affected, as it *legitimately* generates many ARP requests too.

3 The INTERSECTION Framework

The main goal of the INTERSECTION project [1] is to provide an infrastructure for heterogeneous networks to be resilient and secure in the face of intrusions, and to interoperate to achieve that resilience and security. We now present a framework for deploying intrusion-tolerance strategies that can adapt to heterogeneous domains, and is thus able to meet that goal.

The functions of our framework, as shown in Fig. 1, form a control loop that enables automatic intrusion tolerance, and the intervention of a network operator when necessary. To summarise, the network generates raw data about its traffic, Monitoring and Detection reduce this to simpler signals, and the remaining functions feed orders back into the network to either defeat an attack or mitigate its effects.

To address the problem of heterogeneity and enable the development of reusable strategies for detection and remediation, we separate domain-agnostic components Detection and Reaction from domain-specific components Monitoring and Remediation. Detection and Reaction together embody a strategy for determining the existence of an attack and a response to it, while Monitoring and Remediation respectively implement how traffic is collected from the network and how strategies are applied.

The trade-off that determines under what circumstances automated remediation should be used is realised through the configuration of Reaction components. Detection components inform Reaction of the existence and severity of perceived malicious behaviour. Reaction then determines a course of action, also depending on available remediation mechanisms. If appropriate, Visualisation components are used to aid a network operator when making decisions about what remediation activities to invoke, by providing details of the current attack and other network state.

3.1 Monitoring and Detection

IDSs use information on network traffic in order to detect ongoing malicious activities. A wide-scope view of the network traffic as well as a deep knowledge of

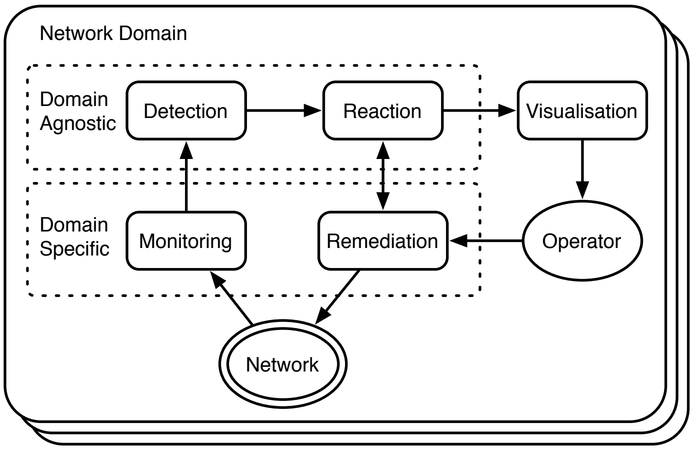


Fig. 1. The INTERSECTION framework

the network status improves the detection process. For example, a Distributed Denial of Service (DDoS) attack is performed by systems that are widely distributed throughout the network. In order to effectively detect such attacks, information regarding a number of active traffic flows and different network entities is required. Such entities share the same "purpose" (i.e., to jeopardize either a service or a single host) and make use of the same resources (e.g., those belonging to the network infrastructure) in order to accomplish their task. They cooperate in order to achieve the same objective. This malicious cooperation can be detected only if a wide-scope analysis of traffic flows is performed.

This kind of analysis requires using coarser grained flow definitions which convey, for example, the traffic from different sources to one destination. Since the amount of resources needed for measurement and reporting increases with the level of granularity required to detect an attack, a multi-step monitoring approach is useful. We call this approach *adaptive monitoring*. An approach to granularity adaptive attack detection is presented in [9].

In the framework, the Monitoring function encompasses all mechanisms for observing traffic in the network and condensing the information about it (e.g., by gathering statistics into flows, or observing simple statistics about the whole network). Detection configures Monitoring to receive data from it, and attempts to interpret that data as anomalous or indicative of an intrusion. As a result, it may simply report that an anomaly is detected, or it may report the degree, plus other parameters such as the end-systems to which it pertains, or the level of confidence it has that a genuine intrusion is taking place. Importantly, it may also reconfigure Monitoring to obtain greater detail temporarily, in order to make a better determination, hence the level of monitoring is adapted according to the need to make more detailed detection decisions.

The choice of Monitoring functions is fixed by the configuration of the network – development here is driven by hardware and associated management software. In contrast, Detection mechanisms may be added to the network dynamically – a programmer works here to improve intrusion tolerance.

3.2 Reaction and Remediation

Reaction represents pre-determined decisions or logic for handling anomalies reported by Detection. Options include reporting the anomaly through IDMEF [8] to other domains, reporting it to a network operator via Visualization, and requesting more information via Detection. Through a chain of detections and reactions, the system may detect more sophisticated intrusions without applying intrusive or resource-hungry network monitoring at all times.

Remediation encompasses all mechanisms provided by the network for defeating an intrusion, or for mitigating its effects. It is an option for Reaction to trigger Remediation having detected a particular kind of intrusion, and might (for example) isolate a compromised node from its peers, or rate-limit its probing attempts. This activity might occur even before an intrusion is fully identified, i.e., at earlier points in the "chain of detections and reactions".

Again, there is a similar distinction between Reaction and Remediation as there is between Monitoring and Detection – the Remediation mechanisms available in a network are fixed, i.e., determined by its configuration, while Reaction mechanisms may be added dynamically.

3.3 Visualization and the Operator

Autonomic remediation will not be enough to deal with the evolution of attacks in the long term. Most forms will involve a certain cost to the network as an alternative to a possible catastrophic cost of network failure, and they may need Operator interaction to determine when to be switched off. The Operator may also be required to deal with new attacks (typically appearing as anomalies) for which a trustworthy autonomic remedy has not yet been devised or deployed. The Visualization function covers the reporting of attacks and anomalies to the Operator, and may receive such signals from Reaction, or additional details from Detection and Monitoring. Visualization together with the Operator could be regarded as a form of remediation – they complete the control loop back to the network with a slow path.

3.4 Recovery

Applying a remedy indefinitely could be costly. Some remedies involve only a one-off cost, but thereafter cost nothing, and could be left applied. Indeed, they could be applied even before any intrusion or anomaly is detected. They consist of patches to fix infrequent bugs and exploits, and changes to newer, more robust protocols.

In contrast, some remedies can only be applied temporarily because they impede the normal operation of some part of the network, and should be withdrawn as soon as possible. These include isolation of compromised nodes (fixed by disinfecting and patching the nodes), and blocking of attack traffic (fixed by detecting the end of an attack) — not eventually withdrawing these remedies renders the affected node's participation in the network pointless. The necessary withdrawal of a remedy to achieve normal operation is referred to as *recovery.*

How should the end of an attack be detected? Two choices are:

1. To re-use Monitoring and Detection, which originally reported the attack, to also report its termination.
2. To get Remediation to perform very anomaly-specific and narrow-scope monitoring at its deployment locations (and perhaps report back to Reaction to make a wide-scope decision to withdraw).

In the first case, there are several options:

Remediation-Monitoring interaction. Remediation could inform Monitoring of what it is doing, e.g., which flows it is blocking, and report flows that Monitoring no longer should see.
A problem with this approach is that it couples Monitoring and Remediation. This potentially requires devising a complex expression of what Remediation is doing, in order for Monitoring to understand it, or would make it harder for Monitoring and Remediation implementations to be developed independently.

Monitoring non-interference. In its reports to Reaction, Detection could indicate the monitoring sources which precipitated those reports. Reaction could use this to deploy remedies at locations that protect the victim but do not affect Monitoring; Detection then will later signal the end of the attack.

Monitoring migration. Reaction informs Detection where it is applying Remediation, and so Detection adjusts the location of Monitoring to be able to continue to detect the attack.
This approach seems impractical. Firstly, the possible locations for Monitoring are likely to be fixed, or if they are good enough when moved, why not simply monitor at the preferred locations anyway? Secondly, two different remedies applied independently at the same time may impose contradictory requirements on Monitoring.

Additionally, all of these approaches do not fit well with recovery that requires a manual trigger (e.g., acknowledging that a compromised node has been cleaned), as it implies that the Operator must also try to convince Monitoring that the threat has passed, instead of simply withdrawing the remedy directly. Furthermore, if Detection is based on traffic from the victim (e.g., anomalous responses to an attack), any successful remedy will surely stop these from happening.

Therefore, we follow the second case, i.e., Remediation should do its own narrow-scope monitoring to detect the end of the attack. This has the following advantages: it leaves Remediation decoupled from Monitoring; the relative positions of Remediation and Monitoring become irrelevant; and Detection can still be based on the victim's response. However, the decision to withdraw can still be placed with Reaction; this will gather reports from Remediation to make a decision based on the wider view it has of the attack, which individual Remediation deployments do not.

3.5 Example Scenario

To demonstrate how the INTERSECTION framework can be applied, we present an example scenario, which is depicted in Fig. 2. In this scenario, a Web server is being subjected to a Distributed Denial of Service (DDoS) attack. The domain that provides connectivity to the Web server is enabled with mechanisms that implement our framework.

The edge router associated with the Web server is able to monitor incoming and outgoing traffic volumes. This summarised information is passed to a detection component, labelled as step one in Fig. 2, which uses this information to detect the onset of problems associated with an attack. Xie *et al.* [15] describe an algorithm that enables the detection of the onset of problems associated with a flash crowd (or similarly, a DDoS attack) on a Web server, which uses a disparity between the expected response traffic volume and the actual traffic seen by monitoring, to suggest either network congestion or system overload. The detection mechanism in this scenario implements this algorithm.

If a problem is detected, Reaction components are informed with details of the attack (e.g., the source addresses of attackers and the severity of the attack, which in this case relates to the magnitude of the disparity of expected and actual response traffic volumes). Reaction can then initiate some form of remediation, if necessary, or inform the Operator via Visualisation (not shown in Fig. 2). Let us assume this attack is particularly intense and the network provider has a contract to protect the Web server. In this instance, Reaction components

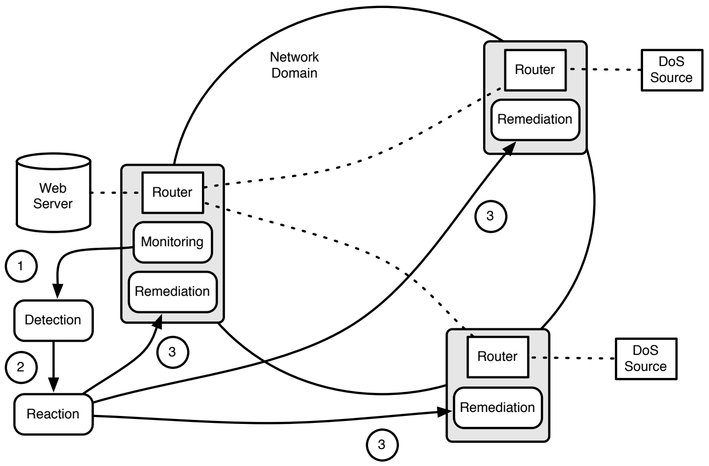


Fig. 2. Use of the INTERSECTION components in an example scenario

initiate a rate-limiting remediation mechanism. The severity of the rate-limiting is dictated by the magnitude of the problem as determined by detection (as Xie *et al.* propose [15]). Rate-limiting is invoked on the edge router associated with the Web server (perhaps, initially) and on the ingress routers to the network providing connectivity to the Web server.

When the rate-limiting is invoked on the ingress routers to the network, the Detection component associated with the edge router will naïvely assume the attack has ended and inform Reaction of the end of the attack. Hence the withdrawal of the remedy has to be determined by information from the remediation components themselves. Here, the rate-limiting at the edge router near the Web server could stop with the invocation of the remedy at the ingress routers.

4 Framework Realisation – Autonomic Distributed IDS

This section describes the design and implementation of Monitoring and Detection components of the INTERSECTION framework. If we assume that the network is aware of itself, security assurance might be regarded as a *service* inherently provided by the network infrastructure. In such a scenario, we can think of a system capable of deploying, both proactively and reactively, on-demand security services.

We propose a system (Fig. 3) which dynamically deploys the entities in charge of providing network security, based on knowledge of the *security status* of the network and its components. The *Preprocessor* and *Broker* entities implement the Monitoring function, collectively the *Detection Engine* and *Decision Engine* entities the Detection function, and the *Autonomic Co-ordinator* implements the Reaction functionality. These entities are described in more detail below.

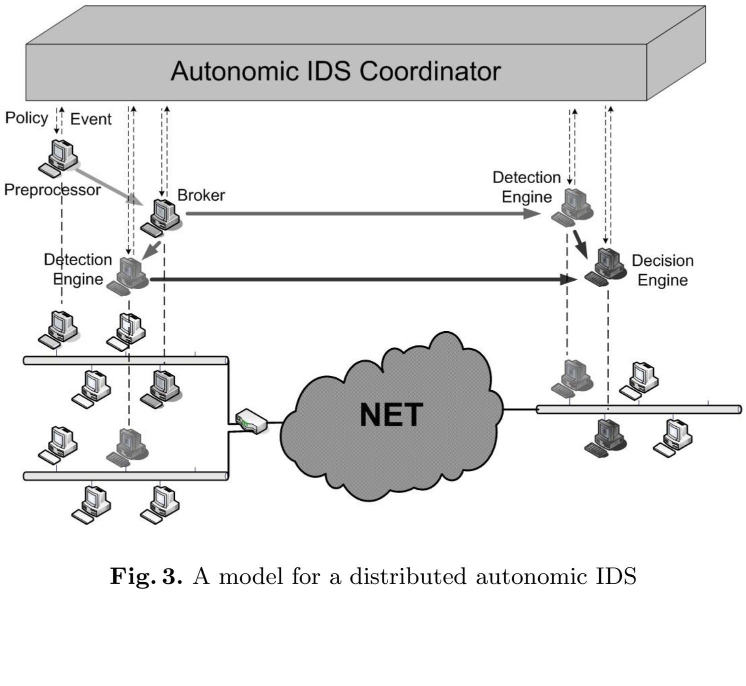


Fig. 3. A model for a distributed autonomic IDS

- *Preprocessor*
  Each such component is in charge of:
  1. capturing raw data from the network;
  2. extracting information (e.g. traffic features, flow information, etc.) from captured packets;
  3. sending the computed information to a *broker* entity.
- *Broker*
  The broker is a mediation component whose main tasks are:
  1. collecting information from all of the preprocessors spread across the network;
  2. distributing the collected information to one or more *detection engines*. The distribution strategy is defined by means of a policy-based approach.
- *Detection Engine*
  Detection engines receive traffic information from the broker and decide on whether or not such information represents a potential attack pattern, based on a specific detection technique.
- *Decision Engine*
  The main task of the decision engine resides in collecting information coming from detection engines and coming up with a final decision concerning the current network context. This might be done through a number of approaches, ranging from simple majority voting to much more complex solutions in which the various inputs provided by the detection engines are appropriately weighted on the basis of their degree of reliability (with respect to their own capability of detecting a particular set of malicious activities). Interaction between each detection engine and the decision engine might be realised using the IDMEF (Intrusion Detection Message Exchange Format) standard data format [8].
- *Autonomic Coordinator*
  This entity is responsible for the coordination of the other system components. It will work on the basis of a notification paradigm: the various system components can be seen as information producers, whereas the coordinator itself acts as an information consumer. Each time something worth reporting happens, an event is generated and captured by the coordinator which triggers the appropriate configuration task, e.g. the invocation of some form of Remediation activity.

In order to improve the system's performance and reliability, a diversity-based approach to the selection of detection techniques might be successfully exploited. In fact, by integrating different mechanisms for attack classification, the system may increase its detection capabilities, since more accurate analysis can be performed in order to detect specific attacks. Support for reliability evaluation can be added when multiple detection techniques are used, thus allowing evaluation of the accuracy of each issued decision. A multi-classification approach can be implemented either in a centralized or a distributed fashion: the combination of multiple detection techniques can be used both locally, in order to increase the reliability of alerts raised at each detection engine, and in a distributed fashion, by combining evidence of events communicated by different detection engines to the decision engine, in order to gain a global knowledge of the overall security status of the monitored networking scenario.

Dynamic deployment of the distributed system components is needed in order to ensure both flexibility of the architecture and robustness in the face of changes in network and traffic conditions. As an example, the IDS might need several preprocessors sniffing traffic in crucial points of the network, as well as several detection engines, each exploiting the best fitting detection technique according to the system status and node location. Such a dynamically distributed system might, for instance, adapt itself at the occurrence of a DDoS attack, by appropriately placing IDS engines in the most critical nodes (i.e., the nodes along the attackers' path) and by coordinating such nodes through a proper protocol (e.g., a protocol for tracing back the attack).

5 Conclusion

We have presented a framework for deploying network intrusion-tolerance systems, taking into consideration: the cost of performing over-detailed, full-time monitoring by adapting the level of monitoring according to current suspicions; the need to deal with different kinds of intrusions and deploy different remedies according to the available facilities of the network; the need to leave some decisions to manual operation; the need to recover the network automatically.

We have shown how the framework is congruent with the implementation plans described in Section 4. To date, a prototype implementation of the distributed IDS is being tested. For the autonomic coordinator, we have adopted an approach based on the Grid services paradigm: it is based on the WS-GRAM (Web Service Grid Resource Allocation and Management) module of the Globus Toolkit[1] and takes care of dynamically deploying the various system components.

Further work will involve investigating the suitability of our chosen approach to recovery, i.e., not to work around the monitoring used to detect intrusions, but to perform narrow-scope monitoring as part of remediation.

Acknowledgements

This research is funded by the European Commission's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 216585 (INTERSECTION Project).

[1] See `http://www.globus.org/toolkit/docs/4.0/execution/wsgram/WSGRAMFacts.html`

References

1. The INTERSECTION Project, `http://www.intersection-project.eu/`
2. Abad, C., Li, Y., Lakkaraju, K., Yin, X., Yurcik, W.: Correlation between netflow system and network views for intrusion detection
3. Abad, C., Taylor, J., Sengul, C., Yurcik, W., Zhou, Y., Rowe, K.: Log correlation for intrusion detection: A proof of concept. In: Proceedings of the 19th Annual Computer Security Applications Conference, ACSAC (2003)
4. Armannsson, D., Smith, P., Hjalmtysson, G., Mathy, L.: Controlling the Effects of Anomalous ARP Behaviour on Ethernet Networks. In: CoNEXT 2005, Toulouse, France, October 2005, pp. 50–60 (2005)
5. Bace, R.G.: Intrusion Detection. Macmillan Technical Publishing, Basingstoke (2000)
6. Baker, A.R., Caswell, B., Poor, M.: Snort 2.1 Intrusion Detection, 2nd edn. Syngress (2004)
7. Balepin, I., Maltsev, S., Rowe, J., Levitt, K.: Using Specification-Based Intrusion Detection for Automated Response. In: Vigna, G., Krügel, C., Jonsson, E. (eds.) RAID 2003. LNCS, vol. 2820. Springer, Heidelberg (2003)
8. Debar, H., Curry, D., Feinstein, B.: The Intrusion Detection Message Exchange Format (IDMEF). Number 4765 in RFC. IETF (March 2007)
9. Gamer, T., Schöller, M., Bless, R.: A granularity-adaptive system for in-network attack detection. In: IEEE / IST Workshop on Monitoring, Attack Detection and Mitigation, Tuebingen, Germany, pp. 47–50 (September 2006)
10. Lakhina, A., Crovella, M., Diot, C.: Mining anomalies using traffic feature distributions. In: Proceedings of ACM SIGCOMM 2005 (August 2005)
11. Li, Z., Taylor, J., Partridge, E., Zhou, Y., Yurcik, W., Abad, C., Barlow, J.J., Rosendale, J.: Uclog: A unified, correlated logging architecture for intrusion detection. In: Proceedings of the 12th International Conference on Telecommunication Systems - Modeling and Analysis, ICTSM (2004)
12. Lindqvist, U., Porras, P.A.: Detecting computer and network misuse through the production-based expert system toolset (p-best). In: Proceedings of the 1999 IEEE Symposium on Security and Privacy, Oakland, California, May 1999, pp. 146–161. IEEE Computer Society Press, Los Alamitos (1999)
13. Mahoney, M.V.: Network traffic anomaly detection based on packet bytes. In: Proceedings of ACM SAC 2003 (2003)
14. Paxson, V., Terney, B.: Bro reference manual (2004)
15. Xie, L., Smith, P., Jabbar, A., Banfield, M., Leopold, H., Hutchison, D., Sterbenz, J.P.G.: From Detection to Remediation: A Self-Organized System for Addressing Flash Crowd Problems. In: IEEE International Conference on Communications (ICC 2008), Beijing, China (May 2008)
16. Yin, X., Yurcik, W., Treaster, M., Li, Y., Lakkaraju, K.: Visflowconnect: netflow visualizations of link relationships for security situational awareness. In: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security, pp. 26–34. ACM Press, New York (2004)
17. Zhang, Y., Luo, J., Lu, H. (eds.): Wireless Mesh Networking Architecture, Protocols and Standards. Auerbach Publications (2007)

Weather Disruption-Tolerant Self-Optimising Millimeter Mesh Networks

Abdul Jabbar, Bharatwajan Raman, Victor S. Frost, and James P.G. Sterbenz

Information and Telecommunication Technology Center,
The University of Kansas, Lawrence, KS, USA
{jabbar,bharat,frost,jpgs}@ittc.ku.edu
http://www.ittc.ku.edu/resilinets

Abstract. Millimeter-wave networks have the potential to supplement fiber in providing high-speed Internet access, as well as backhaul for emerging mobile 3G and 4G services. However, due to the high frequency of operation (70–90 GHz), such networks are highly susceptible to attenuation from rain. In this paper, we present several mechanisms to overcome the disruptive effects of rain storms on network connectivity and service reliability. A resilient mesh topology with cross-layering between the physical and network layer has the capability to self-optimise under the presence of unstable links. We present a novel domain-specific predictive routing algorithm P-WARP that uses real-time radar data to dynamically route traffic around link failures as well as a modified link-state algorithm XL-OSPF that uses cross-layering to achieve resilient routing. Simulations are conducted to evaluate the effectiveness of the proposed algorithms.

Keywords: Resilient, survivable, weather disruption tolerant, millimeter-wave, mesh network, predictive routing, self-optimising.

1 Introduction

With the demand for high-bandwidth wireless communication growing at a rapid pace, coupled with the steady increase in the number of both fixed and mobile users, high-capacity backhaul networking solutions are needed that can quickly deliver large amounts of data as close to the end users as possible. Fiber-optic links provide the necessary bandwidth and reliability for both the backhaul and access links. However, there are two specific cases in which fiber-optic networks may be difficult or cost ineffective to deploy. The first case involves metropolitan areas in which the current fiber deployment is not adequate for the demand, either because existing fiber in a given location is at capacity, or because a particular service provider does not own fiber infrastructure and would have overcome significant deployment barriers. Secondly, in rural areas the subscriber density is generally inadequate to support the infrastructure needed for fiber-based broadband Internet access. With the average cost of new fiber ranging between $50,000 to $300,000 per kilometer, lower-cost fixed wireless links provide an increasingly attractive alternative.

Fixed wireless deployments (e.g. 802.16 links in WiMAX service) are rapidly becoming an integral part of cellular networks that are continually pushing higher speeds towards end users with current 3G services and plans for 3.5 and 4G services. This has led to the current demand of higher bandwidth at lower cost. For commercial vendors, backhaul links must be as reliable as SONET circuits with 50 ms restoration times.[1]

However, until recently fixed wireless has had limited deployment due to three fundamental limitations:

1. *Limited capacity:* Traditionally, the capacity of the wireless networks is an order of magnitude or more lower than fiber.
2. *Reliability:* Wireless links are inherently error prone and lossy.
3. *Shared medium:* The shared medium of wireless networks adds complexity due to contention.

Fixed wireless mesh networks [3,4,5] are emerging that attempt to overcome these limitations and are successful to varying degrees [6]. Notably, 802.16 and other line-of-sight solutions are attempting to emulate wired links. However, the capacity of these links are still significantly lower than that of fiber.

More recently, millimeter-wave transmission [7,8,9] is being explored with frequencies of 71–86 GHz. Millimeter-wave links do not suffer from the capacity and shared-medium limitations: commercial radios can deliver data rates up to 1 Gb/s with the potential to go up to 10 Gb/s [10] using highly directional transmissions resulting in very narrow (pencil) beams. As a result, there can be several links originating or destined at a single node without causing interference, thus allowing for higher degrees of connectivity through spatial reuse. However, millimeter-wave networks (MMWNs) do not address the issue of reliability in wireless networks. On the contrary, millimeter-wave transmission exacerbates the problem of link stability due to its susceptibility to precipitation [7,11,12,13]. Therefore, it is essential for the wireless mesh networks as a whole to be disruption tolerant in the presence of rain storms. When the link experiences partial degradations or complete failure, the network should self-optimise in order to provide high reliability and availability [7].

In this paper, we investigate the effect of weather-related disruptions on millimeter-wave links and the effect of error-prone and failed links on the service availability of the network. Furthermore, we discuss a network-layer approach to overcome link instability by routing around failures within a mesh topology. This requires carefully designed cross-layer mechanisms to leverage physical-layer information at the network layer. We discuss two solutions to based on the link-state algorithm that optimise the network either reactively or predictively. First, we present XL-OSPF, a cross-layered version of the well-known OSPF routing protocol. This reactive approach uses link-cost metrics derived from

[1] While one can debate the merits of such a tight requirement for many data applications, these backhaul links are intended to be used as T1 and SONET replacements for which the SONET APS (automatic protection switching) 50 ms bound [1,2] is considered a hard requirement by at least some large mobile service providers.

physical-layer information to maximise the performance of OSPF in the presence of degraded links. Secondly, we present a *predictive weather assisted routing protocol* (P-WARP), that utilises short-term weather forecasts to reroute *ahead* of an impending link failure. We compare the performance of both approaches in terms of efficiency and the ability to self-optimise under weather disruptions. While some of the findings in this paper regarding disruptive effects of weather are well known, an important contribution is the quantitative characterisation of these challenges and their potential solutions.

1.1 Scope

The effect of a weather disruption on millimeter-wave networks mainly depends on two factors: the rain rate and the geographic footprint of the storm. Both of these parameters vary from one geographic region to the other. For example, the analysis of weather data from southern Great Plains region of the US [14] shows that approximately 78% of all storms are smaller than 25 km in diameter and account for only 1.0% of the precipitation. Furthermore, only 1% of the storms are large (over 40 km diameter) and account for 85% of precipitation. The remaining 20% are medium sized storms (20–40 km diameter) accounting for 14% of precipitation. We draw two conclusions from this study: First, the majority of the storms are small enough for a metropolitan size mesh network (approx. 1000 km^2) to reroute traffic around the storm. Secondly, even moderate sized storms are likely to have small size heavy intensity regions since they do not account for a significant percentage of rainfall. Extensive measurements over a 12-month period in the Topeka – Lawrence – Kansas City corridor confirm these conclusions. This research is most applicable to geographic locations whose climate fits the above-mentioned storm patterns.

The rest of the paper is organised as follows: Section 2 presents the theory behind the effects of weather on millimeter-wave band. Disruption tolerance at the network layer is discussed in Section 3 with XL-OSPF and P-WARP presented in Sections 3.1 and 3.2 respectively. This is followed by performance analysis in Section 4. Finally, a summary of our research and future work is presented in Section 5.

2 Effect of Weather on Millimeter-Wave Links

The primary means of weather disruption on millimeter-wave links is rain fade, which leads to signal attenuation [7,15]. This degradation is due to the effects of scattering and water absorption caused by the rain droplets present in the transmission path. Moreover, rain droplet shape can also affect the amount of attenuation experienced by the transmission. Specifically, if the individual rain droplet shape is more oblate than spherical, then the transmission will suffer substantial attenuation if it is horizontally polarised relative to signals that are vertically polarised. As a result, signal polarisation, center frequency of the signal, and the rain rate are the three major factors that can impact the attenuation of a millimeter-wave transmission.

There are several models that relate millimeter-wave transmission attenuation to rain rates. Most notable among these models are the Crane models [16] and the ITU-R P.530 recommendation [17]. These models employ probabilistic and empirical methods to calculate the attenuation for a given probability distribution of the rain rate. The relationship between the signal attenuation and the rain rate is given by [16]:

$$A(R_p, D) = \alpha R_p^\beta \left[\frac{e^{u\beta d} - 1}{u\beta} - \frac{b^\beta e^{c\beta d}}{c\beta} + \frac{b^\beta e^{c\beta D}}{c\beta} \right], \quad d \leqslant D \leqslant 22.5 \text{ km} \tag{1}$$

$$A(R_p, D) = \alpha R_p^\beta \left[\frac{e^{u\beta d} - 1}{u\beta} \right], \; 0 < D \leqslant d \tag{2}$$

where A is the signal attenuation in dB, D is length of the path over which the rain rate is observed, R_p is the rain rate in mm/h, α and β are numerical constants, and u, b, c, and d are empirical constants defined as:

$$u = \frac{\ln(be^{cd})}{d}, \quad b = 2.3R_p^{-0.17}, \quad c = 0.026 - 0.03 \ln R_p, \quad d = 3.8 - 0.6 \ln R_p$$

Given a weather event with a rain rate of R_p mm/hr and a millimeter-wave link of length D, the received signal strength is decreased by A dB. This increases the effective bit error rate that is a function of received signal strength. In summary, the effect of a weather disturbance on the millimeter-wave network is quantified by the bit error rate and subsequently by the packet error rate on individual links. For example, measurements conducted using off-the-shelf millimeter-wave radios[2] show that rain rates higher than 5 mm/hr result in 100% packet error rate [18].

3 Routing in Millimeter-Wave Mesh Networks

MMWNs are connected in a arbitrary grid topology (Figure 1), in which several access nodes (e.g A, B, and D) are sending data towards a node (e.g C) that has external network access. For generality, we show that the nodes may be connected with alternative links (such as fiber or low frequency wireless for redundancy), even though the selection of appropriate alternatives is not the focus of this paper (see Section 5). Given this scenario, the objective of the MMWN routing protocols is to route around link failures without loosing data packets.

Earlier research has considered several different protocols for routing in mesh networks [19,20]. The focus of the community has been on issues related to shared medium contention and mobility, as well as metrics to quantify their effect on data transmission [21,22,23]. However, MMWNs are neither mobile nor do they have contention issues. Thus, with the exception of link instability, MMWNs share most of their characteristics with *wired* networks.

[2] 72.5 GHz radio with transmit power of 17 dBm, antenna gain of 43dBi, BFSK modulation, and Reed-Solomon FEC code over 10 km link using 1000B frames.

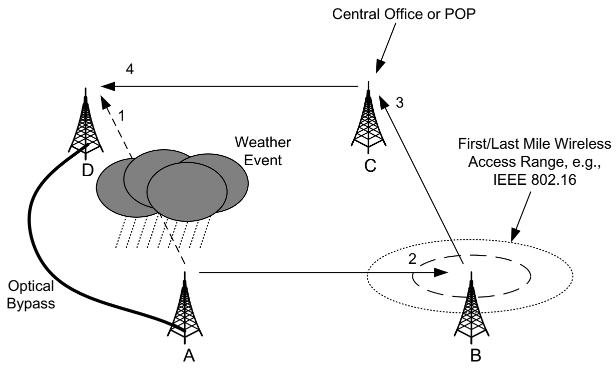


Fig. 1. Schematic of an MMWN with weather system

Conventional wired routing protocols, on the other hand, assume stable end-to-end paths composed of highly reliable links. Millimeter-wave links experience a continually varying link quality (state), particularly during rain storms, which leads to unstable end-to-end paths over time. Since the commonly used link-state protocol OSPF [24] does not mandate or recommend specific link metrics, the default implementations do not instrument explicit mechanisms in the protocol to support cost metrics based on physical link characteristics such as BER. Realising the need for OSPF to handle dynamic network scenarios, several modifications to support mobility are being currently reviewed [25,26]. However, in case of static MMWNs, the network dynamics are a result of link disruptions rather than mobility. In summary, the static, point-to-point, high-capacity but unreliable links of MMWNs present a unique case that can benefit from a domain specific network solution.

Due to the constantly varying link quality, it is necessary for the system to *self-optimise* in order to deliver reliable service. Thus, it is the goal of this paper to explore intelligent cross-layer mechanisms that optimise network routes so as to deliver the highest possible service availability and reliability.

3.1 XL-OSF: Cross-Layered OSPF

A cross-layer approach to link metrics could significantly improve the performance of dynamic link state algorithms [27,28] We chose OSPF (Open Shortest Path First) [24] because it has been widely researched and deployed. OSPF has two basic mechanisms to determine link state. One is the Link State Advertisements (LSAs) generated by each node carrying the status of all its links along with their costs, which are flooded throughout the network. Secondly, hello packets are used to determine if the link to a given neighbor is still alive. A *dead interval* based on the *hello interval* is used to detect dead links. The routes are reactively updated after the LSAs propagate through the network. With rapidly varying link quality, the only mechanism through which OSPF in its default configuration can detect link degradations is when four consecutive hello packets are dropped. Since the size of the hello packets is much smaller than data packets, a BER that results in four consecutive hello drops will correspond to a significantly higher data packet drop rate.

