RvS

Rechnernetze und verteilte Systeme

Stand Oktober 1994

Skriptum zur Stammvorlesung

Heiko Krumm
Universität Dortmund
Fachbereich Informatik, LS IV

1. Rechnernetze, verteilte Systeme und Telekommunikationssysteme 1

1.1 Rechnernetze 1
1.2 Rechnernetze und verteilte Systeme 3
1.3 Rechnernetze und Telekommunikationssysteme 4
1.4 Rechnernetze und Datennetze 5
1.5 Literaturhinweise 5

2. Telekommunikationssysteme 7

2.1 Logische Architektur von Telekommunikationssystemen 9
2.2 Funktionalität von Telekommunikationsdiensten 10
2.2.1 Dienstschnittstelle 11
2.2.2 Dienstnehmer-Rollen 13
2.2.3 Grundformen von Dienstleistungen 13
2.2.4 Verbindungen 14
2.2.5 Auslieferungsdisziplin 15
2.2.6 Richtungsbetrieb 16
2.3 Qualität von Kommunikationsdiensten 16
2.3.1 Leistung 17
2.3.2 Kosten 17
2.3.3 Störungen 18
2.3.4 Schutz 19
2.4 Telekommunikationsprotokolle 19

3. Basisdienste 21

3.1 Übertragungssysteme 22
3.1.1 Kanal 23
3.1.1.1 Signalklassen 23
3.1.1.2 Schrittgruppen 24
3.1.1.3 Parallel- und Serienübertragung 24
3.1.1.4 Synchronisation 24
3.1.1.5 Codierung 25
3.1.1.6 Qualitätsparameter 27
3.1.1.7 Betriebsarten 27
3.1.1.8 Sammelkanäle 27
3.1.2 Medien 27
3.1.2.1 Medienqualität 28
3.1.2.2 Verwendete Medien 29
3.1.2.3 Geteilte Medien 30
3.1.3 Übertragungsverfahren und Umformung 30
3.2 Vermittlungssysteme 31
3.2.1 Logische Schichtenarchitektur 32
3.2.2 Netz-Topologien 34

3.2.3 Durchschaltevermittlung 35
3.2.3.1 Raummultiplex 36
3.2.3.2 Zeitmultiplex 37
3.2.4 Speichervermittlung 37
3.2.5 Zellenvermittlung im statistischen Multiplex-Betrieb (B-ISDN: ATM) ... 39
3.3 Datenübertragungseinrichtungen und Schnittstellen 41
3.3.1 Peripheriegeräteanschluß 41
3.3.1.1 Externe-Schnittstellen 42
3.3.1.2 Gerätezugriff 43
3.3.1.2.1 Programmierte Ein/Ausgabe 43
3.3.1.2.2 Unterbrechungsgesteuerte Ein/Ausgabe 44
3.3.1.2.3 Selbständige Geräte 45
3.3.2 Anschluß an öffentliche Netze 46
3.3.2.1 Schnittstelle nach V.24, V.25 48
3.3.2.2 Schnittstelle nach X.21 50
3.3.2.3 Externe Schnittstellen im ISDN 50
3.3.3 Anschluß an lokale Netze 51

4. Telekommunikationsdienste und Protokolle 54

4.1 Telekommunikationsdienste 54
4.1.1 Beschreibung der Dienstfunktionalität 55
4.1.1.1 Zugangspunktadressen 56
4.1.1.2 Schnittstellenereignistypen 56
4.1.1.3 Weg/Zeit-Diagramme 57
4.1.1.4 Zustandsübergangsdiagramme 58
4.1.2 Verbindungsorientierte Dienste 61
4.1.2.1 Adressen und Verbindungsendpunkte 61
4.1.2.2 Verbindungseigenschaften 63
4.1.2.2.1 Dienstleistungsqualitäten 63
4.1.2.2.2 Richtungsbetrieb 64
4.1.2.2.3 Rechtevergabe 64
4.1.2.2.4 Verbindungsklassen 64
4.1.2.2.5 Rückstau 65
4.1.2.3 Verbindungsaufbau 65
4.1.2.3.1 Parametertypen 65
4.1.2.3.2 Primitive 66
4.1.2.4 Datenaustausch 67
4.1.2.4.1 Unbestätigter Datenaustausch 67
4.1.2.4.2 Bestätigter Datenaustausch 68
4.1.2.5 Verbindungsabbau 68
4.1.2.6 Verbindungsabbruch 69
4.1.3 Datagramm Dienste 70
4.2 Telekommunikationsprotokolle 71
4.2.1 Beschreibung der Protokollfunktionalität 71
4.2.1.1 Ablauffestlegungen 72
4.2.1.2 PDU-Format-Festlegungen 75
4.2.1.2.1 Bitmuster-PDU-Formate 75
4.2.1.2.2 Anwendungsprotokoll-PDU-Formate 76
4.3 Einfaches Beispiel 77
4.3.1 Zieldienst O 78
4.3.2 Basisdienst U 79
4.3.3 Instanz E 81

4.3.4 Schlußbetrachtung 82
4.3.4.1 Protokollkorrektheit 83
4.3.4.2 Protokoll-Leistungsfähigkeit 87
4.3.4.3 Realitätsnähe des Beispiels 93

5. Logische Architektur von Telekommunikationssystemen ... 1

5.1 Grobe Aufgabenaufteilung der Schichten 1
5.2 Transportsystem 3
5.2.1 Transportdienst 3
5.2.2 Aufbau des Transportsystems 5
5.3 Anwendungsorientierte Schichten 9

6. Protokollmechanismen 107

6.1 Zuteilung geteilter Medien 107
6.1.1 Zentrale Zuteilung nach Bedarf 108
6.1.2 Dezentrale Zuteilung unter Konkurrenz 109
6.1.3 Dezentrale Zuteilung durch zirkulierendes Recht 111
6.2 Fehlererkennung 112
6.2.1 Fehlererkennende Codes 114
6.2.2 Sequenzzahl 115
6.2.3 Zeitstempel 116
6.2.4 Quittieren 116
6.2.5 Zeitüberwachung 117
6.3 Fehlerbehebung 118
6.3.1 Fehlerkorrigierende Codes 118
6.3.2 Ignorieren 119
6.3.3 Wiederholung 119
6.3.4 Rücksetzen 119
6.4 Längenanpassung 120
6.5 Flußkontrolle 120
6.6 Weiterleiten 122
6.7 Überlastkontrolle 124
6.8 Übertragungsleistungsanpassung 125
6.9 Schutz und Sicherheit 125
6.10 Verwaltung 127

7. Dienste und Protokolle - Beispiele 130

7.1 Dienstleistungen der Sicherungsschicht 130
7.2 Protokolle der Sicherungsschicht 131
7.3 Dienstleistungen der Netzwerkschicht 132

7.4 Protokolle der Netzwerkschicht ... 132
7.4.1 Lokale Netze ... 134
7.4.1.1 Kopplung physikalischer Medien ... 134
7.4.1.2 Kopplung zugriffskontrollierter Medien ... 135
7.4.2 Subnetz-Kopplung ... 137
7.5 Dienstleistungen der Transportschicht ... 138
7.6 Protokolle der Transportschicht ... 139
7.7 Dienstleistungen der Kommunikationssteuerungsschicht ... 140
7.8 Protokolle der Kommunikationssteuerungsschicht ... 143
7.9 Dienstleistungen der Darstellungsschicht ... 144
7.10 Protokolle der Darstellungsschicht ... 146
7.11 Anwendungsdienste ... 146
7.11.1 Dateitransfer - FTAM ... 147
7.11.2 Gemeinsame Anwendungsdienstleistungstypen - CASE ... 149
7.11.2.1 Assoziationskontrolle - ACSE ... 149
7.11.2.2 Transaktionskontrolle - CCR ... 150
7.11.2.3 'Zuverlässige' Transferdienstleistungen - RTSE ... 154
7.11.2.4 Abgesetzte Operationen - ROSE ... 155
7.12 Protokolle der Anwendungsschicht ... 157
7.13 Anwendungsdienste / Anwendungen ... 158

8. Implementierung ... 159

8.1 Implementierung von Dienstschnittstellen ... 159
8.1.2 Schnittstelle über Prozeßgrenzen ... 160
8.1.2.1 Prozeß ... 160
8.1.2.2 Synchronisation ... 161
8.1.2.2.1 Warten und Fortsetzen ... 162
8.1.2.2.2 Rendezvous-Synchronisation ... 162
8.1.2.2.3 Varianten ... 162
8.1.2.2.4 Realisierung ... 162
8.1.2.3 Datenaustausch ... 163
8.1.2.3.1 Geteilter Speicher ... 163
8.1.2.3.2 Fremdzugreifbarer Speicher ... 164
8.1.2.3.3 Auftragsübergabe ... 164
8.1.2.3.4 Datenaustausch im Rendezvous ... 165
8.1.2.3.5 Nachrichtenaustausch ... 166
8.1.2.3.6 Monitore ... 167
8.1.2.3.7 Geteilte Puffer ... 168
8.1.2.4 Dienstschnittstellenimplementierung ... 168
8.1.2.4.1 Zugangspunktimplementierung durch ein Kanalpaar ... 168
8.1.2.4.2 Lokale Nachrichtenquittungen ... 169
8.1.2.4.3 Verbindungsendpunkte ... 170
8.1.2.4.4 Ausbleibende Partnerreaktion ... 170
8.1.2.4.5 Lange Nachrichten ... 170
8.1.2.4.6 Sehr lange Nachrichten ... 170
8.1.2.4.7 Rendezvous-Austausch ... 171
8.1.2.4.8 Spezielle Monitore ... 171
8.1.3 Prozeßinterne Schnittstelle ... 171
8.1.3.1 Ereignispuffernde Schnittstelle ... 171
8.1.3.2 Aktionsorientierte Schnittstelle ... 172

8.1.3.3 Mischformen ... 173
8.1.3.3.1 Downcall-Schnittstelle ... 174
8.1.3.3.2 Upcall-Schnittstelle ... 174
8.2 Implementierung von Instanzen ... 174
8.2.1 Instanzen ... 175
8.2.1.1 Zustandsdarstellung ... 176
8.2.1.1.1 Hauptzustand ... 177
8.2.1.1.1.1 Datenstruktur-Darstellung ... 177
8.2.1.1.1.2 Befehlszähler-Darstellung ... 177
8.2.1.1.2 Nebenzustände ... 178
8.2.1.1.2.1 Aktuelles Eingabe-Ereignis ... 178
8.2.1.1.2.2 Dienstleistungszusammenhang ... 179
8.2.1.1.2.3 Verbindungszusammenhang ... 179
8.2.1.1.2.4 Nutzdaten-Verwaltung ... 179
8.2.1.1.2.5 Verwaltung des Basisdienstes ... 180
8.2.1.2 PDU-Analyse ... 180
8.2.1.2.1 Paketweise Bearbeitung ... 180
8.2.1.2.2 Segmentweise Bearbeitung ... 180
8.2.1.3 PDU-Aufbau ... 181
8.2.1.3.1 Paketweise Bearbeitung ... 181
8.2.1.3.2 Segmentweise Bearbeitung ... 181
8.2.1.4 Aktivierung ... 181
8.2.1.4.1 Erwarten von Ereignissen ... 181
8.2.1.4.2 Ausführungsstart durch Ereignisse ... 182
8.2.1.5 Selektion ... 182
8.2.1.5.1 Programmierte Selektion ... 183
8.2.1.5.2 Tabellengesteuerte Selektion ... 183
8.2.1.6 Zeitüberwachung ... 183
8.2.1.6.1 Zeitaufträge über Prozeßgrenzen ... 184
8.2.1.6.2 Prozeßinterne Zeitaufträge ... 184
8.2.1.7 Grundlegende Software-Architekturen ... 185
8.2.1.7.1 Zentralverteiler ... 185
8.2.1.7.2 Sammlung von Ereignisbearbeitungsprozeduren ... 188
8.2.2 Instanzen und Prozesse ... 190
8.2.2.1 Prozeßmengen ... 191
8.2.2.2 Leichtgewicht-Prozesse ... 191
8.2.2.3 Integration mehrerer Instanzen ... 192
8.2.2.3.1 Zentralverteiler-Architektur ... 192
8.2.2.3.2 Prozedursammlung ... 192
8.2.2.3.3 Kombination Server/Activity-Thread ... 192
8.2.3 Programm- und Programmierumgebung ... 193
8.2.3.1 Systemimplementierungssprachen ... 193
8.2.3.2 Höhere Programmiersprachen ... 193
8.2.3.3 Einbettungssysteme ... 193
8.2.3.4 Schnelle Prototyp-Erzeugung ... 194
8.2.3.5 Implementierungsgeneratoren ... 194

9. Rechnernetz-Anwendungen ... 195

9.1 Besonderheiten von Rechnernetz-Anwendungen ... 195
9.2 Architektur ... 196
9.2.1 Spektrum ... 197

9.2.1.1 Komponenten 197
9.2.1.2 Funktionen 198
9.2.1.3 Interaktionen 198
9.2.2 Paradigmen 199
9.2.2.1 System kommunizierender Prozesse 199
9.2.2.2 Clients und Server 199
9.2.2.3 Geteilte Objekte und kooperierende Prozesse 200
9.2.2.4 Datenbank und kooperierende Prozesse 201
9.2.2.5 Gemeinsame Aktionen und kooperierende Prozesse 201
9.2.2.6 System aktiver Objekte 202
9.2.2.7 Nachrichtenprozesse 202
9.3 Systemunterstützung 202
9.3.1 Arten den Systemunterstützung 203
9.3.1.1 Netzwerkfähige Stationen 203
9.3.1.2 Verteilte Programmiersprachen 205
9.3.1.3 Verteilte Betriebssysteme 207
9.3.1.4 Netzwerk-Betriebssysteme 209
9.3.1.5 Netzwerk-Betriebssysteme nach Client-Server-Modell 209
9.3.1.6 Verteilte Datenbanken 210
9.3.2 Beispiele 211
9.3.2.1 Unix-Sockets 211
9.3.2.2 Sun-RPC 212
9.3.2.3 OSF: DCE 213
9.3.2.4 ANSA 214
9.3.2.5 OMG OMA 215
9.4 Normung: ODP und DAF 215

10. Verteilte Algorithmen 217

10.1 Basismaschine 217
10.2 Beispiele 221
10.2.1 Echo-Algorithmus 222
10.2.2 Vertreter-Auswahl 223
10.2.3 Maximumsuche 224
10.2.4 Schnappschuß 226
10.2.5 Terminierungserkennung 232

1. Rechnernetze, verteilte Systeme und Telekommunikationssysteme

Die Praxis-Stammvorlesung 'Rechnernetze und verteilte Systeme' behandelt primär die Grundzüge der Architekturen von Rechnernetzen und Rechnernetz-Anwendungen.

Sie gliedert sich inhaltlich grob in 3 Teile:

- Telekommunikationssysteme,
- Rechnernetz-Anwendungen,
- Verteilte Algorithmen.

Der (umfangreichste) Teil 'Telekommunikationssysteme' behandelt die Kommunikationsaspekte im Rechnernetz. Wichtige Begriffe, wie Kommunikationsdienste und -protokolle sowie die sieben Schichten des inzwischen mindestens vom Namen her allgemein bekannten ISO/OSI-Modells werden eingeführt.

Die Nutzung eines Rechnernetzes durch verteilte bzw. Rechnernetz-Anwendungen klingt schon im Telekommunikationssystem-Teil an. Der Teil 'Rechnernetz-Anwendungen' befaßt sich mit der Systemunterstützung für verteilte Anwendungen im Rechnernetz sowie mit Entwurf und Architektur der Anwendungen.

Der letzte, nur in Ansätzen vorhandene Teil 'verteilte Algorithmen' befaßt sich mit Details von Anwendungen, nämlich mit den dort verwendeten Algorithmen (Verteilte Algorithmen finden allerdings nicht nur in verteilten Anwendungen Verwendung, auch Kommunikationsprotokolle, wie sie in Telekommunikationssystemen eingesetzt werden, können als verteilte Algorithmen bezeichnet werden). Es sind nebenläufige Algorithmen, die die besonderen Randbedingungen der verteilten Ausführung berücksichtigen müssen. Hier kann die Vorlesung aus Umfangsgründen nur einige erste Einblicke geben.

1.1 Rechnernetze

In den 60er Jahren entstand der Wunsch, Peripheriegeräte entfernt vom Rechner (z.B. am Ort einer Bankfiliale) aufzustellen. Sie wurden über Datenfernübertragungseinrichtungen mit dem Rechner (z.B. in der Zentrale) verbunden und ermöglichten so den Zugang zum entfernten Rechner. Mit solchen Einrichtungen zur Datenfernverarbeitung konnte einerseits auf einen (teueren) eigenen Rechner in der Filiale verzichtet werden, weil lokale Anwendungen der Filiale den Zentralrechner mitnutzen konnten. Andererseits konnten Anwendungen, die die Gesamtorganisation betrafen, Ein / Ausgaben relativ direkt (sozusagen 'On Line') mit örtlich entfernten Endstellen austauschen.

Mit der Verbesserung des Preis / Leistungsverhältnisses der Rechner in den späten 60er Jahren und dem - durch wachsende Anwendungsfelder - steigenden Bedarf nach Rechenleistung ergab sich, daß Organisationen nicht mehr nur eine einzige zentrale Rechnerkonfiguration einsetzten, sondern daß einzelne Filialen und Abteilungen mit eigenen Rechnern ausgestattet wurden, um den Zentralrechner zu entlasten und die für die Datenfernübertragung entstehenden Kosten zu reduzieren. Anwendungen im Problembereich der Verwaltung einer Organisation insgesamt wurden nach wie vor vom Zentral-

rechner getragen. Lokale Anwendungen konnten isoliert vor Ort abgewickelt werden. Um den Datenaustausch zwischen Zentralrechner und Filialrechnern, später auch zwischen den Filialrechnern zu beschleunigen, war es naheliegend, an die Fernübertragungseinrichtungen auf beiden Seiten einer Verbindung Rechner und nicht Peripheriegeräte anzuschließen. Zunächst ergab sich so eine Sternstruktur mit dem Zentralrechner als Mittelpunkt. Filialrechner waren aus der Sicht des Zentralrechners untergeordnete Peripheriegeräte.

Es wurden schließlich auch (z.B. bei sehr großen Organisationen oder bei Verbänden aus befreundeten Organisationen) Zentralrechner miteinander verbunden. Man kam damit zwangsläufig vom ersten unsymmetrischen Konzept ab. Es entstanden Netze aus gleichrangigen Rechnerknoten und die Kopplung zwischen den Rechnern konnte nicht mehr dem Rechner-Peripherie-Schema folgen. Mit Beginn der 70er Jahre wuchs ferner auch der Bedarf für Filialrechner-Filialrechner-Datenaustausch. Die Filialrechner wurden ebenfalls zu gleichberechtigten Netzknoten. Es entstanden die ersten großen Rechnernetze.

Hauptziel der Netzbildung war dabei in erster Linie die Datenaustauschmöglichkeit. Dateien eines Rechners konnten mit entprechender Netzsoftware per Kopie an andere Rechner übertragen werden (Dateitransfer). Es folgten jedoch sehr schnell weitere Netzdienstleistungen, die an den Datenfernverarbeitungsbetrieb angelehnt waren, man konnte z.B. von einem Rechner aus Prozesse an einem anderen erzeugen (Remote Job Entry), Konsolnachrichten übermitteln und entfernte Rechner urladen (Downline Booting).

Die Ziele bei der Realisierung von Rechnernetzen wurden unter den drei folgenden Schlagwörtern zusammengefaßt:

- **Funktionsverbund**;
  die Vernetzung ermöglicht es Funktionen entfernter Rechner zu nutzen (z.B. komfortabler Zugang zu Rechenleistungen eines Spezialrechners).
- **Datenverbund**;
  Daten, die örtlich im Zugriff von Station A sind, können über das Netz auch von B aus zugegriffen werden,
- **Lastverbund**;
  zwischen relativ gleichartigen Rechnern können Aufgaben verlagert werden um lokale Engpässe auszugleichen.

Hinzu kam bald der Aspekt des gnädigen Leistungsabfalls (Graceful Degradation) eines Netzes bei Ausfällen einzelner Stationen oder Verbindungen. Der Gesamtbetrieb wird nur teilweise gestört. Nach Ausgliederung der gestörten Teile aus dem Rechnernetz verbleibt ein funktionsfähiges Restsystem (**Zuverlässigkeitsverbund**).

Umgekehrt kann ein Rechnernetz in sehr flexibler Weise erweitert werden. Wenn die Gesamtrechenleistung erhöht werden soll, können einzelne Knoten in verhältnismäßig einfacher Weise neu hinzugefügt oder durch leistungsfähigere ersetzt werden (**flexible Erweiterbarkeit**).

Wenn es dabei weniger auf den Bedarf zukünftiger Anforderungen als vielmehr darauf ankommt, eine aktuell geforderte hohe Rechenleistung zu erbringen, spricht man vom **Leistungsverbund**. Man erhofft sich, daß ein Netz aus preiswerten Standardstationen sehr teuere Groß- oder Spezialrechner ersetzen kann.
Die Einsatzgebiete sind unterschiedlich: Verteiltes Hochleistungsrechnen will 'Supercomputer', verteilte Transaktionsbearbeitungssysteme wollen zentrale Datenbasis-verwaltende Großrechner ersetzen. Da lange Nachrichtenlaufzeiten zwischen Stationen die Leistung herabsetzen, handelt es sich bei näherem Hinsehen allerdings sehr oft nicht um Rechnernetze sondern um räumlich eng gekoppelte Mehrrechnersysteme.
Im echten (räumlich weit verteilten) Rechnernetz wird, um bzgl. einer bestimmten Anwendung eine hohe Gesamtleistung zu erzielen, die Anwendung selbst dezentralisiert und als verteilte Anwendung

entworfen (**Dezentralisierung**). Verteilte Anwendungsprogamme werden dadurch ausgeführt, daß verschiedene Ausführungsprozesse an verschiedenen Orten zueinander nebenläufig ablaufen und gelegentlich miteinander kommunizieren. Durch geeignete Algorithmen können Parallelisierungsbarrieren vermindert werden. Durch günstige Ortszuordnung von Datenbasis-Teilen kann der Kommunikationsbedarf vermindert werden (**Lokalität**).

Verteilte Anwendungen können nicht nur aus der Dezentralisierung ehemals zentraler Anwendungen entstehen. Da die Anwendung nun verschiedene Orte erfaßt, können neue Probleme gelöst werden, nämlich solche, die aus einer Kombination von Datenverarbeitungs- und Datentransport-Problemen bestehen (**Ortsverbund**). Hierzu zählen interaktive Telekooperationsanwendungen und verteilte Informationssysteme.

1.2 Rechnernetze und verteilte Systeme

Die beiden Begriffe Rechnernetz und verteiltes System werden häufig in gleicher Bedeutung verwendet.

A. Tanenbaum trifft eine Unterscheidung, nach der Rechnernetz ein System aus gekoppelten aber autonomen und i.a. auch heterogenen Rechnerknoten bezeichnet, während beim Begriff verteiltes System die technische Zusammensetzung des unterliegenden Rechnernetzes wie überhaupt seine Gliederung in einzelne autonome Knoten verdeckt wird. In extremer Sicht ist ein verteiltes System - wie eine traditionelle isolierte Rechnerkonfiguration - eine einzige homogene Maschine mit der Besonderheit, daß unterschiedliche Orte vorhanden sind und Prozesse und Daten an Orten lokalisiert sind.

Eine andere Unterscheidung versteht unter Rechnernetz wie oben ebenfalls ein System aus einzelnen gekoppelten Rechnerknoten, das relativ zweckneutral als besondere Form einer Basismaschine angesehen wird. Beim Begriff verteiltes System steht dagegen ein genauer umrissener Systemzweck im Vordergrund. Ein verteiltes System ist eine Rechneranwendung, bei der Daten und Prozesse an verschiedenen Orten vorhanden sind (verteilte Anwendung). Verteilte Anwendungen werden mittels Rechnernetzen implementiert, so daß die beiden Begriffe verteilte Anwendung und Rechnernetz-Anwendung synonym sind.

Die in der ersten Unterscheidung hervorgehobene Homogenität der Basismaschine 'verteiltes System' sollte nicht überbewertet werden. Sie verspricht zwar eine Vereinfachung des Anwendungsentwurfs, weil Details des aus verschiedenen Stationen bestehenden Rechnernetzes verdeckt werden. Beim Entwurf praxisrelevanter verteilter Anwendungen kann der Rechnernetz-Charakter der Basismaschine aber kaum vernachlässigt werden. Dem stehen die Randbedingungen der technischen Realisierung, d.h. Kosten- und Leistungsaspekte entgegen. Ursachen hierfür sind:

- Lose Kopplung;
  der Datenaustausch zwischen entfernten Rechnern verursacht nicht-vernachlässigbare Kosten; die Laufzeiten für Nachrichten sind verhältnismäßig lang; die Verbindungen können jederzeit getrennt werden; durch die lange Laufzeit können sich zufällige Störeinflüsse viel häufiger auswirken; die Wahrscheinlichkeit dafür, daß Signale auf dem Weg verfälscht werden, kann nicht vernachlässigt werden; die Zeit, die benötigt wird, um eine Nachricht unverfälscht zu übertragen, ist nicht vorhersagbar.
- Nebenläufigkeit;
  es gibt mehrere Stationen, die eigenständig und zeitlich simultan Verarbeitungsprozesse ausführen können. Zur Ausnutzung dieser Fähigkeit sollte eine Problemlösung so entworfen werden, daß sie durch eine Menge schwach synchronisierter Prozesse erbracht wird.
- Dezentrale Kontrolle;
  die Abläufe in den einzelnen Stationen eines verteilten Systems werden möglichst voneinander

entkoppelt. Jede Station steuert ihre eigenen Abläufe soweit als möglich selbst (durch eine lokale Betriebsorganisation). Man verzichtet darauf, daß an einem Ort eine vollständige und aktuelle Sicht des globalen Systemzustands verfügbar gemacht wird. Jede Station hat von den Zuständen an den anderen Stationen nur eine Teilsicht, insbesondere können auch an einzelnen Stationen Ausfälle auftreten, ohne daß dies sofort an den anderen Stationen erkennbar wird.

Diese Eigenschaften bedingen , daß beim Entwurf eines verteilten Systems Kommunikationskosten, -leistung und -störungen sowie die Unabhängigkeit der Abläufe an unterschiedlichen Stationen berücksichtigt werden müssen, wenn effiziente und stabile Lösungen erzielt werden sollen.

Für verteilte Systeme sind ferner folgende Eigenschaften typisch und schlagen sich im Entwurf nieder:

- Systemlebensdauer;
  während eine typische lokale Anwendung innerhalb weniger Stunden terminiert, soll ein verteiltes System oft ein sich in der Realität sehr langfristig stellendes Problem lösen. Ein Flugbuchungssystem sollte z.B. das überhaupt nicht zeitlich terminierte Problem lösen, alle zukünftigen Flüge der angeschlossenen Fluggesellschaften den angeschlossenen Reisebüros buchbar zu machen. Es sollte auf Jahre hinaus arbeiten, ohne daß - die in diesem Zeitraum mit Sicherheit auftretenden - Komponentenausfälle die Systemfunktion stören. Fehler- und Ausfalltoleranzaspekte müssen also in den Entwurf einbezogen werden.
- Systemgröße;
  die hohe Systemgröße (Anzahl der Komponenten, der enthaltenen Funktionen, die logische Verflechtung der Funktionen) kann im Entwurf nur durch sorgfältige Modularisierung bewältigt werden. Im Verbund mit der hohen Lebensdauer muß damit gerechnet werden, daß sich Aufgabenstellung und geeignete Systemauslegung im Lauf der Betriebszeit ändern. Das verteilte System sollte also modular ausgelegt und in Modulen (zur Laufzeit) flexibel in seiner Konfiguration veränderbar sein.

1.3 Rechnernetze und Telekommunikationssysteme

Ein Rechnernetz besteht aus einer Menge von Rechnern, die örtlich voneinander entfernt installiert sind, und aus Einrichtungen, die den Austausch von Daten zwischen den Rechnern ermöglichen.

Vernachlässigen wir, daß jeder einzelne Rechner auch lokal unabhängig vom Gesamtnetz betrieben werden kann, und vernachlässigen wir zunächst auch, daß das Rechnernetz insgesamt als Basismaschine für Rechneranwendungen - dann eben Rechnernetz-Anwendungen - betrachtet werden kann, dann ergibt sich eine Sicht, in der an verschiedenen Orten (den Standorten der einzelnen Rechner) durch dort lokale Vorgänge einerseits Daten erzeugt werden, die an andere Orte zu übertragen sind, und andererseits entfernt erzeugte Daten gelesen werden sollen. Das Rechnernetz ist ein Telekommunikationssystem. Daten werden nicht nur verarbeitet, sondern auch über (oft weite) Strecken transportiert.

Im Vergleich zu traditionellen Telekommunikationssystemen (z.B. Telephon-Dienst, Brief-Dienst) werden im Rechnernetz Informationen digital codiert und transportiert, und die systeminternen Abläufe zur Aufbereitung der Daten und Kontrolle ihres Austauschs können mit allen Vor- und Nachteilen programmgesteuert stattfinden. Die Vorteile überwiegen aus technischer Sicht. So besteht die Tendenz, auch Nicht-Rechner-Rechner-Kommunikation durch digitale Telekommunikationssysteme zu unterstützen (z.B. Telefax, BTX, digitales Telephonieren, Einführung von ISDN). Es bestehen zwei Beziehungen zwischen Rechnernetzen und (modernen digitalen) Telekommunikationssystemen:

- Ein Rechnernetz enthält und nutzt als wichtige systemtragende Bestandteile Telekommunikationssysteme,
- Moderne Telekommunikationssysteme werden per Rechnernetzen realisiert.

Die (im wesentlichen auch mit internationalen Standardisierungsarbeiten einhergehende) Begriffsbildung zu und Betrachtungsweise von Telekommunikationssystemen hat deshalb großen Einfluß auf Konzeption und Architektur von Rechnernetzen.

Ein großer Teil des Vorlesungsstoffs ist also dem Begriff 'Telekommunikationssystem' und der Implementierung von Telekommunikationssystemen mittels vernetzter Rechner gewidmet. Als Zweck des Telekommunikationssystems / Rechnernetzes wird die Erbringung von (Tele-) Kommunikationsdienstleistungen in den Vordergrund gestellt. Der eigentliche, produktive Zweck eines Rechnernetzes, der darin besteht, (verteilte) Anwendungen auszuführen, tritt noch in den Hintergrund, wird aber nicht unsichtbar, weil es wiederum der Zweck der Kommunikationsdienste ist, eben die Kommunikationsanforderungen der verteilten Anwendungen zu erfüllen.

1.4 Rechnernetze und Datennetze

Häufig werden Netze mit den Kürzeln LAN (für Local Area Network bzw. lokales Netz), WAN (für Wide Area Network bzw. Weitverkehrsnetz oder geographisch verteiltes Netz) und MAN (für Metropolitan Area Network) bezeichnet. Auch das Kürzel DAN (für Desktop Area Network bzw. Schreibtisch-Netz) tritt schon stellenweise auf. Man erkennt eine Hierarchie vom räumlich sehr eng gefaßten DAN über LAN und MAN zum WAN.

Der Sprachgebrauch unterscheidet hier nicht zwischen Rechnernetz und reinem Kommunikationssystem, das mit Datennetz bezeichnet wird. Trotzdem sollte man sich dessen bewußt sein, daß ein Rechnernetz, das produktive Anwendungen tragen soll und bei dem es auf die integrale Funktionalität aus Datenverarbeitung und Transport ankommt, mehr ist als ein Datennetz, dessen Aufgaben sich auf die dienstleistende Zurverfügungstellung von Kommunikationsmöglichkeiten beschränkt. Wenn z.B. das Kürzel LAN aus Kommunikationssystem-Sicht benutzt wird, hat es eine andere Bedeutung als in der Rechnernetz-Sicht.

Das Kürzel VAN (für Value-added Network bzw. Netz mit Mehrwertdiensten, manchmal auch intelligentes Netz genannt) steht übrigens dazwischen. Es ist aus der Sicht der Telekommunikationssysteme geprägt, d.h. es geht davon aus, daß "Netz" ein Datennetz bezeichnet, und hebt aus dieser Sicht hervor, daß auch datenverarbeitende Funktionalität (Informationssystem-Funktionen: Auskunftsdienste, Verzeichnisdienste, Datenhaltungsmöglichkeiten, Vorgangsverwaltung) zusätzlich zur Kommunikationsunterstützung im Netz vorhanden ist und den Netznutzern angeboten wird.

1.5 Literaturhinweise

Als ergänzende Lektüre seien folgende Bücher empfohlen.

A : Umfassende Einführungen in den Themenkreis

Lockemann, P.C.; Krüger, G.; Krumm, H.; Telekommunikation und Datenhaltung; Hanser Verlag (1993)

Tanenbaum, A.; Computer Networks, Prentice Hall (1989)

Sloman, M., Kramer, J.; Verteilte Systeme und Rechnernetze, Hanser Verlag (1988)

Kerner,H.; Rechnernetze nach ISO/OSI, Kerner (1989)

B: Vertiefung spezieller Aspekte:

Bocker,P.; ISDN, Springer (1987)
zukünftiges öffentliches Datennetz ISDN

Hammond, J., O'Reilly, P.; Performance Analysis of Local Computer Networks; Addison-Wesley, 1986

Henshall, J., Shaw, S.; OSI Explained, Wiley (1989)
anwendungsorientierte Schichten des ISO/OSI-Modells

Jennings,F.; Practical Data Communications, Blackwell (1986)

Mattern, F.; Verteilte Basisalgorithmen, Springer IFB (1991)

Schwartz,M.; Telecommunication Networks, Addison Wesley (1987)
Leistungsaspekte

Stallings, W.; Data and Computer Communications, Macmillan (1988)
Standards

Stallings,W.; Local Networks - An Introduction, Macmillan (1987)
Lokale Netze, Standards

Stevens, W.; UNIX Network Programming, Prentice Hall (1990)
Programmierung für vernetzte Unix-Stationen

Walke, B.; Datenkommunikation I, Teile 1 und 2, Hüthig (1987)
Nachrichtentechnik zu unteren Schichten des ISO/OSI-Modells.

2. Telekommunikationssysteme

Ein Telekommunikationssystem besteht aus einer Menge von Dienstnehmern und dem Erbringer eines Telekommunikationsdienstes (i.a. können es auch mehrere Diensterbringer sein, und ein Erbringer kann auch mehrere unterschiedliche Telekommunikationsdienste unterstützen). Bild 2.1 zeigt ein entsprechendes Szenario.

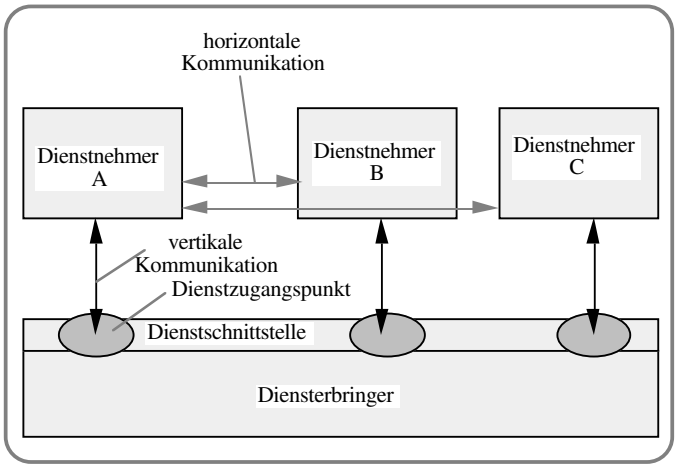

Bild 2.1: Telekommunikationssystem

Dienstnehmer / Nutzer / Teilnehmer im Telekommunikationssystem sind einzelne Handlungsträger / Individuen / Instanzen, die voneinander entfernt sind. Teilmengen der Dienstnehmer arbeiten an einer gemeinsamen Aufgabe zusammen und kommunizieren deshalb miteinander. Da sie voneinander entfernt sind, können sie nicht direkt miteinander kommunizieren sondern müssen Dienstleistungen eines Telekommunikationsdienst-Erbringers in Anspruch nehmen.