One mechanism that can be used to improve the performance of OSPF is a cost metric that is proportional to the bit error rate of the link. However, this is a difficult proposition given the lack of information exchange between the link that detects the actual packet losses and the network layer that determines the routes. Several mechanisms are possible that use in band (packet header) or out-of-band (probe packets) signaling to determine the actual packet error rate.

For the purpose of analysis, we assume such a mechanism that informs the routing layer of the effective packet error rate (PER). We define a cost metric that is proportional the the effective PER, carefully choosing the scaling factor so as to avoid route fluctuations. The exact values of the scaling factor and link weights are given in Section 4.1. The performance of OSPF with this cost metric is discussed in Section 4.2.

Even with the error-based cost metrics, OSPF remains a reactive protocol that requires a finite amount of time before it adapts to changes in link state. If the application or service demands a highly-reliable service, the reactive protocols must have a very short update interval on the order of milliseconds. But this adds unacceptable level of overhead even for broadband networks as discussed in Section 4.2. In the following section, we discuss a predictive alternate for routing.

3.2 P-WARP: Predictive Weather Assisted Routing Protocol

As discussed above, reactive algorithms cannot meet stringent 50 ms restoration service requirements in MMWNs during weather disruptions. Furthermore, it is difficult to measure effective BER or PER at the routing layer without an explicit signaling mechanism. In this section, we investigate the use of information *external* to the network in order to predict the future state of links.

The proposed predictive routing algorithm is a link-state algorithm that utilises weather radar to forecast the future state of the link. In contrast to the OSPF solution mentioned previously, the primary difference is the mechanism through which the link costs are obtained. While XL-OSPF depends on BER measurement from errored packets, we propose P-WARP (predictive weather assisted routing protocol), in which BER of each link is calculated from weather data that is modeled in real-time using the following methodology:

Modeling Effective Rain Rate from Weather Radar. Short-term weather prediction obtained from doppler radar is used to model the predicted rain in the geographical region of the mesh network. Radar data is first passed through a geometric model that maps the areas corresponding to different rain intensities to ellipses. For the sake of simplicity, all precipitation regions are divided in to 3 categories representing heavy, moderate and light, or no rain. The rain rates assigned to each category depends on the characteristics of transceivers such as link budget and the length of the transmission link. For example, measurements on a typical off-the-shelf millimeter-wave system suggest that for a 10 km link, heavy rain corresponds to an average rate of 5 mm/hr or more, moderate rain corresponds to 2–5 mm/hr, and light or no rain corresponds to an average rate less than 2 mm/hr.

Link Cost from Effective Rain Rate. The rain rate obtained in the previous step is then used to determine the effective BER on all the links of the network based on calculations from Section 2. This processing is done at a either a single *core node* or a small subset of nodes that are connected to the external network and are capable of receiving weather-radar data. The topology and physical location of the network is pre-programmed into the software module that does the link BER calculation as well as the PER (packet error rate) for a predefined average packet size. Finally, the link costs are calculated from the PER as discussed in Section 4.1.

Link Status Updates and Route Computation. The weather-based link status updates (WLSUs) in P-WARP are slightly different from the link state advertisements (LSAs) of OSPF. WLSUs are generated from the core nodes and contain the costs of all links in the network based on their *predicted* quality. These weather based updates are flooded throughout the network. When an individual node receive a WLSU, it recomputes routes using the Dijkstra shortest path first algorithm. However, unlike OSPF individual nodes do not generate separate LSAs for the links to their neighbors. This approach significantly reduces the protocol overhead because only *one* update is generated for *all* the links and updates are generated only when a change in one or more link costs is predicted. Thus, the network reroutes traffic *ahead* of the incoming storm thereby minimising the packet loss.

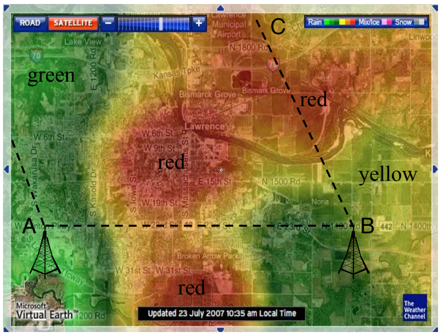
Route Sensitivity. Due to storm movement, the effective BER on each link will vary continuously over the duration of the event. In order to avoid route flaps and false alarms, we use thresholds along with hysteresis. A minimum noticeable change $\text{BER}_{\text{thresh}}$ is defined below which all BER changes are ignored, which determines the resolution of the algorithm. Further, a hysteresis percentage H_{thresh} determines the minimum change in the cost of a link for an update to be generated.

4 Performance Evaluation

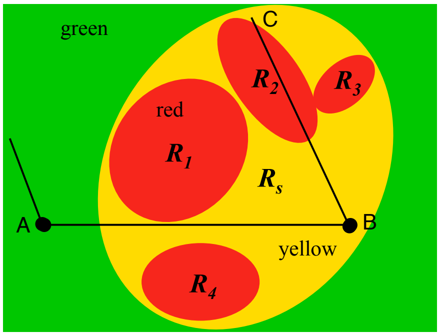
In this section, we present our methodology for modeling and simulating storms. We also present the specific details of XL-OSPF and P-WARP which were used in the simulations. Finally, we quantify the disruptive effect of storms as well as the disruption tolerance of the proposed mechanisms.

4.1 Methodology

Storm Modeling. A typical example of a rain storm snapshot over part of the mesh network is shown in Figure 2(a) as a radar reflectivity map. The link length



(a) Radar image



(b) Geometric model

Fig. 2. Example storm model

in this example is 10 km. The geometric model of the storm at that instant is shown in Figure 2(b). Our model divides the radar reflectivity image in to regions of high (red), low (yellow) and little or no (green) precipitation. Each region is modeled as an ellipse (R_1–R_5) and individual links are represented as line segments. Equation 2 is used to calculate the effective BER and PER based on the intersection between the links and storm ellipses.

This example shows that while certain links (e.g. $\overline{\mathsf{BC}}$) are severely degraded due to the heavy rain, it is possible to re-route traffic on other adjacent links (e.g. $\overline{\mathsf{BA}}$) that are not in an intense rain region. Recall from Section 1.1 that this pattern of rain intensity distribution is typical for a majority of storms in the US Great Plains. In order to model a storm in real time, we continuously calculate the link attenuation and generate WLSUs as the storm moves through the network. The rate of WLSU generation is dependent on the dynamics of the storm relative to the network.

Routing Protocol Configuration. We evaluate the performance of XL-OSPF and P-WARP and compare them against OSFP using common defaults, as well as static routing as the lower bound on performance. For standard OSFP, we use the default values of 10 and 40 seconds for *hello* and *dead* intervals respectively [29]. The LSAs in XL-OSPF are rate limited to 10 seconds, whereas WLSUs in P-WARP are rate limited to 30 seconds, predicting the weather approximately a minute in advance.

For both XL-OSPF and P-WARP the link costs and other protocol parameters are set as follows. Assuming uniform distribution of the bit errors, the cost of a link between two nodes i and j is calculated as:

$$C_{ij} = P \times \mathrm{BER}_{ij} \times \gamma \tag{3}$$

where P is the average packet size on the network, BER_{ij} is the bit error rate observed on the link, and γ is the scale factor. The scale factor determines the sensitivity of the link cost with respect to change in BER and is set to 1000 in our simulations. A $\text{BER}_{\text{thresh}}$ of 10^{-8} is used to define the minimum observable change in BER with hysteresis using H_{thres} of 10% to avoid excessive route flaps in the network. Finally, the value of cost is bounded in the range of $[1, 1000]$ which determines the maximum number of hops a packet can traverse in order to avoid an error-prone or lossy link. Since, the objective here is to avoid disrupted links at all costs, we set this range to 1000.

4.2 Simulation and Results

We conducted simulations using storm modeling software that we developed in Matlab and the ns-2 simulator [30].

Simulation Setup. The simulated topology consists of 16 nodes connected in a square mesh as shown in Figure 3. The millimeter-wave links between any two nodes are 10 km long. The nodes remain fixed at their locations throughout the simulation. Modeling a cellular backhaul network, nodes 0 and 15 are connected to the external network (Internet) and hence are sink nodes. The remaining 14 nodes generate traffic at a rate of 2.4 Mb/s that is destined to one of the two sink nodes, picked randomly. The generated traffic is CBR (constant bit rate) over UDP with a packet size of 1000 bytes. We use two different storms to evaluate the disruption tolerance of the proposed mechanisms. These storms consist of an outer yellow ellipse varying between 20–30 km in diameter and inner red ellipses with a diameter varying between 5–10 km. The first storm consists of four high-intensity regions whereas the second storm contains two high-intensity regions. Both storms last for a duration of 30 minutes over the mesh network. We evaluate the packet delivery ratio and the service availability of the network for four different routing mechanisms.

Packet Delivery Ratio. The packet-delivery ratios averaged over a window of two seconds are shown in Figure 4 for all four routing protocols. This plot shows the instantaneous response of the network to the first simulated storm. As the individual links fail due the storm, the delivery ratio of the network falls rapidly. The time taken by the network to recover from link failures depends on

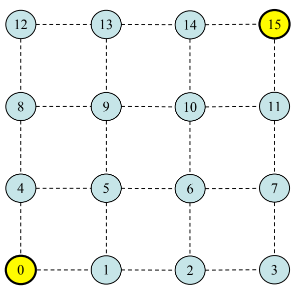


Fig. 3. Simulation topology

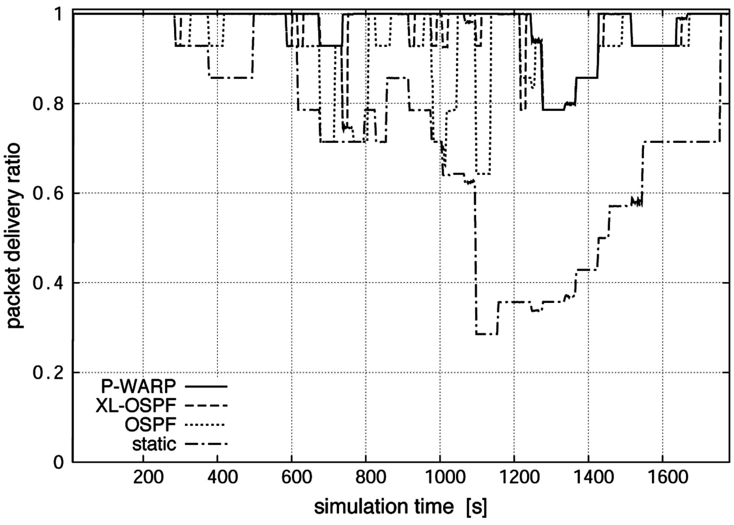
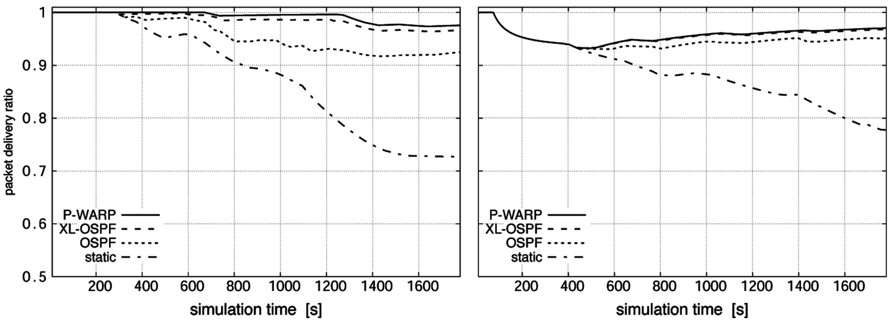


Fig. 4. Windowed average of received packets: first storm

the routing protocol. Static routing, as expected, performs very poorly marking the lower bound. OSPF without any modification performs better than static because it can sense link outages from the loss of four consecutive hello packets. However, the delay in detection and route re-computation results in significant packet loss. XL-OSPF performs better than static and standard OSPF because it can detect degrading links in a shorter time from the their cost, as advertised in the link state updates. Since the cost metric is directly proportional to the link error rate, high PER paths are avoided whenever possible. P-WARP outperforms all three protocols because it can predict an upcoming link failure from weather updates and reroutes traffic ahead of the disruption.

For example, consider the packet delivery ratios at $t = 1100$ s in Figure 4. At 1100 seconds, the storm disrupts several links causing severe packet loss in the case of static routing. Furthermore, the network does not recover until the storm passes at 1600 seconds. OSPF, on the other hand, detects failed links and recovers at 1140 seconds, indicating a 40 sec recovery time which corresponds to the dead interval. XL-OSPF recovers much faster compared to OSPF and static routing. It takes approximately 10 seconds to get back to 100% delivery ratio. Finally, P-WARP does not drop any packets at all confirming its predictive behavior. Accurately predicting the impending disruption, P-WARP preemptively routes data on stable paths, thereby avoiding the failed links completely.

In order to compare the aggregate performance of the protocols, the cumulative average of the packet delivery ratio is shown in Figure 5. The cumulative average of packets delivered by XL-OSPF is very close to that of P-WARP, both of which outperform conventional OSPF. In the case of XL-OSPF, the frequency of LSAs determines the reaction time of the network; strict restoration times would require very short update intervals. Because of its predictive nature, P-WARP has two distinct advantages: first, it has no reaction time which



(a) First storm (b) Second storm

Fig. 5. Cumulative average of received packets

might be necessary if stringent service requirements such as 50 ms restoration are to be met; secondly, the frequency of weather updates does not scale with the restoration times, leading to lower protocol overhead.

Service Availability. In order to evaluate the availability of the network we consider a media streaming-application. The availability of the network is then defined as the percentage of time that the network can deliver data reliably to the destination with *no errors* within a given delay bound. Table 1 shows the availability of the network for the duration of the simulation for two different weather events. We selected a delay bound of 4 ms that is derived from cellular backhaul deployments. Predictive routing outperforms all the other cases since it routes traffic ahead of time, thus increasing the duration during which the service remains available. Among the other three mechanisms, XL-OSPF performs closest to predictive routing.

We have conducted several more simulations with varying update intervals for OSPF and XL-OSPF, and have observed that while the general relationship between the OSPF, XL-OSPF and P-WARP routing remains the same, the performance gap between XL-OSPF and P-WARP decreases with increasing LSA frequency, as expected.

Overhead. Static routing does not generate any overhead traffic. Conventional OSPF generates periodic hello messages as well as LSAs. However, the frequency

Table 1. Service availability

Protocol	Availability Storm-1	Availability Storm-2
P-WARP	0.9700	0.9638
XL-OSPF	0.9666	0.9554
OSPF	0.9514	0.9209
Static	0.7769	0.7304

of the LSA does not affect the performance because the quality of the link is not reflected in the its cost metric. On the other hand, XL-OSPF uses a cost metric that is proportional to link BER and hence generates frequent updates that are flooded in the network. In order to react quickly to weather disruptions, XL-OSPF must generate LSAs at a higher rate, leading to a significant increase in overhead. The number of updates generated in P-WARP is comparatively lower for two reasons. First, a single WLSU update carries the predicted costs for all the links. Hence, individual nodes do not generate link state updates. Secondly, an update is generated only when there is a change in the predicted BER of one or more links. Since the performance of the P-WARP is not dependent on the arbitrary update frequency, WLSUs are rate limited to one in every 30 seconds, representing the weather-prediction interval.

5 Conclusions and Future Work

In this paper, we have shown that millimeter-wave mesh networks must be self-optimising in order to provide reliable service under weather disruptions. It is observed that conventional reactive routing mechanisms do not perform well without a cost metric that reflects the physical status of the link. We presented two domain-specific mechanisms that utilise cross-layering between physical and network layer to improve the availability of network. In XL-OSPF, a cost metric that is proportional to the BER is shown to perform well under lightly loaded conditions. A novel predictive routing algorithm P-WARP based on a short-term weather forecast outperforms reactive routing both in terms of throughput and overheard. The proposed algorithm finds the optimal paths with relatively low frequency of routing updates. Simulations show that MMWNs with the proposed self-optimising mechanisms form a viable low cost solution to high speed backhaul and access networks.

We are currently evaluating the proposed schemes with real storm data collected by the US National Weather Service. Preliminary results based on real storms indicates a good match with the results shown in this paper. We intend to study several more cases to evaluate the performance of the network for different type of applications as well as different type of storms such as tropical, cyclonic. A fiber or low-frequency wireless bypass to overcome node failures remains a part of our future work. In order to support heavily loaded networks, we are investigating metrics that consider link quality as well as available capacity of the active links in its forwarding decisions. Future work also includes image processing of radar data to automatically calculate link attenuation on a linear scale.

Acknowledgments

We would like to acknowledge the meteorological guidance provided by Donna Tucker, the help with illustrations from Dan DePardo, and the technical support from Adam Hock. We also acknowledge the attenuation modeling work done by

Alex Wyglinski and Vinay Muralidharan. This project was funded in part by Sprint; we acknowledge the technical contributions and guidence of Tim Euler.

References

1. ANSI T1.105.01: Telecommunications – synchronous optical network (SONET) – automatic protection switching. American National Standard T1.105.01-2000 (March 2000)
2. ITU-T G.841: Types and characteristics of SDH network protection architectures. ITU-T Recommendation G.841
3. Uchida, D., Sugita, M., Toyoda, I., Atsugi, T.: Mesh-type broadband fixed wireless access system. NTT Technical Review 2(1), 44–54 (2004)
4. Wu, Y., Hui, J., Sun, H.: Fast restoring Gigabit wireless networks using a directional mesh architecture. Computer Communications 26, 1957–1964 (2003)
5. Whitehead, P.: Mesh networks: A new architecture for broadband wireless access systems. In: IEEE Radio and Wireless Conference, Denver, CO, USA, pp. 43–46 (September 2000)
6. Akyildiz, I.F., Wang, X., Wang, W.: Wireless mesh networks: A survey. Computer Networks 47(4), 445–487 (2005)
7. ITU-R F.1704: Characteristics of multipoint-to-multipoint fixed wireless systems with mesh network topology operating in frequency bands above about 17 GHz. ITU-R Recommendation F.1704 (2005)
8. Khan, J.A., Alnuweiri, H.M.: Traffic engineering with distributed dynamic channel allocation in BFWA mesh networks at millimeter wave band. In: Proceedings of the 14th IEEE Workshop on Local and Metropolitan Area Networks, Chania, Greece, pp. 1–6 (September 2005)
9. Ohata, K., Maruhashi, K., Ito, M., Nishiumi, T.: Millimeter-wave broadband transceivers. NEC Journal of Advanced Technology 2(3), 211–216 (2005)
10. Torkildson, E., Ananthasubramaniam, B., Madhow, U., Rodwell, M.: Millimeter-wave MIMO: Wireless Links at Optical Speeds. In: Proceedings of the 44th Allerton Conference on Communication, Control and Computing, Monticello, Illinois, USA (September 2006)
11. Izadpanah, H.: A millimeter-wave broadband wireless access technology demonstrator for the next-generation internet network reach extension. IEEE Communications Magazine, 140–145 (September 2001)
12. Hendrantoro, G., Indrabayu, Suryani, T., Mauludiyanto, A.: A multivariate autoregressive model of rain attenuation on multiple short radio links. IEEE Antennas and Propagation Letters 5, 54–57 (2006)
13. Paulson, K.S., Gibbins, C.J.: Rain models for the prediction of fade durations at millimetre wavelengths. IEE Proceedings - Microwaves, Antennas and Propagation 147(6), 431–436 (2000)
14. Tucker, D.F., Li, X.: Characteristics of warm season precipitating storms in the Arkansas-Red river basin. Journal of Geophysical Research (submitted, 2009)
15. Hou, P., Zhuang, J., Zhang, G.: A rain fading simulation model for broadband wireless access channels in millimeter wavebands. In: Proceedings of the IEEE Vehicular Technology Conference, Tokyo, Japan, May 2000, vol. 3, pp. 2559–2563 (Spring 2000)
16. Crane, R.: Prediction of Attenuation by Rain. IEEE Transactions on Communications 28(9), 1717–1733 (1980)

17. ITU-R P.530: Propagation data and prediction methods required for the design of terrestrial line-of-sight systems. ITU-R Recommendation P.530
18. Folies, J., Raman, B., Jabbar, A., Smith, D., DePardo, D., Sterbenz, J.P., Tucker, D., Euler, T., Frost, V.S.: Experience with frame transfer over a millimeter-wave link. IEEE Communication Letters (submitted, 2008)
19. Waharte, S., Boutaba, R., Iraqi, Y., Ishibashi, B.: Routing protocols in wireless mesh networks: challenges and design considerations. Multimedia Tools Appl. 29(3), 285–303 (2006)
20. Draves, R., Padhye, J., Zill, B.: Comparison of routing metrics for static multi-hop wireless networks. SIGCOMM Computer Communications Review 34(4), 133–144 (2004)
21. Yang, Y., Wang, J., Kravets, R.: Designing routing metrics for mesh networks. In: WiMesh 2005: Proceedings of the IEEE Workshop on Wireless Mesh Networks (2005)
22. Ramachandran, K., Sheriff, I., Belding, E., Almeroth, K.: Routing stability in static wireless mesh networks. In: Proceedings of the eighth Passive and Active Measurement conference, Louvain-la-neuve, Belgium (April 2007)
23. Kim, B.C., Lee, H.S.: Performance comparison of route metrics for wireless mesh networks. IEICE Transactions on Communications 89(11), 3124–3127 (2006)
24. Moy, J.: OSPF version 2. RFC 2328 (Standard) (April 1998)
25. Chandra, M., Roy, A.: Extensions to OSPF to support mobile ad hoc networking. Internet-Draft, draft-ietf-ospf-manet-or-00, Work in progress (February 2008)
26. Spagnolo, P., Henderson, T.: Comparison of proposed OSPF MANET extensions. In: Proceedings of the IEEE Military Communications Conference, Washington, DC, USA, pp. 1–7 (October 2006)
27. Iannone, L., Khalili, R., Salamatian, K., Fdida, S.: Cross-layer routing in wireless mesh networks. In: Proceedings of the 1st International Symposium on Wireless Communication Systems, pp. 319–323 (2004)
28. Pei, G., Spagnolo, P.A., Bae, S., Henderson, T.R., Kim, J.H.: Performance improvements of OSPF MANET extensions: A cross layer approach. In: Proceedings of IEEE Military Communications Conference, Florida, USA, pp. 1–7 (October 2007)
29. Cisco: Cisco IOS IP Command Reference, vol. 2, 4: Routing Protocols. Cisco Systems, Inc. Release 12.3(11)T edn. (2008)
30. Ns-2: The network simulator (July 2008), `http://www.isi.edu/nsnam/ns/`

Bio-inspired Feedback Loops for Self-Organized Event Detection in SANETs

Falko Dressler

Autonomic Networking Group
Computer Networks and Communication Systems, University of Erlangen, Germany
`dressler@informatik.uni-erlangen.de`

Abstract. We focus on the question of self-organized scheduling of event detection in SANETs. This question is especially challenging in very dynamic environments. Recently, a number of self-organization methods have been published that focus on network-centric operation in such networks. Based on our previously developed RSN system, which is a light-weight programming scheme for SANETs, we study the feasibility of promoter / inhibitor based feedback for self-organized schedule management of rule executions. Early simulation results outline the feasibility of the approach.

Keywords: Sensor and actor networks, feedback loop, promoter, inhibitor, network-centric operation, bio-inspired networking.

1 Introduction

Programming of heterogeneous Sensor and Actor Networks (SANETs) is being investigated since several years. Especially, light-weight solutions are demanded for simplified updates of programs during run-time. Besides the requirements and challenges in terms of energy efficiency and the capability to work on low-resource embedded systems, additional coordination among the nodes need to be supported in SANETs.

Figure 1 depicts a typical SANET. Several sensor nodes are shown that directly interact with associated, i.e. co-located actuators. The system-inherent actuation facilities need to be controlled, i.e. activated and driven, by network-inherent sensor measures. This leads to new challenges such as critical real-time operation requirements [1]. Possible solutions can be found in approaches related to the main ideas of *autonomic networking*, i.e. the development of self-managing networks. Accordingly, most recent approaches focus on network-centric operation, i.e. the data management without central base stations, as the key paradigm to handle the mentioned challenges.

In previous work, we developed Rule-based Sensor Network (RSN), a light-weight programming scheme for SANETs [2]. RSN provides all the means for programming heterogeneous SANETs including techniques for updating programmed rule-sets during run-time. An open issue is the scheduling of rule executions, which is especially challenging in dynamic environments, i.e. in case

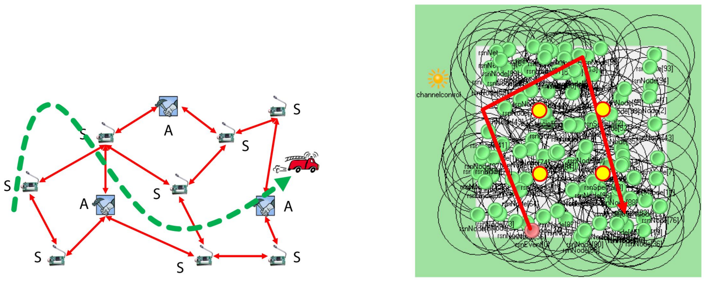


Fig. 1. Network-centric operation in SANETs for dynamic event detection: principle (left) and simulated scenario (right)

of mobility, unforeseeable events, and similar dynamics. The need for optimized scheduling of program execution becomes clear in the following application scenario that – while simplistic from a high lever point of view – shows a number of characteristic challenges for developing self-organizing SANETs. Figure 1 depicts a number of sensors trying to detect one or more mobile targets. Obviously, the detection quality increases if the sensors perform the measurement (and transmit the results) frequently. In contrast, this will lead to many unsuccessful measurements wasting energy and reducing the *network lifetime* [3].

From an algorithmic point of view, first solutions for optimized event detection are available. Akan et al. developed an event detection mechanism for SANETs [4], which provides efficient path selection between the monitoring nodes and the event sink. Similarly, the sensor-actor coordination approach by Melodia et al. [5] includes means for associating sensors to adjacent actor nodes.

This paper presents a bio-inspired promoter / inhibitor based feedback system that allows self-organized schedule management for rule execution. This approach has already successfully been applied to other problem domains [6, 7]. We implemented this scheme for use with RSN. For evaluation purposes, we developed an appropriate simulation model to analyze the behavior and performance. Early results clearly outline the feasibility of the approach.

2 RSN – Rule-Based Sensor Network

2.1 RSN Operation

Recently, we developed RSN, a rule-based programming system for supporting network-centric operation in heterogeneous SANETs [2]. Basically, RSN is an architecture for data-centric message forwarding, aggregation, and processing, i.e. using self-describing messages instead of network-wide unique address identifiers. In this earlier work, we proved that RSN explicitly outperforms other SANET protocols for distributed sensing and network-centric data pre-processing in two dimensions: (a) reactivity of the network, i.e. the response times for network-controlled actuation can be reduced, and (b) communication overhead, i.e. the bandwidth utilization on the wireless transmission channels was improved.

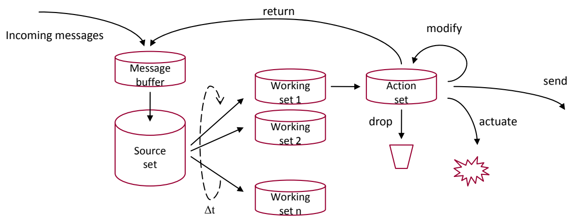


Fig. 2. The working behavior of a single RSN node. Received messages are stored in a buffer. After Δt, they are selected to a working set according to specific criteria, and finally being processed, i.e. forwarded, dropped, etc.

Figure 2 depicts the working behavior of a single RSN node. After receiving a message, it is stored in a message buffer. The rule interpreter is started periodically (after a fixed Δt) or after the reception of a new message. An extensible and flexible rule system is used to evaluate received messages and to provide the basis for the node programming scheme. Thus, the local behavior is controlled by a rule interpreter in form of simple state machines. The interpreter is applying the installed rules to previously received messages.

In several experiments, the period of RSN execution Δt has been identified as a key parameter for controlling the reactivity vs. energy performance of the entire RSN-based network. Basically, the duration of messages stored in the local node introduces an artificial per-hop delay. The optimal value for Δt affects the aggregation quality vs. real-time message processing.

2.2 Feedback Loop Controlled RSN

In the selected scenario, i.e. monitoring of dynamic, mobile entities in a SANET, an optimized value of Δt can be exploited for optimized event detection. In the following, we describe a bio-inspired approach based on dynamic promoter and inhibitors for self-organized event detection. The concept is inspired by the self-regulating process of blood pressure control by the Angiotensin-Renin regulatory process [8]. The adaptation of biological promoters and inhibitors is depicted in Figure 3. Local success is defined as a successful target detection. We exploit the characteristics of the continuous mobility pattern of the monitored targets: in each time step, the target under observation can be expected to be present in a close proximity of its last position. This is depicted as success of neighbors, i.e. identification of a target. Furthermore, we assume to have no knowledge about the specific mobility model, i.e. the exact direction of the target – in the simulation experiments, we employed different mobility patterns of the target to analyze the performance of the approach.

In particular, we used the following RSN rules, which are installed on each sensor node, to determine the current network situation and, therefore, to adapt the local rule execution frequency. If the local measurement was unsuccessful, the rule execution period Δt is set to its maximum value to achieve optimized energy performance (in this example, the maximum period is set to 10 s):

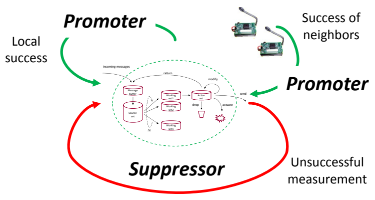


Fig. 3. Feedback loop controlled operation using promoter / suppressor mechanisms

```
if :count = 0 then {
  !controlMgmt($control:=rsnMgmtSetEvalInt, $text:="10s");
  !stop;
}
```

In contrast, if the local measurement was successful, Δt is reduced to the minimum (in the example, it is set to 1 s):

```
if :count > 0 then {
  !controlMgmt($control:=rsnMgmtSetEvalInt, $text:="1s");
  !send($type:=rsnSensorLuminance, $value:=@maximum of $value);
  !drop;
}
```

Finally, the most interesting case is the exploitation of overheard messages from neighboring nodes. If such a node successfully detects a target, the radio message will usually travel faster compared to the target itself. Thus, depending on the distance to the node that detected the event, the rule execution period Δt can be updated. We used the average of the hop count as a basis measure as there is no localization scheme in place in our example. Alternatively, the real distance to the neighboring node could be evaluated.

```
!controlMgmt($control:=rsnMgmtSetEvalInt, $text:=@average of $hopCount);
```

Basically, the shown rules only represent the basic idea of the feedback loops to be used to adaptively set the rule execution period Δt. We experimented with a number of settings as well as algorithms. Selected results of these experiments are presented in the following.

2.3 Simulation Experiments

In the following, we present some early simulation results. In order to evaluate the efficiency of RSN, we compared it to the typical setup used in other sensor network scenarios for event detection. Multiple sensor nodes are continuously measuring environmental conditions, i.e. detect mobile targets, and transmit this information to actors in their neighborhood. In order to evaluate the communication behavior in this scenario, we created a simulation model in which 100 sensor nodes are placed on a rectangular playground. The nodes are either distributed in form of a regular grid or using a random pattern. In addition to these sensor nodes, four actors are included in the middle of each quadrant as depicted in Figure 1 (right). Furthermore, we added a mobile target that moves either on a rectangular trajectory or based on a random waypoint model.

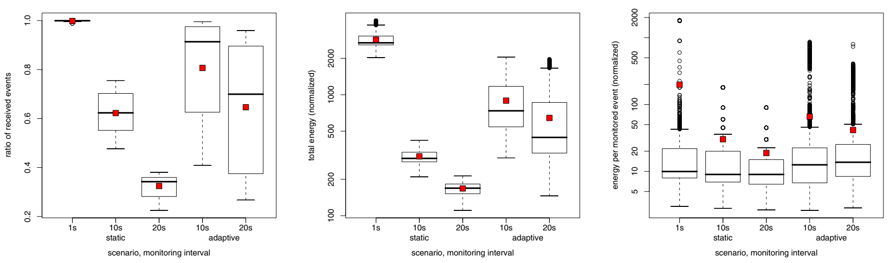


Fig. 4. Ratio of detected (monitored) events compared to all possible detections (left) and energy performance: total energy including all communication activities (middle), and the energy per monitored event (right)

The following scenarios have been analyzed: (a) static configuration of the RSN rule execution period as a baseline measurement to evaluate the adaptive behavior, and (b) two versions of the dynamic feedback based approach (different initial periods). Additionally, we modified the deployment pattern and the mobility model of the mobile target. All results are shown as boxplots. For each data set, a box is drawn from the first quartile to the third quartile, and the median is marked with a thick line. Additional whiskers extend from the edges of the box towards the minimum and maximum of the data set.Additionally, the mean value is depicted in form of a small filled square. In most graphs, the overall mean and median are shown in the middle bar.