Ein Diensterbringer wird durch ein System aus Instanzen und Nachrichtenübertragungsmedien realisiert. Von diesem inneren Aufbau soll aber zunächst abstrahiert werden. Von außen erscheint der Diensterbringer als eine einzige Instanz. Sie hat eine Schnittstelle zu den Dienstnehmern, die Dienstschnittstelle.

Die Dienstschnittstelle ist in Dienstzugangspunkte gegliedert. Ein Dienstzugangspunkt bezeichnet eine an einem Ort gegebene Möglichkeit der Kommunikation zwischen Dienstnehmer und Erbringer (Beispiel: Telephonsteckdose, Briefkasten, Schalter im Postamt) und wird durch eine zugeordnete Adresse identifiziert.

Im Zusammenhang mit Telekommunikationssystemen bedeutet Kommunikation Informationsaustausch durch Nachrichtenübertragung. Grundlegend ist ein Zwei-Partner-Szenario, bestehend aus einem Sender (Quelle) und einem Empfänger (Senke). Der Sender möchte an den Empfänger Information übertragen. Hierzu codiert der Sender die Information in eine Nachricht (Nachricht: zum Zweck der Übertragung aufbereitete Daten; Daten: codierte - mit technischen Mitteln dargestellte - Information). Die Nachricht wird mittels eines Mediums vom Sender an den Empfänger übertragen. Der Empfänger interpretiert die Nachricht um die vom Sender darin hinterlegte Information zu gewinnen.

Im Telekommunikationssystem sind zwei Formen von Kommunikation beobachtbar:

- vertikale Kommunikation; ein Teilnehmer kommuniziert (an einem Ort, d.h. Zugangspunkt) mit dem Diensterbringer,
- horizontale Kommunikation; eine Menge von (zueinander entfernten) Teilnehmern kommuniziert miteinander.

Eine vertikale Kommunikationsaktion findet am Ort eines Dienstnehmers statt und betrifft als Kommunikationspartner diesen Dienstnehmer und den Diensterbringer. Sie dient der Übergabe von Aufträgen an den Diensterbringer oder der Auslieferung von Meldungen des Erbringers an den Dienstnehmer. Sie findet nicht zum Selbstzweck statt, sondern steht in Zusammenhang mit der Abwicklung horizontaler Kommunikation.

Eine horizontale Kommunikationsaktion betrifft eine Menge von Dienstnehmern. Sie stehen in einer Kooperationsbeziehung, d.h. sie arbeiten zur Erfüllung eines übergeordneten Systemzwecks zusammen und tauschen deshalb miteinander Informationen aus. Da die Teilnehmer zueinander örtlich entfernt sind, kann eine horizontale Kommunikationsaktion nicht direkt realisiert werden. Sie ist nur unter Nutzung von Dienstleistungen des Telekommunikationsdiensterbringers möglich, umfaßt also vertikale Kommunikationsaktionen.

Die Funktionalität eines Telekommunikationsdienst-Erbringers kann somit kurz als die eines Mediums zur Übertragung von Nachrichten bezeichnet werden. Ein Telekommunikationsdiensterbringer bietet den Dienstnehmern Dienstleistungen an, mit deren Hilfe Nachrichten an den Erbringer übergeben, vom Erbringer an einen anderen Ort übertragen und dort vom Erbringer an einen anderen Dienstnehmer ausgeliefert werden. Die zwischen Dientstnehmern horizontal ausgetauschten Nachrichten sollen dabei vom Erbringer nur transportiert, nicht aber interpretiert und verarbeitet werden.

Nach dieser kurzen Einführung des Begriffs Telekommunikationssystem werden in den folgenden Abschnitten des Kapitels wesentliche allgemeine Aspekte von Telekommunikationssystemen angesprochen.

In den nachfolgenden Kapiteln 3 bis 7 werden spezielle Aspekte technischer Telekommunikationssysteme konkretisiert. Sie bilden den Teil I 'Telekommunikationssysteme' des Skriptums.

Sichtweise und Begriffsbildung entsprechen dabei weitgehend den Festlegungen des inzwischen allgemein anerkannten Basis-Referenzmodells zur Verbindung offener Systeme der Internationalen Standardisierungsorganisation (dem sog. ISO/OSI-Modell; ISO: Basic Reference Model for Open Systems Interconnection). Ein im ISO/OSI-Sinn offenes System besteht aus unterschiedlichen und in den Details ihrer konkreten Gestalt nicht näher eingeschränkten Teilnehmern, die jedoch über genormte Dienste nach genormten Verfahren geregelt kommunizieren sollen, um im Sinne genormter Kooperationsbeziehungen erfolgreich zusammenzuarbeiten. Die benötigten Standards sind implementierungsunabhängig definiert und orientieren sich an der im Basisreferenzmodell definierten Sicht der logischen Architektur von Telekommunikationssystemen.

2.1 Logische Architektur von Telekommunikationssystemen

Bild 2.1 zeigte die grobe Struktur eines Telekommunikationssystems als System aus Dienstnehmern / Teilnehmern und Diensterbringer / Medium. Nun soll das in Bild 2.1 noch als 'Black Box' gezeigte Medium detaillierter betrachtet werden. Seine logische innere Struktur entspricht dabei einer Hierarchie von Schichten.

Bild 2.2 zeigt dazu ein System aus einer Menge von Teilnehmern $\mathbb{I}_{i+1}$ und einem Medium M_i. Das Medium M_i erbringt Dienstleistungen eines Telekommunikationsdienstes D_i. Zwischen den Teilnehmern $\mathbb{I}_{i+1}$ besteht eine Kooperationsbeziehung A_{i+1}.

In verfeinerter Betrachtung wird M_i durch ein System aus einer Menge von Instanzen $\mathbb{I}_i$ und einem untergeordneten Medium M_{i-1} gebildet. Das untergeordnete Medium M_{i-1} ist wiederum Erbringer eines Telekommunikationsdienstes, nämlich des Dienstes D_{i-1}.

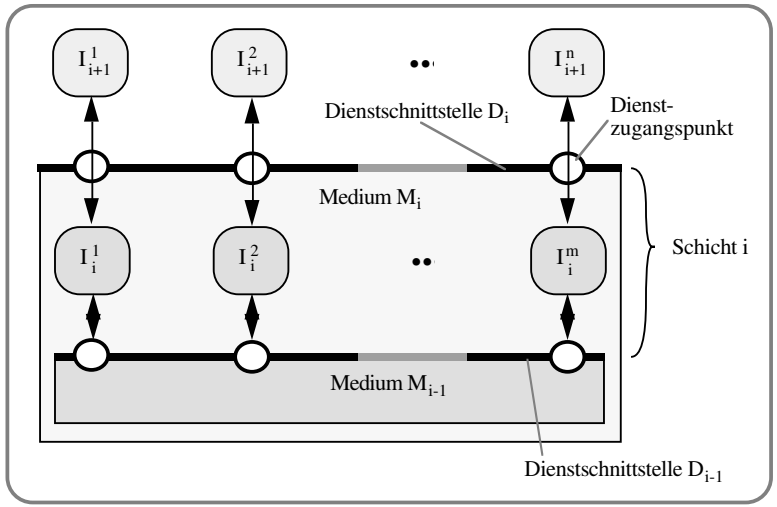

Bild 2.2: Logische Architektur von Telekommunikationssystemen

Die vertikale Kommunikation zwischen Teilnehmern aus $\mathbb{I}_{i+1}$ und dem Medium M_i kann nun verfeinert betrachtet werden. Sie besteht aus einzelnen Kommunikationsaktionen, die jeweils einen Teilnehmer aus $\mathbb{I}_{i+1}$ und eine Instanz aus $\mathbb{I}_i$ betreffen.

Im Blick auf das untergeordnete Medium bilden die Instanzen $\mathbb{I}_i$ die Teilnehmer eines Kommunikationssystems, das dem ursprünglichen hierarchisch untergeordnet ist. Die Kooperationsbeziehung A_i zwischen den Instanzen $\mathbb{I}_i$ dient dem Zweck, den Teilnehmern $\mathbb{I}_{i+1}$ den Dienst D_i zur Verfügung zu stellen, d.h. die gemäß D_i vorgesehenen einzelnen Dienstleistungen abzuwickeln.

Eine Instanz aus $\mathbb{I}_i$ kommuniziert horizontal mit anderen Instanzen aus $\mathbb{I}_i$ im Rahmen der Verfolgung dieser Kooperationsbeziehung A_i. Die horizontale Kommunikation wird über Dienstleistungen des untergeordneten Mediums M_{i-1} realisiert, führt also zu vertikalen Kommunikationsaktionen, die jeweils zwischen einzelnen Instanzen aus $\mathbb{I}_i$ und M_{i-1} stattfinden. Jede Instanz aus I_i kann durch ihr

Verhalten gegenüber den Dienstschnittstellen von M_i und M_{i-1} in Begriffen vertikaler Kommunikationsaktionen beschrieben werden.

Den Verbund aus Gliederung einer Schicht i in eine Instanzenmenge $\mathbb{I}_i$ mit der Festlegung des Verhaltens der einzelnen Instanzen bezüglich der Schnittstellen des zu erbringenden Dienstes D_i und des untergeordneten Dienstes D_{i-1} nennen wir das Telekommunikationsprotokoll P_i. Es wird immer im Zusammenhang mit den betroffenen Diensten D_i und D_{i-1} festgelegt.

Daß das ISO/OSI-Modell 7 Schichten definiert, und die genauere Aufgabenaufteilung der Schichten werden wir in späteren Kapiteln kennenlernen. Vorab nur der Hinweis, daß zwischen transportorientierten (unten in der Hierarchie) und anwendungsorientierten (oben in der Hierarchie) Diensten unterschieden wird:

- Bei einem transportorientierten Teleokommunikationsdienst erfolgt keine Bezugnahme zu einer enger eingegrenzten Kooperationsbeziehung der Dienstnehmer. Man geht davon aus, daß Dienstnehmer existieren und miteinander Nachrichten (sogenannte Nutzdaten, auch Nutzerdaten genannt) austauschen wollen. Die Reihenfolge, Zusammensetzung (von technischen Randbedingungen abgesehen) und Bedeutung der Nachrichten sind nicht weiter reglementiert.
- Bei anwendungsorientierten Diensten steht die Konzeption des Dienstes bereits mit einer Kooperationsbeziehung der Dienstnehmer im Zusammenhang, die in ihrer Art über den Datenaustausch hinaus festgelegt ist und z.B. Reihenfolge, Zusammensetzung und / oder Bedeutung von Nachrichten reglementiert.

Ein Hinweis noch zum Begriff der 'logischen' Architektur. Sie muß unterschieden werden von der konkreten Architektur (Implementierungsarchitektur) eines technischen Systems, wie sie sich aus der Zusammensetzung des Systems aus einzelnen realen Komponenten und der Verteilung der Gesamtfunktion auf die Komponenten sowie der Wechselwirkungen zwischen den Komponenten ergibt. Eine logische Architektur erklärt die Funktionsweise eines (komplexen) Systems durch Angabe einer inneren Struktur und Funktionsaufteilung, die sich am abstrakten Verständnis des Systems orientiert. Sie muß nicht mit der konkreten Architektur einer Implementierung korrespondieren sondern dient nur als Vorgabe im 'als ob'-Sinn: Eine dem logischen Modell entsprechende Implementierung des Systems muß sich an seinen Schnittstellen nach außen so verhalten, wie sich eine 1:1-Umsetzung des logischen Modells verhalten würde (Implementierungen transportorientierter Dienste besitzen häufig eine direkt zum logischen Modell - oder zu einer Verfeinerung des logischen Modells - korrespondierende Architektur; effiziente Implementierungen anwendungsorientierter Dienste dagegen selten).

2.2 Funktionalität von Telekommunikationsdiensten

Ein Telekommunikationsdienst hat die Aufgabe, für eine Menge zueinander örtlich entfernter Dienstnehmer Dienstleistungen zu erbringen, welche den Dienstnehmern Kommunikation gestatten.

Kommunikation räumlich entfernter Dienstnehmer findet im Grundprinzip durch den Austausch von Nachrichten zwischen ihnen statt. Ein Dienstnehmer, der einem anderen eine bestimmte Information übermitteln will, erzeugt eine physikalische Repräsentation hierfür, eine Menge von Daten, welche die Nachricht bildet. Er übergibt diese Nachricht innerhalb eines Übertragungsauftrags zu einem Absendezeitpunkt an einen Telekommunikationsdienst (vertikale Kommunikationsaktion). Der Dienst überbrückt die räumliche Distanz (innerhalb des Mediums stattfindende horizontale Kommunikationsaktionen) und liefert die Nachricht an den als Empfänger vorgesehenen Dienstnehmer aus (vertikale Kommunikationsaktion). Mit der Auslieferung ist die entsprechende Dienstleistung des Telekommunikationsdienstes abgeschlossen. Um auch die horizontale Kommunikationsaktion zwischen den Dienstnehmern erfolgreich zu vervollständigen, muß der Empfänger nun aus dem Zeitpunkt des Empfangs und / oder den Daten der Nachricht die vom Absender gewollte Information gewinnen.

Anschließend sollte der Empfänger, damit die horizontale Kommunikationsaktion auch im Sinne der zwischen den Dienstnehmern bestehenden Kooperationsbeziehung erfolgreich ist, entprechend dieser Information handeln.

Ausgehend von diesem einfachen Ablauf einer horizontalen Kommunikationsaktion und der enthaltenen einfachen Dienstleistung des Telekommunikationsdienstes soll im folgenden eine Übersicht über das Spektrum verschiedener Typen von Dienstleistungen gegeben werden. Hierzu werden im folgenden besprochen:

- die abstrakte Sicht einer Dienstschnittstelle,
- die Rolle eines Dienstnehmers aus der Sicht des Dienstes bei der Abwicklung einer Dienstleistung,
- Grundformen von Dienstleistungen, die in Abhängigkeit zur Anzahl betroffener Dienstnehmer und zur Anzahl erfolgender Nachrichtenübertragungen definiert werden können,
- Dienstleistungskontext,
- die Adressierung betroffener Dienstnehmer,
- Disziplin bei der Auslieferung von Nachrichten,
- Richtung des Nachrichtenaustauschs.

2.2.1 Dienstschnittstelle

Die Schnittstelle zwischen einem Medium und seinen Dienstnehmern ist in ihrer statischen und in ihrer dynamischen Zusammensetzung von Interesse. In der statischen Sicht besteht eine Dienstschnittstelle aus einer Menge von Orten (d.h. von Dienstzugangspunkten), an denen die vertikale Kommunikation zwischen Dienstnehmern und Medium im einzelnen als lokalisierbar angenommen wird. In der dynamischen Sicht müssen die im Ablauf der Zeit möglichen vertikalen Kommunikationsaktionen ersichtlich werden.

Statisch wird eine Dienstschnittstelle durch die Menge von Dienstzugangspunkten erklärt. Jedem Zugangspunkt ist eine Identifikation, die sogenannte Zugangspunktadresse zugeordnet. Zugangspunkte und ihre Adreßzuordnung werden in der Regel als zum Diensterbringer gehörig und unter dessen Verwaltung stehend betrachtet (z.B. Telephonnummern-Zuordnung durch Post). Zusammen mit einem Medium sind seine Zugangspunkte und deren Adressen festgelegt.

Die Unterscheidung zwischen einzelnen Dienstnehmern wird vom Medium ausschließlich nach Zugangspunktadressen vorgenommen. Ein Dienst sieht im Konzept vor, daß an jeden Zugangspunkt genau ein Dienstnehmer fest angeschlossen ist. Sowohl Dienstnehmer als auch Zugangspunkte können mit Zugangspunktadressen identifiziert werden. So wird auch im Telephondienst von der Post her nicht zwischen den einzelnen, tatsächlich ein Gespräch führenden Personen unterschieden, die Post erbringt ihre Dienstleistung nur in Bezug auf die beiden betroffenen, durch Telephonnummern identifizierten Zugangspunkte.

Die dynamische Sicht einer Dienstschnittstelle ergibt sich daraus, daß zwischen Dienstnehmern und Medium im Ablauf der Zeit einzelne vertikale Kommunikationsaktionen stattfinden können. Eine Kommunikationsaktion besitzt folgende relevanten Attribute:

- Zeitpunkt,
- Ort,
- Richtung,
- Typ und Kontrollparameter,
- Nutzdaten / Nutzinformation.

Logisch kann eine vertikale Kommunikationsaktion als zeitlich atomar verlaufend angesehen werden, da ihre detaillierte Abwicklung im Ablauf der Zeit nicht von Interesse ist. Man nennt deshalb auch vertikale Kommunikationsaktionen Dienstschnittstellen-Ereignisse.

Ein Dienstschnittstellen-Ereignis findet an genau einem Ort, d.h. an einem Zugangspunkt statt. Der Ort wird durch die Adresse des betroffenen Zugangspunkts repräsentiert.

Die Richtung eines Ereignisses ergibt sich daraus, auf wessen Initiative hin es erzeugt wird:
- Stimulus, eine von einem Dienstnehmer initiierte Kommunikationsaktion,
- Reaktion, eine vom Medium initiierte Kommunikationsaktion.

So ist der Einwurf eines Briefs durch eine Person in einen gelben Briefkasten ein Stimulus, der Einwurf des Briefs in einen Wohnungsbriefkasten durch den Briefträger eine Reaktion der Briefpost.

Mit einem Stimulus übergibt ein Dienstnehmer Information an das Medium. Zu der implizit übergebenen, vom Medium aus Zeitpunkt und Ort ableitbaren Information kommt in der Regel noch weitere. Sie kann - wie in der Informatik üblich - als durch Datenwerte repräsentiert angenommen werden. Dem Ereignis sind entsprechende Datenparameter zugeordnet. Auch eine Reaktion besitzt solche Datenparameter, um Information vom Medium zum Dienstnehmer zu übermitteln.