The efficiency of the event detection algorithm can be analyzed using the number of detected events. The earlier an event can be detected, the higher the probability another sensor in the local vicinity can detect the same event, i.e. the moving target, again. Figure 4 (left) shows the performance of the static vs. the adaptive approach depicting the ratio of events as received by the actor nodes. As can be seen, the event detection ratio is optimal for the static scenario with 1 s sampling rate. The adaptive approach always outperforms the respective static configurations. Obviously, the feedback loop works in the positive direction, i.e. the amplification via neighbors and successful local measurement.

As a second measure, we analyzed the energy performance, which also describes the energy efficiency of the entire system, and thus, the possible *network lifetime* [3]. Figure 4 (middle) depicts the total energy consumed by the SANET normalized to an energy per operation ratio of one, i.e. one event detection operation costs one energy unit. Both the measurement and the communication energy are considered. For the static scenarios, the event detection rate is exactly defined by the rule execution interval Δt. In the adaptive cases, the energy values also outline the behavior of the feedback loops. In order to compare the energy performance with the quality of the event detection, the energy per monitored event is analyzed in Figure 4 (right). As can be seen, the mean of all energy per monitored event measurements is almost identical (however, the average for the static 1 s scenario and the adaptive scenarios is a bit higher). Therefore, we can conclude that even though more energy is needed in the adaptive solutions compared to the static ones, the overall performance is much better as almost no energy is wasted for monitoring activities while there is no target around.

3 Conclusion

We presented early results of an adaptive solution for rule execution on distributed SANET nodes. The algorithm is based on a bio-inspired feedback loop that uses promoters, i.e. positive feedback, and suppressors, i.e. negative feedback. We implemented this scheme based on our previously developed RSN system, which provides collaborative sensing and processing in SANETs with purely local rule-based programs. Our approach exploits positive feedback from neighboring nodes that already detected the target, i.e. the probability of target detection increases, and from successful local monitoring. Inhibitory effects are introduced by negative feedback from unsuccessful operations. From the simulation results, we can see that the adaptive scenarios provide a good trade-off between energy performance and clearly improved event detection rates.

References

1. Akyildiz, I.F., Kasimoglu, I.H.: Wireless Sensor and Actor Networks: Research Challenges. Elsevier Ad Hoc Networks 2, 351–367 (2004)
2. Dressler, F., Dietrich, I., German, R., Krüger, B.: Efficient Operation in Sensor and Actor Networks Inspired by Cellular Signaling Cascades. In: 1st ACM/ICST International Conference on Autonomic Computing and Communication Systems (Autonomics 2007), Rome, Italy. ACM, New York (2007)
3. Dietrich, I., Dressler, F.: On the Lifetime of Wireless Sensor Networks. ACM Transactions on Sensor Networks (TOSN) (to appear, 2008)
4. Akan, Ö.B., Akyildiz, I.F.: Event-to-Sink Reliable Transport in Wireless Sensor Networks. IEEE/ACM Transactions on Networking (TON) 13(5), 1003–1016 (2005)
5. Melodia, T., Pompili, D., Gungor, V.C., Akyildiz, I.F.: A Distributed Coordination Framework for Wireless Sensor and Actor Networks. In: 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (ACM Mobihoc 2005), Urbana-Champaign, Il, USA, pp. 99–110 (May 2005)
6. Dressler, F.: Bio-inspired Promoters and Inhibitors for Self-Organized Network Security Facilities. In: 1st IEEE/ACM International Conference on Bio-Inspired Models of Network, Information and Computing Systems (IEEE/ACM BIONETICS 2006), Cavalese, Italy. IEEE, Los Alamitos (2006)
7. Neglia, G., Reina, G.: Evaluating Activator-Inhibitor Mechanisms for Sensors Coordination. In: 2nd IEEE/ACM International Conference on Bio-Inspired Models of Network, Information and Computing Systems (IEEE/ACM BIONETICS 2007), Budapest, Hungary (December 2007)
8. Janeway, C.A., Walport, M., Travers, P.: Immunobiology: The Immune System in Health and Disease, 5th edn. Garland Publishing (2001)

A Dynamic Energy-Aware Algorithm for Self-Optimizing Wireless Sensor Networks

Syed I. Nayer and Hesham H. Ali

Department of Computer Science
College of Information Science and Technology
University of Nebraska at Omaha
Omaha, Ne 68182, USA
hali@mail.unomaha.edu

Abstract. Wireless sensor networks are distributed ad-hoc networks that consist of small sensor devices collecting and transmitting data. In such devices, optimizing power consumption due to sensing and routing data is an open area for researchers. In earlier works, approaches based on the Gur game, ant algorithms and evolutionary algorithms were introduced. These approaches have been employed to control the number of active sensors in wireless networks required to guarantee high performance parameters while taking energy conservation into consideration. These studies though ignored the coverage redundancy, which is a key issue in sensor networks. In this paper, we use the mathematical paradigm of the Gur Game in order to achieve the optimal assignment of active sensors while maximizing the number of regions covered by sensor nodes. We use a dynamic clustering algorithm that employs the concept of connected dominating sets. The proposed algorithm addresses this problem by playing Gur Game among the cluster nodes. We also further develop the earlier developed ants algorithm and genetic algorithm to take into consideration node addition and deletion. A simulation study was used to test the proposed algorithms under different network scenarios.

Keywords: Wireless sensor networks; self optimizing; Gur game; Quality of Service (QoS); ants algorithms, genetic algorithms; connected dominating set.

1 Introduction

With advancements in sensor technology and drop in price, it is becoming more common to use large numbers of unattended small sensors in order to monitor remote sites. These sensors are low powered devices with very little computational capability. Wireless sensor networks allow sensors to work together in order to cover wide area in more comprehensive manner. Limited energy supply, low bandwidth, throughput optimization, coverage are some of the major issues in sensor networks. Due to unattended nature of sensors in sensor networks energy supply is limited and it is necessary to conserve energy. Since large number of sensors are used now a days, in order to cover the field comprehensively and accurately, sensors are likely to send redundant data which will be waste of energy and bandwidth.

To guarantee high degree of reliability, sensors may cover overlapping areas to provide better performance parameters via redundancy. In many cases, it is desirable to provide a balance between the reliability of networks as well as reduction in power consumption. Usually, sensors are placed in sensor field randomly it is more likely that one or more sensors may be covering same region and they may end up in gathering similar data which results in too much redundancy of data sent back to base station which is good up to some extent but will result in sensors dying rapidly and will require frequent re-deployment which is very difficult in many of the scenarios in which sensors are used. In order to avoid this it is required that sensors may act together and co-operate in a way that they cover maximum area with minimum number of nodes active at a time.

Numerous techniques have been proposed for nodes placement or topology of sensor networks and mostly primary goal of all these techniques is energy conservation. Every protocol has its own pros and cons. Some are specialized energy aware routing protocols which consumes lot less energy, then others, while routing data [2,4,5]. In addition, incorporating basic fault tolerance measures have been investigated in several studies [1,3].

The greedy approach has been used by allowing the nodes to communicate with base station at a regular interval of time and Packet count, received at regular interval, is used as quality of service parameter in mathematical paradigm of the Gur Game to dynamically activate optimum number of sensors. The result is a robust sensor network that allows the base station to dynamically adjust the resolution of network on basis of feedback it receives from the sensors in network [6].

In this paper, we used the concept of coverage regions in sensor networks and Gur Game. This idea will allow the networks to dynamically self regulate the sensor nodes in order to cover maximum number of regions with minimum number of nodes. Additionally, since sensor nodes usually have a limited life span, it is not always reasonable to assume that the sensor network has a fixed number of nodes. Hence, self optimizing algorithms need to take into consideration additions and deletions of network nodes. Based on algorithms we developed earlier using genetic algorithms and Ants algorithms [7], we further develop these approaches to handle such variability in sensor networks. Simulation studies show that the developed methods deal with nodes additions and deletion effectively.

2 Self-Optimizing Using Gur Game

The algorithm uses the mathematical model of Gur Game in order to maximize the coverage with minimum number of active sensor nodes. This section covers details about Gur Game. Gur game can be described as a function which is random but working in a greedy fashion to achieve global optimization. The key in Gur game is reward function which is responsible to measure the performance of whole system. Higher performance reward functions moves towards value 1. After each iteration, the value of nodes changes probabilistically according to reward function. Regardless of how a node votes, it is rewarded randomly with probability r or penalized with probability *1-r*. Suppose at any instance the number of nodes saying yes is *X* then the reward probability will be *r(X)*. In our case, we have taken x as the ratio between the number of active nodes and the regions covered by the active nodes and it is required to be *0.40,* i.e. the maximum value of reward function will be at $X = 0.40$. The reason behind using the *0.4* value is to target an acceptable level of redundancy so every region is covered by *2.4* nodes. Higher levels of redundancy would be appropriate for critical applications. Nodes in the game do not forecast the behavior of other nodes; instead they use trial and error method to produce the best result. It is clear from the Figure 1 that losing behavior will make the nodes shift the chain towards middle while a winning behavior will shift it outward. The nodes will be active when it is in positive numbered state and will be in-active when it is in negative numbered state.

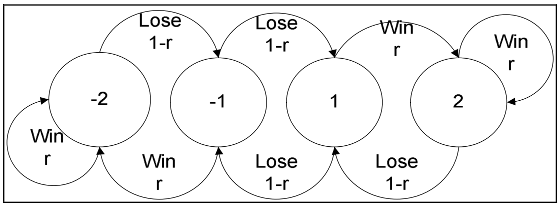


Fig. 1. Gur Memory

Imagine a scenario in which it is required to gather data continuously at a particular interval of time from every single region of the field where several sensors has been deployed. There may be one or more then one sensor covering each region of the field. If all the sensors are active all the time then it will consume un-necessary amount of energy because of detecting and transmitting redundant data along with producing high interference which will cause the nodes to die very quickly resulting in shortening the life of sensors.

Let us assume collection of *n* number of sensor nodes which are deployed in sensor field. The field is divided into 'm' number of small regions which is covered by one or more sensors. Every sensor has a small memory reserved for saving finite state which is described in previous section. We can plot a bi-partite graph between sensor nodes and region covered by each node. At a regular interval of time sensors will send data along with the information about the region it covered. On the basis of numbers of sensor active and number of regions covered by these active sensors feedback will be sent randomly by the base station in form of reward or penalty.

As the number of sensors and regions increased it takes more time to converge to a point where the ratio between number of regions covered and active nodes is 0.40. In order to avoid this delay we created clusters of small number of regions and then each branch in cluster has its own sink. This sink is assumed to act as router and theses routers are assume to be more powerful and are capable of sending feedback function to the nodes in its branch of cluster.

Sensor nodes will send status and region it covers to base station, regardless of any stimuli it sense, at a regular interval of time. Each time base station receive any stimuli from sensor nodes in field it may or may not necessarily receive messages from optimum number of nodes per region which was desired to cover every region in sensor field. When ever a base station will receive the number of nodes active at a time and a region covered by active sensors nodes it will generate reward function

$r(X_t)$ where *t* is the ratio between the regions covered and number of active sensor nodes at any given time. Finally base station will broadcast reward probability *$r(X_t)$* and sensors will reward itself with probability *r* or penalized with probability *1-r*. The sensor S_i will be active when it is in positive state and sleeping or in-active when in negative state. It will not be taking part in routing data when it is in sleep mode or in negative state. In sleep mode or negative state sensors will be listening to the network but will not be actively sending data which will save energy.

3 Distributed Ants Algorithm and Genetic Algorithms

In addition to the Gur Game based algorithm, we also used distributed Ants algorithms and genetic algorithms to optimize the number of active nodes in the networks. In the ants algorithm, a given number of ants or a distributed agents visit the nodes of the sensor nodes, modeled as vertices in the underlying graph and changes the color of each visited node according to a local criterion. At a given iteration each ant moves from the current node to the adjacent node with the maximum number of violations, and replaces the old color of the node with a new color that minimizes this number. For a given node *i*, the number of violations is computed as the number of adjacent nodes to *i* with the same color than *i*. This action is randomly repeated for each ant the ant moves to the worst adjacent node with a certain probability p_n (otherwise it moves to any other adjacent node randomly chosen), and assigns the best possible color with a probability p_c (otherwise any color is assigned at random). Probabilistic nature of the algorithm allows the ants to avoid local minima and reach the absolute minimum. This process, which is carried out simultaneously by the set of ants, is repeated until the optimal solution is found or the algorithm converges.

The algorithm starts by generating an arbitrary random coloring of the underlying graph of the network and each node identifies itself as active or inactive. Once it has been initialized, the process begins to evolve by assigning to every node a value calculated as the difference between the number of adjacent vertices with the same status and the number of adjacent nodes with different status. Next, every ant moves to the adjacent vertex that has maximum value, changes the status of this node and recalculates the values for it and its adjacent vertices, all that with a certain probability to avoid local minima. Once each ant has moved, the process is repeated until the algorithm converges to a near optimal solution. This method, our multi-agent system, used up to 10 agents. The criteria for all our algorithm is that ratio between the number or regions covered by active sensor and number of active sensor. For all our simulations we used 0.4 as convergence number such that each area is covered by 2.5 nodes with the optimization criteria of maximizing the number of covered regions.

The third self organizing algorithm we use is based on Constructive Genetic Algorithms (CGA). CGA allows working on schemata instead of actual structures which helps us in simulating genetic algorithm on a dynamic population. CGA is an alternative to genetic algorithm which works on schemata and populations generating good schemata along with good structures. P-median is a classical graph theory problem in which the main goal is to select P nodes in a graph so as to minimize the sum of distance between each node to its nearest median node. This problem is NP hard problem. Suppose a graph *$G=(V, E)$* composed of n vertices and distance weight matrix u_{ij}

for all *i,j* ε *V*. we will try to find *p* medians where *p* will be equal to (Area of the region / average coverage by each node)*2. Theoretically this will result in each region covered by 2 sensor nodes. Schemata can be defined like an array with *1-0-X* where *1* denotes a node which is a median and *0* its not and *X* denotes a node which is not present at moment [7].

4 Simulation Results

We now present the simulation results for our algorithm. In our experiments, we assume the memory size is 4 and sensors will pick regions randomly in the beginning of simulation and will not change its region during simulation. The number of regions covered by sensor nodes in all our simulations is kept 35. We performed the simulation with 100 sensor nodes spread randomly in sensor fields. Each region is covered by random number of sensors and sensors will not die during the simulation. In the beginning each sensor will pick a random state and random region. We assume that base station will send feedback 1 when the ratio for regions covered to active sensor nodes is 0.40 i.e. QoS will have a peak when 2.5 nodes will cover each region. The reward function used for the simulation is f, as calculated below. The parameter α is an integer defined as a function of the number of regions covered and is used to enhance the factor regions covered. Since the ratio between 20 regions to 60 active nodes will be the same as 10 regions to 30 nodes. But the second case with 20 regions covered is more desirable then the first one in which only 10 regions were covered. Following table shows how the number of regions covered change when the value of α changes. In each iteration, the algorithms calculates the number of active nodes and number of regions covered from the packets it will received from the sensor nodes to base station. After receiving number of packets it will use the feedback function mentioned above in order to reward nodes with probability *r* or penalized the nodes with probability *1-r*.

$f = 0.2 + 0.8\, e^{v}$

$v = (-0.002) * (X_t * 100 - 40)^2$

X_t = (#number of regions covered by active sensors)$^{\alpha}$ / (# of active sensors),

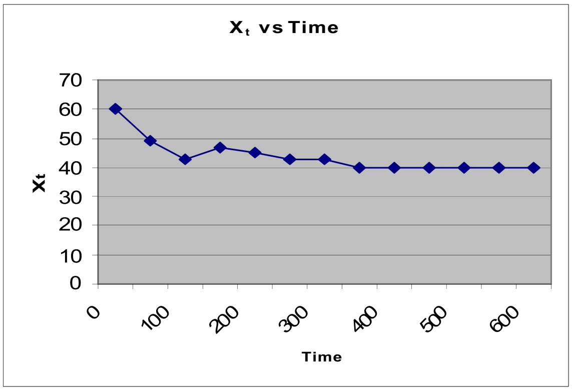


Fig. 2. Value of X_t vs. time with 100 Nodes

Figure 2 shows that it takes almost 400 iterations to reach the value of $X_t=0.40$. We ran the simulation several times and it converges to peak value of 0.40 between 340 and 550 iterations. It is clear that the value of X_t oscillates in the beginning but converges to optimal. Nodes die after being active for more then 50 seconds and will remain dead for a period of 50 to 90 seconds randomly before charging itself back to life. Here it will reach peak several times but due to death, random re-charging and random birth of new nodes it will be fluctuating. The graph in Figure 3 shows the simulation results in which nodes are added and removed randomly.

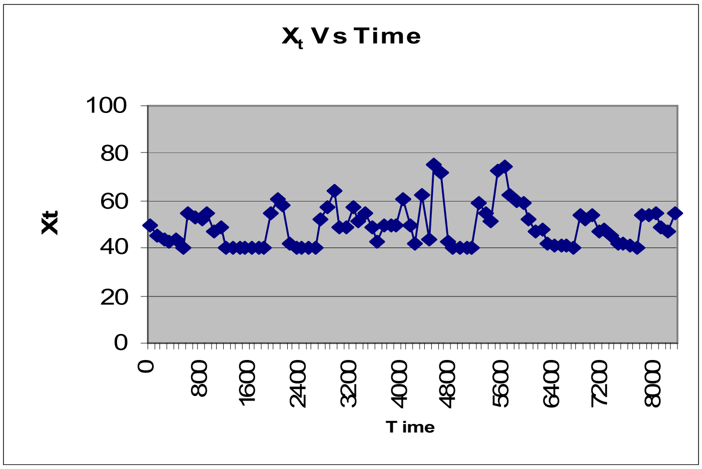


Fig. 3. Simulation Results for Dynamic Networks

The results show that the network will try to converge to a point in which on an average 2.5 nodes will be covering each region hence we maximize the number of regions covered with minimum number of active sensors hence conserve energy. Based on the simulation results, it is clear that when we double the number of nodes even with same number of regions the time to converge increase. We can infer that an increase in number of nodes will increase the convergence time. For large networks, we use clustering to reduce the convergence time with the help of connected dominating set between the regions. In our simulation, we used tree growing algorithm to create dominating sets between the regions and make sure the dominating nodes remain active and act as cluster head to exchange messages. The connected dominating set formation phase includes exchange of information between the nodes of the regions. After the dominating set is formed every dominating node's region will have information about the other dominating region. After formation of dominating set, every region will start playing Gur game with its dominator region.

Figure 4 shows that increasing the number of nodes will not effect the time it takes to converge to maximum value. This is due to the formation of sets which have small number of regions then the original case. The time depends on how connected dominating sets are formed. It desirable that the connected dominating set should not be the minimum. The reason is that we don't want our set to be large as it will be much easier to converge in case of small sets of region. This algorithms works on principal that local optimization will result in global optimization.

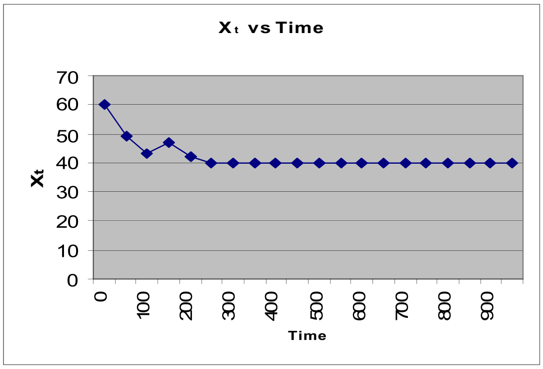


Fig. 4. Value of X_t vs. time with 100 Nodes

5 Conclusion

In this paper, we introduce self optimizing algorithms to regulate the process of activating sensors while maximizing the number regions covered by sensor nodes. We proposed a dynamic clustering algorithm that employs the concept of connected dominating sets. The proposed algorithm then resolves the problem by playing Gur Game among the cluster nodes. We also improved on earlier ants algorithm and genetic algorithm to take into consideration the dynamic nature of wireless sensor networks. To achieve the much desired redundancy in covering the monitored regions, the algorithm attempts to find the best assignment of active sensors while maximizing the number of regions covered by sensor nodes.

References

1. Cerpa, A., Estrin, D.: AASCENT: Adaptive Self-Configuring Sensor Networks Topologies. In: Proceedings of INFOCOM 2002 (2002)
2. Intanagonwiwat, C., Govindan, R., Estrin, D.: Directed diffusion: A scalable and robust communication paradigm for sensor networks. In: Proceedings of MobiCOM 2000 (August 2000)
3. Krishnamachari, B., Mourtada, Y., Wicker, S.: The Energy-Robustness Tradeoff for Routing in Wireless Sensor Networks. In: IEEE, Proceedings of IEEE International Conference on Communications (ICC 2003) (May 2003)
4. Li, Q., Aslam, J., Rus, D.: Online power-aware routing in wireless ad-hoc networks. In: Proceedings of MOBICOM 2001 (2001)
5. Dulman, S., Nieberg, T., Havinga, P., Hartel, P.: Multipath Routing for Data Dissemination in Energy Efficient Sensor Networks. TR-CTIT-02-20 (July 2002)
6. Deb, B., Bhatangar, S., Nath, B.: A Topology Discovery Algorithm for Sensor Networks with Applications to NW Management. In: IEEE CAS Workshop 2002 (2002)
7. Nayer, I., Ali, H.: On Employing Distributed Algorithms and Evolutionary Algorithms in Managing WSN. In: Proc. of the Intl Workshop on Theoretical and Algorithmic Aspects of Wireless Ad Hoc, Sensor, and P2P NWs, Chicago (2004)

SQUIRREL: Self-Organizing Qos-roUting for IntRa-flow Contention in Ad-Hoc wiRELess Networks

Abdelouahid Derhab

Department of Computer Engineering, CERIST, Rue des 3 frères Aissou, Ben-Aknoun, BP 143 Algiers, 16030 Algeria

Abstract. In this paper, we use the self-organizing paradigms presented in [1] to design a new QoS-routing intra-flow contention-based protocol for wireless ad-hoc networks, called SQUIRREL. The admission control component of SQUIRREL, called: Scalable and Accurate Admission control (SAICAC), has two variants: SAICAC-Power and SAICAC-CS. SAICAC-Power estimates channel bandwidth availability through high power transmissions and SAICAC-CS through passive monitoring of the channel. Contrary to the existing intra-flow contention-based protocols, SQUIRREL can ensure all the properties of a self-organizing system and can achieve the best results in terms of of message overhead, delay, and scalability.

Keywords: Ad-hoc network, self-organization, intra-flow contention, admission control, scalability.

1 Introduction

In wireless ad-hoc networks, the wireless channel is shared between all nodes within a certain range, called *Carrier-Sensing Range (CSR)*. This range is typically much larger than the transmission range. Nodes that are within carrier sensing range detect a transmission but may not be able to decode the packet. Nodes within the sender's transmission range are considered its neighbors, and those which are within the *CSR* of the sender are called its *Carrier-Sensing Neighbors (CSN)*. In IEEE 802.11 MAC protocol, a node's transmission consumes bandwidth at all nodes within its vicinity (i.e., carrier-sensing range). Let us consider a flow f with a bandwidth requirement, B_{req}; multiple nodes on the route may locate within the carrier-sensing range of a given node S, and they all contend for bandwidth. The number of these nodes is called the *contention count* of the route and is denoted as CC. To make admission control decisions over a multi-hop path, the bandwidth availability, B_{av}, must be larger than the effective bandwidth consumed by the flow at node S, $(CC \times B_{req})$.

Using the four self-organizing design paradigms proposed in [1], which will be presented in Section 3, we propose a new QoS-routing intra-flow contention-based protocol for wireless ad-hoc networks, called SQUIRREL. SQUIRREL consists of the following components: (1) admission control, called Scalable and Accurate Admission control (SAICAC), (2) bandwidth management, (3) flow restoration, and (4) congestion control[1]. SAICAC has two variants: SAICAC-Power and SAICAC-CS. SAICAC is compared against intra-flow admission control methods that can accurately estimate CC, which are: CACP and MAC-MAN [3]. From this comparison study, we show that using self-organizing design paradigms is translated into high scalability and optimal performance in terms of message and delay.

The rest of the paper is organized as follows: Section 2 presents SQUIRREL. In Section 3, we analyze the performance of SAICAC, CACP [7], and MACMAN [3]. Section 4 concludes the paper.

2 Self-Organizing Qos-roUting for IntRa-flow Contention in Ad-Hoc Wireless Networks (SQUIRREL)

Our admission control, called SAICAC, is integrated with a route discovery procedure of a reactive routing protocol similar to AODV [4]. In SAICAC, each node i maintains for each flow f circulating in its carrier sensing range: (1) the contention count $CC_{i,f}$, and (2) the list of the CSNs which transmit the flow f. If the available bandwidth at a given node is smaller than the bandwidth requirement of the flow, the admission control fails. The bandwidth reservation is only carried out during the route reply phase.

When a source node wants to send a data flow f to a destination node, it broadcasts a RREQ packet to its neighbors. The RREQ contains the bandwidth requirement $B_{req,f}$. When the destination node receives the RREQ packet, it sends a RREP(target), such as target is the next hop toward the source. If multiple requests arrive at the destination, the destination only sends the RREP along one route. The other routes are cached for a short period of time as backup in case the first RREP does not reach the source due to link breakage or admission failure. In the reply phase, SAICAC can use one of the two variants: SAICAC-Power or SAICAC-CS. In Figure 1, node A wants to introduce a new traffic flow $f1 = [A, B, C, D, E]$ requiring $B_{req,f1} = \frac{B_c}{7}$ bits/s, such that: B_c denotes the channel capacity. In SAICAC-Power, intermediate nodes as well as the source node send reply packets using a larger transmission power level than the transmission power level used for normal data transmission. In figure 1(a), the first intermediate node toward the source (i.e, node D), sets $CC_{D,f1}$ to 1 and checks if $B_{av} < B_{req,f1}$. If so, it sends RREP(target) using a high power transmission. After receiving RREP(target), each node i executes the following actions, (1) it increases $CC_{i,f1}$ by 2 if $i = target$ and by 1 otherwise, (2) checks if $(B_{av} < CC_{i,f1} \times B_{req,f1})$ holds true, (3) If node i belongs to the route, it sets target to the next hop toward the source if $i \neq A$ and $\varnothing$ otherwise, and sends RREP(target) using a high power transmission. The respective contention counts of the flows are shown adjacent to each node. (See Figures 1(b), 1(c), 1(d)).

[1] Due to page limitation, only SAICAC is presented in this paper.

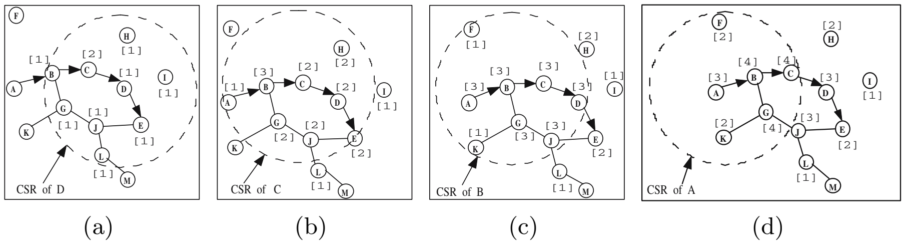


Fig. 1. Admission control acceptance in SAICAC-Power

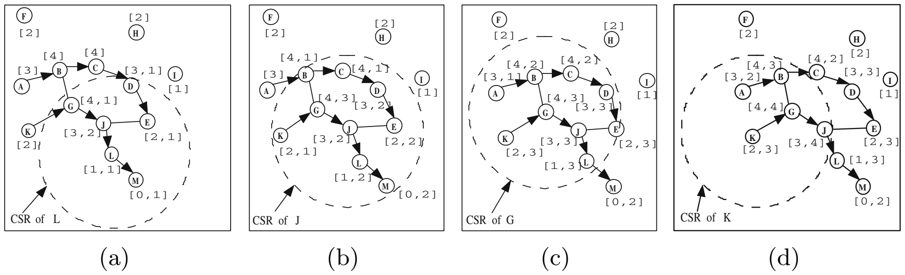


Fig. 2. Admission control rejection in SAICAC-Power

In Figure 2, node K wants to introduce a new traffic flow $f2 = [K, G, J, L, M]$ requiring $B_{req,f2} = \frac{B_c}{7}$ bits/s. After the route request phase, node M broadcasts a reply packet toward node K (See Figures 2(a), 2(b), 2(c)). Upon receiving the reply packet, the source node K in Figure 2(c) finds that the total reserved bandwidth is: $(2B_{req,f1} + 3B_{req,f2}) = (\frac{5}{7})B_c$. So, it broadcasts a reply packet using a high power packet transmission. When node G receives such a message (See Figure 2(d)), it finds that $(4B_{req,f1} + 4B_{req,f2}) = (\frac{8}{7})B_c$. It concludes that flow $f2$ will hinder the existing flow $f1$. Then, it sends a *Reject* packet to K, which will send an *Error* packet to M and K to release the allocated bandwidth.

In SAICAC-CS, a passive approach is used to obtain c-neighborhood available bandwidth. The node that receives the route reply directly estimates its c-neighborhood available bandwidth using the equation presented in [7] and compares it with the bandwidth consumption of the flow to make admission decisions.

3 Comparison of Intra-flow Contention-Based Admission Control Methods

In this section, we compare the admission control of SQUIRREL, i.e., SAICAC, against methods that can accurately estimate the contention count, which are: CACP and MACMAN. The comparison is studied under the following metrics: the number and size of control packets, the additional delay incurred in making the flow admission control decision, the design paradigms applied by each

Table 1. Comparison of intra-flow contention-based admission control methods

Metrics	CACP-Power	CACP-CS	MACMAN	SAICAC-Power	SAICAC-CS
Paradigms	None	None	2,3, 4	1,2,3,4	1,2,3,4
Routing protocol	DSR	DSR	DSR	AODV	AODV
Localized	No	No	No	Yes	Yes
Adaptability	No	No	Yes	Yes	Yes
Robustness	Yes	Yes	Yes	Yes	Yes
RREQ sent	N	N	N	N	N
RRRP sent	M	M	M	M(High Power)	M
Other packet	M(High Power)	0	N(High Power)	0	0
RREQ size	$Q + M \times I$	$Q + M \times I$	$Q + M \times I$	Q	Q
RREP size	$P + M \times I$	$P + M \times I$	$P + M \times I$	P	P
Other packet size	$M \times I$	0	I	0	0
Extra delay	$M \times D_1$	0	0	0	0
$X_{ov}(N)$	$O(N^2 \log_2 N)$	$O(N^2 \log_2 N)$	$O(N^2 \log_2 N)$	$O(N^{1.5})$	$O(N^{1.5})$
Scalability	No	No	No	Yes	Yes

method, routing protocol integrated with the method, the properties ensured by the method, such as: localized interaction, adaptability, robustness, and scalability. In the following, we show how our QoS-routing protocol attempts to satisfy the design paradigms compared to other protocols and how these paradigms can improve the protocol performances and ensure the self-organization properties. The results of comparison is summarized in Table 1.

1. *Design local behavior rules that should be able to achieve global properties (Paradigm 1):* In our protocol, each node uses only the information obtained from CSNs to calculate the contention count $CC_{i,f}$. CACP and MACMAN both do not apply paradigm 1 since to make an admission control decision they need to know all nodes along the flow, which leads to high cost in terms of message overhead, and hence they are not localized.
2. *Do not aim for perfect coordination, exploit implicit coordination (Paradigm 2):* In CACP-Power, when a node receives a route reply packet, it broadcasts, using high power transmission, an admission request message, which carries the full route of the flow, to its c-neighborhood. Upon reception of such a message, if a node finds that the bandwidth consumption of the flow is larger than the node's local available bandwidth, it sends an admission rejection message back to the initiator(i.e., node sending the admission request message), and the admission control fails. Otherwise, if the initiator does not receive any admission rejection message after a given time period, it admits the flow and sends the route reply packet toward the source node. Although, the bandwidth is allocated in a coordinated manner, CACP-Power incurs more cost in terms of message overhead and delay. In SAICAC-Power, a node, which receives a route reply packet, compares, using the known CC value, the bandwidth consumption of the flow at its location with its local available bandwidth. If the former is less than the latter, it admits the flows and sends a route reply packet using high power transmission to the next hop toward the source node. Here, SAICAC-Power does not wait for its cs-neighbor's approval to accept the flow. This choice increases significantly the performance of the protocol in terms of message overhead and delay.

However, such a design choice may hinder the existing flows circulating in the node's CSR and may lead to conflicts. Such conflicts can be easily contained because each node that receives or overhears the route reply packet, updates its CC and checks if it can accept the new flow. If not, it sends a reject message back to the initiator. The initiator in turn sends an error message toward the source node to stop the flow generation. As MACMAN CACP-CS and SAICAC-CS, extend their bandwidth measurement range, they do not need coordination between nodes to make an admission control decision.

3. *Minimize long-lived state information (Paradigm 3):* It is known that source nodes in DSR caches multiple routes at the time of route discovery. However, over time the routes are likely to become stale and be maintained by the source nodes for a long time period. To deal with issue, MACMAN continuously monitors each alternative route in the cache. However, this comes at the cost of additional message overhead. On the other hand, each node in SAICAC needs only to maintains information about its c-neighborhood, and which can be updated as topology changes.
4. *Design protocols that should adapt to changes (Paradigm 4)*: Except CACP that does not propose any strategy to handle mobility and loss of QoS guarantees, SAICAC and MACMAN can adapt to topology changes.