Die in Datenparametern repräsentierte Information kann in zwei Klassen gegliedert werden:
- Kontrollinformation, die vom Medium ausgewertet oder bestimmt wird,
- Nutzinformation, die von Dienstnehmern sowohl bestimmt als auch ausgewertet wird.

Ereignistyp und Kontrollparameter eines Stimulus werden vom Medium selbst interpretiert. Hiermit teilt der erzeugende Dienstnehmer dem Medium z.B. die Art und spezielle Auslegung der gewünschten Dienstleistung mit. Auf einem Brief stehen so als Kontrollparameter z.B. die Adresse und zur Typbestimmung der Vermerk 'Expresszustellung'.

Bei einer Reaktion werden Ereignistyp und Kontrollparameter vom Medium festgelegt und sind vom Dienstnehmer auszuwerten. Der Vermerk 'Empfänger unbekannt verzogen' kann so z.B. als Kontrollparameter angesehen werden.

Nutzinformation wird i.a. durch einen Parameter namens Nutzdaten mit einem Ereignis verbunden. Die Nutzdaten eines Stimulus bilden i.a. eine Nachricht, die der Dienst an einen anderen Dienstnehmer zu übermitteln hat und diesem per Nutzdaten-Parameter einer Reaktion übergibt. Eine Dienstleistung 'Briefübermittlung' der Post, zeigt sich an der Dienstschnittstelle am Zugangspunkt eines gelben Briefkastens als Stimulus des Typs 'Briefeinwurf', dessen Nutzdaten im Inhalt des Briefumschlags hinterlegt sind, sowie als Reaktion des Typs 'Briefauslieferung' an einem Wohnungsbriefkasten.

Bei der Beschreibung transportorientierter Dienste genügt es die Nutzdaten eines Ereignisses durch einen einzigen, in Zusammensetzung und Bedeutung nicht näher aufgeschlüsselten Parameter darzustellen. Bei anwendungsorientierten Diensten wird im Gegensatz hierzu der Dienst im Zusammenhang mit einer näher spezifizierten Kooperationsbeziehung der Dienstnehmer gesehen, und die Nutzdaten werden diesbezüglich aufgeschlüsselt und zerfallen in der Regel in verschiedene Nutzdaten-Parameter.

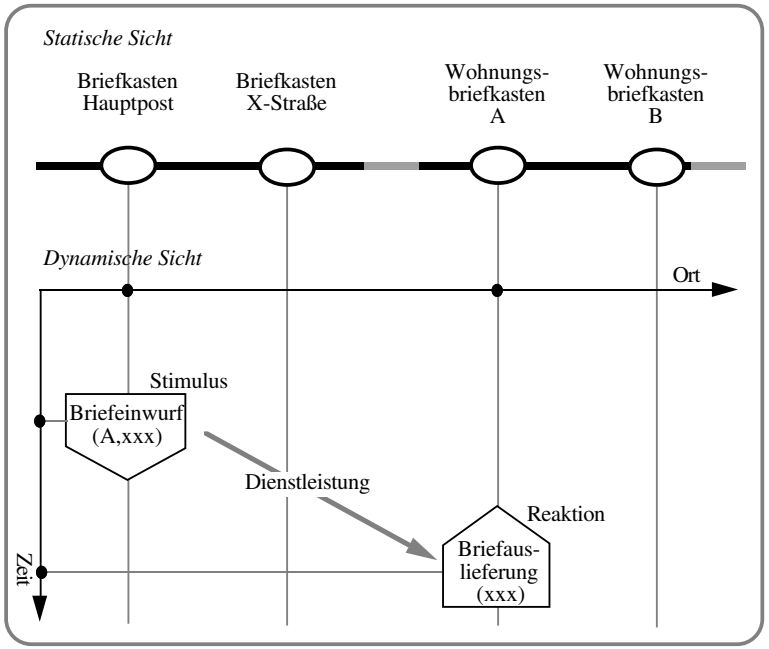

Bild 2.3: Dienstschnittstelle des Briefdienstes der Post

Bild 2.3 verdeutlicht die statische und dynamische Sicht einer Dienstschnittstelle am Beispiel des Briefdienstes und einer entsprechenden Dienstleistung graphisch.

2.2.2 Dienstnehmer-Rollen

Ein Medium erbringt eine Dienstleistung nur auf Anforderung eines Dienstnehmers. Für eine bestimmte Dienstleistung gibt es deshalb immer einen Dienstnehmer, der sie (durch einen Stimulus) initiiert hat. Er wird Initiator der Dienstleistung genannt.

Alle anderen Dienstnehmer, die zusätzlich zum Initiator noch von einer Dienstleistung berührt werden, werden zur Unterscheidung hierzu Beantworter (Responder) genannt. Sie werden hauptsächlich durch Reaktionen des Mediums in eine Dienstleistung einbezogen. Es gibt jedoch auch Dienstleistungen, die auch durch Beantworter-Stimuli beeinflußt werden können.

2.2.3 Grundformen von Dienstleistungen

Dienstleistungen können in Entsprechung zur Anzahl betroffener Dienstnehmer und zur Anzahl einbezogener Nachrichtenübermittlungen in Grundformen eingeteilt werden.

Zunächst soll von zwei beteiligten Dienstnehmern, einem Initiator und einem Beantworter ausgegangen werden. Es werden unterschieden:

- unbestätigte Dienstleistung,
- bestätigte Dienstleistung.

Eine unbestätigte Dienstleistung sieht eine einzige Nachrichtenübermittlung vom Initiator der Dienstleistung zum Beantworter vor. Sie zeigt sich an der Dienstschnittstelle mit zwei Ereignissen, einem Stimulus auf Initiatorseite und einer nachfolgenden Reaktion auf Beantworterseite. Der Stimulus wird i.a. Anforderung (Request), die Reaktion Anzeige (Indication) genannt.

Als Beispiel für eine unbestätigte Dienstleistung kann eine einfache Briefübermittlung - wie in Bild 2.3 gezeigt - dienen.

Eine bestätigte Dienstleistung sieht dagegen auch in umgekehrter Richtung eine Informationsübermittlung zwischen Dienstnehmern vor. Sie zeigt sich an der Dienstschnittstelle mit vier Ereignissen. Der Initiator fordert die Dienstleistung mit einem Anforderung (Request) genannten Stimulus an. Der Beantworter wird dann mit einer Anzeige (Indication) genannten Reaktion unterrichtet und ihm werden die vom Initiator gesendeten Nutzdaten übergeben. Nach Auswertung dieser Nachricht erzeugt der Beantworter eine Rückantwort. Er übergibt diese als Nutzdaten eines Stimulus, der Antwort (Response) genannt wird, an den Dienst. Erst wenn der Dienst anschließend auf Initiatorseite eine Reaktion als Bestätigung (Confirmation) erzeugt hat, gilt die Dienstleistung als abgeschlossen. Die Bestätigung liefert die Nutzdaten der Antwort an den Initiator aus.

Die folgende Unterscheidung nach der Anzahl beteiligter Dienstnehmer geht zur vereinfachten Erläuterung von einem unbestätigten Dienst, also dem Schema einer Benachrichtigung, aus:
- Zwei-Partner-Benachrichtigung,
- Gruppen-Benachrichtigung,
- Rundruf.

Die Zwei Partner-Benachrichtigung entspricht der oben erläuterten Form. Neben dem Initiator ist nur noch ein einziger Beantworter von der Dienstleistung betroffen.

Bei der Gruppen-Benachrichtigung (Multicast) gibt es eine Menge von Beantwortern. Auf die Anforderung des Initiators hin, die als Kontrollparameter eine Spezifikation der Zielgruppe enthält, wird vom Medium bei jedem Mitglied der Zielgruppe (d.h. an den zugeordneten Zugangspunkten) je eine Anzeige erzeugt. Jede dieser Anzeigen liefert die Nutzdaten der Anforderung aus.

Der Rundruf entspricht einer Gruppen-Benachrichtigung mit einer Zielgruppe, die außer dem Initiator alle Dienstnehmer des Mediums umfaßt. Ist auch der Initiator enthalten, spricht man vom Rundruf mit Mithören.

2.2.4 Verbindungen

Bisher wurde - auch in den Beispielen - von einfachen Dienstleistungen ausgegangen, bei welchen eine gewünschte Dienstleistung vollständig mit der Anforderung definiert wurde. So entnimmt die Post alle Kontrollinformationen zur Übermittlung eines Briefs den ihr mit dem Einwurf auf dem Umschlag übergebenen Kontrollparametern. Die Dienstleistung wird - wie sie den Dienstnehmern sichtbar wird - nicht zusätzlich durch andere, vorher abgewickelte Dienstleistungen und dort u.U. übergebene Kontrollparameter beeinflußt.

Solche Dienstleistungen werden Datagramm-Dienstleistungen (manchmal auch verbindungslos / connectionless) genannt, sie stehen nicht in einem sowohl für Dienstnehmer als auch für das Medium relevanten Zusammenhang mit anderen Dienstleistungen.

Im Gegensatz hierzu stehen Dienstleistungen, deren genaue Ausprägung auch von einem Kontext abhängt, der mit früheren Dienstleistungen aufgebaut wurde. Ein derartiger - sowohl Medium als

auch Dienstnehmer ersichtlicher - Zusammenhang hat häufig die Ausprägung einer Verbindung. Entsprechende Dienstleistungen und auch die sie umfassenden Dienste heißen verbindungsorientiert.

Verbindungsorientierte Dienste haben für die Dienstnehmer den Vorteil, daß sie weniger Kontrollinformation mit dem Medium austauschen müssen, weil sie sich auf frühere Festlegungen implizit beziehen können. Ferner können auch die Nutzdaten kürzer ausfallen, wenn der Dienstleistungskontext Verbindung per Konvention der Dienstnehmer mit einem zwischen ihnen - laut Kooperationsbeziehung bestehenden - logischen Zusammenhang gleichgesetzt wird. Einzelne Nachrichten müssen dann nicht mehr eine vollständige Beschreibung des Bezugs für den empfangenden Dienstnehmer enthalten, sondern er kann ihn aus vorher in derselben Verbindung empfangenen Nachrichten mitableiten.

Die verschiedenen Dienstleistungen eines verbindungsorientierten Dienstes sind sehr von ihrer traditionellen Realisierung durch Leitungsvermittlung geprägt (Telephon) und werden im allgemeinen in vier Klassen gegliedert:
- Dienstleistungen zum Aufbau einer Verbindung,
- Dienstleistungen zum Datentransfer über eine (aufgebaute) Verbindung,
- Dienstleistungen zum Abbau einer Verbindung,
- Dienstleistungen im Zusammenhang mit dem Abbruch einer Verbindung.

Die Nutzung der Dienstleistungen gliedert sich in drei Phasen:
- Verbindungsaufbau,
- Datenaustausch,
- Verbindungsabbau.

In der Aufbauphase wird i.a. mit einer bestätigten Dienstleistung eine Verbindung hergestellt. Mit erfolgreichem Ausgang der Dienstleistung 'Verbindungsaufbau' beginnt die Austauschphase. Nun können die verschiedenen zum Daten- oder Informationsaustausch zwischen den Dienstnehmern vorgesehenen Dienstleistungen angefordert und abgewickelt werden. Den Abschluß bildet die Abbauphase, sie beginnt damit, daß einer der beteiligten Dienstnehmer die Dienstleistung 'Verbindungsabbau' anfordert, und endet mit der entsprechenden Anzeige. Danach ist die Verbindung nicht mehr existent.

Bei echten, über größere Entfernungen angebotenen Telekommunikationsdiensten muß darüberhinaus immer der vorzeitige Abbruch einer Verbindung berücksichtigt werden. Einerseits kann das Medium selbst jederzeit die Fähigkeit zur weiteren Aufrechterhaltung einer Verbindung verlieren (z.B. Bagger durchtrennt Kabel zum Teilnehmer, Fernleitung muß für wichtigere Zwecke freigemacht werden). Andererseits kann auch ein beteiligter Dienstnehmer aufgrund nicht vorhersehbarer Einflüsse die Fähigkeit oder auch den Willen zur weiteren Aufrechterhaltung der Verbindung verlieren. Im ersten Fall bricht das Medium die Verbindung ab und zeigt dies mit der Dienstleistung Erbringer-Abbruch (Provider-Abort) an, die sich in je einer Abbruch-Anzeige je beteiligtem Dienstnehmer äußert. Im zweiten Fall fordert ein Dienstnehmer die unbestätigte Dienstleistung Dienstnehmer-Abbruch (User-Abort) an. Sie wird vom Medium durch Erzeugung von Abbruch-Anzeigen bei allen übrigen betroffenen Dienstnehmern erbracht.

2.2.5 Auslieferungsdisziplin

Unter Auslieferungsdisziplin soll verstanden werden, ob und welcher Bezug zwischen der zeitlichen Reihenfolge von Stimuli und der zugeordneter Reaktionen nach dem Konzept eines Dienstes besteht.

Bei Datagramm-Dienstleistungen besteht kein solcher Zusammenhang. Die Reihenfolge des Einwurfs mehrerer Briefe auch in denselben Briefkasten erlaubt es nicht, auf die Reihenfolge ihrer Auslieferung zu schließen.

Bei verbindungsorientierten Diensten wird sehr oft mit einer Verbindung auch eine Auslieferungsdisziplin festgelegt. Und zwar je Richtung durch eine eigene FIFO-Beziehung. Alle Reaktionen auf einer Seite erfolgen zeitlich entsprechend der Reihenfolge, in der die verursachenden Stimuli auf der anderen Seite der Verbindung aufgetreten sind.

Manche Dienste sehen darüberhinaus die Priorisierung von Dienstleistungsanforderungen vor. Üblich sind zwei Stufen 'vorrangig (expedited / urgent / out of bound)' und 'normal'. Nun ist je Prioritätsstufe und Richtung eine eigene FIFO-Beziehung definiert, mit der Maßgabe, daß höherpriorisierte Anforderungen vor niederprioren durch das Medium abgewickelt werden. Daß höherpriore Anforderungen u.U. dennoch von niederprioren überholt werden, wird allerdings i.a. nicht ganz ausgeschlossen.

2.2.6 Richtungsbetrieb

Interessant bei einem Kommunikationsdienst ist es ferner, ob er allen Dienstnehmern in gleicher Weise zur Verfügung steht, oder ob bestimmte Dienstleistungen nur unter bestimmten Umständen von bestimmten Dienstnehmern angefordert werden können. Eine wichtiger Aspekt hierzu ist der Richtungsbetrieb in der Austauschphase verbindungsorientierter Dienste. Es werden drei Betriebsarten unterschieden:

- Simplex-Betrieb; nur ein Partner hat das Recht zu senden,
- Duplex-Betrieb; beide Partner haben das Recht zu senden,
- Halbduplex-Betrieb; zu einem Zeitpunkt ist das Senderecht nur in den Händen eines Partners, es kann jedoch an den anderen übergeben werden.

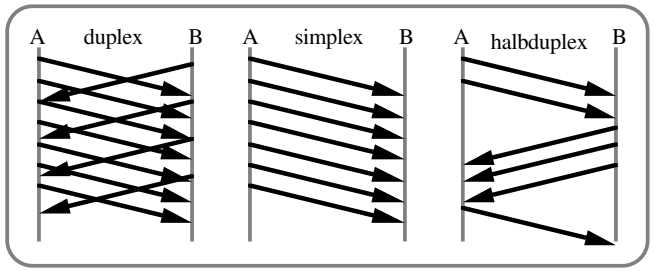

Bild 2.4: Richtungsbetrieb

Bild 2.4 verdeutlicht den zeitlichen Ablauf bei den drei Betriebsarten.

2.3 Qualität von Kommunikationsdiensten

Von hohem Interesse und einsatzentscheidend für einen Telekommunikationsdienst sind nicht nur seine Funktionalität, d.h. die Menge und Auslegung der vorgesehenen Dienstleistungen sondern auch

seine qualitativen Eigenschaften. Sie können je Dienstleistung in sogenannten Qualitätsparametern gemessen werden. Beispiele dafür sind die Laufzeit eines Briefs vom Zeitpunkt des Einwurfs bis zu seiner Auslieferung, die Porto-Kosten, die Wahrscheinlichkeit für den Verlust des Briefs auf dem Transportweg oder die Wahrscheinlichkeit dafür, daß Unbefugte den Brief öffnen und lesen können. Damit ergeben sich auch die vier Hauptaspekte für die Qualität eines Dienstes:

- Leistung,
- Kosten,
- Störungen,
- Schutz.

Die entsprechenden Parameter spezieller Dienstleistungen können im allgemeinen - wie auch im obigen Beispiel - nicht im voraus garantiert werden. Sie hängen in der Regel von zahlreichen erbringerinternen Gegebenheiten, von der Menge anderer gleichzeitig anhängiger Aufträge und nicht über die Dienstschnittstelle stattfindender anderweitiger Einflüsse auf das Medium ab. Sie sind deshalb großenteils nicht allein durch die Kontrollparameter eines Auftrags determiniert. Es wirken sich auch als zufällig anzusehende Einflüsse aus. Qualitätsangaben erfolgen deshalb per Grenz- oder Mittelwerten.

2.3.1 Leistung

Der Leistungsaspekt unterscheidet sich danach, ob der Standpunkt des Erbringers oder der des Dienstnehmers eingenommen wird.

Für einen Dienstnehmer ist es wichtig, mit welcher Leistung das Medium seine Aufträge bearbeitet. Als wichtige Kenngrößen hierzu treten auf:

- die Nachrichtenlaufzeit (Zeitdauer zwischen Anlieferung und Auslieferung einer Nachricht),
- die Übertragungsleistung (Bitrate / Durchsatz),
- die überbrückte Entfernung.

Für einen Diensterbringer sind die vorgenannten dienstnehmer-bezogenen Kenngrößen natürlich unter dem Aspekt der Akzeptanz der von ihm angebotenen Dienste ebenfalls wichtig. Unter wirtschaftlichen Erwägungen interessiert ihn darüberhinaus auch das Kosten/Nutzen-Verhältnis zwischen eingebrachten Ressourcen und der Summe den Dienstnehmern insgesamt erbringbarer Dienstleistungen. Auch im Sinne einer übergeordneten Bewertung eines Mediums sind z.B. Größen wie der Nutzungsgrad von Interesse, der die Übertragungsleistung von medien-intern in tieferen Schichten eingesetzten Medien zu der Nutzdaten-Übertragungsleistung in Bezug setzt, die den Dienstnehmern verfügbar gemacht werden kann.

2.3.2 Kosten

Auch bei den Kosten ist der Standpunkt des Erbringers von dem des Dienstnehmers zu unterscheiden.

Ein Erbringer bemißt seine Kosten an den eingesetzten Investitionen und den zur Aufrechterhaltung des laufenden Betriebs benötigten Aufwendungen. Ist er von den Dienstnehmern organisatorisch getrennt, erhebt er von den Dienstnehmern Gebühren um die Kosten abzudecken.

Ein Dienstnehmer hat einerseits ebenfalls direkte Investitions- und Betriebskosten für die unter seiner Verantwortung stehenden Dienstzugangseinrichtungen. Andererseits sind die vom Diensterbringer verlangten Gebühren zu berücksichtigen. Sie gliedern sich i.a. in eine Grundgebühr und Nutzungsgebühren. Die Grundgebühr wird allein für die Verfügbarmachung von Zugangspunkten unabhängig von der Nutzung des Dienstes fällig. Die Nutzungsgebühr wird in der Regel je tatsächlich ausgeführ-

ter Dienstleistung erhoben und richtet sich nach der Menge übertragener Nutzdaten und anderen Qualitätsparametern der Dienstleistung.

2.3.3 Störungen

Unter dem Stichwort Störungen sollen nicht über die Dienstschnittstelle laufende Einflüsse auf die interne Funktion des Erbringers in ihren Auswirkungen auf Dienstleistungen behandelt werden. Solche Einflüsse können im wesentlichen der Ausfall oder Fehlfunktionen von Geräten sowie elektromagnetische, induktive oder kapazitive Einkopplungen von Fremdsignalen in Übertragungswege sein. Sie führen dazu, daß die tatsächliche Funktionalität des Mediums bei der Erbringung von Dienstleistungen von der gewünschten abweicht.

Es ist hervorzuheben, daß echte, längere Distanzen überbrückende Kommunikationsdienste i.a. von Medien erbracht werden, bei denen die Wahrscheinlichkeit für solche Störungen nicht als vernachlässigbar klein angesehen werden kann. Die möglichen Auswirkungen von Störungen müssen deshalb schon im Konzept eines Dienstes, bei der Auslegung seiner Funktionalität berücksichtigt werden. Für jeden Dienstleistungstyp wird deshalb nicht nur der normale, eigentlich gewünschte Verlauf in Form einer Abfolge entsprechender Ereignisse und ihrer Parametrisierung an der Dienstschnittstelle definiert. Ferner sind die Auswirkungen von Störungen als indeterministisch gewählte, alternativ ebenfalls mögliche Ereignisabfolgen und / oder als Abweichungen von Parameterwerten in die Beschreibung der Funktionalität eines Dienstes ausdrücklich mitaufzunehmen. Der Aspekt Störungen betrifft also direkt bereits die Funktionalität eines Dienstes und kann nicht allein unter dem Thema Qualität abgehandelt werden.

Als funktionelle Auswirkungen von Störungen ergeben sich im wesentlichen:
- der Abbruch von Verbindungen,
- der Verlust von Nachrichten,
- die Erzeugung von Phantomnachrichten,
- das Duplizieren von Nachrichten,
- die Reihenfolgevertauschung von Nachrichten,
- die Verfälschung von Nachrichten.

Der qualitative Aspekt von Störungen betrifft ihre stochastischen Eigenschaften, d.h. mit welcher Wahrscheinlichkeit bzw. unter welcher Wahrscheinlichkeitsverteilung treten sie auf. Während ihre Auswirkungen bei der Funktionsbeschreibung nur als generelle Möglichkeiten mitaufgeführt sind, man also hierdurch weiß, was passieren kann, will man hier wissen, in welchem Umfang damit zu rechnen ist.

Von besonderem Interesse ist hierbei oft die Verfälschung und als entsprechender Qualitätsparameter wird die Bitfehlerwahrscheinlichkeit angegeben. Die oft zu findende Angabe 10^{-7} besagt so z.B., daß in den Nutzdaten bei der Übertragung im Mittel jedes 10.000.000-te Bit vertauscht wird. Zur genaueren Bestimmung der Qualität eines Dienstes hinsichtlich Verfälschung wird sie oft nicht nur im Mittelwert sondern auch in ihrer genauen Verteilung - auch unter Berücksichtigung verschiedener Störungstypen (Einfachbitfehler, Bündelfehler) - untersucht.

Abschließend soll nocheinmal darauf hingewiesen werden, daß das Auftreten von Störungen bei Telekommunikationsdiensten nie vernachlässigt werden kann. So ist auch die Qualitätsverbesserung hinsichtlich Störauswirkungen eine wichtige Aufgabe medieninterner Schichten, d.h. dort ablaufender Telekommunikationsprotokolle. Die Störwahrscheinlichkeit 0 kann allerdings grundsätzlich nicht erreicht werden. Deshalb werden immer auch Teilnehmer auf Anwendungsebene mit Störauswirkungen konfrontiert.

2.3.4 Schutz

Unter dem Aspekt Störungen wurden zufällige Fremdeinflüsse behandelt, die aufgrund technischer medieninterner Gegebenheiten auftreten können. Sie stören die Dienstfunktion, sind aber nicht in dem Sinne bösartig, daß dahinter ein Verursacher steht, der Dienstnehmer in der Verfolgung ihrer Kooperationsbeziehungen bewußt und gezielt stören will.

Unter dem Aspekt Schutz und Sicherheit ist es dagegen vor allem interessant, welche Möglichkeiten ein solcher Eindringling durch Eingriffe über die reguläre Dienstschnittstelle (z.B. durch die trickreiche Gestaltung von 'normalen' Aufträgen) oder über Seitenwege in die interne Organisation des Mediums (z.B. durch Anzapfen von Leitungen) besitzt:
- Stören der Funktionalität und / oder der Leistungsfähigkeit,
- Mithören von Nachrichten und / oder Dienstleistungskontrollparametern,
- Abfangen von Nachrichten,
- Verfälschen von Nachrichten,
- Unterschieben von Nachrichten.

Die Angriffe können also die Verfügbarkeit, die Integrität und / oder die Vertraulichkeit der Kommunikationsdienste stören.

2.4 Telekommunikationsprotokolle

Nach den Erläuterungen zur logischen Architektur von Telekommunikationssystemen besteht eine Schicht i aus einer Menge $\mathbb{I}_i$ von Instanzen. Die Instanzen sind die Dienstnehmer eines untergeordneten Dienstes D_{i-1}, und ihre Aufgabe ist es, an der Schnittstelle zwischen Schicht i und Schicht i+1 einen Dienst D_i zur Verfügung zu stellen, d.h. einzelne an der Dienstschnittstelle von D_i angeforderte Dienstleistungen zu erbringen (vgl. Bild 2.2). Hierzu nutzen sie ihrerseits Dienstleistungen des Dienstes D_{i-1}.

Die Art und Weise der Erbringung des Dienstes D_i durch die Instanzen $\mathbb{I}_i$ auf der Basis der Nutzung von Leistungen des untergeordneten Dienstes D_{i-1} soll unter dem Begriff Telekommunikationsprotokoll der Schicht i angesprochen werden. Wesentlich hierfür ist, wie sich die einzelnen Instanzen gegenüber den beiden Dienstschnittstellen von D_{i-1} und D_i an den ihnen jeweils zugeordneten Zugangspunkten verhalten. Ein derartiges Verhalten wird beeinflußt von Stimuli der Dienstnehmer des Dienstes D_i und Reaktionen des Dienstes D_{i-1}, d.h. von Existenz, Ort, Zeitpunkt, Typ, Kontrollparametern und Nutzdaten solcher Ereignisse. Es äußert sich seinerseits in der Erzeugung von Reaktionen des Dienstes D_i und Stimuli für den Dienst D_{i-1} sowie in der Festlegung der spezifischen Attribute dieser Ereignisse. Von sonstigen Einflüssen und Seiteneffekten, wie z.B. den von einer Instanz lokal benötigten Betriebsmitteln und ihrer Inanspruchnahme, soll abstrahiert werden.

Die Definition eines Protokolls erfolgt durch folgende Festlegungen:
- Verweis auf die Definition des zu erbringenden Dienstes (Zieldienstes / service provided),
- Verweis auf die Definition des unterliegenden Dienstes (Basisdienstes / service used),
- Beschreibung des Ablaufverhaltens der Instanzen,
- Festlegung der Nachrichten, die zwischen den Instanzen ausgetauscht werden (der sog. PDUs, protocol data units).

Man kann ein Protokoll auch als Algorithmus ansehen, dessen Anforderungen mit dem zu erbringenden Dienst D_i gegeben sind. Es handelt sich um einen nebenläufigen Algorithmus. Er soll zeitlich parallel von den einzelnen $\mathbb{I}_i$-Instanzen ausgeführt werden, wobei jede Instanz für sich zeitlich sequen-

tiell agiert. Wichtige vom Algorithmus benutzte Grundoperationen sind die Dienstleistungen des Dienstes D_{i-1}. Nur unter ihrer Nutzung können die $\mathbb{I}_i$-Instanzen miteinander kommunizieren.

Ein Telekommunikationsprotokoll dient der Erbringung eines gewünschten Telekommunikationsdienstes auf der Basis der Nutzung eines untergeordneten vorhandenen Dienstes. Grob gesprochen soll ein Protokoll funktionelle und qualitative Unterschiede zwischen einem vorhandenen Dienst D_{i-1} und einem gewünschten Dienst D_i überbrücken.

Folgende Arten funktioneller Unterschiede zwischen Telekommunikationsdiensten D_{i-1} und D_i treten im wesentlichen auf:

- D_i unterscheidet Dienstleistungstypen im Vergleich zu D_{i-1} genauer,
- D_i bietet bestätigte Dienstleistungen an, D_{i-1} nur unbestätigte,
- die Anzahl der jeweils betroffenen Orte einzelner Dienstleistungen von D_i und D_{i-1} ist unterschiedlich,
- D_i unterstützt Dienstleistungen zwischen Orten, zwischen denen D_{i-1} direkt keine Dienstleistungen anbietet,
- D_i sieht einen Dienstleistungskontext vor, der von D_{i-1} nicht unterstützt wird,
- die Auslieferungsdisziplinen von D_i und D_{i-1} sind unterschiedlich,
- die Modalitäten des Richtungsbetriebs von D_i und D_{i-1} sind unterschiedlich.

Folgende Arten qualitativer Unterschiede zwischen Telekommunikationsdiensten D_{i-1} und D_i treten im wesentlichen auf:

- D_i bietet Dienstleistungen mit anderer Leistung als D_{i-1} an,
- D_i besitzt eine andere Kostenstruktur im Vergleich zu der von D_{i-1},
- D_i verdeckt ein gewisses Maß an Störungsauswirkungen von D_{i-1},
- die Dienstleistungen von D_i sind gegenüber denen von D_{i-1} sicherer gegen Stör- oder Abhörmaßnahmen gezielt vorgehender Eindringlinge.

3. Basisdienste

Kommunikationsdienste können unterschiedlichste Funktionalität besitzen. Diensterbringer, d.h. Medien, können von unterschiedlichstem inneren Aufbau geprägt sein.

Das vorliegende Kapitel konzentriert sich zunächst auf die untersten Schichten der logischen Architektur von Telekommunikationssystemen. Von ihnen werden ausschließlich transportorientierte Dienste angeboten. Sie werden in ihrer Funktionalität und Qualität sehr unmittelbar durch die Eigenschaften technischer Einrichtungen zur Nachrichtenübertragung geprägt und sind in ihrer detaillierten Konzeption traditionell Gegenstand der Elektrotechnik / Nachrichtentechnik. Mindestens ein grundlegendes Verständnis der wichtigsten Begriffe, Zusammenhänge und Verfahren aus der Nachrichtentechnik ist aber auch für den schwerpunktmäßig an Rechnernetzen und verteilten Anwendungen interessierten Informatiker wichtig. Im Abschnitt 'Übertragungssysteme' werden die Grundzüge der technischen Übertragung von Nachrichten über Entfernungen vorgestellt. Der Abschnitt 'Vermittlungssysteme' befaßt sich mit einigen Prinzipien der Vermittlung von Nachrichten bzw. Nachrichtenwegen zwischen Dienstnehmern.

Anschließend wird im Abschnitt 'Datenübertragungseinrichtungen und Schnittstellen' erläutert, wie Rechner mit Übertragungs- und Vermittlungssystemen gekoppelt werden.

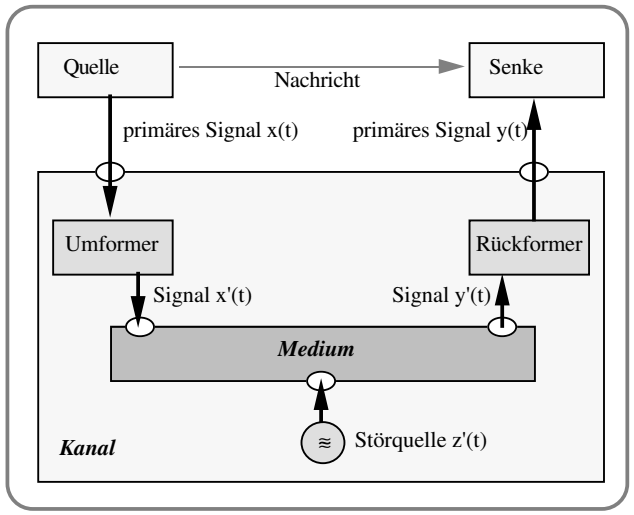

Bild 3.1: Ubertragungssystem

3.1 Übertragungssysteme

Bild 3.1 zeigt ein einfaches Übertragungssystems. Zwei Dienstnehmern Quelle und Senke wird über zwei Zugangspunkte ein einfacher Nachrichtenübertragungsdienst angeboten.

Im Übertragungssystem sind bereits drei Schichten erkennbar:
- Nachrichtenquelle und Senke,
- nachrichtentechnischer Kanal,
- physikalisches Medium.

Das physikalische Medium - man denke z.B. an eine aus einem Adernpaar bestehende elektrische Leitung - überbrückt die Entfernung zwischen Quelle und Senke. Es kann am Ort der Quelle über eine, Eingabesignal genannte, physikalische Größe x'(t) im Verlauf der Zeit t beeinflußt werden (z.B. quellenseitig durch die Leitung fließender elektrischer Strom). Am Ort der Senke ist eine, Ausgabesignal genannte, physikalische Größe y'(t) in ihrem zeitlichen Verlauf beobachtbar (z.B. senkenseitig an einem Lastwiderstand auftretender Spannungsabfall). Da die Entfernung zwischen Quelle und Senke groß ist, ist ferner mit zahlreichen Störeinflüssen auf das physikalische Medium zu rechnen. Sie seien in der Störgröße z'(t) zusammengefaßt. Das Ausgabesignal ist in seinem zeitlichen Verlauf nach physikalischen Gesetzmäßigkeiten vom zeitlichen Verlauf des Eingabesignals und dem der Störgröße bestimmt "y'(t) = F(x'(t), z'(t))" und diese Funktionalität des physikalischen Mediums kann z.B. anhand eines entsprechenden Differentialgleichungssystems untersucht werden.

Der nachrichtentechnische Kanal wird aus den Komponenten Umformer, Rückformer und physikalisches Medium gebildet. Als Produzent der nicht vernachlässigbaren Störgröße z'(t) wird ferner eine Störquelle berücksichtigt. Der Umformer wird am quellenseitigen Dienstzugangspunkt des Kanals im Verlauf der Zeit durch eine, primäres Quellensignal genannte und i.a. elektrische, physikalische Größe beeinflußt und erzeugt in Abhängigkeit hierzu das Eingabesignal x'(t). Der Rückformer beobachtet das Ausgabesignal des Mediums y'(t) und erzeugt ein entsprechendes primäres, i.a. elektrisches, Signal y(t). Unter Abstraktion vom internen Aufbau ist es interessant, welche Formen von Primärsignalen x(t) von einem Kanal mit welcher Leistung übertragen werden können.

3.1.1 Kanal

Nachrichtenquelle und Senke sind die Dienstnehmer des nachrichtentechnischen Kanals. Aus seiner Sicht interessieren Zweck und logische Einbettung der zu übertragenden Nachrichten sowie auch ihre abstrakte Syntax nicht. Betrachtet wird die konkrete Syntax der Nachrichten, wie sie als primäre Signale x(t) und y(t) dargestellt werden.

3.1.1.1 Signalklassen

Ein nachrichtentechnischer Kanal kann zunächst nach der Klasse unterstützter primärer Signale klassifiziert werden. Bild 3.2 zeigt hierzu die 4 Hauptklassen. Die linke obere Signalform, das klassische analoge Signal, wird bei der Sprachübertragung verwendet (z.B. Rundfunk, Telephon). Die rechte obere Form findet sich z.B. im Gebiet der Lenkung technischer Prozesse, wenn analoge Prozeßgrößen nur zu bestimmten Zeitpunkten gemessen und ab dann bis zur nächsten Messung unverändert übertragen werden (Sample and Hold).

Für die digitale Datenübertragung sind die beiden unteren Klassen, die digitalen Signalformen, und zwar in der Ausprägung zweiwertiger Signale wichtig. Zu übertragen sind aus '0'/'1'-Folgen bestehende binäre Nachrichten.

Bei zeitkontinuierlichen Digitalsignalen ist nur ein Mindestzeitintervall zwischen Signaländerungen vereinbart. Ansonsten können die Signalwerte zu beliebigen Zeitpunkten wechseln. Bei zeitdiskreten Digitalsignalen ist dagegen ein - in der Regel isochrones - Taktraster vorgegeben, so daß Signaländerungen nur zu festen Zeitpunkten erlaubt sind.