In [5], the scalability of a protocol is calculated using the following definitions:

– The *minimum traffic load* of a network under parameters λ_1, λ_2, etc., denoted by $T_r(\lambda_1, \lambda_2, \ldots)$, is the minimum amount of bandwidth required to forward packets over the shortest paths available, assuming all the nodes have a priori full topology information. The network scalability factor Ψ_{λ_i} with respect to parameter λ_i is defined to be $\Psi_{\lambda_i} = \lim_{\lambda_i \to \infty} \frac{\log T_r(\lambda_1, \lambda_2, \ldots)}{\log \lambda_i}$
– If $X_{ov}(\lambda_1, \lambda_2, \ldots)$ is the total overhead due to the admission control method X under parameters λ_1, λ_2, etc., then the protocol scalability factor $\rho^X_{\lambda_i}$ of protocol X with respect to λ_i is defined to be $\rho^X_{\lambda_i} = \lim_{\lambda_i \to \infty} \frac{\log X_{ov}(\lambda_1, \lambda_2, \ldots)}{\log \lambda_i}$
– A protocol X is said to be scalable with respect to parameter λ_i, if and only if $\rho^X_{\lambda_i} \leq \Psi_{\lambda_i}$.

We assume that N nodes are randomly distributed in a region of area A. As N increases, the average node density remains constant. If each node generates λ_t bits per second and the average path length increases as $\sqrt{N}$ hops, then $T_r(\lambda_t, N) = \lambda_t N^{1.5}$. Thus, $\Psi_{\lambda_t} = 1$, and $\Psi_N = 1.5$. In [2], the authors study the capacity of a fixed ad-hoc network in which the nodes' locations are fixed but randomly distributed. They prove that, as the number of nodes N per unit area increases, the throughput per session λ_t can at best be $\frac{1}{\sqrt{N}}$. Let λ_s be the average number of new sessions generated by a node per second, it is obvious that $\lambda_s < \lambda_t$, so $\lambda_s < \frac{1}{\sqrt{N}}$. In Table 1, we use the notations given in [6], which are as follows: M denotes the number of nodes on the path. Q, P, and I denote the size of RREQ, RREP, and node address respectively. D_1 is a constant used in CACP-Power. We assume that I is proportional to $\log_2 N$. In CACP and

MACMAN, RREQ and RREP carry the IDs of the nodes on the route. Thus, the control information piggybacked onto the packets are of the size of $M \times I$. Our method, on the other hand, do not piggyback any additional control information onto RREQ or RREP. The forwarding of the RREP in CACP-Power is delayed at each intermediate node by D_1 time units. So, the extra delay incurred to make multi-hop admission control decision is $M \times D_1$. CACP-CS, MACMAN and SAICAC all of which require no additional delay over that incurred by the route discovery procedure. As sessions are generated at a rate λ_s per second per node, The total overhead cost, $SAICAC_{ov}$ is computed as follows: $SAICAC_{ov} = \lambda_s N(N \times Q + M \times P) < O(N^{1.5})$. As $\rho_N^{SAICAC} \leq 1.5$, SAICAC is considered to be scalable. The total overhead complexities of the other admission control methods can be computed in the same way as SAICAC.

4 Conclusion

In this paper, we have proposed SQUIRREL, a new QoS-routing intra-flow contention-based protocol for wireless ad-hoc networks. By applying the design paradigms presented in [1], SQUIRREL can ensure all the properties of a self-organizing system contrary to the existing intra-flow contention-based protocols. To make a multi-hop admission control decision, SAICAC does not incur high message overhead or extra delay over that incurred by a regular route discovery. SAICAC has an asymptotic cost of $O(N^{1.5})$, and thus scales well with the increase in the number of nodes in the network.

References

1. Prehoferand, C., Bettstetter, C.: Self-organization in communication networks: principles and design paradigms. IEEE Communications Magazine 43(7), 78–85 (2005)
2. Li, J., Blake, C., De Couto, D.S.J., Lee, H.I., Morris, R.: Capacity of ad hoc wireless networks. In: Proceedings of the 7th annual international conference on Mobile computing and networking (MobiCom 2001) (2001)
3. Lindgren, A., Belding-Royer, E.M.: Multi-path admission control for mobile ad hoc networks. In: 2nd Annual International Conference on Mobile and Ubiquitous Systems (MobiQuitous 2005), pp. 407–417 (2005)
4. Perkins, C.E., Royer, E.M.: Ad hoc on demand distance vector (aodv) algorithm. In: Systems and Applications (WMCSA 1999), pp. 90–100 (1999)
5. Santivanez, C., McDonald, B., Stavrakakis, I., Ramanathan, R.: On the scalability of ad hoc routing protocols. In: IEEE INFOCOM (June 2002)
6. Sanzgiri, K., Chakeres, I.D., Belding-Royer, E.M.: Pre-reply probe and route request tail: approaches for calculation of intra-flow contention in multihop wireless networks. Mobile Networks and Applications 11(1), 21–35 (2006)
7. Yang, Y., Kravets, R.: Contention-aware admission control for ad hoc networks. IEEE Transactions on Mobile Computing 4(4), 363–377 (2005)

Use Cases, Requirements and Assessment Criteria for Future Self-Organising Radio Access Networks[*]

Mehdi Amirijoo[1], Remco Litjens[2], Kathleen Spaey[3], Martin Döttling[4], Thomas Jansen[5], Neil Scully[6], and Ulrich Türke[7]

[1] Ericsson, Linköping, Sweden
[2] TNO ICT, Delft, The Netherlands
[3] IBBT, Antwerp, Belgium
[4] Nokia Siemens Networks, Munich, Germany
[5] TU Braunschweig, Braunschweig, Germany
[6] Vodafone, Newbury, United Kingdom
[7] Atesio, Berlin, Germany
mehdi.amirijoo@ericsson.com, remco.litjens@tno.nl, kathleen.spaey@ua.ac.be, martin.doettling@nsn.com, jansen@ifn.ing.tu-bs.de, neil.scully@vodafone.com, tuerke@atesio.de

Abstract. Self-organisation (self-optimisation, self-configuration, and self-healing) methods are a promising concept to automate wireless access network planning, deployment and optimisation. This paper contains a mind setting exercise. First the mechanisms for which self-organisation is anticipated to be effective and feasible are identified. Then technical and non-technical requirements that need to be taken into account for the successful development of self-organisation functionalities are discussed. Furthermore, a set of metrics and appropriate reference cases (benchmarks) are presented, which allow to do on one hand a quantitative comparison of the different algorithms developed for a given use case, and on the other hand to evaluate the gains from self-organisation by comparing self-organisation solutions with the case of manual network operations.

1 Introduction

As recognised by the standardisation body 3rd Generation Partnership Project (3GPP) [1] and the operators' lobby Next Generation Mobile Networks (NGMN) [2], future wireless access networks, such as the 3GPP Long Term Evolution (LTE) radio access, will exhibit a significant degree of self-organisation. The principal objective of introducing Self-Organising Network (SON) functionalities in wireless access networks is to reduce the costs associated with network operations, while enhancing network performance. By improving the effectiveness of manual effort in network operational tasks, significant Operational Expenditure (OPEX) reductions are expected, while

[*] This paper was written in the context of the SOCRATES project, which is funded by the European Union within the 7th Framework Program and will run from 2008-2010. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the SOCRATES project.

because of the better adaptation to changing network characteristics and failures, it is anticipated that SON features will enhance the global network capacity, coverage and service quality experienced by the users.

Before starting on detailed technical work and the development of SON methods for future wireless access networks, it is essential to first perform a mind setting exercise to identify the mechanisms for which self-organisation is anticipated to be effective, to obtain a clear view on the (non-)technical requirements put on SON solutions, and to define criteria that can be used to evaluate the feasibility and performance of the developed SON methods. This is the topic of the current paper. All presented material is developed within the context of the European FP7 research project SOCRATES (Self-Optimisation and self-ConfiguRATion in wirelEss networkS) [3,4]. A main objective of SOCRATES is providing dedicated SON solutions, i.e., methods and algorithms, as a step towards the implementation of SON functionality into future wireless access networks, where 3GPP Evolved UTRAN (E-UTRAN), which is the 3GPP LTE radio access, has been selected as the technology of focus.

2 Use Cases and Requirements for Self-Organising Access Networks

Use cases are an established means of describing what a solution to a particular problem shall achieve. Within the SOCRATES project, over twenty-five use cases that focus on self-organisation in 3GPP E-UTRAN have been identified. These include for example the self-configuration use cases ‘intelligently selecting site locations’ and ‘automatic generation of default parameters for network element insertion’, the self-optimisation use cases ‘packet scheduling optimisation’, ‘interference coordination’, ‘admission control parameter optimisation’ and ‘load balancing’, and the self-healing use case ‘cell outage management’. See [5] for an extensive description of these and the other use cases.

The classification of the use cases in the three categories, i.e., self-configuration, self-optimisation and self-healing, is in line with the framework SOCRATES envisions regarding the use of self-organisation methods in future radio networks [3,4]. Newly added NEs (Network Elements) like e.g., base stations (eNodeBs), *self-configure* in a ‘plug-and-play’ fashion, while existing NEs continuously *self-optimise* their operational algorithms and parameters in response to changes in network, traffic and environmental conditions. The adaptations are performed in order to provide the targeted service availability and quality as efficiently as possible. In the event of a cell or site failure, *self-healing* methods are triggered to alleviate the performance effects due to the resulting coverage/capacity gap by appropriately adjusting radio parameters in surrounding sites. In general, human involvement shall only be triggered when absolutely necessary, e.g., when manual repairs are needed.

For the successful development of SON functionalities, various *technical requirements* must be considered. In [6], we present several technical requirement categories: performance and complexity, stability, robustness, timing, interaction, architecture and scalability, and required inputs (performance counters and measurements). As the same principles apply to many use cases, while the details vary, these requirement categories are discussed in general, but also in detail for every identified use case.

Since solutions that are good from a purely technical point of view may not necessarily meet *business requirements*, it is also important to consider these requirements. They comprise cost efficiency requirements (e.g., SON solutions should reduce OPEX and CAPEX (Capital Expenditure)) and LTE deployment requirements (e.g., the roll-out of LTE networks should be sped up, new services should easily be deployed, the end user should benefit). For more details, we again refer to [6].

3 Assessment Criteria for Self-organising Access Networks

In future SOCRATES will develop self-organisation methods and algorithms. An adequate assessment of the benefits from developed self-organisation methods requires a set of well-defined metrics and appropriate reference cases (benchmarks). With regard to the reference cases, on one hand different self-organisation algorithms may be compared with one another, while on the other hand an appropriate 'manual reference case' needs to be defined to allow evaluation of the gains from self-organisation with respect to contemporary and manually operated networks.

3.1 Metrics

Key metrics that are relevant in the assessment of self-organisation methods are [7]:

- **Performance metrics:** These metrics express the service level experience from the user perspective and include *grade of service (GoS)* metrics, e.g., call blocking ratio, call dropping ratio, and *quality of service (QoS)* metrics, e.g., packet delay statistics, packet loss ratio, throughput statistics, mean opinion score, fairness.

- **Coverage metrics:** Different coverage metrics exist, e.g., the *service coverage*, i.e., the fraction of area where a given service can be supported with adequate service quality and the *data rate coverage*, i.e., the fraction of area where a user can experience at least some specified data rate.

- **Capacity metrics:** Cell (or network) capacity is not unambiguously defined and different sensible perspectives are applied in the literature, such as maximum number of concurrent calls, maximum supportable traffic load, and spectrum efficiency.

- **CAPEX:** In general, CAPEX encompasses the investments needed in order to create future benefits, including e.g., radio and core network elements. An approach we propose to estimate CAPEX is to determine the number of network elements that is needed to cover a certain service area with pre-specified GoS and QoS requirements, and multiply this with the corresponding costs. Given a certain service demand per km^2, the required number of network elements can be determined by maximising the cell radii such that traffic demand per cell and cell capacity are sufficiently well balanced to meet the GoS and QoS requirements.

An additional aspect to consider is that the introduction of self-organisation features themselves may lead to an increase in equipment cost (per unit). This additional CAPEX is hard to estimate, but depends on the nature and complexity of the self-organisation algorithm, the transmission bandwidth requirements that may be higher due to increased signalling overhead, and additional costs related to needed site equipment, e.g., electrical antenna tilt and additional circuitry for enabling power savings.

• **OPEX:** The costs associated with the network operations and, in particular, the reduction of these costs due to the introduction of self-organisation functionalities, are rather difficult to assess. Noting that actual OPEX reductions depend on the degree of self-organisation that is deployed, in an extreme implementation, all OPEX related to manual adjustment of a given parameter set (associated with a use case) is removed. In order to develop an approach to assess the OPEX level, we distinguish between three main phases in effectuating parameter adjustments, i.e., gathering input data, e.g., via performance counters, drive tests or planning tools; determining new parameter settings, using (some combination of) manual adjustments and/or computer-aided adjustments using planning tools or advanced simulation models; and applying new parameter settings, which may be done remotely or requires a site visit.

Depending on the applied methods, a use case-based estimation of the human effort in man hours involved in the three distinct phases can be made by the operator[1]. Multiplying this by the effective cost per expert hour, the number of times per year such a parameter adjustment is needed and a multiplication factor that reflects the number of cells (or cell classes) for which separate parameter adjustments need to be made, yields the OPEX per year for the considered use case.

In case self-organisation functionalities are applied, their specific impact on the above-mentioned distinct components that contribute to OPEX should be assessed, based on the properties of the developed solutions. Note that for some components the required human effort is significantly reduced, while for others it remains unchanged.

3.2 Benchmarking

A key objective in the development of methods for self-organisation is to do a quantitative comparison of different methods developed for a given use case, and to compare their achieved performance, capacity and cost to a case with manual network operation. Below we describe an approach for such benchmarking. The approach is primarily outlined from the perspective of a self-optimisation use case, although it is readily converted to cover self-configuration and self-healing use cases as well.

Starting point is a specific scenario in terms of e.g., the propagation environment, service mix, traffic characteristics and spatial traffic distribution. For such a scenario, the achievements of the different self-optimisation algorithms can be expressed in terms of the metrics mentioned in Section 3.1, including an estimation of the optimisation effort that is based on the observed number of automatic parameter adjustments per time unit. Note that if the optimisations were done manually, this would indicate the OPEX level.

Example values of the metrics obtained at the end of simulation studies are shown in Fig. 1[2]. Observe that e.g., self-optimisation algorithm SO_A achieves the highest performance, which can be exploited to achieve the lowest CAPEX, but in order to achieve this, a large optimisation effort is required. In contrast, algorithm SO_D is significantly less complex but consequently achieves worse performance and CAPEX.

In general, it is hard to compare the different self-optimisation schemes given the conflicting performance objectives. One possible approach to enforce a strict overall

[1] Note that this depends on the operator policy: a cost-driven operator is likely to spend less effort on network planning and optimisation than a quality-driven operator.

[2] Note that all figures shown in this section are only speculative examples to illustrate and explain the developed benchmarking approach.

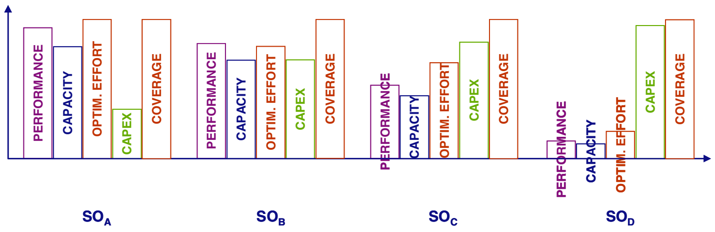


Fig. 1. Example values of obtained metrics

ranking is to weigh/combine the different measures into some utility function and rank the algorithms based on the obtained utility values.

Whereas the above discussion outlines an approach to compare different self-optimisation algorithms, a more difficult challenge is to compare a self-optimisation algorithm with a case of manual optimisation. In an extreme case, one could assume that a 'manual operator' freezes permanent settings of his radio parameters, which should then be chosen such that the overall performance of a given scenario is optimised. In practice, however, a network operator will upon observed need or sensibility adjust the radio parameters. Depending on the operator's policy this may happen more or less frequently: a quality-oriented operator is likely to do more frequent adjustments than a cost-oriented operator. In order to model this in a reasonable way, we propose to define 'manual optimisation algorithms' MO_A through MO_D (continuing the above example) such that they manually adjust radio parameters at the same time and to the same values as would the corresponding self-optimisation algorithms with the same label. Fig. 2 visualises an example comparison of SO_A with MO_D, concentrating on CAPEX and OPEX related measures.

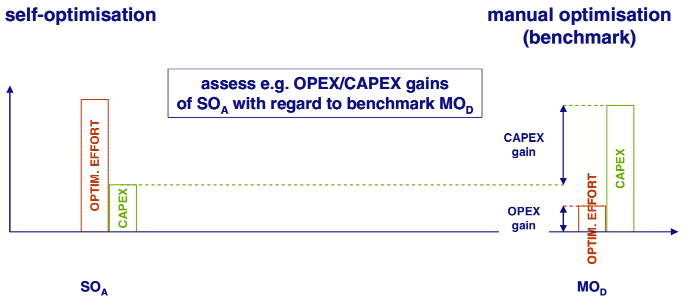

self-optimisation
manual optimisation (benchmark)
assess e.g. OPEX/CAPEX gains of SO_A with regard to benchmark MO_D
OPTIM. EFFORT
CAPEX
CAPEX gain
OPEX gain
SO_A
MO_D

Fig. 2. Comparison of SO_A with regard to benchmark MO_D

Assuming that self-optimisation reduces OPEX to zero (which may be too extreme, but is fine for illustrative purposes), the OPEX and CAPEX gains are indicated in Fig. 2. Note that the OPEX gain is determined by the optimisation effort that applies in the manual case. Continuing this approach for different combinations of SO_X and MO_Y we could generate tables such as Table 1, where the '+', '-' and '0's are just qualitative indicators; actual numerical values should be determined via simulations. Observe that introducing self-optimisation in the network of a quality-oriented operator is likely to establish the highest OPEX gains, but the lowest CAPEX gains.

Table 1. Comparing self-organisation algorithms with manual algorithms

		CAPEX GAINS				OPEX GAINS			
		SO_A	SO_B	SO_C	SO_D	SO_A	SO_B	SO_C	SO_D
quality oriented operator	MO_A	0	-	--	---	++++			
	MO_B	+	0	-	--	+++			
	MO_C	++	+	0	-	++			
cost oriented operator	MO_D	+++	++	+	0	+			

4 Concluding Remarks and Future Work

In this paper we have identified use cases, requirements and assessment criteria for future self-organising radio access networks. Within the SOCRATES project, these will form the basis of a framework for the development of self-organisation methods and algorithms, which describes among other things the relation and dependencies between the different components of SON. As future work, SON algorithms for the use cases will be developed, while the identified requirements will be taken into account. The proposed assessment criteria will then be used to evaluate the developed algorithms and solutions. As at this early stage of LTE development, a live test of developed SON algorithms in the field is not yet possible, the actual comparison and assessment of the different developed SON solutions will be done by performing simulation studies. The future work also includes the integration of the developed SON solutions, to ensure consistent behaviour when the developed algorithms are operated simultaneously.

References

1. 3GPP TR 32.816: Telecommunication Management; Study on Management of Evolved Universal Terrestrial Radio Access Network (E-UTRAN) and Evolved Packet Core (EPC) (Release 8) v1.3.1 (2007)
2. NGMN: Next Generation Mobile Networks Beyond HSPA & EVDO. White Paper (2006), `http://www.ngmn.org`
3. SOCRATES project, `http://www.fp7-socrates.eu`
4. Van den Berg, J.L., Litjens, R., Eisenblätter, A., Amirijoo, M., Linnell, O., Blondia, C., Kürner, T., Scully, N., Oszmianski, J., Schmelz, L.C.: Self-Organisation in Future Mobile Communication Networks. In: Proceedings of ICT Mobile Summit 2008, Stockholm, Sweden (2008)
5. SOCRATES deliverable D2.1: Use Cases for Self-Organising Networks, `http://www.fp7-socrates.eu`
6. SOCRATES deliverable D2.2: Requirements for Self-Organising Networks, `http://www.fp7-socrates.eu`
7. SOCRATES deliverable D2.3: Assessment Criteria for Self-Organising Networks, `http://www.fp7-socrates.eu`

Distributed Self-Optimization of Handover for the Long Term Evolution

André Schröder, Henrik Lundqvist, and Giorgio Nunzi

NEC Laboratories Europe, NEC Europe Ltd., Kurfürstenanlage 36,
69115 Heidelberg, Germany
{Andre.Schroeder,Henrik.Lundqvist,Giorgio.Nunzi}@nw.neclab.eu

Abstract. The 3GPP Long Term Evolution (LTE) is defining the next generation radio access network which introduces advanced service capabilities in base stations. Consequently, self-management is seen as an enabling technique for the deployment of LTE. This paper discusses handover optimization options for base stations and evaluates one possible approach of distributed self-optimization. After a review of the key aspects related to handover optimization, a self-optimization technique for handover based on trial-and-error is presented. The interference between neighboring nodes, a typical problem of distributed systems, is analyzed in more detail and a technique to cope with it is presented. Simulation results show that the control algorithm can achieve the expected optimal configuration through iterative steps and the effect of interfering neighboring nodes can be mitigated at the expense of a longer optimization time.

Keywords: Self-optimization, handover, 3GPP, E-UTRAN, LTE, distributed systems.

1 Introduction

The control and optimization of performance related to radio resources is a challenging and costly task for mobile operators. Optimization of handover is of major interest for operators, and the Next Generation Mobile Networks[1] (NGMN) consortium has defined a specific use case [1] to adopt self-optimizing techniques for handover configuration in the next generation 3GPP radio access network (RAN) [2][3], the so-called Long Term Evolution[2] (LTE). This effort aims at implementing optimization operations as automated control loops in the network to reduce manual intervention in the context of operation and maintenance.

There are several factors that make an optimization process for handover difficult. The first is the analysis of performance indicators that are available on the base station, the eNB. Such indicators provide elementary information about the current status of an eNB, but it is required to correlate different indicators and to compare their combined values with respect to an operator's objectives. Moreover, such indicators

[1] http://www.ngmn-cooperation.com/
[2] http://www.3gpp.org/Highlights/LTE/LTE.htm

vary with time, depending on user activity and radio environment. So it is important to capture their trend in the given traffic conditions during the optimization process and to ignore temporal fluctuations. The second factor is the different effects that a change in the radio configuration can have on system performance. A change of a parameter, like a threshold, can affect different performance indicators and, consequently, it is important to avoid negative impact on other performance indicators. Moreover, the change of a parameter at one eNodeB can also affect the performance of adjacent eNBs.

This paper provides a short study on handover optimization through a self-optimizing module for LTE. The proposed solution is decentralized and is based on a local module implemented on each eNB. Simulation results show the behavior of such a control loop as well as the effects of optimization on network performance.

2 Handover Optimization

This paper targets the network controlled handover, where the mobile terminal (MT) provides measurement reports to eNBs, but the decision for the handover is taken by the network. The behavior and performance of the RAN are evaluated by operators through a set of aggregated metrics, the so-called key performance indicators (KPIs). We identified a set of KPIs that are relevant for handover and that will be used for the self-optimization algorithm of handover. Their values are the sum of past event occurrences over a sliding window of length l_{sl} (in seconds); the end of the time window is the point in time of evaluation (present). For example, the number of incoming call requests accumulated during the recent period of time of length l_{sl}, e.g. in the last 60 seconds.

Handover performance indicators can be indirectly affected by a set of control parameters. *Can* means in this context that there is a certain probability that the modification has an effect on performance indicators, because the impact of the modification of a control parameter on the KPIs is not deterministic. The reason is that part of the system state, such as the users' positions, is not visible and future user activity is unknown. Therefore, the cause of a performance indicator's value can only be guessed or better be based on experience, but the exact reason remains hidden.

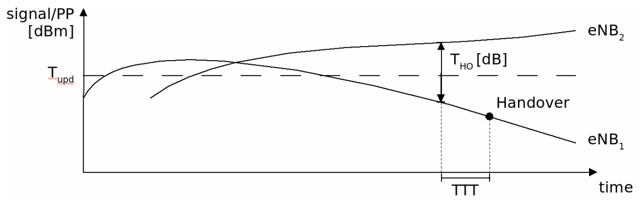


Fig. 1. Illustration of handover thresholds

Fig. 1 illustrates a selection of parameters involved in the handover decision: the figure shows an example of the signals of two base stations received by an MT. A handover decision is taken by the serving base station and is based on the measurements reported by the MT. The threshold T_{HO} means for the handover decision to require a minimum difference in signal strength between the serving and the potential new base station. The time-to-trigger (TTT) is a time interval during which the T_{HO}-condition must be fulfilled. T_{upd} controls in which range of the current received signal the MT should send measurement reports. This parameter is used in practice to reduce the message overhead in the RAN. In the following, we describe a method to approach an optimal state with a heuristic algorithm that continuously changes the current thresholds to discover the optimal configuration. A cost function is used to evaluate and aggregate the KPIs. Assuming that a *KPI* depends on a set of control parameters $p_0,...,p_n \in P_{KPI}$ plus the uncertain environmental conditions *E*, the *KPI* can be defined as a function:

$$KPI = f(P_{KPI}, E) \tag{1}$$

A global cost function is a linear combination of cost functions of individual KPIs:

$$C_{global} = \sum c_j(KPI_j) \tag{2}$$

where c_j is a weighting function that reflects a performance priorization of indicator KPI_j given by the operator. The objective of the optimization is to minimize C_{global}. Such cost functions can be used by operators to adjust the network performance, e.g. to find trade-offs between the number of call drops and the number of handovers by varying T_{HO}. By trend, a high T_{HO} decreases the number of handovers but increases the number of call drops and vice versa.

The optimization algorithm which is inspired by [4] is executed on each base station and is implemented as a continous trial-and-error loop. One or more parameters can be optimized, but in each step a single parameter *p* is considered. The optimization is done by measuring the system's performance for the currently set parameter *p* as well as for *p*+Δ and *p*-Δ in series, where Δ is an incremental step. In the next cycle of the algorithm, *p* is set to the value which resulted in the best performance of *p, p*+Δ and *p*-Δ. Through these continuous cycles of trials the algorithm can achieve the optimal configuration for that parameter.

3 Evaluation of Control Loop

This section evaluates simulation results of the the previously explained control loop for optimization of handovers with respect to the call drop ratio, using T_{HO} as parameter. The evaluation is based on a simulator that models and implements most of the handover characteristics. The call drop ratio can simply be defined as

$$CDR = \frac{C_D}{C_S + C_I} \tag{3}$$

where C_D is the number of dropped calls, C_S is the number of active connections at measurement start and C_I is the number of incoming calls and handovers during the current measurement period or sliding window.

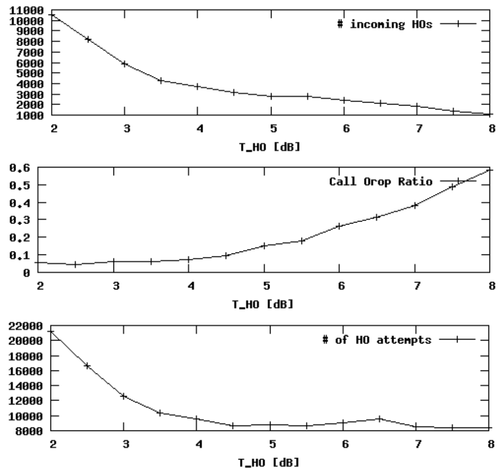


Fig. 2. Handover threshold dependency with call drop ratio

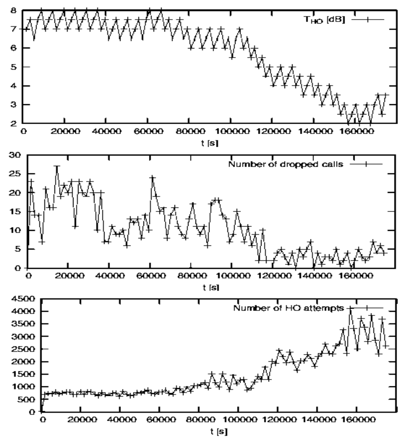


Fig. 3. Handover optimization on a single eNB

We have studied how handover perfomance depends on T_{HO} through simulations. Our scenario consisted of 9 eNBs in a 3x3 grid. The mobility model was a 12x9 Manhattan Grid with 200 moving users. In the simulation, bad radio conditions have been chosen, resulting in many failing handovers and dropped calls, in order to illustrate the handover optmization process. In a first simulation, all base stations have been assigned the same T_{HO} values and different T_{HO} values have been tested subsequently. Fig. 2 shows CDR, the number of handover attempts as well as the number of incoming handovers plotted over T_{HO}. The results shown are averages of all basestations in a scenario. It is obvious that the numbers of incoming and outgoing handovers decrease with an increasing T_{HO} since the T_{HO} is the threshold for handovers. In contrast to that, the number of dropped calls increases for large values of T_{HO}. This is due to an increasing risk that the user will move out of coverage before the handover is completed.

Fig. 3 shows the behavior of the self-optimization loop of the previous chapter on a base station over time. To make the convergence visible, the starting value of threshold T_{HO} is set much higher than the target value. T_{HO} is then gradually adjusted downwards by the algorithm. The trend in the call drop ratio follows the threshold downwards to a large extent, which is the targeted behavior. However, the used approach does not immediately show a trend towards the optimum which is due to noisy measurements on the one side and also due to optimization interferences between base stations. The optimization loop used intervals on a relatively short time scale which causes noise. The trends that are visible on the graphs are timescales spanning several hours, whereas the decisions are made based on measurements over 30 minutes.

It is obvious on first sight that a quick reaction time in terms of a low T_{HO} is beneficial. However, handovers cannot happen arbitrarily fast which is why disadvantageous handovers might result in call drops, in case T_{HO} is too low. An example is street canyons in which cars pass crossing roads with better signal.

4 Local Serialization

Self-optimization modules can be run individually and in parallel at each eNB. In practice, changing the settings at a single eNB is likely to indirectly affect the adjacent eNBs as well. This is evident if the pilot power of a base station is changed, because it immediately affects the level of interferences with its neighbors as well as the coverage area which influences the number of attached MTs.

According to the effect of a configuration change on an adjacent cell, optimization efforts of neighboring cells interfere with each other. Our simulations have shown that the optimization convergence time is higher on an eNB if adjacent eNBs are performing optimizations on the same parameter as well, e.g. T_{HO}.

One solution to minimize the inter-cell optimization interferences is to send messages to neighbors to give them the opportunity to react appropriately. However, the challenging question remains, how to appropriately react on the reception of such a message and how to incorporate this reaction into ongoing optimization or measurements. Therefore, we follow a different approach. Our approach tries to take advantage of the accuracy of non-simultaneous optimization of neighboring eNBs while still subsequently performing optimizations on all eNBs. We call this method local serialization.

In this approach, only one eNB in a neighborhood of eNBs is allowed to perform optimizations at a time. After one cycle of optimization, another eNB performs optimization, while the one that performed optimization in the previous run is in waiting state which is added between the cycles of the algorithm described in section 2. Fig. 4 shows an exemplary series of three steps of local serialization in a simple network structure. Each circle with a number represents an eNB, gray ones are performing optimizations. The large circle exemplary indicates a neighborhood.

It is obvious that this approach requires some kind of coordination amongst eNBs. To accomplish this, any common election mechanisms can be used. To make the process fair, counters for the number of optimization periods performed by an eNB should be used and exchanged during an election process.

The expected gain from the approach is higher accuracy in a single optimization period. An obvious drawback to be expected is the convergence time since the eNBs do optimization less often. A simulation to show the feasibility of this method has resulted in the expected behavior. Fig. 5 shows a per eNB performance comparison between simultaneous (all) optimization and locally serialized optimization.

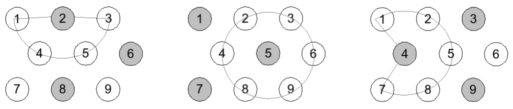


Fig. 4. Three exemplary steps of local serialization

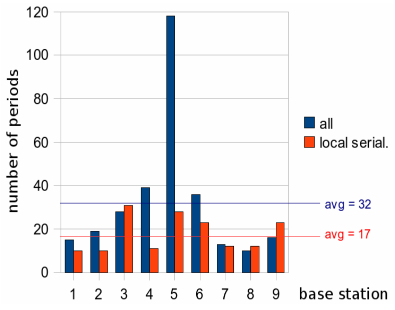


Fig. 5. Parallel vs. serialized

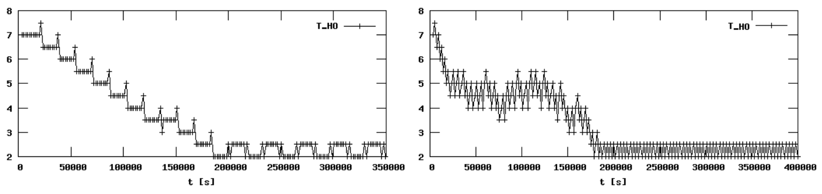


Fig. 6. The simultaneous optimization in (a) needed 39 periods, serialization of optimization in (b) need 11 optimization periods

The high value at eNB 5 means that the optimization loop needed an extra-ordinary long time for convergence in the simultaneous case, trapped in a local minimum, which did not happen in the locally synchronized case. The average number of needed optimization periods is lower in the case of local serialization. The figures 6 a) and b) show an example of the optimization of T_{HO} of an eNB. In the first example, optimization is done simultaneously, in the second local serialization is used. Concerning the convergence time, the average convergence needs much more time using local serialization, although this is not obvious from this shown example. Although convergence even showed quicker in some cases, it was up to 4 times slower in worst cases. In the end, it is up to the operator to decide a trade-off between the number of configuration changes, optimization time, and the resulting performance.