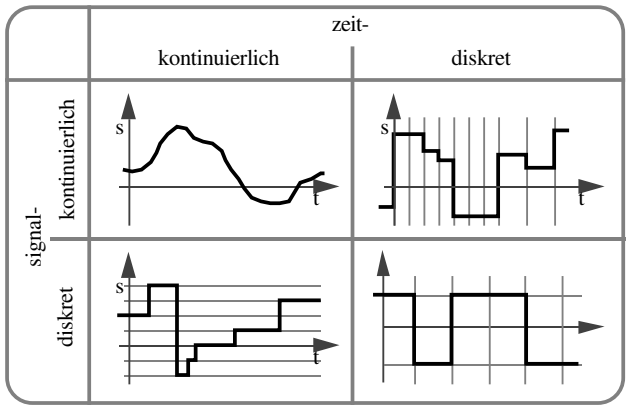

Bild 3.2: Signalklassen

3.1.1.2 Schrittgruppen

Ein digitales Signal x(t) wird durch eine Folge von Signalschritten gebildet. Im allgemeinen werden aufeinanderfolgende Schritte eines Signals zu einzelnen Gruppen zusammengefaßt:

- Wort,
- Block.

Gruppen kleinerer (5, 7, 8) und in der Regel fester Anzahl von Schritten werden üblicherweise Wörter genannt. Bei modernen binären Kanälen findet sich eine Gliederung in Gruppen von 8 aufeinanderfolgenden Binärschritten, den Oktetten.

Gruppen größerer und manchmal auch variabler Anzahl von Schritten werden Blöcke genannt. Sie dienen i.a. der Darstellung logisch zusammengehöriger Nachrichten und können sich bei zweistufiger Gliederung aus einer Menge aufeinanderfolgender Wörter zusammensetzen. Damit bei variabler Blocklänge die Senke das Ende eines Blocks erkennen kann, muß entweder im Block Längeninformation übertragen werden, oder es sind besondere Signalformen zur Markierung eines Blockendes vorgesehen.

3.1.1.3 Parallel- und Serienübertragung

Bei Fernübertragung über tatsächlich längere Entfernungen (>100m) wird von einem Quellen / Senken-Paar für einen bestimmten Nachrichtenstrom aus Aufwandsgründen (z.B. Leitungskosten) üblicherweise nur ein nachrichtentechnischer Kanal genutzt. Eine Nachricht wird z.B. in Form einer Bitfolge als Block über einen einzigen binären Kanal übertragen. Ist eine Wortstruktur erkennbar, reihen sich die Wörter im Block seriell aneinander an, und auch die einzelnen Bitstellen eines Worts werden nach einer festgelegten Reihenfolge seriell übertragen. Hiervon soll im folgenden ausgegangen werden.

Bei kürzeren Entfernungen werden zum Teil mehrere Kanäle für dieselbe Quellen / Senken-Verbindung parallel geschaltet. Ist die Wortlänge z.B. auf 8 bit festgelegt und stehen 8 gleichartige binäre Kanäle zur Verfügung, dann kann ein Wort in einem Schritt durch Nutzung eines Kanals je Bitstelle übertragen werden. Die Parallelübertragung wird eingesetzt, um auch mit einfachen Mitteln höhere Übertragungsleistungen zu erzielen.

3.1.1.4 Synchronisation

Wesentlich für das Gelingen einer digitalen Übertragung ist die Bestimmung der Abtastzeitpunkte, d.h. der Zeitpunkte an denen die Senke das Signal y(t) für das vorangegangene Zeitintervall als Signalschritt auswertet. Hierzu bestehen im wesentlichen folgende Möglichkeiten:

- Sende- und Empfangstakt unterliegen gemeinsamen Konventionen und werden diesen folgend von Quelle und Senke unabhängig voneinander bestimmt,
- Übertragung des Taktrasters auf einem eigenen Kanal,
- Übertragung des Taktrasters mit dem Signal,
- Punktuelle Synchronisation eines weitgehend unabhängigen Taktgenerators bei der Senke durch das Signal.

Die Bestimmung des Taktrasters durch voneinander unabhängige Taktgeneratoren, einen bei der Quelle und einen bei der Senke, besitzt den Vorteil, daß zur Taktübertragung keine Übertragungsleistung benötigt wird. Sie würde aber äußerst stabile Taktgeneratoren voraussetzen. Ferner bedarf es einer initialen Maßnahme zum abgestimmten Start beider Generatoren.

Die Übertragung des Taktrasters auf einem eigenen parallelen Kanal findet nur im Nahbereich statt.

In der Fernübertragung werden bei nicht-isochronem - oder auch bei im Prinzip isochronem aber tatsächlich starken Schwankungen unterliegendem - Taktraster Signalformen eingesetzt, die es der Senke erlauben, das Taktraster aus dem Signalverlauf abzuleiten.

Mit vertretbarem Aufwand sind Taktgeneratoren realisierbar, deren Frequenzkonstanz für eine beschränkte Anzahl von Schritten ausreichend ist. Ein spezieller Signalverlauf dient zur Markierung des Beginns einer solchen Schrittfolge. Die Schrittfolge fällt in der Regel mit einer Schrittgruppe (Wort oder Block, bei Blöcken variabler Länge ist eine maximale Länge vorgegeben) zusammen. Eine Schrittgruppe darf, wenn eine minimale Wartezeit nach Ende der vorangegangenen Gruppe eingehalten wird, zu beliebigen Zeitpunkten beginnen (Gruppen zueinander asynchron). Innerhalb der Gruppe liegt eine synchrone Übertragung vor, den Signalschritten unterliegt ein isochrones Taktraster. Von großer Bedeutung sind die beiden Verfahren:

- Zeichenweiser Start / Stop-Betrieb,
- Blocksynchronisation.

Beim zeichenweisen Start / Stop-Betrieb bestehen die Schrittgruppen aus Wörtern (Sie heißen Zeichen, weil traditionell ein Wort zur Darstellung eines Zeichens aus einem festen Zeichenvorrat dient). Er ist üblicherweise für einen binären Kanal mit zwei zulässigen Signalpegeln 0,1 definiert. Im Ruhezustand werde z.B. die 1 übertragen. Als Signalform zur Kennzeichnung eines Zeichenbeginns dient der Wechsel nach 0 und die Übertragung der 0 für die Dauer eines Schritts (Startbit). Flanke und diese Zeitdauer dienen der Senke zum Einsynchronisieren ihres Taktgenerators. Nun folgen, in isochronem Raster zum Startbit, eine festgelegte Anzahl von Datenbits, dargestellt durch 0- oder 1-Schritte. Nach dem letzten Datenbit wird eine kleinere Anzahl von Stopbits mit dem Ruhepegel 1 übertragen (i.a. 1 oder 2 Stopbits). Sie dienen nicht zur Kennzeichnung des Wortendes (es ist durch die feste Schrittanzahl bekannt), sondern sollen eine Ruhephase vor Beginn der nächsten Zeichenübertragung garantieren, die die Senke zur Nachbereitung der Zeichenübertragung (interne Verarbeitung, Weitergabe, beim Fernschreiber: Drucken des Zeichens) und zur Wiederherstellung der Empfangsbereitschaft nutzen kann. Dieses Verfahren findet sich vornehmlich bei älteren Systemen. Es verlangt von Quelle und Senke nur die Aufrechterhaltung eines isochronen Taktes über kurze Zeitperioden, und die Daten müssen nur in kleinen Portionen (nämlich Zeichen) bei Quelle und Senke zwischengepuffert werden. Nachteil ist, daß z.B. bei 1 Start-, 5 Daten- und 2 Stopschritten, von 8 Schritten nur 5 für die Übertragung von Nutzdaten eingesetzt werden.

Die Blocksynchronisation nutzt die Fähigkeiten moderner quarzstabilisierter Oszillatoren und digitaler Schaltkreise und Speicher um dieses Verhältnis wesentlich zu verbessern. Hier sind alle Oktette eines ganzen Blocks und deren einzelne Schritte in isochronem Raster zum Blockbeginn. Bei binärem Kanal dient üblicherweise eine Folge mehrerer Schritte zur Kennzeichnung des Blockbeginns und zur hier etwas anspruchsvolleren Einsynchronisation des Taktgenerators der Senke. Dieser Signalverlauf dient ebenfalls zur Kennzeichnung des Blockendes und ist innerhalb eines Blocks deshalb verboten. Um beliebige Bitmuster innerhalb des Blocks übertragen zu können, werden diese von der Quelle untersucht und bei Bedarf so modifiziert, daß die Senke dies rückgängig machen kann (Bittransparenz durch 'Bitstopfen').

3.1.1.5 Codierung

Codierung befaßt sich mit der Abbildung eines bei der Quelle gegebenen Symbolvorrats, d.h. der Menge dort für Nachrichten oder ihrer Komponenten berücksichtigten Datenwerte, auf den hierfür auszugebenden Signalverlauf x(t). Sie ist mit der Decodierung eng verknüpft, die umgekehrt den Signalverlauf y(t) wieder in den Symbolvorrat rückabbilden muß.

Codierung bzw. Decodierung finden außerhalb des Kanals in der Quelle bzw. der Senke statt. Man unterscheidet:

- Quellencodierung,
- Kanalcodierung.

Aufgabe der Quellencodierung ist es, für jedes Symbol überhaupt ein entsprechendes übertragbares Signalmuster festzulegen. Bei Verwendung binärer Kanäle heißt dies, daß den einzelnen Datenwerten ein zur Übertragung zu verwendendes Bitmuster zugeordnet wird. Im Bereich traditioneller Telekommunikationssysteme finden sich hierzu z.B. die Fernschreib-Zeichencodes. Pulscodemodulation PCM zur binären Darstellung von Sprache und Musik ist ein moderneres Beispiel. In der logischen Architektur eines Kommunikationssystems nach dem ISO/OSI-Modell befaßt sich die Schicht 6, die Darstellungsschicht mit den wesentlichen Aspekten der Quellencodierung. Die unterhalb dieser Schicht zu übertragenden Nachrichten sind Bitmuster in Grenzen beliebiger Länge und werden als Blöcke mit beliebigem internem Aufbau übertragen (Bittransparenz).

Die Kanalcodierung ist dagegen sehr eng am Kanal anzusiedeln. Sie wird auf den Kanal und sein spezielles Übertragungsverhalten zugeschnitten. Aufgabe der Kanalcodierung ist es, die Synchronisation der Senke sowie senkenseitige Maßnahmen zur Erkennung oder Behebung von Übertragungsfehlern zu unterstützen. Synchronisationsmaßnahmen wurden im vorangegangenen Abschnitt schon besprochen. Fehlererkennungs- und Behebungsmaßnahmen werden durch Hinzufügen redundanter Information auf Quellenseite und deren Auswertung auf Empfangsseite verwirklicht. Zur Verbesserung des Erkennungs- bzw. Behebungsverhaltens können ferner Umordnungen der Bitmuster durchgeführt werden, um die Auswirkungen von Bündelfehlern zu mildern.

Zwei häufig angewendete Verfahren zur Fehlererkennung sind:
- Zeichenparitätssicherung,
- Blocksicherung.

Die Zeichenparitätssicherung ist die ältere Methode und wird gern bei der schon angesprochenen zeichenweisen Übertragung eingesetzt. Hierzu wird jedes Wort im Vergleich zum Quellencode um eine weitere Binärstelle, das Paritätsbit, ergänzt. Es wird von der Quelle so belegt, daß die Quersumme über alle Stellen des Worts gerade (bei manchen Systemen auch ungerade) wird. Die Senke kann durch Quersummenbildung eine ungerade Anzahl von Bitfehlern je Zeichen erkennen. Eine gerade Anzahl von Bitfehlern bleibt unerkannt.

Bestehen die Nachrichten nicht nur aus einzelnen Wörtern sondern sind Blöcke erkennbar, kann auch für den Block insgesamt ein redundantes Sicherungsbitmuster erzeugt und von der Quelle hinten angefügt werden.

Im Verbund mit der Zeichenparitätssicherung kann so eine Kreuzsicherung erzielt werden. Neben der Sicherung eines jeden Zeichens durch dessen Paritätsbits, werden auch je Bitstelle über alle Zeichen Paritätsbits erzeugt und als Blocksicherungszeichen am Ende des Blocks übertragen. Hier sind (wenn Mehrfachbitfehler vernachlässigt werden) Einfachbitfehler sogar korrigierbar. Insgesamt gesehen ist die Paritätssicherung jedoch ineffizient. Moderne fehlerkorrigierende Kanalcodes bieten gemessen an der Redundanz wesentlich bessere Erkennungs- und Korrigierfähigkeiten.

Ein sehr wichtiges der moderneren Verfahren, das eine sehr gute Fehlererkennung bei geringer Redundanz ermöglicht, ist die zyklische Blocksicherung. Wenige, i.a. 16 und am Ende des Blocks in der sogenannten Frame Check Sequence (FCS) übertragene, Bitstellen ergänzen einen beliebigen Binärblock, zu einem erlaubten Block-Codewort.

Die Kanalcodierung ist im ISO/OSI-Modell Aufgabe der Schicht 2, Sicherungsschicht. Hierfür werden in späteren Kapiteln noch Protokollmechanismen vorgestellt. Auch die zyklische Blocksicherung wird später noch genauer erläutert.

3.1.1.6 Qualitätsparameter

Die wichtigsten Qualitätsparameter eines Kanals sind:
- Schrittgeschwindigkeit (auch Schrittrate genannt); sie wird durch die Anzahl von Signalschritten pro Zeiteinheit bestimmt (Einheit: bd = 1/s nach Baud) und entspricht bei isochronem Takt der Taktfrequenz.
- Übertragungsgeschwindigkeit (auch Bitrate genannt); sie wird durch die Anzahl übertragbarer Bitstellen pro Zeiteinheit bestimmt (Einheit: bit/s) und entspricht bei binärem Kanal mit isochronem Takt der Schrittgeschwindigkeit. Bei nicht-isochronem Takt sind die Verhältnisse komplizierter, weil hier u.U. Information auch durch Zeitspannen dargestellt werden kann. Bei nicht-binären digitalen Kanälen mit isochronem Takt geht die Schrittgeschwindigkeit als Faktor in die Übertragungsgeschwindigkeit ein.
- Signallaufzeit; Zeitdauer, die ein Signal zur Fortpflanzung über Umformer und Medium bis zur Senke benötigt.
- Schrittverzerrung; Mittelwert der Abweichung des Taktrasters des Signals von der festgelegten isochronen Schrittzeit bezogen auf diese. Hier wirken sich Störeinflüsse auf das Medium und auch nicht-ideale Taktgeneratoren aus.
- Bitfehlerwahrscheinlichkeit; Wahrscheinlichkeit, mit der bei binären Kanälen ein gesendetes Bit von der Senke nicht in seinem korrekten Wert erkannt wird. Hier wirken sich Störeinflüsse auf das Medium aus, und es ergeben sich in Abhängigkeit zum eingesetzten Medium (und seiner Umgebung) die unterschiedlichsten Fehlerverteilungen.

3.1.1.7 Betriebsarten

Bild 3.1 zeigt den einfachsten Fall eines Übertragungssystems, genau eine Quelle und eine Senke sind über einen Kanal verbunden. Nachrichten werden nur von der Quelle zur Senke übertragen. Dies entspricht dem Simplex-Betrieb des Kanals. Es sind ferner auch Kanäle möglich, die bereits Duplex- oder Halbduplex-Betrieb unterstützen.

3.1.1.8 Sammelkanäle

Eine weitere Besonderheit bilden Sammelkanäle, die auch kurz Bus genannt werden. Hier sind mehr als zwei Dienstnehmer über Um- und Rückformer mit dem Medium verbunden. Prinzipiell können i.a. alle Dienstnehmer des Kanals ein eigenes Quellensignal $x_i(t)$ einspeisen und ein Signal $y_i(t)$ abtasten. Wenn über eine Zeitspanne nur ein Quellensignal eingespeist wird, wird es allen anderen über die entsprechenden $y_i(t)$-Signale beobachtbar. Auf diese Weise ist der Rundruf die von diesem Kanal angebotene Grundform der Nachrichtenübertragung.

Zum geregelten Betrieb eines Sammelkanals müssen jedoch Maßnahmen zur Zugriffskontrolle bei den Quellen eingesetzt werden. Wenn mehrere Quellen innerhalb derselben Zeitspanne Quellsignale einspeisen, überlagern sich die Signale im Medium, und dies kann dazu führen, daß Senken die Nachrichten nicht mehr korrekt empfangen können. Zugriffskontrollverfahren sollen dies nach Möglichkeit vermeiden. Sie sind in ihrer Auslegung sehr vom eingesetzten Medium abhängig. Sie sind deshalb kanal-nah anzusiedeln und werden später bei der Behandlung der den Kanälen direkt übergeordneten Schicht 2 des ISO/OSI-Modells besprochen.

3.1.2 Medien

Dieser Abschnitt soll einen kurzen Überblick über wichtige zur Signalübertragung eingesetzte physikalische Medien geben. Die Auswahl eines Mediums richtet sich nach dessen Beschaffungs-, Installa-

tions- und Wartungskosten, nach den Kosten der entsprechenden Um- und Rückformer und der darauf aufbauend für den Kanal erreichbaren Qualität.

Bei aller Unterschiedlichkeit der Medien und der eingesetzen Übertragungsverfahren, gibt es auch gemeinsames, das in der gewünschten Funktionalität von Medien allgemein, nämlich der Fortpflanzung physikalischer Größen begründet ist. Dies soll zur Nennung der entsprechenden Qualitätsparameter von Medien zunächst kurz umrissen werden. Abschließend soll auf die geteilte Nutzung desselben Mediums für verschiedene Kanäle eingegangen werden.

3.1.2.1 Medienqualität

Die Funktionalität eines Mediums besteht darin, daß ein Ausgabesignal y'(t) in Abhängigkeit zu einem beeinflußbaren Eingabesignal x'(t) und weiteren Störeinflüssen z'(t) gebildet wird: y'(t)=F(x'(t), z'(t)). Idealerweise sollte gelten $y'(t) = 1/k * x'(t-t_0)$, wobei die Zeitkonstante t_0 der Laufzeit der Signale im Medium und der Dämpfungsfaktor k der Abschwächung des Signals im Medium entspricht. Leider ist das aufgrund medien-interner Gegebenheiten bei der Signalfortpflanzung und aufgrund der äußeren Störeinflüsse z'(t) nur ungefähr der Fall.