5 Conclusions

This paper presented distributed handover optimization techniques for the 3GPP LTE. We showed how a trial-and-error approach can be used for optimization of a set of parameters that we identified to influence handovers, especially T_{HO}. Simulations showed that this algorithm can be used to improve handover performance in a distributed manner with iterative steps. We verified that the interference between local algorithms of neighboring nodes has an impact on the global convergence and we propose a method that we call local serialization. This reduces the number of optimization steps per eNB but increases the total optimization time.

We believe that this technique can be introduced to increase robustness of self-management in LTE and therefore contribute to achieve LTE's goals of a scalable radio access network with reduced maintenance operations.

References

[1] NGMN: Use Cases related to Self Organising Network - Overall Description. V2.02 (2007)

[2] 3GPP: E-UTRAN Overall description; Stage 2. TS 36.300, V8.5.0 (May 2008)

[3] 3GPP: E-UTRAN Self-configuring and self-optimizing network use cases and solutions. TR 36.902, V0.0.1 (Febraury 2008)

[4] Flanagan, J.A., Novosad, T.: WCDMA Network Cost Function Minimization for Soft Handover Optimization with Variable User Load. In: 56th IEEE Vehicular Technology Conference, vol. 4, pp. 2224–2228 (2002)

ADAPT: A Semantics-Oriented Protocol Architecture

Stefan Götz, Christian Beckel, Tobias Heer, and Klaus Wehrle

Distributed Systems Group
RWTH Aachen University, Germany
{goetz,heer,wehrle}@cs.rwth-aachen.de,
beckel@informatik.uni-tuebingen.de

Abstract. Although modularized protocol frameworks are flexible and adaptive to the increasing heterogeneity of networking environments, it remains a challenge to automatically compose communication stacks from protocol modules. The typical static classification into network layers or class hierarchies cannot appropriately accommodate cross-cutting changes such as overlay routing or cross-layer signaling.

In this paper, we discuss how protocol composition can be driven by functionality and demand at runtime based on extensible semantic models of protocols and their execution environment. Such an approach allows to reason about the functionality and quality of automatically composed and adapted protocol compounds and it is open to existing and future protocols.

1 Introduction

The static nature of the classic TCP/IP protocol stack and its increasing complexity has prompted research in the area of dynamic and modularized communication subsystems from a number of different angles [1,2]. Ideally, a communication subsystem should resemble a configurable, dynamic framework of individual protocols that, in their entirety, provide a target functionality. To maximize modularity, protocol implementations should adhere to a uniform interface while information about protocol semantics is represented separately [3].

In addition to protocol functionality, a large number of factors play a role in composing a protocol stack that best matches its operational environment: dependencies among components, the network configuration (e.g., requiring authentication or tunneling), capabilities of communication partners (such as their support for specific protocols), user preferences (privacy, cost, etc.), and classic Quality-of-Service (QoS) metrics (e.g., application requirements or device capabilities). The length and incompleteness of this list illustrates the main deficiency of existing approaches: they describe protocols through strict design-time classifications, such as class hierarchies or languages with a fixed vocabulary, and are thus only poorly extensible. Consequently, they cannot incorporate new protocols, functionalities, or requirements.

In this paper, we discuss automated protocol composition in our dynamic protocol framework ADAPT. It leverages a semantic description format based on ontologies that provides an abstract notion of a protocol's functionality and properties to guide protocol composition. Transparently to the user, mobile applications can use dynamically composed protocol stacks that permanently adapt to their needs as well as to changes in the execution and network environment, as caused by, for example, roaming and network hand-offs. ADAPT's generic protocol model is not bound to established layering conventions and consistently integrates tunneling, encapsulation, and transformation (e.g., for VPNs, overlays, encryption), and significantly simplifies protocol deployment. Despite these significant differences from a traditional TCP/IP stack, ADAPT strives for transparency to existing protocols and applications.

2 Related Work

In modularized protocol frameworks, the composition of a protocol stack needs to satisfy criteria like protocol dependencies or functional requirements (for example to include loss handling or to perform compression before encryption). F-CSS [4] and DaCapo [5] explicitly separate this information from the implementation and represent it through custom languages. However, these description formats fail to achieve a clean separation since they expose implementation aspects. Also, they rigidly encode the protocol-related knowledge at design time and do not lend themselves to later extension.

In knowledge representation, ontologies have received wide-spread attention in particular around the semantic web and for web service management [6,7], but also innumerable other fields of information science. Ontologies strike a compromise between formalism and expressiveness that allows for being intuitive, generic, extensible, and powerful for reasoning and querying [8]. Zhou and associates apply this approach to the protocol domain [9] but their framework and modeling centers on a classic stack design of monolithic protocols ignoring decomposition and protocol extensions. Their ontology bases on the Internet protocol layers to reduce redundancy in the description and to reduce the complexity of the orchestration process. Consequently, this approach cannot support non-classic protocol arrangements such as overlay routing. ADAPT lifts this restriction by removing the layer structure from the protocol model. Thus, the complexity of protocol orchestration increases significantly and forms the challenge which we address in this paper.

3 Design

The basis of ADAPT is an end-system communication framework in which protocols operate as software components which share uniform interfaces and which can be instantiated, configured, and replaced at runtime. This framework follows the basic concepts of other componentized OS and network systems such as the x-Kernel [2] so that individual protocols can be composed into *protoocol chains.*

3.1 Semantic Protocol Modeling

The semantic model of protocols and their execution environment formally describes protocols and the orchestration criteria. Adapt uses *OWL DL*, a sublanguage of the Web Ontology Language (OWL) [10], for its high expressiveness and decidability in reasoning. In OWL DL, knowledge is represented by classes with class properties, individuals belonging to classes, and relations between them. Our ontology models not only protocols as individual classes but also the orchestration criteria and the relationships between them. This allows for extensibility with new criteria and forms of relationships as opposed to a model based on class properties.

To support the orchestration process and its individual stages, we distinguish three categories of models: functionalities, dependencies, and qualitative information. The functionality model derives directly from individual protocol functionalities, such as session support or loss handling. By sub-classing, it expresses a refinement of a more abstract functionality (e.g., packet retransmission is a sub-class of the loss handling class). The dependency model establishes associations between protocols, functionalities, and user-defined criteria. It also establishes *and* and *or* relationships between multiple dependencies (e.g., to enforce the inclusion of one of two specific encryption protocols). Furthermore, the ontology provides *qualitative information* about such aspects as protocol resource demands. Here, we distinguish between *requirements* a protocol imposes on its protocol chain or the environment (e.g., the existence of a DNS server) and information that solely affects the *ranking* of different protocol chains.

Based on the explicitly defined *asserted model*, the reasoning process derives an *inferred model* that represents additional knowledge. Adapt employs the following reasoning capabilities, primarily as fundamental means to integrate future protocols, orchestration criteria, and metrics.

Type inheritance: by inference, individuals of class X inherit the types of X's super classes. Thus, protocols describe their functionality precisely (e.g., RSA encryption) and can later be classified more generically (e.g., as providing confidentiality) through newly introduced knowledge.

Inverse properties: a symmetric relation between X and Y specified only for X is inferred to apply to Y. Thus, the inferred knowledge base remains consistent despite the incorporation of new knowledge.

Instance classification: *defined classes* classify individuals through a set of logic expressions. Consequently, the knowledge about the criteria of class membership receives an explicit representation.

Rule support: user-defined rules allow to assert new facts about individuals. Protocols can be asserted to support reliability, for example, if they provide ordered data delivery, retransmission, and a checksum algorithm.

OWL DL represents information in an abstract syntax or *RDF/XML* format. It allows to describe protocol semantics as a single unit which can be easily exchanged among endsystems and merged with other pre-existing ontologies at runtime.

3.2 Orchestration Criteria

The process of protocol orchestration in Adapt is driven, on the one hand, by the functionality and inter-dependencies of the protocols and, on the other hand, by the properties of the execution and network environment. Although many of these factors relate to typical QoS parameters, our research focus lies not on a QoS framework but on supporting arbitrary protocols at all layers. We distinguish four broad categories in the ontology:

Network capabilities primarily influence the orchestration of the network-related lower-level elements of chains. They range from local properties, such as a mandatory link protocol, across remote factors (e.g., the necessity of authentication) to QoS aspects, such as link loss rates or latencies.

Device capabilities are of a similar qualitative nature and primarily reflect information about the available CPU, memory, and energy resources.

Application requirements stem directly from application requests, e.g. to establish a new connection. They comprise functional requirements, such as session semantics, and qualitative aspects (a preference for low latency, for example).

The user influences via their *user preferences* how the above qualitative factors are traded against each other in the orchestration process. Typical trade-offs concern security, cost, and performance aspects. User preferences also provide immediate configuration information, for example fixed IP addresses, authentication information, and wireless network priorities. Finally, they allow users to influence the orchestration process such that, e.g., VPN tunneling is enforced for specific networks.

3.3 Orchestration Process

The fundamental goal of protocol orchestration is to determine the protocol chain best suited to the given application and environment requirements. Finding such a chain in the full set of all possible protocol combinations (a selective approach) suffers from limited scalability with a growing number of protocols. Adapt thus follows a constructive approach in two phases. First, it composes only the protocol chains that are functionally viable and fulfill all requirements imposed by the application request and the current execution environment. In the second phase, an expert systems ranks each resulting chain to determine the one which matches the environment best.

Composition. The semantic composer consecutively relies on three types of composition information contained in the semantic protocol descriptions. First, it evaluates the functional requirements of the application and the execution environment to obtain all protocols that are necessary to satisfy these demands. Next, it recursively resolves the dependencies among protocols and constructs partial protocol chains (*stubs*) from this information. Finally, it merges the partial chains into complete functional compounds that can later be instantiated for packet processing.

For the dependency resolution of the second step, the semantic composer distinguishes direct dependencies, which need to be satisfied by the next protocol in the chain (e.g., a specific address format), and indirect dependencies, which can be fulfilled further down the chain (e.g., encryption). For each protocol, resolving dependencies is a recursive process that creates individual stubs for each alternative. To avoid a state explosion during stub merging, the composer obeys the ordering imposed by the dependencies, immediately discarding chains contradicting that order. Since mechanisms like tunnels or overlays change and potentially contradict regular ordering constraints, so-called *connector modules* allow overriding them. Finally, the composer validates the chains so they satisfy their internal dependencies and the external requirements and discards non-matching chains.

Ranking. The ranking phase aggregates the quality metrics of individual protocols to determine a single chain to instantiate. While such qualitative information is available for protocols from their semantic model, the ADAPT runtime provides the equivalent information for the network, the endsystem, and the user preferences. The latter also specify optional weight factors for each of these properties to bias their influence on the result. Based on these inputs, the expert system first matches the corresponding protocol and environment properties to derive per-protocol metrics. Then, it feeds them to a multi-criterion decision making system, which aggregates the metrics for each chain to arrive at a ranking order of all chains. Thus, the highest-ranking protocol chain emerges as the best match for the current communication requirements.

4 Results

As an initial evaluation, we tested the protocol orchestration with three typical functionality requests: support for a reliable connection, for multicast, and for name resolution. The ontological model contained 14 protocols and protocol extensions (e.g., an extension to TCP which makes it perform better in scenarios with high bit error rate), tunnels (e.g., in case IP Multicast is not supported by the network the host resides in), and network requirements (existence of a DNS server). The matchmaker, the composition engine, and the stub expert system are implemented in Java based on the Jena semantic web framework which manages the ontology and provides a reasoner, a query engine, and basic rule support. The composition engine uses SPARQL [11] to perform queries on the description repository. All measurements were performed on a Linux system with a 1.80 GHz Intel Pentium M processor and 1GB RAM, employing Sun's Java 1.5.0 runtime environment and version 2.5.5 of the Jena library.

Table 1 lists the duration of the matchmaking process, the protocol composition process, and the number of resulting protocol chains for queries which specify the desired functionality concisely. More loosely specified queries are satisfied by more chains but they also increase the composition time significantly, as Table 2 illustrates. We see considerable room for improvement in our current implementation, e.g., via additional, though less generic, rules in the composition process, pre-computation, and caching of chain stubs. Although an evaluation of the expert

Table 1. Protocol composition based on restrictive queries

Query	Matching	Compos.	Sum	# chains
Reliability	43 ms	117 ms	166 ms	6
Name resolution	30 ms	136 ms	166 ms	4
Multicast query	53 ms	62 ms	115 ms	18

Table 2. Protocol composition based on non-restrictive queries

Query	Matching	Compos.	Sum	# chains
Reliability	33 ms	222 ms	255 ms	226
Name resolution	29 ms	375 ms	404 ms	655
Multicast query	54 ms	112 ms	166 ms	88

system is still outstanding, these initial results strongly suggest the practical viability of a functional composition of protocol modules based on ontological models.

References

1. Muhugusa, M., Di Marzo, G., Tschudin, C.F., Harms, J.: ComScript: An Environment for the Implementation of Protocol Stacks and their Dynamic Reconfiguration. In: International Symposium on Applied Corporate Computing (1994)
2. Hutchinson, N.C., Peterson, L.L.: The x-Kernel: An Architecture for Implementing Network Protocols. IEEE Transactions on Software Engineering 17(1) (1991)
3. O'Malley, S.W., Peterson, L.L.: A Dynamic Network Architecture. ACM Trans. Comput. Syst. 10(2), 110–143 (1992)
4. Zitterbart, M., Stiller, B., Tantawy, A.N.: A Model for Flexible High-performance Communication Subsystems. IEEE Journal on Selected Areas in Communications 11(4), 507–518 (1993)
5. Plagemann, T., Vogt, M., Plattner, B., Walter, T.: Modules as Building Blocks for Protocol Configuration. In: 1993 International Conference on Network Protocols 1993. Proceedings, pp. 106–113 (1993)
6. Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K.P.: Semantic Matching of Web Services Capabilities. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 333–347. Springer, Heidelberg (2002)
7. Li, L., Horrocks, I.: A Software Framework for Matchmaking Based on Semantic Web Technology. In: Proceedings of the 12th International Conference on World Wide Web, New York, NY, USA, pp. 331–339 (2003)
8. Wang, X.H., Zhang, D.Q., Gu, T., Pung, H.K.: Ontology Based Context Modeling and Reasoning Using OWL, 18–22 (2004)
9. Zhou, L., Pung, H.K., Ngoh, L.H., Gu, T.: Ontology Modeling of a Dynamic Protocol Stack. In: 31st IEEE Conference on Local Computer Networks, pp. 353–360. IEEE Computer Society, Los Alamitos (2006)
10. McGuinness, D.L., van Harmelen, F.: OWL Web Ontology Language Overview. W3C Recommendation (February 2004)
11. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF. W3C Recommendation (January 2008)

Cross Layer Dynamics in Self-Organising Service Oriented Architectures

Martin Randles, A. Taleb-Bendiab, and David Lamb

School of Computing and Mathematical Sciences, Liverpool John Moores University, UK
{m.j.randles,a.talebbendiab}@ljmu.ac.uk, d.lamb@2005.ljmu.ac.uk

Abstract. This paper assesses and analyses the in and cross layer dynamics engendered by the engineering of self-organisation in one layer of a global Service Oriented Architecture currently emerging as the Internet of Services. A resource allocation algorithm is implemented at the application (business services) layer and its impact is investigated across this layer and the resource (business function) layer through the analysis of the service (server) compositions (digital ecosystem) arising and its associated partitioning into task specific teams (communities). Beneficial self-organisation is shown to ensue in a remote layer from the initially instigated engineered self-organisation.

1 Introduction

At present, much research work is devoted to engineering various examples of self-organising behaviour, observed in non-equilibrium natural or large-scale artificial systems [1]; for instance, in solving complex networking problems, to engender resilient network topologies or configurations [2] or provide *ad-hoc* routing in sensor and actuation overlays [3]. There is, however, a further notable effort devoted to analysing the consequential effects of self-organisation, which has not been specifically engineered to occur [2, 4 and 5]; for instance, in predicting or exploiting membership models of community/organisation, arising as a side-effect of self-organisation within or across layers. Intuitively, for example, in the layered Internet of Services (IoS) model [6], self-organisation engineered at the application layer will inevitably have consequences at other layers and within the applications that it is supporting. The effects, however, are hard to predict and/or guard for or against. Furthermore to fully understand the functions of the system, to realise potential gains or opportunities, the structure ought to evolve dynamically in response to the emerging function.

This paper will address the usual problem of load-balancing, in server clusters, by the application of a self-organising strategy based on beehive dynamics. The strategy is assessed for effectiveness, to show the beneficial effects of the self-organisation at the application layer. A developed algorithm for community detection, comprising some early findings on runtime detection of emerging sub-structures (communities) in the resource layer, is then applied to analyse the utilisation of resources, which arises in response to the induced self-organisation in the IoS application layer, yet which were not specifically programmed for in the executing algorithm. Accordingly in this paper Section 2 gives a basic overview for formalisation and community detection to reason on self-organisation. Section 3 reports on the formalisation and implemented simulation. Section 4 gives some general conclusions and suggestions for future work.

2 Formalisation for Community Detection and Self-Organisation

In order to have a consistent reasoning model, for component actions and interactions (local level) up to whole system behaviour (global awareness), not dependent on explicit pre-defined outcomes, it is necessary to formalise an Observer System [7] (and associated system model) in logic, to facilitate deliberation over self-organisation in the systems and systems of systems (multi-layered) states. Specifically the benefits of employing mathematical logic in this instance include [8]: The removal of the need to explicitly enumerate states and their transition functions. Behaviour is a consequence of deduction from the systems' description and a propositional account provides an abstract specification to prove properties of the system, entirely within the logic. Where deduction is efficient the system specification is also executable; thus rendering a simulator for the system as a side effect of the specification. To handle the dynamism within the domain Situation Calculus is used.

2.1 Situation Calculus

In Situation Calculus fluent values, stating what is true in the system, are initialised in the starting situation (S_0) and change from situation to situation according to effect axioms for each action. The partial solution to the resultant frame problem [8] gives successor state axioms that largely specify the system together with action precondition axioms and the initial situation. So an initial situation, S_0 is the start of the Situation Calculus representation. An action, a, then changes this situation from S_0 to do(a,S_0) with the next action, a_1 say, changing the situation to do(a_1,do(a,S_0)) and so on. Thus a situation S_n is comprised of a simple action history: $a_1a_2.....a_n$.

2.2 Detecting Communities

This paper presents a special case of community detection for network topologies with power law connectivity. The range for the degree distribution in a network is given by a probability distribution *P(k)*, the likelihood that a node has degree *k*. The observed distributions for many large-scale networks; the World Wide Web [9], and Internet (router level graphs) [10] for example, display a degree distribution with a power law tail: The distribution, for some $\lambda>0$ with $k=m,....M,$ is given by

$$P(k) = ck^{-\lambda} \tag{1}$$

where m and M are the lower and upper bound respectively of the nodes' connectivity and c is a constant normalisation factor. In such systems there is a lack of scaling around the mean connectivity leading to the topology being termed *scale-free*.

One of the most noticeable features of scale-free topology is the formation of high degree hub nodes. So if the set of links for the system is C, meaning that if node *i* is connected to node *j* then $(i,j)\in C$ for any nodes *i* and *j*

$$\sum_{(i,j)\in C} k_i k_j \tag{2}$$

is maximised when high degree nodes are connected to other high degree nodes [11]. The Hub Connection Density (HCD) is defined for a graph consisting of a set of vertices, V, with |V| = N and a set of edges E linking the vertices, as:

$$\frac{1}{N}\sum_{(i,j)\in E} k_i k_j \tag{3}$$

for any vertices i and j with degrees k_i and k_j respectively. In particular when such circumstances prevail communities can be identified around the hubs of the system: The edges between hubs display high betweenness. Thus integrated communities may be isolated (identified) through selective breaking of the network at hub-to-hub edges.

3 Case Study

The Internet of Services (IoS) provides an important example of a current trend seeing the convergence of Web 2.0 technologies and Service Oriented Architectures (SOA) into a global SOA [12]: In this illustrative case-study, in line with the envisaged IoS, a simulation of a self-organising, beehive-based load-balancing algorithm is used at the application (business services) layer for optimum resource allocation to further the understanding of cross layer dynamics. This addresses the server allocation problem in a large-scale SOA; allocating servers to Web applications to maximise profit.

3.1 Load Balancing Based on Bee Hive Dynamics

This case study follows the specification, given in [13]; for the coordination of a cluster of servers, hosting the (Web) services. As the demand for Web services fluctuates it is desirable to enable dynamic service allocation to regulate the system based on user demand. Servers $s_1,\ldots..s_n$ are arranged into M virtual servers $VS_0,\ldots.VS_{M-1}$ with service queues $Q_1,\ldots,Q_{M-1}$ respectively. The reward for a server $s_i \in VS_j$ serving a request from Q_j is c_j. Each server is either a forager or a scout and the advert board is where servers, successfully fulfilling a request, may place adverts, with probability p; a forager server may read the advert board, a scout simply chooses a random virtual server group VS_j.

The server s_i assesses its profit by comparing its profit, P_i to the colony's profit, P_{colony} and adjusts the probability of reading the advert board accordingly. This permits a fully formal specification of the domain, necessary for deliberation on emergent outcome. It is necessary to specify the deterministic outcomes for each stochastic action. For example the *readAdvertBoard* action can succeed or fail denoted *s_readAdvertBoard* and *f_readAdvertBoard* as appropriate, for a forager F:

$choice(readAdvertBoard(F),a) \equiv a = s\_readAdvertboard(F) \vee a = f\_readAdvertBoard(F)$

For each of nature's choices associated with an action we specify the probability with which it is chosen, given the stochastic action was performed in situation s, for example for the successful reading of the advert board we may have:

$$\begin{aligned}&prob(s\_\ readAdvertboard(F),\ readAdvertBoard(F),s)=p \equiv perf(F,s)\leq 0.5 \wedge p=0.6 \vee\\ &\quad (perf(F,s)>0.5\wedge perf(F,s)\leq 0.85\wedge p=0.2)\vee(perf(F,s)>0.85\wedge perf(F,s)\leq 1.15\wedge p=0.02)\end{aligned}$$

where perf(F,s) is the performance of the forager server F in situation s as measured by the ratio of forager profit (P_F) against colony profit (P_{colony}).

The action precondition axiom is:

$$poss(s\_readAdvertBoard,s) \Rightarrow forager(F,s)\wedge atAdvertboard(s)$$

The successor state axioms can be stated, for example:

$$\begin{aligned}&perf(F,do(a,s))=n\Leftrightarrow(perf(F,s)=n\wedge\neg\exists s_i a=finishrequest(s_i))\vee\exists i,j[(s_i\in VS_j)\wedge\\ &\quad[(a=finishrequest(s_i)\wedge((F=s_i\wedge P_i(s)= perf(F,s)P_{colony}(s) - c_j perf(F,s) + c_j))\vee\\ &\qquad(F\neq s_i\wedge P_i(s)= perf(F,s)P_{colony}(s) - c_j perf(F,s))]\end{aligned}$$

3.2 Evaluation

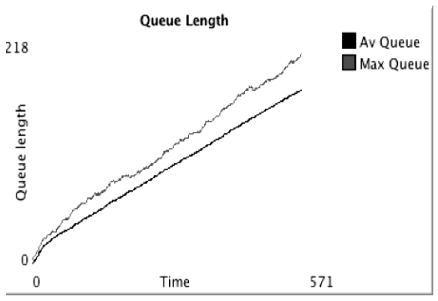
The Netlogo [14] simulation environment has been used to test the implementation of this specification under variable conditions. The servers possess an attribute number that is used to combine the optimum servers for the tasks in the queues at the virtual servers; simulating heterogeneity in the servers covering a large-scale SOA or IoS. The simulation starts with 99 forager and 71 scout servers connecting with virtual servers and their associated task queues. During operation servers dynamically connect to and service virtual server queues then reconfigure according to the algorithm.

Figure 2 shows the results for the server cluster, after 500 ticks, performing tasks of varying lengths up to 50 ticks. The task arrival at the virtual server queues was randomized with a 0.8 probability of arrival. The top line on the graphs is the maximum queue length at the virtual servers whilst the lower line is the average queue length: This demonstrates the significant improvement in load balancing using this self-organisational algorithm: There is a 27% decrease in average queue length, a 40% decrease in the maximum queue length and a smaller difference between the average and maximum queue lengths (low variance/standard-deviation=better balanced load) produced when the beehive foraging algorithm is used (Fig. 2(b)) compared to when it is not in operation (Fig. 2(a)).

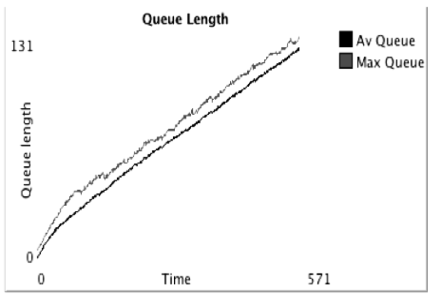
From this an underlying community structure can be extracted comprising the servers that work together on the specific tasks. Figure 3 shows the extracted underlying community structure for the resource layer (servers) with connections between those cooperating to complete a specific task, indicating dependencies.

An initial breakdown of the community structure is then possible based on the removal of edges with high betweenness, as previously described. The edge betweenness is calculated based on the hub-hub connectivity established via the HCD Measure.

Further experimentation has been completed placing markers at the automatically recognised hubs to either facilitate the propagation of data to the whole community or identify monitor points for specific communities.

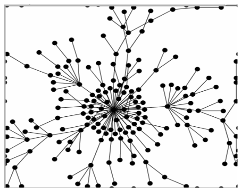


(a)

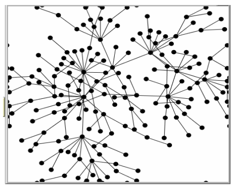


(b)

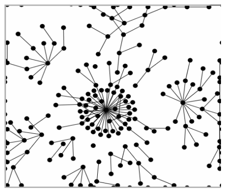
Fig. 2. Average queue lengths: (a) 165 with a standard deviation of 20.45 without foraging algorithm. (b) 121 with a standard deviation of 5.07 with foraging algorithm, at t=500.



(a)



(b)



(c)

Fig. 3. Community structure of servers: (a) When the foraging algorithm is used with a HCD calculated at 80 against an expected value of 19 for a random arrangement. (b) Corresponding community when the foraging algorithm is not used with a HCD calculated at 28. (c) Initial community structure breakdown from (a).

4 Conclusion

This paper has sought to address the engineering of self-organisation in a manner that accounts for the wider systemic implications of the implemented behaviour; thus additional benefits of self-organisation, in a SOA or the envisioned Internet of Services (IoS), aiding the identification of emergent resilient service compositions in the layer and its super/sub-layers, can be seen. It can also be observed that self-organisation at one layer affects other layers: This may be useful to take into account the upper layer requirements when the provisioning layer is self-organising or evolving; indicating the level of intervention required to either promote the organisational behaviour or provide some guards against it, resolving conflicts between organisational layers. There are many scenarios where detection of the underlying or emergent structures is both desirable and necessary to ensure appropriate resource provision and the optimisation of some system-wide value or utility. For instance the guaranteed provision of an emergent service feature can be dependent on the topological nature of the arrangement and specific procedures may utilise the properties of the topology.

Much further work is required to investigate the overall effects of self-organisation and the most beneficial system location in which to engineer emergence. For instance, in the described case study, would the engineering of scale-free connectivity in the resource layer give a similar beneficial result for the load-balancing in the application layer? Work is also ongoing validating these results in a real setting.

References

1. Mamei, M., Menezes, R., Tolksdorf, R., Zambonelli, F.: Case Studies for Self-Organization in Computer Science. J. Syst. Archit. 52, 443–460 (2006)
2. Doyle, J.C., Low, S., Carlson, J.M., Paganini, F., Vinnicombe, G., Willingerand, W., Parillo, P.: Robustness and the Internet: Theoretical Foundations. In: Jen, E. (ed.) Robust Design: A Repertoire of Biological, Ecological, and Engineering Case Studies (Santa Fe Institute Studies on the Sciences of Complexity). Oxford University Press, Oxford (2005)
3. Baldoni, R., Marchetti, C., Virgillito, A., Vitenberg, R.: Content-Based Publish-Subscribe over Structured Overlay Networks. In: 25th IEEE International Conference on Distributed Computing Systems (ICDCS 2005), pp. 437–446 (2005)
4. Stromberg, S., Carlson, J.: Robustness and Fragility in Immunosenescence. PLoS Computational Biology 2(11), 160 (2006)
5. Strassner, J., Foghlu, M.O., Donnelly, W., Agoulmine, N.: Beyond the Knowledge Plane: An Inference Plane to Support the Next Generation Internet. In: 1st International Global Information Infrastructure Symposium (GIIS 2007), pp. 112–119 (2007)
6. Ruggaber, R.: Internet of Services SAP Research Vision. In: 16th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2007), p. 3 (2007)
7. Randles, M., Zhu, H., Taleb-Bendiab, A.: A Formal Approach to the Engineering of Emergence and its Recurrence. In: 4th IEEE Conference on Autonomic Computing ICAC 2007-EEDAS Workshop, USA (2007)
8. Reiter, R.: Knowledge in Action. MIT Press, Cambridge (2001)
9. Albert, R., Jeong, R.H., Barabasi, A.-L.: Diameter of the World-Wide Web. Nature 401, 130 (1999)
10. Faloutsos, M., Faloutsos, P., Faloutsos, C.: On Power Law Relationships of the Internet Topology. ACM SIGCOMM Computer Communication Rev. 29(4), 251–262 (1999)
11. Li, L., Alderson, D., Doyle, J.C., Willinger, W.: Towards a Theory of Scale-Free Graphs: Definition, Properties and Implications. Internet Mathematics 2(4), 431–523 (2005)
12. Schroth, C., Janner, T.: Web 2.0 and SOA: Converging Concepts Enabling the Internet of Services. IT Professional 9(3), 36–41 (2007)
13. Nakrani, S., Tovey, C.: On Honey Bees and Dynamic Server Allocation in Internet Hosting Centers. Adaptive Behavior 12, 223–240 (2004)
14. NetLogo Simulation Software Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL (2008), http://ccl.northwestern.edu/netlogo

Can Solutions Emerge?

Michael Zapf and Thomas Weise

Universität Kassel, Wilhelmshöher Allee 73
D-34121 Kassel, Germany
{zapf,weise}@vs.uni-kassel.de

Abstract. Emergence engineering is a novel approach in Software Engineering which targets at triggering emergent phenomena in groups of individuals in order to exploit those phenomena for engineering solutions. We impose the requirements of functional adequateness to a dynamic system and wait for it to adapt. In this article we discuss the effects of the expressiveness of the behavioral description in terms of reliability of the solutions. Can we expect Emergence Engineering to produce solutions in the proper meaning of the term at all?

1 Introduction

Recent developments in ubiquitous computing and autonomous distributed systems show that standard engineering approaches reach their limits when we have to cope with a large number of interacting, autonomous platforms. Instead of the traditional divide-and-conquer approach, realizing self-organization capabilities as found in nature have gained substantial interest. Swarms demonstrate that large collections of individuals may produce a useful group behavior whereas the individual behavior may be difficult to determine. We call these phenomena *emergent* when we consider it unfeasible to analyze the contribution of the individuals to the overall effect. Utilizing emergent phenomena within an engineering process is called Emergence Engineering, a rather new area in software engineering. A characteristic property of Emergence Engineering is that one cannot guarantee whether the results of this process actually reliably fulfill the requirements of the scenario, that is, whether the results are really “solutions” as we understand the term. The question arises whether such an emergent process can create reliable solutions at all.

In the following sections we approach this question by briefly introducing our concept of Offline Emergence Engineering (OEE), enhanced with a more expressive behavioral language. We discuss the contribution of the base language to the evolution success, and we comment on the overall practicability of creating solutions by emergence.

2 Offline Emergence Engineering

Our approach to Emergence Engineering [1] is based on Genetic Programming which is a class of evolutionary algorithms for breeding programs, algorithms, and similar constructs. In GP, a population of individuals with a totally random genome is created. All individuals of the population are tested for fitness according to the objective functions.

A subsequent selection process filters out the individuals with low fitness and allows those with good fitness to enter the mating pool with a higher probability. In the reproduction phase, offspring is created by varying or combining these solution candidates and integrated into the population. If the termination criterion is met, the evolution stops here, otherwise the process is iterated, continuing by evaluating the new population.

2.1 Engineering Process

In principle, our engineering process consists of five phases:

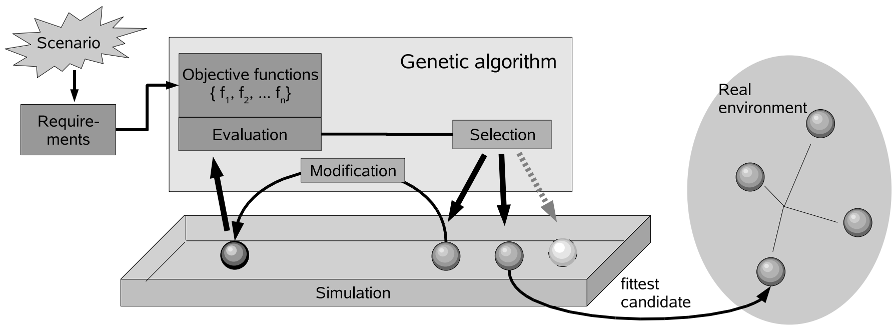


Fig. 1. The Offline Emergence Engineering Process

1. The scenario must be analyzed, resulting in a collection of requirements.
2. Suitable objective functions must be found which determine the fitness of a solution. These functions should allow for a gradual evaluation, not just be boolean.
3. The Genetic Algorithm repeatedly creates new versions of the individuals and selects the fittest ones.
4. At some time, an individual is picked out. If the Genetic algorithm converges, this individual will be most likely better suited than any individual before.
5. The individual is put into a component as its behavior and so deployed in the real environment.

To reduce complexity for the evolution process, we let only one type of agent behavior evolve, which must be put into all participating agents. The appropriate part of the behavior may be selected by means of a state variable. As we have only one type of individual (determined by its behavior), the selection pressure on the individuals is propagated to the rule set of the individuals. As the set of rules excerts some effect on the behavior of the agent, the individual behavior contributes to the overall behavior which can be measured by the fitness function. Hence, within OEE, we can witness emergence in different levels. The behavior of the group somehow emerges from the behavior of the individuals, and the behavior of the individuals emerges from the rule set.

2.2 Expressiveness of the Behavioral Language

Can we somehow make sure that this process eventually produces a behavior which is adequate to our scenario? Which are enhancing or limiting factors of this procedure?

– In order to approach this issue we examined the influence of the behavior language in our recent experiments. One potential reason for the process to fail is that the solution (as we can conceive it) requires some specific features, for example, sorting a list, but the language is unable to implement it.

Our OEE concept uses a behavioral description which features an unordered set of rules instead of sequentially executed instructions. We call this *Rule-based Genetic Programming* (RBGP). The RBGP language defines rules with a condition part, consisting of two conditions connected by *and/or*, and an effect part [1,2]. RBGP is powerful enough to express many of the constructs known from high-level programming languages, but it lacks important capabilities like indexed memory access. Thus, to extend the expressiveness, we created new primitives for indirect memory access, using the notation $[a(t)](s)$, which stands for the value of the $a(t)^{th}$ symbol (at time step $s = t$ or $s = t + 1$) in the ordered list of all symbols. This new form of Rule-based Genetic Programming named eRBGP allows the evolution of list-sorting algorithms and makes it Turing-complete.[1] eRBGP allows for conditions with more than two expressions, which enables the process to create more complex rules without the need for intermediate variables. Terms may be used as boolean expressions, being compared to zero.

```
========= Election (RBGP) ==========
false or (start(t)=incomingMsg(t)) => start(t+1)=1-b(t)
false or (b(t)>=a(t)) => a(t+1)=a(t)+id(t)
(out(t)<=start(t)) or (id(t)!=b(t)) => send
false or (a(t)!=out(t)) => out(t+1)=a(t)
(id(t)=0) and (out(t)>=0) => id(t+1)=id(t)/b(t)
(0=id(t)) or (id(t)<in(t)) => a(t+1)=in(t)

========= Election (eRBGP) ==========
id(t) => send
[incomingMsg(t)](t) => out(t+1) = id(t)
(out(t) - (incomingMsg(t) or [a(t)](t))) => a(t+1) = id(t)
(in(t) / id(t)) => id(t+1) = in(t)
```

Fig. 2. Comparing solutions from RBGP and eRBGP

As the expressions do not have a common structure anymore, we lose the ability of using Genetic Algorithms with fixed-size genes for its evolution. Instead we apply Genetic Programming with a tree-shaped genome, representing the hierarchical expression structure of the rules. The evolutionary operations modify the tree structure directly.

3 Experiments

We evaluated some experiments in order to determine preconditions for a successful emergence engineering. We have a look at two specific scenarios.

[1] The proof for Turing Completeness of Genetic Programming languages with indexed memory [3] can be easily adapted to the case of eRBGP.

3.1 Election

The challenge of an election procedure where one agent is elected as *leader* and all agents agree to this choice is the distributed nature: It must be ensured that all agents get a consistent view of the election. In our realization, an agent may send a message containing a single number stored in the variable **out** with the command **send**. Whenever a message is received, its contents appear in the symbol **in** and the variable **incomingMsg** is set to 1. The agents have unique identifiers (variable **id**) and two multi-purpose variables **a** and **b**. We expect the identifier of the elected agent to be stored in variable **a** after about 5000 simulated time units.

The objective functions are defined as follows: f_1 counts the number of different IDs found when comparing all the values stored in the **a** variables of the agents after the simulation and penalizes values that do not denote valid IDs. f_2 determines the behavior size in terms of the number of rules, f_3 counts the time units used for active computations (penalizing useless computation when the node could sleep instead), and f_4 counts the number of messages exchanged. For the best overall fitness, all four functions shall yield minimal values.

Some of the results look like well-known algorithms (one of them resembling *Message Extinction*), other are incomprehensible but obviously work. After using the RBGP process in the first run, we went for eRBGP in the second run. The solutions are shown in Figure 2. We evaluated the reliability of multiple solutions delivered by these two approaches by the fraction of scenarios where the election process proceeds correctly. Programs generated with RBGP are reliable in 60% of the scenarios and those from eRBGP achieve correct behavior in 91% of the network simulations. This seems to be an indication that the expressiveness of the language can help to find better solutions.

3.2 Critical Section (CS)

Code that accesses a shared resource is called *critical section*. Processes running concurrently on different nodes have to decide whether they are allowed to access the critical section or whether they have to wait, using message exchange to coordinate themselves.

The first objective function f_1 evaluates the number of violations of the mutual exclusion criterion. Concretely, to increase pressure, we sum up the square of the number of nodes inside the CS for each time step. The second objective function f_2 represents the number of times each process could enter the critical section *at least* in the fixed time span of the simulation. We add a value proportional to the total number of accesses of the CS. Finally, f_3 counts the number of rules and exerts pressure to drop unnecessary rules. f_1 and f_3 are subject to minimization, f_2 is to be maximized.

Unfortunately, the RBGP process did not converge to a specific set of adequate solutions, even after more than 2100 generations. We evaluated the behavior of the best individuals in 200 random network configurations. For RBGP, we found a solution avoiding collisions in 98.5% of all given environments. In 52.5% of the environments, the solution was also fair, allowing more than one process to enter the critical section. Our tests also showed that lower rates of collisions correlate with lower fairness. Involving more agents obviously increases the chances of triggering violations.

We compared solutions from the RBGP framework with the results of the enhanced eRBGP framework. With six rules compared to 15, the best solution we found is much more compact as the solution of RBGP. But while it fulfilled the first criterion (exclusiveness) in 99% of all simulations, it was highly unfair, allowing more than one agent to enter the CS in only 1% of all cases. It seems as if fairness was only achieved for that moment when it failed to ensure the exclusiveness – the evolution was obviously trapped in a suboptimal state.

4 Evaluation

In some scenarios like election, the engineering process may profit from the higher expressiveness, while in others, it fails to create better solutions. An important key to a productive usage of this engineering process is to find out early which scenario is well suited, and which is not. Analyzing our experiments, we found special cases for the result of the evolution as *any-agent* and *all-agent behaviors*. The former one refers to the observation that eventually, one of the agent instances has got some property or expresses some behavior. The latter case refers to the situation where eventually, the whole collective adops the same properties and behaviors.

Load balancing [1] and election seem to be suitable problems for an emergence engineering approach. We believe that this is due to the fact that both problems require *all-agent behaviors*. For load balancing, each agent migrates as soon as there is a host with lower load, while for election, all agents eventually share the same property at the end, namely knowing the ID of the winner.

Critical Section is more than an election problem, due to the fact that we also want to reach fairness. More specifically, the CS problem requires an iterative election, and it requires the behavior of the group to change in order to achieve fairness. This is neither an any-agent nor an all-agent behavior. Obviously, the agents somehow need to *learn* that some agent was already allowed to enter the CS, and so it does not qualify to enter again for some time. While it is not impossible that agents develop learning behavior with the evolutionary process, it is obviously fairly unlikely. Hence, for the applicability of this approach, the analysis must take care whether the group behavior implies learning capabilities or not.

5 Related Work

Cramer was the first one to utilize genetic algorithms and tree-like structures to evolve programs [4]. For agent systems, Genetic Programming is also a well-known approach in the context of foraging simulations [5] or rendezvous scenarios [6]. These concepts all address either optimization problems or solutions for specific problem areas. There is currently only few related work concerning emergence engineering for agent societies. The most notable work here is ADELFE [7] which is an engineering approach explicitly exploiting emergence among a set of cooperative agents. We call ADELFE an *online emergence engineering* approach, due to the fact that the agents are situated in the real environment and need to self-organize to respond to changes in the environment.

6 Conclusions

Our research is still in an early state where we need to conduct more experiments to find out characteristics of problems which may be suitably handled by this approach. However, we must face the fact that generating programs in this way is extremely time-consuming, literally taking days to weeks until a solution is found.

We have presented some simple criteria which may indicate whether EE is likely to produce suitable solutions for a problem. If we have considered a problem to be suitable for EE, we must find appropriate objective functions for the evolutionary algorithms to measure the fitness of individuals. Finally, we have to decide on the expressiveness of the behavioral language and the capabilities of the agents. We found that increasing the expressiveness of the implementation language of the agent behavior need not necessarily yield better solutions. Although we increased the scope of the possible agent behavior, some problems seem to trap the evolution in sub-optimal areas. We believe that defining some more scenarios will provide us with more experience on the behavior of the evolution process and whether it may be required to introduce additional modifications.

References

1. Zapf, M., Weise, T.: Offline emergence engineering for agent societies. In: Proceedings of the Fifth European Workshop on Multi-Agent Systems EUMAS 2007, Elmouradi Hotel, Hammamet, Tunisia (2007), `http://www.vs.uni-kassel.de/publications/2007/ZW07a`
2. Weise, T., Zapf, M., Geihs, K.: Rule-based genetic programming. In: Proceedings of 2nd International Conference on Bio-Inspired Models of Network, Information, and Computing Systems (BIONETICS 2007), December 10-13 (2007)
3. Teller, A.: Turing completeness in the language of genetic programming with indexed memory. In: Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, Piscataway, New Jersey, June 27–29, pp. 136–141. IEEE Press, Los Alamitos (1994)
4. Cramer, N.L.: A representation for the adaptive generation of simple sequential programs. In: Proceedings of the 1st International Conference on Genetic Algorithms and their Applications, Mahwah, NJ, USA, July 24-26, pp. 183–187 (1985)
5. Bennett, F.H.: Emergence of a multi-agent architecture and new tactics for the ant colony foraging problem using genetic programming. In: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior: From animals to animats 4, pp. 430–439 (1996)
6. Qureshi, M.A.: Evolving agents. In: Genetic Programming 1996: Proceedings of the First Annual Conference, pp. 369–374 (1996)
7. Bernon, C., Gleizes, M.-P., Peyruqueou, S., Picard, G.: ADELFE: A methodology for adaptive multi-agent systems engineering. In: Petta, P., Tolksdorf, R., Zambonelli, F. (eds.) ESAW 2002. LNCS, vol. 2577. Springer, Heidelberg (2003)

On the Use of Linear Programming in Optimizing Energy Costs

Fahad Javed and Naveed Arshad

LUMS School of Science and Engineering Lahore, Pakistan
fahadjaved@lums.edu.pk, naveedarshad@lums.edu.pk

Abstract. Efficient energy consumption in large sets of electric devices is a complex problem since it requires a balance between many competing factors. Presently, self-optimization techniques work expeditiously on small and relatively less complex problems. However, these techniques are not shown to be scalable on large and complex problems. In this paper we have used linear programming to optimize the use of energy in a typical environment that consists of large number of devices. Our initial results show that LP is fast, predictable and scalable. Moreover, we have also observed that modeling in LP is quite simple as compared to other self-optimization techniques.

1 Introduction

Self-optimization, the goal of enabling a system to autonomically optimize itself, necessitates a methodology that is able to handle input domain for any given system. This requires an ability to handle a hyper-dimensional variable space involving hundreds of variables with complex relationships.

Some of the recent approaches for self-optimization problems have used traditional methods such as control theory [5] [3] to optimize a given system. However these solutions are limited as it has not been shown that these techniques can scale to hyper-dimensional variable space or handle the complex relationships appropriately.

In contrast our interest centers around the problems of self-optimization for large and complex systems involving hyper-dimensional input and output variable space with complex relationships. We approach this problem using linear programming to find the extremum of the system.

In this approach we have used linear programming to optimize the power consumption of a heterogenous system of machines under variable demand. Our initial results have shown upto 80% savings.

Our contributions in this paper are:

1. A scalable methodology for self-optimizing a system which involves hyper-dimensional variable input and output space as long as the system is linear or can be interpolated to linear domain.
2. A unique time-variate modeling schema for planning with linear programming.

3. Identification of a class of problems where optimization techniques such as linear programming could be used.

2 Related Work for Cost Optimization

Efficient energy consumption is a critical issue that has become the focus of academia, industry and general public in recent times. This has roots in ecological as well as cost management issues. Electric consumption for commercial entities such as cyber-cafes, server farms and other facilities where a considerable number of computers are installed face problem of optimization of cost of operations.

A typical cost of operations in a lab has two mostly conflicting components. The first component is the energy cost incurred by running the machines. The second component is the cost of repairs. Both these costs have a weekly inverse relationship. As it is observed that frequent restarts of machines is one of the most common cause for breakages in machines. Labs, including managed by our own IT department, would rather keep the machines "on" rather than pay for costly breakages. The machines are scheduled to go to "hibernate"mode if machine is the machine is not used for a given period of time. However "hibernate" cost is usually around 20% of a typical "on" cost. There is no methodology available that balances the growing cost due to hibernate with the cost of breakages for a more robust systrm.

Optimizing a system consisting of homogenous machines with or without indistinguishable power consumption profiles has been proposed in previous work [4], [3] [5] . But our target system, computer labs usually employ computers with varying configurations, demands and power profiles. Therefore we need a more robust approach that is able to handle the extra requirements.

Our technique for optimization is a balance between [1] and [2], [4]. Almeida and colleagues provides a heuristics based mathematical technique which is scalable but does not guarantee an optimum solution and is extremely complex to grasp[1]. In contrast Nathuji and colleagues used a simplistic greedy algorithm to balance power in a heterogeneous data-center environment[4]. This was possible due to the nature of problem which allowed mapping of input to a single variable domain thereby making greedy algorithm a possibility. Although Femal and Freeh used LP to derive an optimal solution for boosting performance[2], it can be shown that a greedy technique would have been more suitable to find the solution.

We felt that a sizeable number of problems in the autonomic computing domain can be handled by a far less complex system than [1] and with a much better guarantee. At the same time not all problems can be mapped to an input domain which is solvable through greedy algorithms. Our optimization technique targets the class of problems that lies between these two extremes.

3 Approach

Our approach to reduce the operation costs of an environment which consists of large number of machines is based on a hypothesis. This hypothesis states that

$ON_{i_{tj}}$	i type of machine on at time j
$OFF_{i_{tj}}$	i type of machine off at time j
$HIBERNATE_{i_{tj}}$	i type of machine hibernating at time j
$SWITCHON_{i_{tj}}$	i type of machine to be switched on at time j

Fig. 1. LP Variables

although a restart cycle has a cost component, and a "hibernate" and "on" also has a cost component. However, there is a an optimum point which balances these two competing cost components. In our approach we take this optimum point to be the reduction of the total cost of operations of the environment.

To model such a system our input domain includes various classes of machines with their demands and costs. Our output domain is the number of machines in "on" "off" or "hibernate" state for each time period. We applied linear programming (LP) to our problem to find the optimum utilization of such an environment based on the reduction of the total cost of operations.

3.1 Modeling in Linear Programming

An LP model requires a series of linear equations constraining the problem domain and an optimization function. Our cost function defines the optimization function and the constraint equations are discussed shortly.

Our system consists of various classes of machines. Each machine has states and each state carries a cost per unit time. Our explicit objective is to reduce cost but implicitly we require a solution that meets our demands for each time period and also satisfies the relational constraints such as the demands for a particular class of machines. The variables for our model are given in figure 1.

We will achieve optimization by reducing the two cost components over a period of time. LP selects the least cost combination of machines which will satisfy the requirements (equation 1). The decision that we are interested in is which machines to shut down to minimize the cost. The requirement here are the demands for each usage class for a specific time period (equation 3). We are bounded by the number of machines for each class (equation 2). We put it all together, the "on", "off", "hibernate" and "switch-on" costs, in our optimization function as given in figure 2 equation 1. To complete the model we added the non-negativity constraints for all the variables (not shown in the figure). This is because in our system a negative *on* or off machine does not make sense .

4 Evaluation

We ran several simulations using equations in figure 2 on data collected from one of our labs and compared it with simulated cost of other techniques available. We ran our tests on a week's data. We considered a window of 24 hours and compared our LP results with the existing setup. In our tests we assumed that we will have knowledge of usage for each class priori.

$$z = \sum_{i,j} ON_{i_{tj}}.cost_{On_i} + \sum_{i,j} HIBERNATE_{i_{tj}}.cost_{Hibernate_i} + \sum_{i,j} SWITCHON_{i_{tj}}.cost_{Restart_i} \tag{1}$$

$$\forall_{i,j}\{ON_{i_{tj}} + HIBERNATE_{i_{tj}} + OFF_{i_{tj}} = supply_i\} \tag{2}$$

$$\forall_{k,j}\{\sum_{i:X_i \subset S_i} ON_{i_{tj}} \geq demand_{ktj}\} \tag{3}$$

$$\forall_i \forall_j \{ON_{i_{tj+1}} + HIBERNATE_{i_{tj+1}} - ON_{i_{tj}} - HIBERNATE_{i_{tj}} \leq SWITCHON_{i_{tj}}\} \tag{4}$$

Fig. 2. LP Equations

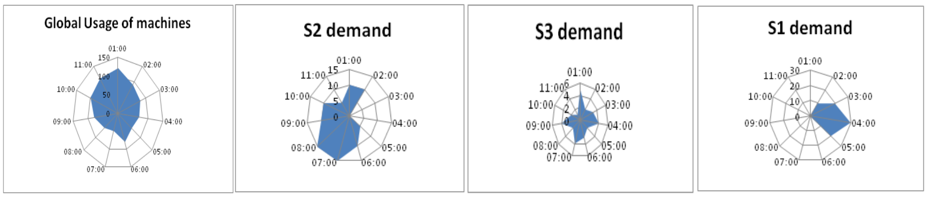


Fig. 3. Demand pattern

4.1 EnergySaver++

Our application, EnergySaver++, uses LP to provide a plan for the number of machines to shutdown or hibernate in each 1 hour period over a period of 24 hours. We classify the machines in our lab according to their power consumption profile and configuration of each system.

The power consumption profile is derived by calculating the power needed for the CPU, monitor and periphery devices for the three states concerned, namely: *On*, *Hibernate* and the breakage cost due to restart. Breakage cost of restart is calculated by finding the average cost of repair for machines which were diagnosed with power fluctuation related faults divided by number of times an average machine was rebooted in the past one year.

Our second classification criterions, for configuration classes, are the special software or hardware installation on machines for which students make special demands. Due to various issues some facilities are available only at specific number of machines. We classified machines based on these special configurations and measured the usage of the "special" resource for each system configuration class.

Using the above two classification methods, we define our set of general classes X for LP. X is defined as partitions of U created by C and intersections of C and S. Here U is the set of all the machines, $\{C_i \subset C : C_i$ is a consumption class$\}$ and $\{S_i \subset S : S_i$ is a usage class$\}$ Consumption classes are disjoint classes as a computing can belong to only a single profile. Whereas usage classes can overlap one or more consumption classes.

In our initial test-bed consists of 120 machines. There were 6 different power consumption profiles for those machines. We identified 3 configuration classes.

Table 1. Cost function of machine classification

State	On						Hibernate						Restart					
Class	C1	C2	C3	C4	C5	C6	C1	C2	C3	C4	C5	C6	C1	C2	C3	C4	C5	C6
Cost	10	5	10	15	20	20	2	1	4	2	3	2	10	16	18	18	10	17

Table 2. Daily cost of operations for alternative methods and percentage saving in comparison to LP method

Plan	day1		day2		day3		day4		day5		day6		day7	
	Cost	Savings	Cost	Saving	Cost	Saving	Cost	Saving	Cost	Saving	Cost	Saving	Cost	Saving
Alwayon	3.36	62%	3.36	61%	3.36	61.6%	3.36	62.0%	3.36	62.2%	3.36	80%	3.36	81%
Hibernate	1.56	19%	1.58	18.2%	1.57	18.3%	1.56	18.7%	1.56	18.9%	1.13	42%	1.11	43.3%
1 hour off	1.86	32%	1.89	31%	1.88	31.6%	1.86	31.8%	1.87	32.4%	1.36	52%	1.36	53.4%

For LP model, with overlapping of consumption classes, we had 10 (X) classes. We collected historical data to predict the workload. Table 1 defines the cost for each class for each of the 6 consumption profile classifications.

We collected the usage of systems by noting the logon/ logoff times for machines. A second script noted if a special resource was used in the last hour. We observed that the usage of machines was highly cyclic. In-fact the usage repeats itself after 7 days. We predicted demand for the 24 hour period for each time period as shown in Fig 3. The global demand is for a general purpose machines where as demands $S1$, $S2$ and $S3$ corresponds to the demand for Matlab, SPSS, and scanner respectively for each time period.

We evaluated our system by comparing the cost of operations incurred by our system with the cost incurred by the three plans provided by OS vendors. Namely: Keep machines always in on state or Hibernate machines after x minutes ($x < 60$) or Switch-off machine if machine has been in hibernate state for 1 hour.

We used data for 7 days of a week and applied LP based planning. We simulated the three plans that are used for optimizing power consumption on the same 7 day dataset. Table 2 shows the result of the cost of operations that these systems incurred given the same usage pattern. The table also shows, in savings column, the percentage savings that our LP based system attains in comparison.

We observed upto 80% savings in comparison to an always on policy for weekend days (day 6 & 7) and upto 43.3% and 53.4% savings for hibernating plan and switching off after one hour of hibernate plans, respectively.

For weekdays(day 1 - 5), we saw a maximum of 62% savings for always on and up to 19% and 32% savings for hibernating plan and switching off after one hour of hibernate plans respectively.

Since our lab was using policy where the machines switched to hibernate, our net savings were 2.43 units or 24%. This means that we are able to slash our costs by 1/4th by using an LP based planner.

Our reason for using LP was scalability. To test our claim that LP is scalable, we evaluated our application against a variety of input parameters. We generated random data to set up scenarios. Our variation in data can come from change in number of machine and change in the classes of machines. These parameters

Table 3. Scalability results of LP (time in seconds)

Classes	10			50			100		
Machines	100	1000	10000	100	1000	10000	100	1000	10000
Running Time	0.23	0.12	0.13	0.5	0.53	0.51	1.48	1.38	1.38

and their results are shown in table 3. It could have been inferred from the LP equations themselves that a change in the number of machines only will not effect the running time. We can see this in our simulations as well. The time for 100, 1000, and 10000 machines is nearly same for the different classes.

5 Future Work and Conclusion

In this paper we have provided the groundwork for a possible future direction in self-optimization. We have described the issues with using traditional methods for self-optimization.

We have provided an alternative approach for self-optimization by using conventional mathematical techniques. As a sample we have described a hyper-dimensional problem, EnergySaver++, and applied linear programming to derive a solution which is optimal and is derived in polynomial time.

Our future direction is two pronged. First we will build on this work to formulate a better way to handle prediction errors and current load transitions by using some iterative methods on LP. Second, we will try to provide a solution for non-linear time variant system planning.

Our targeted system for these future direction is to manage the power supply at the metropolitan level which will optimize the power consumption while maintaining a basic quality level for the users.

References

1. Almeida, J., Almeida, V., Ardagna, D., Francalanci, C., Trubian, M.: Resource management in the autonomic service-oriented architecture. In: IEEE International Conference on Autonomic Computing, 2006. ICAC 2006, June 13-16, pp. 84–92 (2006)
2. Femal, M.E., Freeh, V.W.: Boosting data center performance through non-uniform power allocation. In: Second International Conference on Autonomic Computing, 2005. ICAC 2005. Proceedings, June 13-16, pp. 250–261 (2005)
3. Lefurgy, C., Wang, X., Ware, M.: Server-level power control. In: Fourth International Conference on Autonomic Computing, 2007. ICAC 2007, June 11-15, p. 4 (2007)
4. Nathuji, R., Isci, C., Gorbatov, E.: Exploiting platform heterogeneity for power efficient data centers. In: Fourth International Conference on Autonomic Computing, 2007. ICAC 2007, June 11-15, p. 5 (2007)
5. Wang, M., Kandasamy, N., Guez, A., Kam, M.: Adaptive performance control of computing systems via distributed cooperative control: Application to power management in computing clusters. In: IEEE International Conference on Autonomic Computing, 2006. ICAC 2006, June 13-16, pp. 165–174 (2006)

Instantiation of a Generic Model for Load Balancing with Intelligent Algorithms

Vesna Sesum-Cavic and Eva Kühn

Vienna University of Technology, Institute for Computer Languages,
Space Based Computing Group, Argentinierstraße 4, 1040 Wien
{vesna,eva}@complang.tuwien.ac.at

Abstract. In peer-to-peer networks, an important issue is the distribution of load having an impact on the overall performance of the system. The answer could be the application of an intelligent approach that leads to autonomic self-organizing infrastructures. In this position paper, we briefly introduce a framework model for load balancing that allows various load-balancing algorithms to be plugged-in, and that uses virtual shared-memory-based communication known to be advantageous for the communication of autonomous agents in order to enable the collaboration of load-balancing agents. As the main contribution, we show how the biological concepts of bees can be mapped to the load-balancing problem, explain why we expect that bee intelligence can outperform other (un)intelligent approaches, and present an instantiation of the model with the bee intelligence algorithm. This load-balancing scheme focuses on two main policies: a transfer and a location policy for which we suggest some improvements.

Keywords: load balancing, bee intelligence, autonomous agents.

1 Introduction

The significance of Load Balancing (LB) in distributed systems is well known. The goal is to obtain an improved performance of the complete system. Dynamic LB algorithms can contribute towards efficiently distributing a load at run time, i.e., appropriately transferring work from heavily loaded nodes to idle or lightly loaded nodes.

Many different approaches cope with the LB problem. We will briefly enumerate the most important ones and classify them. The first group consists of different conventional approaches that do not make use of any kind of intelligence; for example, sender- and receiver-initiated negotiation [25], a gradient model [19], random algorithm [10], diffusion algorithm [7], hierarchical approaches [22], an economic-based model [2], and game theoretical approaches [11]. The second group includes theoretical improvements of LB algorithms using different mathematical tools and estimations [3]; however, with a lack of implementations and real benchmarks. The third group contains approaches that use intelligent algorithms: an evolutionary approach [5] and an ant colony optimization approach [12]. These intelligent approaches successfully cope with LB. Nevertheless, the improvement of several issues (scalability, the quality of the solution, general model, flexibility, etc.) is still open. In a general investigation of non-pheromone-based algorithms (bee intelligence), in order to compare them to pheromone-based algorithms, Lemmens et al. [18] concluded that the former were significantly more efficient in finding and collecting food and also that they were more scalable.

Our intention is to build a general LB self-organized architecture with exchangeable and combinable algorithms. The communication is based on the shared data model supported by space-based architecture [15]. We have developed such a general model for LB [17]; and now, as the second step, we propose to apply an intelligent algorithm to this model. In this paper, we propose an adaptation of bee intelligence for the realization of LB policies. In Section 2, we give a short summary of a general LB model. Section 3 contains a description of the bee intelligence and the motivation why to use it for LB. Section 4 describes algorithms and the mapping to the LB problem. Section 5 concludes this position paper with an outlook for our future work on this topic.

2 Overview of the SILBA Model

SILBA [17] is a model for LB and stands for "self-initiative load-balancing agents". Instead of having a central coordinator, autonomous agents exist in SILBA. They operate in a network with a dynamically changing amount of nodes, and decide on their own when to pick up or push back work, providing the necessary basis for self-organization. The main idea is to have a framework that allows the plugging of different LB algorithms for comparison. The benchmarks are instantiated with different algorithms, policies and parameters. Moreover, an execution environment for SILBA exists which uses shared data structures[1] for the collaboration of the LB worker agents. This indirect communication allows for more autonomy of the agents, [23] and [13]. The shared data structures are the Internet addressable resources [16]; they maintain collaboration information like pheromones, overlay topology, and other LB relevant parameters. The concurrently running agents either retrieve, or subscribe to this information being notified in near-time about changes, or modify it. Requestors continuously put tasks to any node in the P2P infrastructure, i.e., to a so-called shareable task-list data structure at this node.

In a first implementation, [24] and [17], we have populated the model by not yet intelligent algorithms and have compared it with a centralized approach [14], demonstrating that autonomously deciding agents can improve performance and scalability. The different benchmarks were promising and showed that the agents based approach achieved better results in all test examples (about 27% on average).

In this paper, we suggest how to extend the LB algorithms towards intelligent ones, based on biological self-organized systems. An LB algorithm consists of policies. Every policy has its own autonomous goal. The most important policies are the transfer policy (TP) and the location policy (LP). TP determines whether (and in which form) a resource should participate in load distribution and in that sense, how the classification of resources is done, using certain parameters (described below); whereas the LP determines a suitable partner of a particular resource for LB [25]. The LB algorithm regulates the coordinated interplay of these policies. The parameters of TP

[1] We have implemented the data structures with the XVSM shared data space (www.xvsm.org).

that we want to consider in this paper are the two thresholds T_1 and T_2 that are continuously adapted by each node. They allow for a classification of nodes according to their current load into three groups: under-loaded nodes (UL) with load $\leq T_1$, OK nodes with a load between T_1 and T_2, and over-loaded nodes (OL) with load $\geq T_2$.

3 Motivation for Bee Intelligence Algorithm

As mentioned in Section 1, [18] investigated swarm algorithms and their properties, and concluded that bee-inspired algorithms outperform ant algorithms when finding and collecting food, and are more scalable as they require less computation time to complete a task. These facts motivate us to apply bee algorithms to the LB problem.

3.1 Biological Background

The biological background of bee behavior is characterized by autonomy and distributed functioning, and self-organization [4]. A honeybee colony of one hive contains bees with different roles: foragers, followers, and receivers. Self-organization of bees relies on two main strategies, *navigation* and *recruitment*. The navigation strategy is concerned with searching for nectar of flowers in an unknown landscape. A *forager* scouts for a flower with good nectar and after finding and collecting, it returns to the hive and unloads the nectar. Afterwards, the forager performs a recruitment strategy (a so-called "waggle dance"), meaning that it communicates the knowledge about the visited flowers to other bees. This serves to inform other members of the hive about the quality, distance and direction of the found nectar [4]. A *follower* chooses to follow a forager at random and visit the flower that has been "advertised". It does not need a decision about navigation on its own and therefore, is more efficient. A forager can choose to become a follower in the next step of navigation, if it observes better information about nectar (through the recruitment process of some other forager), i.e., foragers and followers can change their roles. A receiver always stays in the hive (stationary) and processes the nectar.

Bee-inspired algorithms have been applied to several computer science problems like Traveling Salesman Problem [26], Job Shop Scheduling [6], Routing and wavelength assignment in all-optical networks [20].

3.2 Motivation for Using Intelligent Algorithms

Different dynamic processes characterize the LB scenario. Nodes can dynamically join and leave, the information about load changes permanently, and tasks are dynamically added and continuously processed. Employing P2P overlay networks [1], a *structured* P2P network has an overlay topology that is controlled; there is a mapping from content to a location, and therefore it scales well. At the other hand side, the support of dynamics is not so good; the queries can be only simple key/value mappings, i.e., exact match queries instead of more complex queries. From these reasons, they are not suitable for the LB problem at hand. In an *unstructured* P2P network, a placement of information can be done independently of an overlay topology (i.e., unstructured), but the content must be localized explicitly (for example, through brute force mechanisms or flooding – cf. Gnutella). It is very well suitable for dynamic populations, and queries that are more complex are possible. Therefore, an unstructured P2P network fits better to our problem. The negative point is that it does not scale so well, which is the starting point for an improvement.