Das Eingabesignal x'(t) kann als Überlagerung von Sinus-Verläufen unterschiedlicher Frequenz aufgefaßt werden (jede tatsächlich erzeugbare Signalform kann durch eine solche Summe harmonischer Schwingungen dargestellt werden). Deshalb kann das Übertragungsverhalten eines Mediums bezogen auf eine harmonische Schwingung $x'(t) = \sin(\omega*t)$ mit der Kreisfrequenz ω als Eingabesignal charakterisiert werden. Als Ausgabesignal sollte sich dann idealerweise ebenfalls eine Sinus-Schwingung $y'(t) = 1/k * \sin(\omega*(t-t_0))$ ergeben. Tatsächlich weicht y'(t) durch Überlagerung von Störsignalen und der Frequenzabhängigkeit von k und t_0 davon ab.

Medien-intern bedingte Abweichungen werden üblicherweise durch folgende Kenngrößen charakterisiert:

- Bandbreite; die Dämpfung ist in der Regel derart frequenzabhängig, daß Schwingungen unterhalb einer unteren Grenzfrequenz f_u und oberhalb einer oberen Grenzfrequenz f_o (mit $f_u<f_o$) so stark gedämpft werden, daß eine sinnvolle Auswertung des Ausgabesignals nicht mehr möglich ist. (f_o-f_u) ist die Bandbreite des Mediums. Das Medium wird deshalb nur in einem bestimmten Frequenzband $<f_1, f_2>$ (mit $f_u \le f_1 < f_2 \le f_o$) genutzt, d.h. es werden mit x'(t) nur Schwingungen aus diesem Frequenzbereich gezielt eingespeist. Hat ein Eingabesignal ungewollt außerhalb liegende Anteile (z.B. Oberschwingungen bei Rechtecksignalen), so werden diese, um davon ausgehende Störungen zu vermindern, üblicherweise schon im Umformer durch Bandfilter reduziert.
- Dämpfungsverzerrung; die Dämpfung kann auch bei gleicher Frequenz nicht konstant bleiben sondern sich mit der Zeit verändern, so daß Amplitudenschwund und Amplitudensprünge auftreten können.
- Laufzeitverzerrungen können zur Frequenzverwerfung und zu Phasenschwankungen führen.

Modelle für die Störeinflüsse z'(t) sind im wesentlichen:

- weißes Rauschen; ein im gesamten Frequenzband mit gleicher Leistungsdichte auftretendes Signal modelliert einen gewissen Grundstöranteil,
- Echo-Bildung; Störsignale aus (zeitverschobenem) Eingabesignal,
- Nebensprechen; Anteile von Signalen auf räumlich nah liegenden anderen Medien,
- Brumm; niederfrequente Störsignale wie z.B. Netzbrummen aus dem Übersprechen von Stromversorgungsleitungen,
- Störimpulse; vereinzelt auftretende Störsignale i.a. kürzerer Dauer aber größerer Amplitude, sie können durch Umschaltvorgänge im Medium, in benachbarten Medien oder Stromversorgungsleitungen entstehen.

Die Störeinflüsse bilden additive Komponenten des Ausgabesignals. Sie sind in ihrem genauen Verlauf nicht vorab bestimmbar sondern gehen in eine genauere Analyse eines Mediums als stochastische

Einflüsse ein. Zur groben Charakterisierung dient der (minimale / mittlere / maximale) Signal / Störabstand (auch Signal/Rauschabstand genannt), der sich auf die Amplitude der Störanteile im Ausgabesignal bezieht.

Bandbreite eines physikalischen Mediums und Signal/Störabstand begrenzen auch die maximal erreichbare Übertragungsleistung:

Nyquist-Theorem:
Wenn B [1/s] die Bandbreite eines Mediums ist, können damit nur Kanäle mit einer Schrittgeschwindigkeit < 2*B realisiert werden.

Shannon-Theorem:
Es sei R der Signal/Rauschabstand (R = mittlere Signalleistung / mittlere Rauschleistung') eines nur durch weißes Rauschen gestörten Kanals, dann können pro Schritt maximal nur 0.5*ld(1+R) Bit übertragen werden.

3.1.2.2 Verwendete Medien

Folgende Arten von Medien werden in Telekommunikationssystemen im wesentlichen eingesetzt:

- elektrische Leiter,
- Funk,
- Hohlleiter,
- Lichtwellenleiter.

Die am weitesten verbreitete Form des Mediums ist der elektrische Leiter. Möglichkeiten sind hier eine einzelne Ader (Stromkreis wird durch Erdung geschlossen) oder ein Adernpaar, das u.U. zur Abschwächung von Störeinflüssen verdrillt oder mit einer Abschirmung versehen ist. Höhere Bitgeschwindigkeiten sind mit teuereren Koaxialkabeln (1 Ader mittig zu einer umgebenden Abschirmung) erzielbar. Hier werden dann vornehmlich hochfrequente Wechselströme verwendet. Über lange Strecken müssen aktive Elemente als Verstärker vorgesehen werden.

Als Alternative zum elektrischen Kabel war lange Zeit nur die Übertragung hochfrequenter elektromagnetischer Wellen per Funk vorhanden (Rundfunk). Hier fallen keine Kosten für das Medium selbst an. Allerdings sind die verwendbaren Frequenzbereiche beschränkt, so daß ihre Nutzung durch zwischenstaatliche und nationale Konventionen reglementiert wird, um gegenseitige Störungen zu vermindern. Sonderformen sind der Richtfunk mit sehr stark bündelnden Antennen bei sehr hohen Frequenzen. Im Rundfunk / Fernsehbereich, aber auch für interkontinentale Fernsprech-, Telegraphie- und sonstige Datenübertragung werden neuerdings geostationäre Satelliten als Relais eingesetzt. Auch vermittelnde Satelliten sind in Diskussion. Ferner wird - um größere Einstrahlungswinkel auch für äquator-ferne Gebiete und bessere Bodennähe zu erzielen - auch über den Einsatz mehrerer sich ergänzender nicht-geostationärer Satelliten (z.B. für den Mobilfunk dünn besiedelter Gebiete) nachgedacht.

Die Verwendung elektromagnetischer Wellen im Licht- oder Wärmefrequenzbereich ist im Nahbereich kaum reglementiert und kann für kurze Entfernungen als Alternative zu Kabeln dienen, wenn Sichtverbindung gewährleistet ist und Standort-Flexibilität erzielt werden soll.

Die Fortpflanzung elektromagentischer Wellen sehr hoher Frequenz kann auch in Hohlleitern 'kanalisiert' werden. I.a. mit Luft gefüllte metallische Röhren beschränken die Ausbreitung der Wellen.

Glasfasern werden in der letzten Dekade als kostengünstige Alternative zu Kabeln angeboten (Rohstoffpreise Glas / Kupfer). Sie sind jedoch aufwendiger im Verlegen (minimaler Biegeradius, Dehnbarkeit). Es werden aufwendigere Umformer erforderlich und Anschlüsse können nicht mehr kostengünstig mit dem Lötkolben erfolgen.

Hier werden Lichtwellen (z.B. durch LEDs oder Laser-Dioden erzeugt) durch das Medium übertragen. Es sind im Vergleich zu Kupfer sehr geringe Dämpfungen erzielbar (-> größere Entfernungen ohne Zwischenverstärker). Das Übersprechen zwischen benachbart verlaufenden Fasern sowie sonstige Störeinkopplungen von außen sind überaus gering.

Trotz der niederen Rohstoffpreise sind die Preise für Glasfaserkabel aufgrund der Verwendung spezieller Gläser, die u.U. einen bestimmten Schichtaufbau haben, sowie enger Fertigungstoleranzen für die Fasern noch hoch. Als kostengünstigere Alternative werden bereits robustere Kunststoff-Fasern mit größerem Querschnitt und damit geringeren Anschlußkosten angeboten.

3.1.2.3 Geteilte Medien

Bei Medien mit größerer Bandbreite, und wenn mehrere gleichlaufende Kanäle vergleichbar geringer Übertragungsleistung benötigt werden, empfiehlt sich die geteilte Nutzung desselben Mediums für verschiedene Kanäle. Man spricht vom Multiplex-Betrieb:

- Frequenzmultiplex,
- Zeitmultiplex.

Beim Frequenzmultiplex wird das nutzbare Frequenzband des Mediums in mehrere Teilbänder aufgeteilt. Sie dürfen sich nicht überlappen, und es werden in der Regel sogar ungenutzte Abstände zur Verringerung des Nebensprechens vorgesehen. Auf diese Art entstehen mehrere weitgehend unabhängige Medien, die zur Realisierung verschiedener Kanäle dienen können. Zur Realisierung sind, in das jeweilige Frequenzband abbildende, Umformer sowie Bandfilter vorzusehen.

Beim Zeitmultiplex ist an jedem Ende des Mediums ein Umschalter angebracht, der jeweils eine von n Quellen bzw. eine von n Senken mit dem Medium verbindet. So wird das Medium für bestimmte Zeitperioden einem Quellen-Senken-Paar exklusiv zugeordnet, wenn der Gleichlauf der beiden Umschalter sichergestellt ist. Die Übertragung zwischen einem Paar kann jetzt nicht mehr kontinuierlich sondern nur noch in entsprechenden Zeitperioden stattfinden. Hierzu müssen Zwischenspeicher für primäre Signale zur Geschwindigkeitsanpassung vorgesehen werden.

3.1.3 Übertragungsverfahren und Umformung

Das eingesetzte Übertragungsverfahren bezieht sich auf die Umformung des primären Quellensignals in das Eingabesignal des Mediums und die Rückformung des Ausgabesignals in das primäre Senkensignal. Es wird grob zwischen Basisband- und Trägerfrequenztechniken unterschieden:

- Basisbandübertragung; sie bezeichnet die direkte Übertragung des rechteckförmigen Quellensignals über das Medium und ist mit einfachen Mitteln realisierbar. Zur Milderung des Übersprechens müssen die steilen Flanken der Rechtecke durch Tiefpass-Filter verflacht werden. Das Medium muß - jedenfalls in der Grundform - zur Gleichstromübertragung in der Lage sein, wenn z.B. eine lange Folge konstanter Quellensignalschritte zu übertragen ist.
- Amplitudentastung; den einzelnen Pegeln des Quellensignals werden verschiedene Amplitudenhöhen der Trägerschwingung zugeordnet. Das Verfahren ist sehr störempfindlich, weil die Amplitude des Ausgabesignals der am meisten durch Störungen beeinflußte Parameter ist.
- Frequenztastung; den einzelnen Pegeln des Quellensignals werden verschiedene Frequenzen der Trägerschwingung zugeordnet. Dies ist das bei Übertragung digitaler Daten über herkömmliche Telefonleitungen am meisten eingesetzte Verfahren.
- Phasentastung; den einzelnen Pegeln des Quellensignals werden verschieden starke Phasensprünge der Trägerschwingung zugeordnet. Mit diesem Verfahren lassen sich - allerdings bei aufwendigerer Um- und Rückformung die besten Übertragungsleistungen erzielen.

Die Um- und Rückformer zu Trägerfrequenztechniken werden Modulator und Demodulator (bei Duplex- oder Halbduplex-Betrieb kurz Modem) genannt.

3.2 Vermittlungssysteme

Aufgabe eines nachrichtentechnischen Kanals ist es, Nachrichten darstellende Signale zwischen einer i.a. kleineren Anzahl von Dienstnehmern auf einem Weg zu übertragen, der durch das benutzte physikalische Medium festgelegt ist. Eine Quelle kann nicht ausdrücklich wählen, welche anderen Dienstnehmer eine Nachricht erhalten sollen. Bei einem Sammelkanal können alle anderen Dienstnehmer die entsprechenden Signale vom Medium abtasten, bei einem Kanal zwischen zwei Punkten ist der einzige andere Dienstnehmer die Senke.

Aufgabe eines Vermittlungsnetzes ist es dagegen, Nachrichten zwischen einer i.a. größeren Anzahl von Dienstnehmern in einer Weise zu übertragen, daß einzelne Nachrichten jeweils nur eine kleinere von der Quelle bestimmbare Menge von Senken (üblicherweise eine einzige Senke) erreichen.

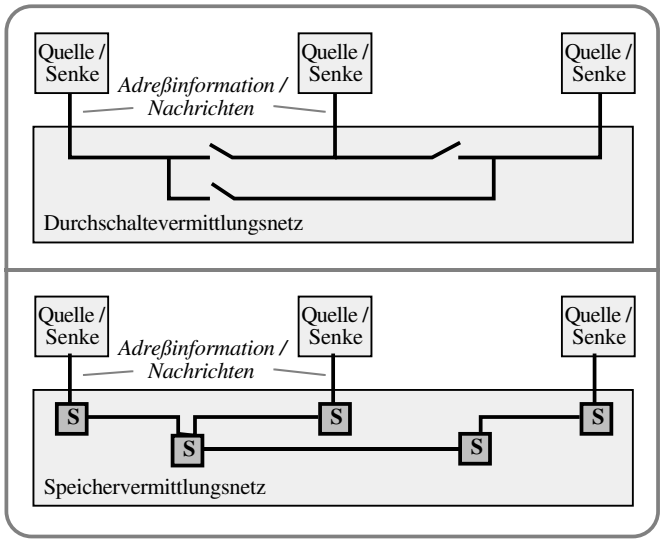

Bild 3.3: Vermittlungssysteme

Bild 3.3 zeigt zwei Vermittlungssysteme beispielhaft. Zwischen Dienstnehmern und Vermittlungsnetz werden neben den eigentlich zu übertragenden Nachrichten (Nutzdaten) auch Adreßparameter zur Senkenauswahl ausgetauscht. Im Vermittlungsystem sind die beiden zu seiner Realisierung einsetzbaren grundlegenden Verfahren skizziert:

- Durchschaltevermittlung,
- Speichervermittlung.

Bei der Durchschaltevermittlung fordert eine Quelle vom Vermittlungsnetz die Herstellung eines durchgängigen Mediums (einer Verbindung) zu einer gewünschten Senke an und kann anschließend hierüber Nachrichtenaustausch bis zur Freigabe der Verbindung durchführen. Beispiel für ein Durchschaltevermittlungsnetz ist das öffentliche Telephonnetz.

Bei der Speichervermittlung übergibt eine Quelle eine Nachricht mit Adreßangabe an das Vermittlungsnetz. Das Netz speichert die Nachricht intern und leitet sie u.U. über mehrere Teilstrecken und unter weiterer Zwischenspeicherung an den Ort der gewünschten Senke weiter. Hier sind das Senden und Empfangen von Nachrichten zeitlich zueinander und von der momentanen Verfügbarkeit geeigneter Medien weitgehend entkoppelt. Als Beispiel für ein speichervermitteltes Netz kann die Briefpost dienen.

Es sind auch Kombinationen möglich wie z.B. die untergeordnete Verwendung von Durchschaltevermittlungsnetzen zur Realisierung eines Speichervermittlungsnetzes.

In allen Fällen sind im Vermittlungssystem Orte und durch Medien gegebene direkte Übertragungswege zwischen ihnen erkennbar. Die Orte können nach Ort eines Dienstnehmers, Ort einer Vermittlungseinrichtung oder Kombination beider klassifiziert werden. Die Übertragungswege sind traditionell Wege zwischen zwei Orten. Unter Topologie eines Systems wird die Form eines Graphen verstanden, in dem Orte als Knoten und Übertragungswege als Kanten dargestellt werden. Nur unidirektional genutzte Medien werden dabei durch gerichtete Kanten vertreten. Für geteilte Medien werden Sonderformen von Kanten zur Verbindung von mehr als zwei Knoten eingeführt.

Im folgenden sollen zunächst die wichtigsten Topologien angesprochen werden. Abschließend sollen die beiden Vermittlungsverfahren am Beispiel behandelt werden. Vorher soll der Bezug zum ISO/OSI-Modell hergestellt werden.

3.2.1 Logische Schichtenarchitektur

Bei der Behandlung von Übertragungssystemen ergab sich bereits eine Hierarchie zweier Schichten, ein nachrichtentechnischer Kanal wird auf der Basis eines physikalischen Mediums gebildet.

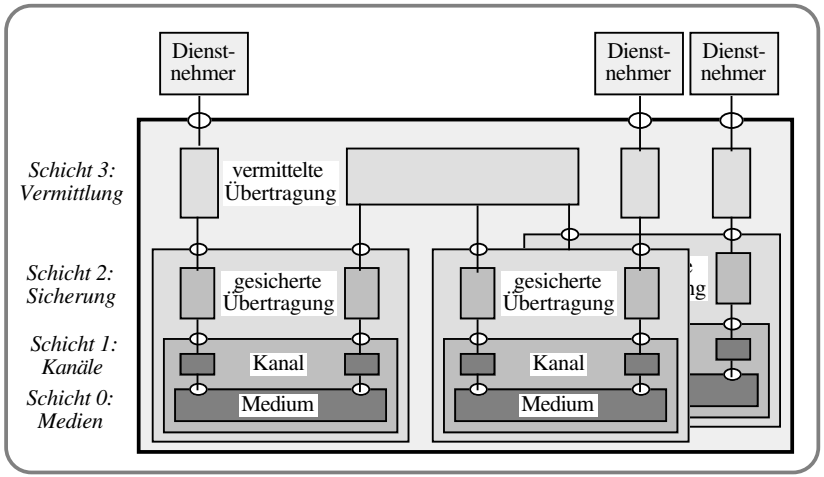

Bild 3.4: Logische Architektur eines Vermittlungssystems

Im physikalischen Medium, dem die Schichtnummer 0 zugeordnet wird, breiten sich physikalische Größen aus. Die Dienstschnittstelle eines entsprechenden Kommunikationsdienstes besteht aus verschiedenen Stellen am Medium, an denen Eingabesignale aufgeprägt und / oder Ausgabesignale abgetastet werden können. Da dies zeitkontinuierlich erfolgt, ist das Konzept zeitlich punktueller Dienstschnittstellenereignisse hier noch nicht anwendbar.