In order to point out the arguments for the potential of using bees for the LB problem, we give a short comparison of Gnutella and swarm-based systems:

Gnutella [1] operates on a query flooding-based protocol. For the communication between servants, it supports the following types of messages: ping (discover hosts), pong (reply to ping), query (search request), and query hit (reply to query). Gnutella uses an unintelligent flooding to find a particular node. It needs many concurrent agents for one (exhaustive) search, as for each branch, a new agent is required.

Bees [18] search the network and build up routes as overlays. If a bee finds the required information, it directly flies back to its hive, i.e. it informs the "starting place" of the search directly in a P2P way, like in Gnutella. Bounding the number of bees is possible, and this is one indication that this algorithm would scale better than Gnutella. However, in the first iteration step, there is no guarantee to find a solution, but one will find a solution in upcoming iterations through learning and sending further bees. Knowledge distribution takes place in the own hive. Bees of different hives do not communicate with each other, and bees that are out of their hive do not communicate with other bees.

Ants [9] leave information (pheromones) at all nodes on the backward trip. We can say that the forward trip is comparable to the bees' forward movement (navigation), but the backward trip is different – the ant does not directly contact the "starting place" in a P2P way but must go the entire way back.

4 Mapping Bee Intelligence to the LB Problem

In SILBA, software agents act in swarms, inspired by bee colonies [18] and [26]. An *agent* can play the role of a *bee*[2] and reside at a particular node. In the following, we assume that the number of bees will not change.

We define a *node* to consist of exactly *one hive* (incl. bees) and *one flower* (incl. nectar) in its environment. We abstract the existence of many flowers to one flower that can contain many "nectar units". For each hive and flower, it is deterministic to which node it belongs. In addition, we assume that a bee can take nectar units out of flowers and can transfer it between flowers.

A *task* is *one nectar unit* in a flower. Note that there can be n ($n \geq 0$) nectar units in a flower; if a flower is empty it is not removed from the system. A new task can be put at any node in the network. We assume that all nodes are addressable (cf. task container of the node [17]).

Initially, a hive has k ($k \geq 1$) stationary (= receiver) bees and l ($l \geq 1$) outgoing (i.e., forager plus follower) bees. We assume that these numbers are static. If we allow for dynamic addition and removal of bees, this would only influence how to determine the role of a forager and a follower, but it has no influence on the navigation part of the LB algorithm. Initially, all outgoing bees are foragers. Foragers scout for an LP

[2] For brevity, instead of using an "agent with the role of a bee", we speak of a "bee".

partner node (to pull, or to push to nectar to it) of their node, inform followers, and finally recruit followers. Receivers process tasks at their node and have no influence on the algorithm itself. The main actors are foragers and followers. The goal is to find the *best* LP partner node by taking the *best* path. In our case, the best path is the shortest path. We define the best LP partner with a suitability function δ(see below).

A navigation strategy can be realized in two ways or in a combination of both: 1) process migration of the software agent, or 2) the software agent can remotely contact the information stored at another site as the SILBA uses shared data structures that offer internet addressable URLs. Note that the SILBA execution environment hosts a software agent as a thread, which can only communicate with the environment using shared data structures that it can reach from any node. Therefore, a "process migration" is not a costly operation – which is crucial to avoid unnecessary overhead. We propose to use a state transition rule, adopted from Wong et al. [26], for the navigation strategy of a bee to decide which node to visit next.

The recruitment strategy leads to building up a knowledge base in the own hive through aggregation of knowledge in order to derive the behavior of further bees. During the recruitment, bees communicate using the following parameters: i) path (distance), and ii) quality of the solution. From these, we can derive a fitness function f_i for a particular bee i as $f_i = \frac{1}{H_i}\delta$, where H_i is the number of hops on the tour, and δ is the suitability function of the solution. Let n be the number of outgoing bees. Then we can compute the colony's fitness function as the average of all single bees' fitness functions: $f_{colony} = \frac{1}{n}\sum_{i=1}^{n} f_i$. f_i evaluates to a good value, if a bee finds a highly suitable LP partner node while using a "fast route" for traveling. After a trip, an outgoing bee determines how "good it was", by comparing its result with the average value, and based on that decides its next role. For example, if the bee's success is low, compared to the average fitness function of the colony [21], then the bee will become a follower. The success of a bee affects the credibility of its recruitment, expressed as a quotient between f_i and f_{colony}. As both UL node and OL node can start LP, we would like to make a general analysis of two cases: In the first case, a bee of a UL node searches for a suitable task belonging to some OL node. The bee carries the information about how complex a task the node can accept. In the second case, a bee of an OL node searches for a UL node that can accept one or more tasks from this OL node. It carries the information about the complexity of tasks this OL node offers and compares it with the available resource of the current UL node that it just visits. We assume that the information about UL and OL is available to each bee visiting a node (through shared data structures published at this node [17]).

In both cases, we have to compare the complexity of the task with the available resources at a node. For this purpose, we need the following definitions: a task complexity c, a host load hl and a host speed hs [8]. A host speed represents the speed of the host and its value is relative in a heterogeneous environment. A host load represents the fraction of the machine that is not available to the application. Finally, a task complexity c is the time necessary for a machine with $hs = 1$ to complete a task when $hl = 0$. We can calculate the argument x of the suitability function δ as follows:

$$x = \frac{\frac{c}{hs}}{1 - hl}.$$

We define the suitability function (which e.g. can be a linear function) as $\delta = \delta(x)$. If $x = 1$, then we have an ideal situation. The main intention is to find a good LP partner. For example, when a UL node with high resource capacities is taking work from an OL node, a partner node offering tasks with small complexity is not a good partner as other nodes could perform these small tasks as well; taking them would mean to waste available resources. Referring to the SILBA model [17], all mentioned parameters that contribute to the bee algorithm, are configurable.

5 Conclusion

In this position paper, we have proposed an adaptation of bee intelligence for the realization of location policies to improve the overall load balancing (LB) in a system. In previous work, we have shown a decentralized LB based on active agents, realized without intelligent algorithms. Benchmarks were carried out in previous work too, with a general LB framework called SILBA, and have already shown promising results compared to the centralized approach, demonstrating that autonomously deciding agents can improve the performance and scalability. Therefore, we propose keep the idea of autonomy and expand it further towards intelligent algorithms. In this paper, we have argued why bee algorithms can be more suitable than unstructured P2P networks and ant algorithms for a highly dynamic problem like the LB scenario, and have show a possible instantiation of such an algorithm and its policies.

In our further work, we will include the implementation of the proposed algorithms into SILBA and perform benchmarks. The comparison will comprise the instantiation of SILBA with other intelligent approaches as well as with Gnutella. Further work will also consider a comparison of the algorithms with regard to security issues, as the agents in the network must be trustworthy.

Acknowledgments. We would like to thank Martin Murth and the anonymous reviewers for the many valuable comments to this paper.

References

1. Androutsellis-Theotokis, S., Spinellis, D.: A survey of peer-to-peer content distribution technologies. ACM Comput. Surv. 36, 335–371 (2004)
2. Backschat, M., Pfaffinger, A., Zenger, C.: Economic-Based Dynamic Load Distribution in Large Workstation Networks. In: 2nd Int. Euro-Par Conf. on Parallel Processing, France, pp. 631–634 (1996)
3. Bronevich, A.G., Meyer, W.: Load-balancing algorithms based on gradient methods and their analysis through algebraic graph theory. Parallel and Distr. Comp. 68, 209–220 (2008)
4. Camazine, S., Sneyd, J.: A model of collective nectar source selection by honey bees: Self-organization through simple rules. J. of Theoretical Biology 149(4), 547–571 (1991)
5. Chen, J.C., Liao, G.X., Hsie, J.S., Liao, C.H.: A study of a contribution made by evolutionary learning on dynamic load-balancing problems in distributed computing systems. Expert Systems with Application 34, 357–365 (2008)

6. Chong, C.S., Sivakumar, A.I., Low, M.Y., Gay, K.L.: A bee colony optimization algorithm to job shop scheduling. In: Proc. of the 38th Conf. on Winter Simulation, California, pp. 1954–1961 (2006)
7. Cortes, A., Ripoll, A., Cedo, F., Senar, M.A., Luque, E.: An asynchronous and iterative LB algorithm for discrete load model. Parallel and Distr. Comp. 62, 1729–1746 (2002)
8. Da Silva, D.P., Cirne, W., Brasileiro, F.V., Grande, C.: Trading Cycles for Information: Using Replication to Schedule Bag-of-Tasks. In: Kosch, H., Böszörményi, L., Hellwagner, H. (eds.) Euro-Par 2003. LNCS, vol. 2790, pp. 169–180. Springer, Heidelberg (2003)
9. Dorigo, M., Di Caro, G., Gambardella, L.: Ant colony optimization: A new meta-heuristic. In: Proc. of the Congress on Evolutionary Computation, USA, vol. 2, pp. 1470–1477 (1999)
10. Eager, D.L., Lazowska, E.D., Zahorjan, J.: Adaptive Load Sharing in Homogeneous Distributed system. IEEE Trans. on Software Engineering 12(5), 662–675 (1986)
11. Grosu, D., Chronopoulos, A.T.: A Game-Theoretic Model and Algorithm for Load Balancing in Distributed Systems. In: APDCM 2002, USA, pp. 146–153 (2002)
12. Ho, C.K., Ewe, H.T.: Ant Colony Optimization Approaches for the Dynamic Load-Balanced Clustering Problem in Ad Hoc Networks. In: Swarm Intelligence Symp., Hawaii (2007)
13. Huang, Y., Garcia-Molina, H.: Publish/Subscribe in a Mobile Environment. In: 2nd Int. Workshop on Data Engineering for Wireless and Mobile Access, USA, pp. 27–34 (2001)
14. Kraus, K.: Development and Evaluation of a Load Balancer Based on Corso (in German), Praktikum, Institute for Computer Languages, TU Wien (2004)
15. Kühn, e.: Virtual Shared Memory for Distributed Architecture. Nova Science (2001)
16. Kühn, e., Mordinyi, R., Schreiber, C.: An Extensible Space-based Coordination Approach for Modeling Complex Patterns in Large Systems. In: Proc. 3rd Int. Symposium on Leveraging Applications of Formal Methods, Verification and Validation, Greece, October 13-15 (2008)
17. Kühn, e., Šešum-Cavic, V.: A Model for Self-Initiative Load Balancing Agents with Support for Swarm Intelligence and Genetic Algorithms (submitted for publication) (2008)
18. Lemmens, N., de Jong, S., Tuyls, K., Nowe, A.: Bee Behaviour in Multi-agent Systems. In: Tuyls, K., Nowe, A., Guessoum, Z., Kudenko, D. (eds.) ALAMAS 2005, ALAMAS 2006, and ALAMAS 2007. LNCS, vol. 4865, pp. 145–156. Springer, Heidelberg (2008)
19. Lin, F.C.H., Cellars, R.M.: The gradient of modelling Load-balancing Method. IEEE Trans. on Software Engineering 13(1), 32–38 (1987)
20. Markovic, G., Teodorovic, D., Acimovic-Raspopovic, V.: Routing and wavelength assignment in all-optical networks based on the bee colony optimization. AI Commun. 20(4), 273–285 (2007)
21. Nakrani, S., Tovey, C.: On honey bees and dynamic server allocation in the Internet hosting centers. Adaptive Behaviour 12(3-4), 223–240 (2004)
22. Pollak, R.: A Hierarchical Load Balancing Environment for Parallel and Distributed Supercomputer. In: Int. Symposium on Parallel and Distr. Supercomputing, Japan (1995)
23. Rodrigues, J.A.N., Monteiro, P.C.L., de Oliveira Sampaio, J., de Souza, J.M., Zimbrao, G.: Autonomic business processes scalable architecture. In: Business Process Management Workshops, pp. 78–83 (2007)
24. Rohner, M.: Load Balancing for Grid Computing (German), dipl. thesis, TU Wien (2005)
25. Shivaratri, N.G., Krueger, P.: Adaptive Location Policies for Global Scheduling. IEEE Trans. on Software Engineering 20(6), 432–444 (1994)
26. Wong, L.P., Low, M.Y.H., Chong, C.S.: A Bee Colony Optimization for Traveling Salesman Problem. In: 2nd Asia Int. Conf. on Modeling & Simulation, Malaysia, pp. 818–823 (2008)

On Properties of Game Theoretical Approaches to Balance Load Distribution in Mobile Grids

Karin Anna Hummel and Harald Meyer

Department of Distributed and Multimedia Systems
University of Vienna, Austria
{karin.hummel,harald.meyer}@unvie.ac.at

Abstract. Mobile devices can be integrated into grids to access grid resources but also to provide resources, such as CPU cycles or memory, building *mobile grids*. To exploit the potential of mobile grids, we propose an opportunistic job scheduling approach to harness cycles among mobile devices. Mobile nodes decide autonomously and locally which job to take by matching the job's requirements against their capabilities and coordinate with one another by means of shared job queues. A prototype implementation has been presented in previous work. In this work, we introduce selfish nodes which are expected to occur among mobile devices with limited energy sources. To react to selfishness, we introduce three strategies of game theory, that are, Tit For Tat (TFT), generous TFT, and Go-By Majority (GBM), to our approach and investigate the emerging behavior of the system by means of simulation. First results show, that the TFT and GBM implementations converge fast to fully selfish systems, while generous TFT exhibits self-healing characteristics.

Keywords: Self-organization, Mobile Grids, Game Theory.

1 Introduction

Mobile devices are ubiquitously available and allow the spontaneous construction of mobile distributed systems which can share their processing power and memory. Although connected mobile devices cannot compete with high performance computing clusters or desktop grids, they still may support the sharing of computing cycles. The envisioned *mobile grids* are expected to be either integrated into stationary grid infrastructures or consist of rather small ad-hoc networked systems useful for mobile communities or mobile field work scenarios which occur in domains like archeology, ecology, or biology, where pre-analysis of data might be desirable.

In accordance with Satyanarayanan [1], among the most important challenges for mobile distributed computing are lower wireless network bandwidth when compared to wired networks and varying link quality, limited capabilities of devices and energy sources, and varying availability of resources caused by mobility. Due to these characteristics a fully decentralized scheduling approach has been chosen to assure robustness in particular in spite of disconnection failures.

In principle, mobile nodes decide locally which jobs to take depending on their capabilities and the jobs' requirements and coordinate among each other using shared job queues to keep track of a job's status. The shared queue management is supported by a distributed Virtual Shared Memory (VSM). We described a prototype implementation and the achievable speedup in mobile test scenarios (intermittent connectivity) in previous work [2]. Based on the promising results, we can now further investigate how to achieve fairness, energy efficiency, and a low communication overhead in mobile grids. Additionally, due to possibly selfish nodes, it cannot be assumed that cooperation will automatically emerge – which will be addressed in this paper.

We describe and investigate strategies that allow mobile nodes to adapt to selfish behavior of other nodes. The basic scheduling approach and related work (Section 2) is extended by applying strategies of game theory to job scheduling in mobile grids (Section 3). Depending on the strategy and the behavior of other mobile nodes, individual nodes either *cooperate* with one another by taking jobs or *defect* from each other by declining jobs. Each node's decision is communicated to other nodes residing in the same group. The performance of the strategies is analyzed in stationary situations under varying group sizes as well as the spreading of defecting nodes over time for each strategy based on a discrete event simulator (Section 4). We conclude our work and present an outlook on future work in Section 5.

2 Related Work

Distributed job scheduling among mobile clusters and grids has recently attracted several research efforts, addressing challenges such as energy efficiency [3] and intermittent connectivity. Due to limited resources (energy, bandwidth, etc.) user preferences may lead to an unfair load distribution in mobile grids. The situation worsens, in cases where nodes act both as grid clients and grid resources from time to time but refuse to contribute resources.

Related to our scheduling approach, in [4], job scheduling is proposed via agents that individually decide which tasks to run on which machine based on simple local scheduling rules. Results show that a tree-based network evolves in which important tasks are assigned to more reliable resources. This approach is similar to the introduction of reliable super-peers in our approach but differs in the degree of autonomous decision making. In [5], a framework for opportunistic cluster computing using JavaSpaces is implemented. This framework uses a shared communication space and autonomous scheduling decisions but does not consider transient and persistent disconnections that are likely to occur due to node mobility. Based on self-monitoring of utilization, workers decide upon job execution. A distributed load balancing approach is discussed in [6]. Mobile agents monitor the utilization of nodes and encourage the most and the least utilized nodes to balance their workload. Similar to our previous work [2], this decentralized solution has not yet considered non-cooperative nodes.

Using game theory to ensure a cooperative but fair system has been considered, e.g., for peer-to-peer systems [7]. By exploiting the potential of traditional game theory for mobile grids, our work is related to general results of game theory. In [8], different game strategies were tested against each other that tried to maximize the benefit of the player by reacting to the gaming partner's previous decision in terms of cooperation or defection. The game is only meaningful if the competition is repeated and lasts long enough to gain the benefits for cooperation. Tit For Tat (TFT), a strategy that first cooperates and defects only if the gaming partner defected in the last round, turned out to be the best overall strategy, but is also very sensitive to errors (defections by mistake). In [9] this issue is addressed by introducing the stochastic strategy generous TFT, which forgives defections with certain probability.

3 Approaches from Game Theory for Load Distribution

Assuming that one mobile device will generate more load than it can handle, it might desire an available grid to submit its jobs to. The scheduling approach for mobile grids follows the concept of decentralized opportunistic batch scheduling based on the principles of *autonomous local scheduling decisions* and *job queue based coordination* supported by a distributed Virtual Shared Memory (VSM). We chose to base the scheduler on the VSM paradigm because it supports asynchronous communication and persistence of data that is beneficial for unreliable mobile nodes. The job model has been simplified by assuming no inter-job dependencies and equal job priorities to reduce complexity, but also to model situations where non real-time tasks should be simply migrated to a more powerful device.

3.1 Decentralized Scheduling Approach

The scheduling decisions are taken locally by the mobile peers. The algorithm analyzes the current and predicted future node status in terms of connection quality, remaining capacity (battery), utilization, and static resource capabilities. This status is matched against the job's requirements (as given by the job's meta-data) considering user-configurable policies that define the desired degree of resource contribution. The approach allows reaction to self-description of jobs providing a basic best-effort approach if this data is missing (e.g., the expected execution time of a job on a particular computer architecture). To manage distributed coordination, shared job queues are used. Basically, jobs are submitted to a queue by a client from where a worker retrieves a job and its corresponding job meta-data in first come first serve manner. To communicate the job's current processing status, jobs are stored in queues corresponding to the job's state.

We constructed a prototype implementation to demonstrate the feasibility of the approach including fault tolerance mechanisms to handle intermittent connectivity due to lack of remaining battery power and moving out of communication range of one another [2]. Each node is ready to serve as a worker as well as ready to consume computing resources as a client. Non-cooperative nodes have not been addressed so far.

3.2 Introduction of Game Strategies

We now distinguish between *cooperative* (job taking) nodes and selfish, *defective* nodes (non job taking). The motivation for non-cooperative nodes to enter the grid is to just use resources to fulfill their own processing tasks in the role of clients and refuse to contribute as a worker (although they could due to their capabilities). Note that if nodes do not benefit from cooperation incentives (e.g., the possibility to submit jobs to others in the future), selfishness will be the optimal strategy for each node.

The basic scheduling approach is now extended with the game theoretic strategies *Tit For Tat* (TFT), *generous TFT*, and *Go-By Majority* (GBM) to react to selfish nodes. TFT first cooperates and then mirrors a node's opponent's actions. We expect that selfishness would propagate among nodes (as this is the optimal state in our scenario). Generous TFT and GBM do not blindly mirror the opponent's actions. Generous TFT forgives a defection with a probability p – we chose $p = 1/3$ for the experiments after [9]. GMB first cooperates and then defects, but only if the opponent's number of past defections is higher than the opponent's number of past cooperations.

There are two possibilities to apply these game strategies. First, each node could store a history about each other node's behavior (indicating its defection or cooperation) in order to penalize selfish nodes while still cooperating with cooperative ones. Second, each node changes its behavior according to the observed behavior of any other node. By choosing the second option, the whole system is influenced by selfish nodes and we want to investigate how long the system can tolerate selfish nodes without the need of keeping $N(N-1)$ history entries in a mobile grid of size N.

In our model, the nodes communicate within non-disjoint groups of size n about their cooperating or defecting behavior (where n might vary between 1 and N). The group forming algorithm used should assure that the network is not partitioned, which would prevent network wide self-organization. In the current implementation, we assume a unique node ID (like the MAC address) known by the other nodes in range of the ad-hoc network. After sorting (the highest node ID should be the predecessor of the lowest one to build a ring), each node selects the $\lfloor (n-1)/2 \rfloor$ nodes preceding its ID and the $\lceil (n-1)/2 \rceil$ nodes following to create its neighbor group. If $n = N$, every node will send its current cooperation status to each other node causing the worst case communication overhead.

3.3 Simulation Framework

We used a simulation framework based on Discrete Event Simulation (DES) for our investigations. Since we wanted to focus on the complex middleware layer and the distributed game strategies, we abstracted from lower network layers, and thus preferred a custom implementation. Node mobility can be simulated, but for the experiments in this paper, mobile devices only determine the performance footprint of the device model while the nodes remain stationary. We validated the basic approach against the previous prototype implementation to assure compliance with real-world behavior [2].

Table 1. Number of processed jobs / ART (in seconds) / number of defecting nodes for three different game strategies and varying group size n at the end of each experiment (number of always-defecting nodes $d_e = 5$)

	n = 2	**n = 3**	**n = 5**	**n = 10**	**n = 15**	**n = 20**
TFT	58/269/20	66/293/20	34/184/20	27/166/20	15/118/20	15/118/20
g-TFT	80/338/5	80/338/5	80/338/5	80/338/5	80/338/5	80/338/6
GBM	58/269/20	25/166/20	27/172/20	29/170/20	21/139/20	21/138/20

4 Experiments

Experiments were conducted based on simulation to investigate the effects of the different game strategies in terms of job *Average Response Times (ART)*, measured as the delay between the submission and the successful processing end of a job (the ART is only calculated for finished jobs), total *number of processed jobs*, and *number of defecting nodes*.

The job processing time is assumed to be 116.632 seconds (as has been measured for an example test RC5 key decoding application for calculating a 24 bit key on a worker notebook [2]). The simulated mobile scenario consisted of one client submitting 80 jobs (one job per second), and $N = 20$ workers.

We varied two parameters of the simulator: the *number of always-defecting nodes* d_e ranging from 0 to N and the *group size* n for gossiping the cooperation state ranging from 1 to N. We will now present the results for a constant $d_e = 5$ and varying group sizes. The always-defecting nodes were chosen at random out of the set of nodes (uniformly distributed among the groups). We performed one simulation run for each experiment, which we plan to extend in future work to capture different neighboring conditions. Each run was terminated after all jobs were processed or when no job has been taken for more than 1000 seconds.

In Table 1, the results for samples of different group sizes n are summarized.[1] The number of processed jobs reflects the impact of the propagation of selfishness on the final system performance. Only *generous TFT* (g-TFT) could process all jobs independent of n and with small variations within the ART. The reasons are twofold: first, this strategy propagates selfishness only slowly; and second, the frequency chosen for reconsidering cooperation vs. defection is high, which leads only to a short delay in job processing even when a node choses to defect. We observed that nodes that changed to being defective changed back to cooperative mode quickly. *TFT* and *GBM* could never process all jobs and the number of defecting nodes increased towards N for both strategies.

We will now investigate how the number of defecting nodes changed over time as depicted in Fig. 1(a)–1(d) for different group sizes n. The higher the number of defecting nodes, the lower the performance of the mobile grid (lower number

[1] $n = 1$ is omitted, because none of the chosen strategies influences job processing.

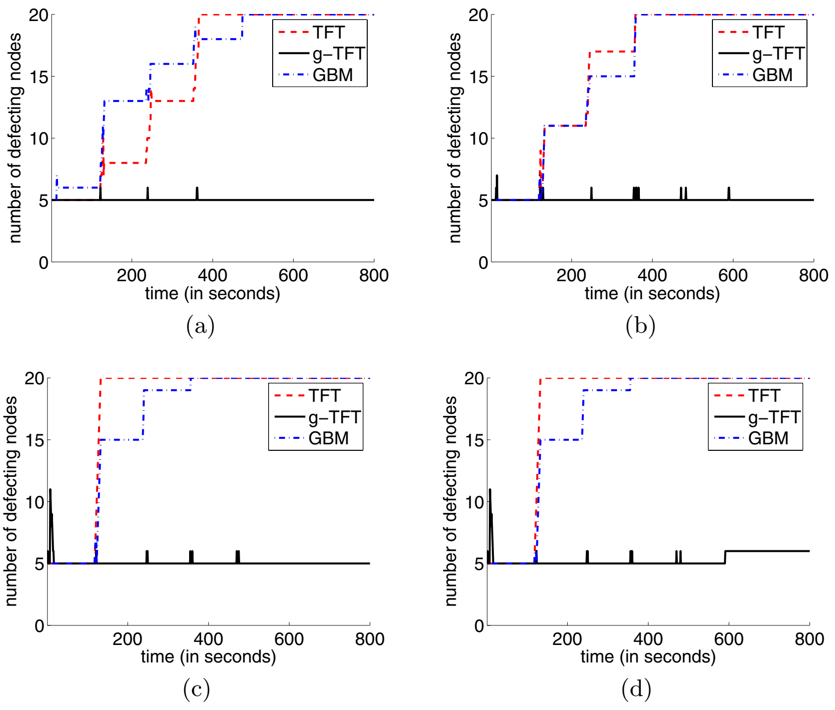


Fig. 1. Number of defecting nodes over time for $d_e = 5$ and varying group sizes (a) $n = 5$, (b) $n = 10$, (c) $n = 15$, and (d) $n = 20$

of processed jobs). For all different group sizes, *TFT* and *GBM* propagated the selfishness of only 25% of nodes quickly. The larger the group size, the faster the selfishness has been propagated. For $n = 5$, *GBM* fostered the increase of the number of defecting nodes slightly faster than *TFT* (Fig. 1(a)). For larger group sizes this behavior is inverted for these two strategies.

All four figures show similar curves for *generous TFT*. The small peaks occur when one or more nodes reacted with defection to other defecting nodes with probability 2/3. The good results are due to the forgiving behavior, which tolerates 25% always-selfish nodes). In Fig. 1(d) one previously cooperating node stayed in defecting state after ≈ 600s. We ran the same simulation with more jobs and a longer simulation time, and observed that this node fell back to cooperating state quickly again. The forgiving *generous TFT* can thus change a node's defective behavior back to cooperation (self-healing), which can be exploited in future work.

To generalize these first results, further investigations in terms of varying neighborhood relationships (distribution of selfish nodes) and dynamically changing selfishness are required and planned for future work. Hereby, the nodes' selfishness is assumed to be a result of policies or different dynamic properties such as workload or remaining battery, which may vary over time.

5 Conclusions and Outlook

We investigated the game strategies *TFT*, *generous TFT*, and *Go-By Majority* (GBM) for a self-organizing job scheduling approach capable of reacting to selfish nodes in mobile grids. We have analyzed the propagation of selfishness introduced by some nodes (25% of the mobile devices) while the defecting or cooperating behavior was communicated among non-disjoint sub-groups of the mobile grid. First results show that TFT and GBM lead to similar results. For both strategies, the number of defecting nodes quickly converged to the total number of nodes and thus job processing halted. This shows the vulnerability and unforgiveness of the strategies. On the other hand, generous TFT could process all jobs, independent of the size of the gossiping sub-groups and did not propagate selfishness.

We plan to work on strategies that are capable of reacting as fast as TFT or GBM to selfish nodes (e.g., in case the number of selfish nodes exceeds a reasonable number) and to show forgiving and self-healing behavior as given by generous TFT in case only a few selfish nodes disturb the system. We plan to approach this by introducing a benefit and cost for cooperation including workload and energy costs/benefits. Finally, we plan to include mobility related intermittent connectivity and node churns in our investigations.

Acknowledgment

The work described in this paper is partially supported by the Austrian Grid Project, funded under contract GZ BMWF-10.220/0002-II/10/2007.

References

1. Satyanarayanan, M.: Fundamental Challenges in Mobile Computing. In: 15th Annual ACM Symposium on Principles of Distributed Computing, pp. 1–7 (1996)
2. Hummel, K.A., Jelleschitz, G.: A Robust Decentralized Job Scheduling Approach for Mobile Peers in Ad-hoc Grids. In: 7th CCGRID, pp. 461–470 (2007)
3. Zong, Z., Nijm, M., Manzanares, A., Qin, X.: Energy Efficient Scheduling for Parallel Applications on Mobile Clusters. Cluster Computing 11(1), 91–113 (2008)
4. Chakravarti, A.J., Baumgartner, G., Lauria, M.: The Organic Grid: Self-Organizing Computation on a Peer-to-Peer Network. In: 1st ICAC, pp. 96–103 (2004)
5. Batheja, J., Parashar, M.: A Framework for Opportunistic Cluster Computing Using JavaSpaces. In: 9th HPCN, pp. 647–656 (2001)
6. Cao, J.: Self-Organizing Agents for Grid Load Balancing. In: 5th IEEE/ACM Int. Workshop on Grid Computing, pp. 388–395 (2004)
7. Feldman, M., Chuang, J.: Overcoming Free-riding Behavior in Peer-to-peer Systems. ACM SIGecom Exchanges 5(4), 41–50 (2005)
8. Axelrod, R.: The Evolution Of Cooperation. Basic Books (1985)
9. Nowak, M.A., Sigmund, K.: Tit for Tat in Heterogeneous Populations. Nature 355, 250–253 (1992)

Web-Based Monitoring and Visualization of Self-Organizing Process Control Agents

Grzegorz Polaków and Mieczyslaw Metzger

Faculty of Automatic Control, Electronics and Computer Science
Silesian University of Technology,
Akademicka 16, 44-100 Gliwice, Poland
{grzegorz.polakow,mieczyslaw.metzger}@polsl.pl

Abstract. This work addresses the problem of visualizing interactions between agents self-organizing in a non-codified way. This kind of behavior emerges from a human factor, which is heavily involved in the presented case of multi-agent system. Tasks and intentions of users present in the system are unknown and the proposed method of analyzing and debugging temporary structures emerging in the system is the eavesdropping of the communication between the actors. A separate software service is proposed, which makes the inferred structure data accessible by an external higher scale system. An example of visualization is provided in the form of SVG-based interactive dynamic block diagram.

Keywords: Self-organizing multiagent system, industrial process control, visualization, human factor, producer-distributor-consumer.

1 Introduction

Visualization of self-organizing [1] agent systems is a difficult task, due to the distribution of the information on the system structure. Each agent is aware of its local surroundings only, while the system's capabilities are an effect of its overall global configuration [2]. Therefore, a visualization task consists of gathering scattered information and consolidating it into consistent description at first, before presenting it to the user. There are many engines for visualization of a topology of agent system structure, fitted to varying capabilities of agent environments, e.g. VizScript [3], VAST [4], MASON visualization tools [5].

Most of the existing approaches assume that the self-organizing system being visualized is purely artificial, and the data needed for the visualization may be gathered by implementing additional services in the nodes (as in [6]). These artificial self-organizing systems are based on natural ones incorporating societies of living organisms (ants, bees, even humans) [7]. In case of these natural systems, there is no possibility of implementing visualization services in the nodes, only a passive observation of the system is possible, as tasks and intentions of the actors are generally unknown. Therefore, the visualization has to rely on passive observations of the actors behavior. In case of physical natural systems, observations may consist of simple measurements of real-world physical variables (e.g. spatial location of the actors in environment).

On the other hand, logical systems hardly have any physically measurable properties and the only source of information regarding the system structure is content of communication between the actors.

This work addresses the issue of visualizing the structure of such a system, in which the self-organization logic comes from the human users. Modeling and visualization of human behavior has been actively researched for a long time ([8][9]). In this paper a practical method of self-organizing joint natural-artificial distributed system observation is presented, consisting of capturing and analyzing the communication between the nodes of the system. Similar passive observation method of IP traffic was introduced in [10]. While the traffic data still doesn't show the intentions of the actors, it at least allows for graphical presentation of the data flow in the network in the form of graphical mesh of connections, allowing for clustering of previously uniformly distributed nodes (see [11]). Since the communication in various instances of the class of self-organizing networks with embedded intentions may be performed in varying ways, a unified service is proposed. While the industry still lacks an accepted standard of the Web Services location and description ([12]), the service proposed seems to be promising approach.

2 Motivation

During last few years a system of multiple laboratories has been installed at the Silesian University of Technology enabling advanced experimentation on real-world industry-grade continuous processes. Possibilities of research and teaching enabled by this system are unique as the system was designed to be highly reconfigurable. While the hardware of the plants stays the same, logical structure of the system is changed to suit the users. Each of the users is able to connect self-developed pieces of software to the static hardware system, according to currently conducted experiments. Typical roles of user programs include: signal generators, control algorithms, user interfaces, etc.

The system is integrated using the holonic approach [13]. Each of the plants is treated as a holon consisting of lightweight software agents (see [14]) assigned to the hardware sensors and actuators. Such a holon is visible from outside as a simple set of agent communication interfaces, keeping the hardware fully embedded. Users develop their programs as agents with similar interfaces, so the interaction between the hardware and the users' software takes place in the shared agents' communication pool. The hardware plants and interfaces offered by the lightweight agents may be considered as an environment for organizational-centered heavyweight agents developed by the users (see [15]).

The communication between the agents is based on the custom high-bandwidth low-latency protocol [16], due to the specific requirements ([17]) of the continuous processes control systems. This data-centric real-time protocol is based on the ppPDC data distribution scheme (parallel processing producer-distributor-consumer), which arranges the agents as a blackboard system and allows them to communicate in a fully time determined manner by the switched Ethernet network.

The system of laboratories performs very well, however due to its distributed and dynamic nature it became hard to monitor. A temporary structure of software in the system is a dynamic effect of multiple elements: hardware structure, software agents, and the human factor, i.e. researchers' and students' tasks and intentions (Fig. 1).

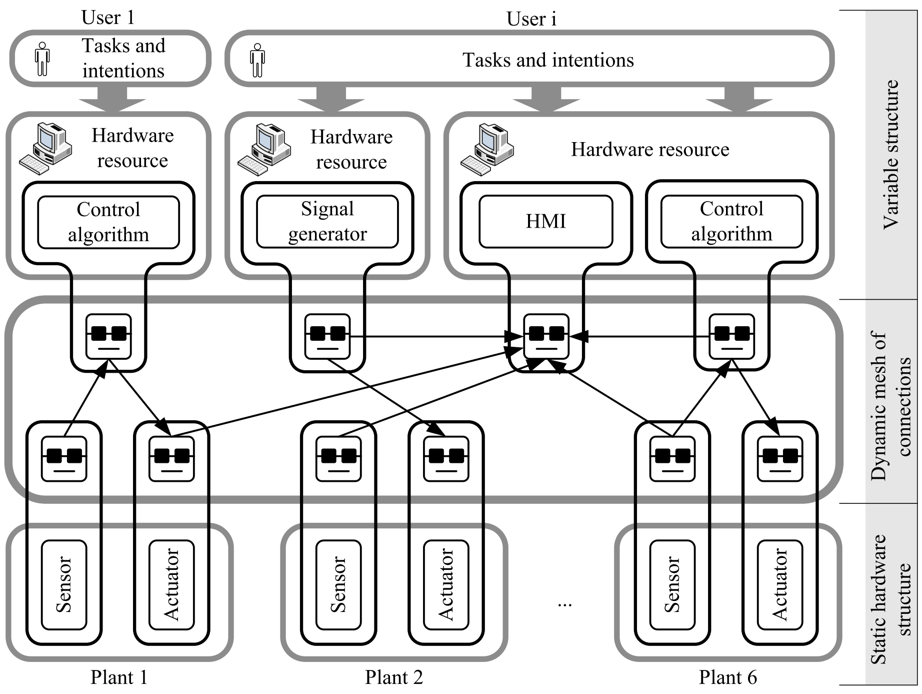


Fig. 1. A mesh of dynamic inter-agent connections as an effect of the hardware structure and users' tasks

When the system is considered from the outside point of view, without any knowledge of users' tasks, it consists simply of a set of software agents executing their software threads in the distributed hardware, displaying self-organization behavior when grouping into clusters. Within the clusters they exchange numerical data in an ordered way. At any given moment the structure is logically organized, however, in a broader time frame (in order of hours), it is highly variable.

To make the visualization tasks achievable, it was required to develop a system that exploits low level characteristics of the used inter-agent communication protocol. The protocol is based on the producer-distributer-consumer data distribution scheme, employing the distributor node to maintain the data exchange. Due to distributor's role, it is capable of intercepting inter-agent communication, and performing the system's visualization accordingly to the captured network traffic.

The approach to the visualization of the agentified system takes into the account a presence of the specialized visualization interface. Such an interface is usually implemented in all control systems using some type of data access standard (Web Service, OPC standard). A typical structure of a SCADA (Supervisory Control and Data Acquisition) system connection with the control system is shown in Fig. 2a.

The control system's data in the form of numerical values of measurements are available through the standardized interface, designated here as Raw Data service. The content of the transmission stream gives no knowledge on current state of the

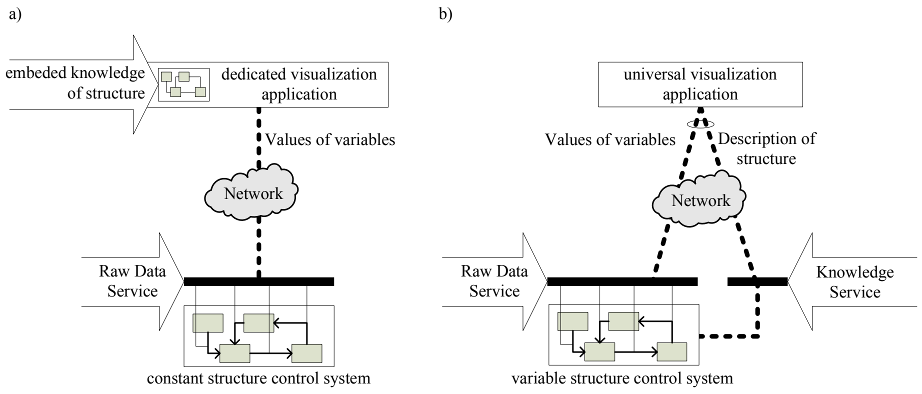

a)
embeded knowledge of structure
dedicated visualization application
Values of variables
Network
Raw Data Service
constant structure control system
b)
universal visualization application
Values of variables
Description of structure
Network
Raw Data Service
Knowledge Service
variable structure control system

Fig. 2. a) Architecture of visualization for constant structure systems b) Architecture with complementing knowledge service for systems with variable structure

system until raw data are correctly placed in the system's structure according to its meaning. In a traditional visualization application, this knowledge on data's meaning is embedded in the executable code of an application, because the mesh of connections in the system is constant, and can be stored once and for all. In case of a system with variable structure, a method of a structure description has to be developed. The proposed architecture shown in Fig. 2b introduces an additional interface, called the Knowledge Service, whose main task is serving the knowledge on the current state of the structure. The union of abstract knowledge from the proposed service and the raw data from the classic interface gives the full state of the system in a given moment.

Implementation of the proposed Knowledge Service enables the system to be treated as a part of higher-level large-scale system, which may incorporate multiple local self-organizing systems, visualizing and comparing them.

3 Implementation

The proposed Knowledge Service was implemented in the existing real-world system. Characteristics of the framework made it impossible to predict the agenda of self-organization, so only the communication stream can be observed. Human factor based parts of the logical connection mesh are completely embedded, and only the artificial part of the network are possible to gather. However, all the actors acting on behalf of human users and on behalf of the control system's hardware are exchanging information with the blackboard, so it is the single point of network in which all the meaning messages are passing. The system was augmented with the agent communication traffic capture capability. The traffic is analyzed in the real-time, and the structure of the connection mesh is inferred on this basis. This temporal data is then served with the Knowledge Service, while the raw content of the blackboard is served with the Raw Data Service.

The Services are implemented in this case as typical Web Services, i.e. they serve XML data through the HTTP protocol. Such a method was chosen because the system is designed to be able to be integrated with the global Semantic Network, which is then supposed to fulfill the role of higher-level large scale system. According to the assumed specifics of the higher-level system, a visualization engine was developed as a web-bot capable of using both the Services. The application is Web-based and it uses the AJAX technique to provide interactive experience to an end-user. From technical point of view the user interface of the bot is an SVG document while its logics are programmed as an embedded ECMAScript program. SVG was chosen instead of HTML because its capabilities of software driven vector graphics already proved to be perfect for ontological diagrams visualisation ([18]) and it has capability of implementing much appreciated rich-graphics human-machine interfaces.

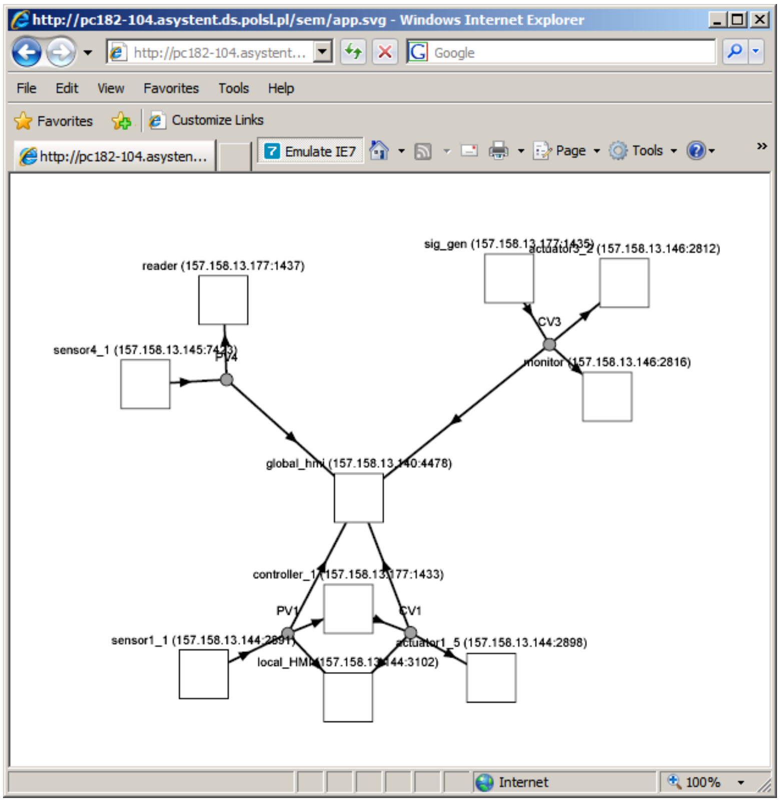

http://pc182-104.asystent.ds.polsl.pl/sem/app.svg - Windows Internet Explorer
File Edit View Favorites Tools Help
Favorites
Customize Links
Emulate IE7
Page
Tools
Internet
100%
reader (157.158.13.177:1437)
sig_gen (157.158.13.177:1435)
actuator3_2 (157.158.13.146:2812)
CV3
sensor4_1 (157.158.13.145:7443)
PV4
monitor (157.158.13.146:2816)
global_hmi (157.158.13.140:4478)
controller_1 (157.158.13.177:1433)
PV1
CV1
sensor1_1 (157.158.13.144:2891)
actuator1_5 (157.158.13.144:2898)
local_HMI (157.158.13.144:3102)

Fig. 3. A diagram drawn by the monitoring application

To put stress on the logical bonds between the agents, they are graphically clustered into groups by use of the gravitational model (see [11] and [19]). A screenshot of a working application with a sample system's structure is presented in Fig. 3. As it is seen in the figure, those agents which exchange data at the moment are graphically grouped, which allows the system's supervisor to estimate the current level of system organization.

4 Concluding Remarks and Future Work

This work focuses at the task of visualizing the system in which the self-organization rules are a result of non-deterministic factors, i.e. unpredictable human behavior driven by unknown intentions. As an example of such a system the real-world laboratory for research and education of distributed control systems was presented. The method of the system state visualization was proposed, consisting of capturing the network traffic caused by user-programmed software agents and drawing the communication dependencies between the actors. The resulting diagram of clustering allows the supervisor to infer the character of the currently performed tasks. It is planned to expand the presented method to take the human owners of visualized software agents into account. Such a modification will add an additional logical layer of the visualization by bonding the nodes of the visualization diagram to a specific user.

Acknowledgement. This work was supported by the Polish Ministry of Scientific Research and Higher Education using funds for 2008-2010 under grant no. N N514 296335.

References

1. Ashby, W.R.: Principles of the self-organizing dynamic system. Journal of General Psychology 37, 125–128 (1947)
2. Ndumu, D.T., Nwana, H.S., Lee, L.C., Haynes, H.R.: Visualization and debugging of distributed multiagent systems. Applied Artificial Intelligence 13(1-2), 187–208 (1999)
3. Jin, J., Maheswaran, R.T., Sanchez, R., Szekely, P.: VizScript: Visualizing complex interactions in multi-agent systems. Technical report, AAAI Spring Symposium, pp. 64–65 (2007)
4. Oberheide, J., Karir, M., Blazakis, D.: VAST: Visualizing autonomous system topology. In: Proceedings of the 3rd international workshop on Visualization for computer security, pp. 71–80. ACM, New York (2006)
5. Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., Balan, G.: MASON: A multiagent simulation environment. Simulation 81(7), 517–527 (2005)
6. Lim, A.: Distributed services for information dissemination in self-organizing sensor networks. J. Franklin Inst. 338(6), 707–727 (2001)
7. Van Dyke Parunak, H., Brueckner, S.: Entropy and self-organization in multi-agent systems. In: Proceedings of the fifth international conference on Autonomous agents, pp. 124–130. ACM, New York (2001)
8. Hlavacs, H., Kotsis, G.: Modeling user behavior: a layered approach. In: 7th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, pp. 218–225. IEEE Press, New York (1999)
9. Moya, J.L., McKenzie, F.D., Nguyen, Q.-A.H.: Visualization and rule validation in human-behavior representation. Simulation and Gaming 39(1), 101–117 (2008)
10. McPherson, J., Ma, K.-L., Krystosk, P., Bartoletti, T., Christensen, M.: PortVis: A tool for port-based detection of security events. In: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security, pp. 73–81. ACM, New York (2004)

11. Staron, R.J., Tichý, P., Sindelár, R., Maturana, F.P.: Methods to observe the clustering of agents within a multi-agent system. In: Mařík, V., Vyatkin, V., Colombo, A.W. (eds.) HoloMAS 2007. LNCS, vol. 4659, pp. 127–136. Springer, Heidelberg (2007)
12. Forster, F., De Meer, H.: Discovery of web services with a P2P network. In: Bubak, M., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2004. LNCS, vol. 3038, pp. 90–97. Springer, Heidelberg (2004)
13. Metzger, M., Polaków, G.: Holonic multiagent-based system for distributed control of semi-industrial pilot plants. In: Mařík, V., Vyatkin, V., Colombo, A.W. (eds.) HoloMAS 2007. LNCS (LNAI), vol. 4659, pp. 338–347. Springer, Heidelberg (2007)
14. Van Dyke Parunak, H., Nielsen, P., Brueckner, S., Alonso, R.: Hybrid multi-agent systems: Integrating swarming and BDI agents. In: Brueckner, S.A., Hassas, S., Jelasity, M., Yamins, D. (eds.) ESOA 2006. LNCS, vol. 4335, pp. 1–14. Springer, Heidelberg (2007)
15. Weyns, D., Van Dyke Parunak, H., Michel, F., Holvoet, T., Ferber, J.: Environments for multiagent systems state-of-the-art and research challenges. In: Weyns, D., Van Dyke Parunak, H., Michel, F. (eds.) E4MAS 2004. LNCS (LNAI), vol. 3374, pp. 1–47. Springer, Heidelberg (2005)
16. Polaków, G., Metzger, M.: Agent–based approach for labVIEW developed distributed control systems. In: Nguyen, N.T., Grzech, A., Howlett, R.J., Jain, L.C. (eds.) KES-AMSTA 2007. LNCS (LNAI), vol. 4496, pp. 21–30. Springer, Heidelberg (2007)
17. Sterbenz, J.P.G.: High-speed networking: a systematic approach to high-bandwidth low-latency communications. In: 13th Symposium on High Performance Interconnects, pp. 8–9. IEEE Press, New York (2005)
18. Samper, J.J., Tomás, V.R., Carrillo, E., Do Nascimento, R.P.C.: Visualization of ontologies to specify semantic descriptions of services. IEEE Transactions on Knowledge and Data Engineering 20(1), 130–134 (2008)
19. Golovchinsky, G., Kamps, T., Reichenberger, K.: Subverting Structure: Data-driven Diagram Generation. In: Proceedings of the 6th IEEE Visualization Conference, pp. 217–223. IEEE Computer Society Press, Washington (1995)

Spatial Self-Organization in Networks of Things

Kashif Zia and Alois Ferscha

Institut für Pervasive Computing
Johannes Kepler University Linz, Austria
{zia,ferscha}@soft.uni-linz.ac.at

Abstract. Miniaturized, wirelessly networked embedded systems combined with Peer-to-Peer computing principles have started to pervade into objects of everyday use, like tools, appliances or the environment, thus implementing ensembles of autonomous, interacting "networked things". With the development of the Peer-it framework, integrating a self-contained, miniaturized, universal and scalable embedded systems hardware platform, basically containing sensors, actuators, computing and wireless communication facilities, and a Peer-to-Peer (P2P) based software architecture, we have proposed a "stick-on" solution for the implementation of networks of things (NoTs). The Peer-it design and miniaturization ultimately aim to yield a "smart label", ready to be sticked on to literally every "thing" as a NoT enabler. The paper addresses the issue of spatial awareness of objects within NoTs, and proposes abstractions of space based on (i) topology, (ii) distance and (iii) orientation. Experiments are conducted to investigate on the ability of objects in a NoT to self-organize based on their spatial orientation.

Keywords: Autonomic Computing, Sensor/Actuator Systems, Context Awareness, Self-organization, Spatial Abstraction, Networks of Things.

1 Introduction

Pervasive computing has promoted a new era of computing, where the "computer" is no longer understood as a single device or a network of devices, but rather the entirety of services delivered through ad-hoc ensembles of electronic devices or information appliances. Such ensembles of objects are assumed to sense the physical world via a huge variety of cooperative and coordinated sensors, and acting via a plethora of (again) coordinated actuators. The nature and appearance of such objects appears to be hidden in the fabric of everyday life, like tools, appliances, machinery, furniture, clothing, etc., yet invisibly networked and omnipresent. Individually built with miniaturized embedded systems technology, usually for a specialized purpose, and engaging (short range) wireless technology for spontaneous communication, these objects raise the challenge of an operative, and semantically meaningful interplay among each other. "Meaningful" here referring to a variety of purposes for which a group of objects is spontaneously configured into a NoT service ensemble, be it to achieve a certain service goal, to improve service quality, dependability or performance, to increase fault-tolerance, etc.

An acknowledged approach to address the challenge of "meaningful interplay" is by design, for example to design objects to be able to manage themselves in a more or less autonomous way, while at the same time designing ad-hoc ensembles of objects to self-organize. Self-management here stands for the ability of single object to describe itself, to select and use adequate sensors to capture information describing its situation, to find an interpretation for the situation (context), and to be ready to spontaneously interact with other objects in the vicinity, defined by the range of the involved wireless communication technology. Self-organizing (in the context of this paper) stands for the ability of a group of (possibly heterogeneous) objects to engage into a spontaneous ensemble based on interest, purpose or goal, to agree a common ensemble interest or negotiate a common ensemble goal, and to enforce individual objects to act so as to achieve the ensemble goal.

With previous work [1, 2] we have proposed to implement NoTs based on a miniaturized stick-on embedded computing systems, the Peer-it system, integrating sensor, actuator and wireless communication facilities, which connect these objects within limited vicinity. The Peer-it software stack implements features of self-management within a component based software architecture local to a peer, and features of self-organization in a totally distributed style. Interaction among peers (or NoT objects) at the application level is invoked based on the analysis of self-describing profile data exchanged across objects in vicinity. Peer-it based NoTs thus represent spontaneous ensembles of coordinated nodes in a wireless network, exhibiting features of autonomy like self-management at the node level, and self-organization at the network level.

Since NoTs operate in physical space, with nodes having locations in physical space, interesting questions arise with respect to the spatial distribution of nodes, particularly the impact of their distribution onto their ability to communicate, or even more challenging, to coordinate their activities. With this paper, therefore, we address issues on how the spatial distribution of nodes in NoTs, and their mutual spatial relationship impacts self-organization in NoTs. Specifically, we study the self-organization capability of NoTs with a certain number of objects at a certain density within a terrain of certain size. Section 2 motivates a scenario of investigation for NoTs within which objects are assumed to be able to sense their position and orientation. Section 3, based on simulation results, reports to which extent self-organization can be achieved among an ensemble of objects, if objects are only assumed to change their orientation, but not their position. We report evidence, that self-organization of "global" orientation in NoTs can be achieved with only local coordination (Section 4).

2 The Self-Organization of "Things" wrt. Orientation

The spatial orientation of objects (or "things") in physical space can be basically abstracted wrt. extrinsic or intrinsic frames of reference. In an extrinsic frame of reference, the direction of an object is defined in relation to fixed bearings like the cardinal directions north, south, east, west, and any arbitrary resolution of accuracy, or gravity (like high or low). The technological sensors in Peer-its to capture orientation of an object according to the extrinsic scheme are compasses and gyroscopes. Within an intrinsic frame of reference, the orientation of an object is defined in relation to part of itself or part of another object (e.g. front, back, left, rigt, etc.). The technological sensors in Peer-its to capture orientation according to the intrinsic scheme are ultra-sound sensors or laser scanners. For the rest of the paper we consider orientation wrt. to an extrinsic frame of reference, and study cases of self-organization where the orientation information of objects (randomly placed in physical space) is the only control parameter for ensemble self-organization. Further, we concentrated on the re-adjustment capabilities of the whole population with respect to orientation.

Our research hypothesis is that the self-organization of orientation of things in NoTs can be achieved at global level with the knowledge of local information about neighborhood and corresponding orientations only. In addition, we study the effects onto self-organization when varying the node density, the number of nodes eligible for "re-organization" (i.e. with the ability of an object to autonomously change its orientation, e.g. with a built-in actuator like a motor), and the shape, size or dimension (2D, 3D) of the NoT terrain (here, only results for 2D, quadratic terrains are reported).

In a wider context, spatial self-organization has been a subject of interest in agent technology [5] and robotics [6]. Research in AI and cognitive science has also focussed on flocking behavior [7] and swarm intelligence [8]. Our research is not focussed on any of these areas which are specific application domains. Instead, we formulated dependencies between driving forces (node density and number of re-adjustments required) to gauge expectations. In an environment of varying settings, the known expectation can help understand the resultant behavior. It can also motivate the user to change the settings to get the expected result within acceptable range.

3 Experiment Settings and Results

In a scenario having randomly placed objects, we assume objects having actuators to re-adjust the orientation if required. Objects can sense and actuate a reaction within a constraint. We take this constraint to be 30°. It means that a node can sense orientation from 0 to 330 degrees, and an actuator can invoke a re-adjustment in orientation by steps of +30° or -30°. In this research, we consider a static network, in which nodes are not moving (change of orientation is not considered as a motion). In an environment of nodes having only orientation sensors, it would be insignificant to consider moving nodes, not knowing measures of their motion. But motion, alone can have dynamics (a constantly volatile neighborhood) which can impact the self-organization capabilities of network. We have retained this aspect for future research.

The investigation terrain, or Global Space (GS), is defined by the size, shape and dimensions of the space. For simplicity, we have considered 2D terrain space of 500 * 500 units of distance. Further we define the Zone of Influence (ZoI) of an object to be the number of other objects within interaction range. This zone can be of any shape. For simplicity we consider a 2D circular region as the shape being considered. As the ZoI expands, the number of objects an object can interact with increases. We conducted experiments for a varying number of objects (10, 25, 50, 100, 200, and 500), and increased the object coverage of ZoIs periodically for each case (percentages of covered objects to be 0.5%, 3%, 10%, 33%, and 100%). For each such subset, we increased the percentage of objects which needed re-adjustment from 1 to 50 % (1%, 2%, 5%, 10%, 20%, 30%, 40%, and 50%). At this lowest level each such experiment was performed for a 1000 times to meet statistical significance. The pseudo code described below sketches the simulation algorithm:

```
for each node                   // we refer to objects as nodes
     switch_destination = get_max_orient() // counts median of
            neighbors' orientation
     // if neighbors' count > 1, median is unique, and different from
            node's own orientation, do following
     switch_factor = get_min_factor() // returns either +30 or -30
     is_switching = TRUE
     for each node
            while (is_switching)
                   orientation=orientation + switch_factor
                   if orientation >= 360 orientation = 0
                   if orientation = switch_destination
                          is_switching = FALSE
                          has_switched = TRUE
get_min_factor()
     if (orientation!=switch_destination)
            if (Math.Abs(switch_destination-orientation) < 180)

                   return 30
            else
                   return -30
```

We refer to nodes which require re-adjustments to be the 'wrong' (‘wrongly directed’) nodes, as opposed to nodes which do not require re-adjustments ('right' nodes). After injecting a certain amount of wrong nodes into the terrain, each node in parallel calculates the median of neighbors' orientations within its ZoI. If the median is greater than 1 and different from the node's orientation, a re-adjustment is performed, referred to as a node 'switch'.

The main purpose of this experiment is to analyze the inter-relation between the number of nodes, the number of neighbors and the number of wrong nodes in a setting to question the possibility of reaching to a stabilized condition (no more switching required), and to inquire about relative improvement and limitations, from one iteration to the other. To analyze the inter-relationship among these factors, we concentrate on following indicators:

- **Wrong-Wrong** (WW) percentage.
  percentage of nodes which were wrong before simulation and still are wrong after simulation.
- **Wrong-Right** (WR) percentage.
  percentage of nodes which were wrong before simulation but are right after simulation.
- **Right-Right** (RR) percentage.
  percentage of nodes which were right before simulation and still are right after simulation.
- **Right-Wrong** (RW) percentage.
  percentage of nodes which were right before simulation but are wrong after simulation.

To analyze the relative improvement, we performed one iteration after the other and compared the results (up to 3 iterations).

Fig 1. shows a graphical view of a sample node distribution for 25 nodes, ZoI adjusted to include 10% of the neighbor nodes and 10% wrong nodes. Fig 2. shows a graphical view of a sample node distribution for 50 nodes, ZoI adjusted to include 3% of the neighbor nodes and 30% wrong nodes (nodes with orientation 0° are referred to as right).

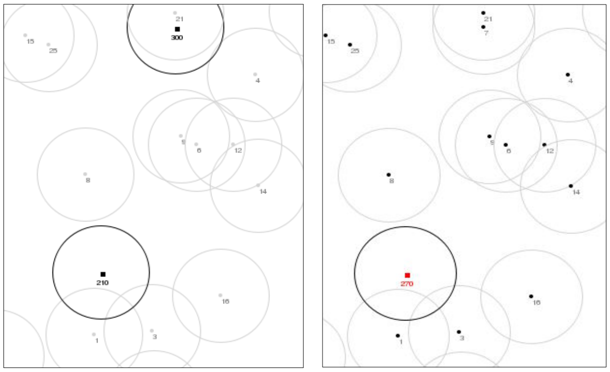


Fig. 1. Part of the scenario with 25 nodes. Left: Node 7 (top) and 19 (bottom) are wrong nodes represented by darker ZoI circles, having orientation equal to 300 and 210 respectively. Right: After two switches, node 7 has settled to 0; node 19 would also be settled to 0 after 3 more switches.

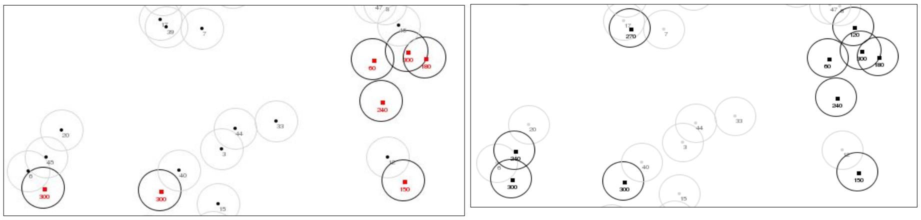


Fig. 2. Part of the scenario with 50 nodes. Node 39 (top-middle) shows expected behavior and settled to 0 after 4 switches. Node 45 (bottom-left) shows expected behavior but restricts other node with orientation 300 to perform switching. The same is the case with clusters of nodes (top-right). Only one node (top most) is to perform the switching whereas the other remains the same due to variety of neighbors' orientations (with no clear winner).

Analysis was done for each of 10, 25, 50, 100, 200, and 500 nodes. For limited space reasons, in Fig. 3 we only report on the results for the case with 25 nodes. The x-axis of graph in Fig. 3 is divided into five basic chunks representing cases where the ZoI is adjusted to get 0.5%, 3%, 10%, 33% or 100% neighbors. Each of these chunks is subdivided into levels of 1%, 2%, 5%, 10%, 20%, 30%, 40% or 50% of “wrong” nodes. The y-axis in Fig. 3 represents the possible (%-age) levels of WW, WR, RW and RR. Irrespective of actual quantities, we have stacked these values according to contributing percentages. It is important to note that the statistics of WW, WR, RW and RR are percentages stacked only for the 1st iteration, whereas improvement lines represent improvement after the 1st (thick gray dotted line), 2nd (gray line) and 3rd (black line) iteration. As for example, in the case of 50% wrong nodes inserted, and 10% neighborhood conditions (12 nodes out of 25 are wrong), lets say in the 1st iteration, we have the following results: WW = 28% (7 nodes), WR = 20% (5 nodes), RW = 4% (1 node), RR = 48% (12 nodes). This result is expressed in the 50% column of the 10% chunk on the x-axis. The percentage of improvement (1st iteration) line shows approximately 35% improvement (which is understandable when we see that after the 1st iteration, the number of wrong nodes decreased from 12 to 8). In the next iterations, the aggregated value of improvement has marginally increased.

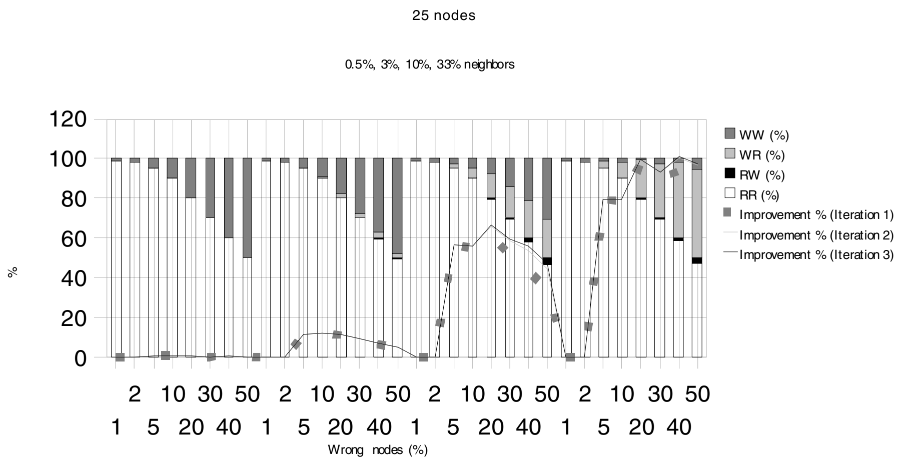


Fig. 3. Case 2: Number of nodes = 25; Percentage Stacked values (WW, WR, RW, RR) against 0.5%, 3%, 10% and 33% neighbors. Each neighborhood is further sub-divided into 1%, 2%, 5%, 10%, 20%, 30%, 40% and 50% wrong nodes.

In summary, from the experiments involving all the scenarios (10, 25, 50, 100, 200, and 500 nodes, not only the ones reported in Fig. 3), the following conclusions could be drawn:

- Finding 1: With an increasing number of wrong nodes, the improvement decreases for sufficiently large value of wrong nodes (more than 5%) and RW increases (thus decreasing RR).
- Finding 2: With an increasing number of neighbors, the improvement increases and RW increases (thus decreasing RR) followed by a decrease.
- Finding 3: With an increasing number of nodes, the improvement increases and RW increases (thus decreasing RR) followed by a decrease.

4 Conclusion

Self-organization of orientation in NoTs can be achieved at global level with a local level coordination mechanism only. In addition, the simulation results show that self-organization depends on number of factors including node density, percentage of nodes eligible for self-organization and global space specifications. Irrespective of the application domain, studying the behavioral inter-play between these factors is important to forecast effectiveness of self-organization. Alternatively, if applicable, application expectations can be tailored by re-adjustment of one or more of these factors.

Acknowledgements

We wish to thank the reviewers for their comments on the longer version of this work, and our shepherd for valuable suggestions on how to compact its contribution into less pages. She was very guiding wrt. to finding the right focus, and thoroughly and patiently helping to even get the paper error and typo free.

References

1. Ferscha, A., et al.: Peer-it: Stick-on solutions for networks of things. Das, S.K., Contri, M., Shirazi, B. (eds.) Pervasive and Mobile Computing Journal 4(3), 448–479 (2008)
2. Ferscha, A., et al.: Building Flexible Manufacturing Systems Based on Peer-its. Embedded Systems Design in Intelligent Industrial Automation, EURASIP Journal on Embedded Systems (October 2007) (special issue)
3. Gervasi, V., Prencipe, G.: Flocking by a set of autonomous mobile robots. Technical report TR-01-21, University of Pisa (2001)
4. Mataric, M., Zordan, V., Williamson, M.: Making complex articulated agents dance: an analysis of control methods drawn from robotics, animation and biology. Autonomous Agents and Multi-Agent Systems 2(1), 23–44 (1999)
5. Reynolds, C.W.: Flocks, herds and schools: a distributed behavioral model. Computer Graphics 21(4), 25–34 (1987)
6. Mondada, F., Guignard, A., Bonani, M., Floreano, D., Bar, D., Lauria, M.: Swarm-bot: from concept to implementation. In: Lee, G., Yuj, J. (eds.) Proc. of the IEEE/RSJ International Conference on Intelligent Robot and Systems, pp. 1626–1631 (2003)

Author Index