Der darüber per Um- und Rückformern gebildete nachrichtentechnische Kanal bietet als Dienst die Übertragung von primären Signalen an. Ihm wird die Schichtnummer 1 zugeordnet. Handelt es sich, wie bei rechnergestützter Telekommunikation üblich, um einen digitalen Kanal, kann man hier die Übergabe einzelner Signalschritte jeweils als Dienstschnittstellenereignis werten.

Unter dem Stichwort Kanalcodierung wurde bereits der Aufgabenkomplex der Sicherung der Informationsübertragung gegen störungsbedingte Verfälschungen angesprochen und es wurde, wie auch bei der Quellencodierung, auf ein späteres Kapitel verwiesen. In der logischen Architektur wird die Herstellung von Kanälen zur gesicherten Übertragung als den nachrichtentechnischen Kanälen direkt übergeordnete Aufgabe angesehen und diesem Aufgabenkomplex wird die Schichtnummer 2 zugeordnet. Er soll hier noch nicht näher erläutert werden. Erwähnt sei nur, daß der an der Dienstschnittstelle der Schicht 2 verfügbare Dienst Primitive zur Anforderung der Übertragung digitaler Datenblöcke und zu deren Auslieferung unterstützt. Die Übertragung erfolgt in gesicherter Weise über den im Diensterbringer enthaltenen Kanal. Sehr oft muß dies in einem Dienstleistungskontext, der sogenannten gesicherten Verbindung, erfolgen, die über zusätzliche Primitive auf- und abgebaut werden kann.

Die Aufgaben der Vermittlung werden der darüberliegenden Schicht 3 zugeordnet. Die Instanzen dieser Schicht sind über Schicht-2-Dienste entsprechend der Topologie des Systems miteinander verbunden. Sie unterstützen Zugangspunkte des Schicht-3-Dienstes an den verschiedenen Orten der Dienstnehmer und bieten diesen unabhängig von der Topologie Dienstleistungen an zur Übertragung digitaler Datenblöcke zwischen i.a. beliebigen Orten des Systems. Auch hier sehen manche Vermittlungsdienste hierfür den Kontext einer Verbindung, der sogenannten Netzwerkverbindung, vor. Das

aus den Schichten 0 bis 3 bestehende Telekommunikationssubsystem wird häufig auch Datennetz genannt.

Bild 3.4 verdeutlicht diese Sicht der logischen Architektur. Es sei erwähnt, daß auch die Vermittlungsschicht, die von den Instanzen abgewickelten Protokolle und deren Mechanismen sowie Vermittlungsdienste in einem späteren Kapitel noch genauer vorgestellt werden.

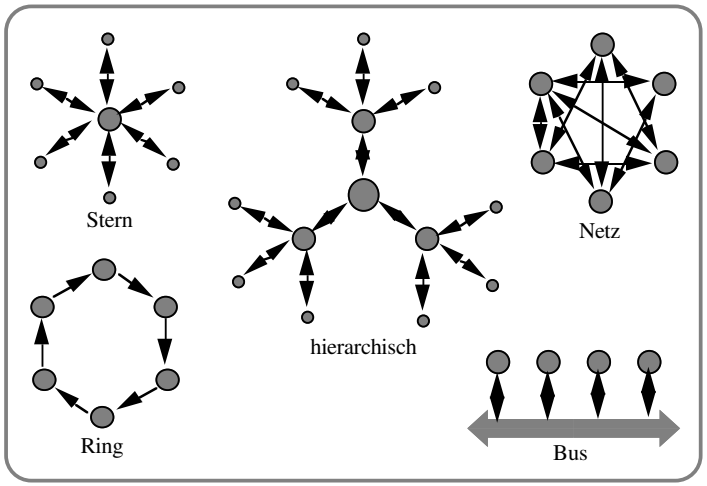

Bild 3.5: Topologie

3.2.2 Netz-Topologien

Die wichtigsten in Vermittlungssystemen verwendeten Netz-Topologien sind in Bild 3.5 dargestellt. Die Sternstruktur sieht nur einen einzigen zentralen Ort zur Vermittlung vor. In der hierarchischen Variante sind dienstnehmernahe Zentralen wiederum sternförmig mit einer Hauptzentralen verbunden. Im gezeigten vermaschten Netz fallen Vermittlungsknoten und Dienstnehmerknoten zusammen, jeder Knoten ist mit mehreren anderen verbunden. Eine Sonderform wäre hier das sehr aufwendige voll vermaschte Netz, in dem zwischen jedem Knotenpaar eine direkte Verbindung besteht.

Demgegenüber enthält ein Netz in Ringform bedeutend weniger Verbindungen. Jeder Knoten ist nur an zwei Medien angeschlossen. Im gezeigten unidirektionalen Ring liegt sogar weitergehend fest, daß eines der beiden nur als Senke und das andere nur als Quelle betrieben werden darf. Beim Ausfall eines Mediums ist hierdurch das ganze Netz betroffen. Beim vermaschten Netz kann dagegen sogar bei Ausfall mehrerer Medien u.U. noch volle Konnektivität gegeben sein. Bei der Sternstruktur ist je Ausfall nur eine Anschaltungsinstanz nicht mehr erreichbar.

Die bisher angesprochenen Formen basieren auf Zwei-Punkt-Verbindungen und es sind Funktionen zu Verwaltung und Betrieb von Teilstrecken notwendig. Rechts unten im Bild ist eine auf der Verwendung eines geteilten Mediums basierende Topologie dargestellt. Um z.B. Zwei-Parteien-Benachrichtigung zu unterstützen, sind hier nur adressierende und selektierende Funktionen und zwar am Ort der Dienstnehmer notwendig.

3.2.3 Durchschaltevermittlung

Die typische Form eines Durschaltevermittlungsnetzes ist ein Leitungsvermittlungsnetz. Leitung ist hierbei gleichbedeutend mit Medium, d.h. das Netz stellt direkte Kopplungen zwischen Medien her. Wichtiger Vertreter ist das herkömmliche Telephonnetz, das noch bis vor wenigen Jahrzehnten ohne ausgesprochene Datenverarbeitungskomponenten allein mit Kupferleitungen als Medien, analogen Signalverstärkern und Filtern sowie mechanischen Umschalteinrichtungen (z.B. Heb/Dreh-Wähler) realisiert wurde. Digitale Informationsdarstellung wurde nur bei Wählimpulsen und internen Abrechnungssignalen verwendet.

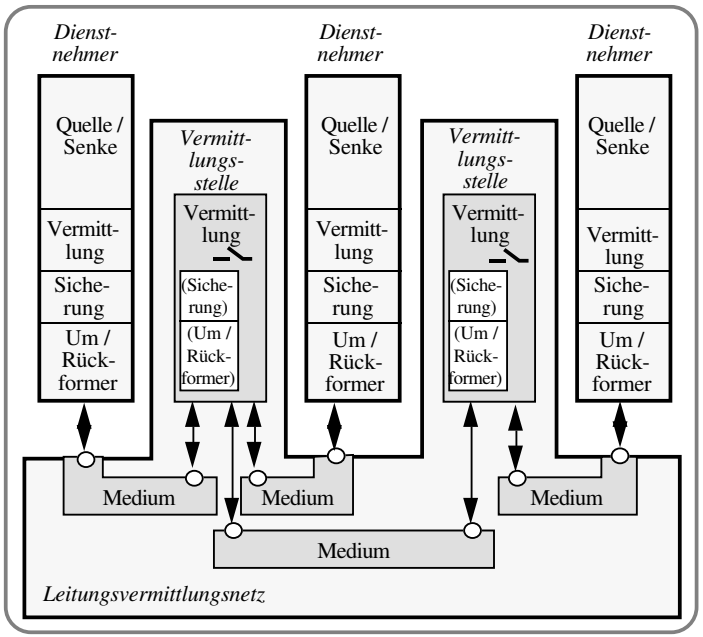

Bild 3.6: Leitungsvermittlungssystem

Bild 3.6 zeigt ein solches Leitungsvermittlungssystem in einer Form, die die mit Bild 3.4 eingeführte Schichtenarchitektur erkennbar werden läßt. Das eigentliche Netz besteht im wesentlichen aus Medien und Kopplungselementen. Die Kopplungselemente sind räumlich in Vermittlungsstellen konzentriert und greifen direkt auf die Medien durch. Die Schichten 1 und 2 sind in den Vermittlungsstellen bezogen auf die Übertragung von Nachrichten zwischen den Teilnehmern im Netz 'leer'.

Ein Netzteilnehmer (Telephon + Mensch) enthält Um- und Rückformer (Mikrophon, Hörkapsel, Wählimpulsformer: Wählscheibe), Sicherungsfunktionen (Redundanz der natürlichen Sprache, 'Wie bitte?') und Adressierungsfunktionen (Bedienung der Wählscheibe).

In der Verbindungsaufbauphase (Hörer abnehmen, wählen, Rufzeichen hören, warten beim Initiator bzw. Klingeln hören, abnehmen beim Beantworter), im wesentlichen während des Wahlvorgangs, und während der Abbauphase wird aber auch Information zwischen Teilnehmer und Netz ausge-

tauscht, die vom Netz selbst und zwar in den Vermittlungsstellen interpretiert oder bestimmt wird. Sie betrifft den beim Telephonnetz 'Signalisierung' genannten Bereich der Wähl- und Abrechnungsinformation (Signalisierung umfaßt ferner hier vereinfachend nicht betrachteten Austausch weiterer netzinterner Verwaltungsinformationen). Sie muß übertragen werden, und die Übertragung muß zumindest rudimentär gesichert werden. Deshalb wurden in der Abbildung der Vermittlungsstellen für den Bereich der Signalisierung entsprechende Instanzen angedeutet.

Inzwischen bilden auch analoge Telephonnetze ein großes Anwendungsgebiet der Informatik, weil herkömmliche Baugruppen immer stärker durch rechnergesteuerte Elemente ersetzt werden. Dies wird sich insbesondere auch im Zuge der Digitalisierung der Übertragung im Rahmen der Einführung des ISDN verstärken. Umgekehrt ist zur Zeit noch das analoge Telephonnetz ein wichtiges Hilfsmittel zur Verwirklichung räumlich weit verteilter Rechnernetze.

Es soll erwähnt werden, daß das Konzept der Durchschaltung auch auf höhere Ebenen übertragbar ist. Es sind auch Systeme möglich, in denen Übertragungswege abstrakterer Natur als physikalische Medien bei Bedarf zwischen Dienstnehmern durchgeschaltet werden. Charakteristisch bleibt auch hier die Unterstützung eines verbindungsorientierten Dienstes, weil die Aufbauphase für diese Maßnahmen genutzt wird.

Im folgenden sollen zwei grundlegende Verfahren zur Kopplung von Medien skizziert werden.

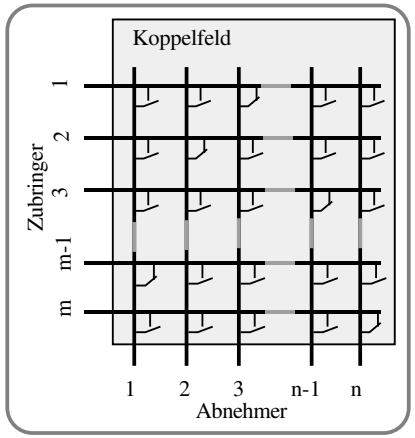

Bild 3.7: Raummultiplex-Durchschaltung

3.2.3.1 Raummultiplex

Gegeben ist eine Menge von Zubringer-Leitungen und eine Menge von Abnehmer-Leitungen. Ein Raummultiplex-Koppelfeld soll nun Zubringer-Leitungen jeweils mit einer beliebigen freien Abnehmer-Leitung koppeln können. Bild 3.7 zeigt einen entsprechenden Aufbau, der je Paar aus Zubringer- und Abnehmer-Leitung ein Schaltelement vorsieht. Die Schaltelemente können einzeln angesteuert werden.

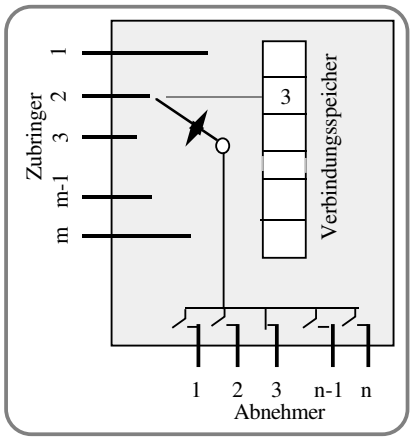

Bild 3.8: Zeitmultiplex-Durchschaltung

3.2.3.2 Zeitmultiplex

Beim Zeitmultiplex-Verfahren werden ebenfalls Zubringer- und Abnehmer-Leitungen unterschieden. Auch werden Zubringerleitungen mit Abnehmer-Leitungen gekoppelt. Allerdings kann zur selben Zeit nur ein Paar gleichzeitig gekoppelt sein. Der zeitliche Verlauf wird in Zeitschlitze eingeteilt, die jeweils einer einzigen Zubringer-Leitung zugeordnet sind. Während des Zeitschlitzes wird die zugeordnete Zubringer-Leitung mit einer wählbaren Abnehmer-Leitung verbunden.

3.2.4 Speichervermittlung

Speichervermittlungsnetze enthalten Pufferspeicher für Nutzdaten und tragen somit zur Entkopplung des Nachrichten-Sendens von der momentanen Medienverfügbarkeit bei.

Typischerweise werden auf der Basis speichervermittelter Netze Datagramm-Dienste erbracht. Wenn einzelne Dienstleistungen nicht in einen umfassenderen Dienstleistungskontext und vor allem auch Adressierkontext eingebettet sind, ergibt sich größtmögliche Flexibilität zum effizienten Einsatz der Betriebsmittel des Netzes.

Neben der Briefpost soll das DATEX-P-Netz der Bundespost als Beispiel genannt werden. Das Datenpaketvermittlungsnetz der Bundespost folgt den CCITT-Empfehlungen X.25 und X.75 und soll die Übertragung von Blöcken digitaler Daten (Datenpakete) vornehmlich zwischen digitalen Datenverarbeitungsgeräten unterstützen. Bild 3.9 gibt eine vereinfachte Darstellung.

Der an der Dienstschnittstelle der Vermittlungsschicht bei den Teilnehmern angebotene Dienst ist allerdings (untypischerweise) verbindungsorientiert. Es ist der Dienstleistungkontext einer Verbindung mit Auf- und Abbauphase vorgesehen. Für den Teilnehmer hat er den Charakter eines Adressier- und Reihenfolgekontextes. Partneradressen werden nur beim Verbindungsaufbau genannt, Pakete werden in derselben Verbindung und Richtung reihenfolgetreu übertragen. Der Erbringer nutzt ihn zur Festlegung des, für alle in der Verbindung ausgetauschten Pakete gleichen, Übertragungswegs und zur Vorabreservierung von Pufferspeicherplatz in den betroffenen Stationen.

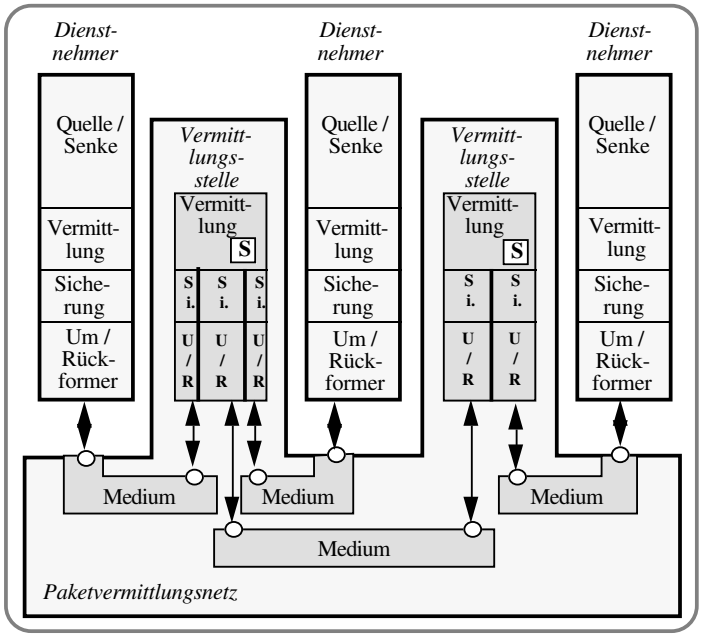

Bild 3.9: Paketvermittlungssystem

Im Unterschied zum Telephonnetz findet sich hier eine sowohl für Steuerinformation als auch für Nutzdaten gleichermaßen anwendbare logische Schichtenarchitektur. Die Vermittlungsstellen enthalten je Medium einen Instanzenturm der Schichten 1 und 2, über den Nutzdaten und Vermittlungssteuerdaten gleichartig geschleust werden. Eine Instanz der Schicht 3 wertet die Steuerdaten aus, puffert die Nutzdaten und leitet sie schnellstmöglich weiter. Dienstnehmerseitig findet sich ebenfalls je Teilnehmer ein Instanzenturm.

An diesem Beispiel soll noch auf folgendes hingewiesen werden. Aus Zuständigkeitsgründen wird ein öffentliches Vermittlungsnetz in die beiden organisatorischen Bereiche Netz und Teilnehmer gegliedert. Der Teilnehmer beginnt und das Netz endet an den Endpunkten der Teilnehmerleitungen. Dies zeigen auch die dicken Umrandungen in den Bildern 3.6 und 3.9. Für den geregelten Betrieb eines Netzes darf der Einflußbereich des Netzbetreibers - als Gesamtverantwortlichem für den von der Vermittlungsschicht angebotenen Dienst - hier aber noch nicht enden. Der mit dem Netz realisierte Vermittlungsdienst bedarf auch am Ort des Teilnehmers eines Instanzenturms für die Schichten 1 bis 3. Nur wenn sie ordnungsgemäß mit den Partnerinstanzen in den Vermittlungsstellen zusammenarbeiten kann der Dienst, wie vorgesehen, erbracht werden. Deshalb sind mit der Zurverfügungstellung eines Endpunkts einer Teilnehmerleitung i.a. auch weitergehende Auflagen an den Teilnehmer verknüpft, deren Einhaltung z.B. durch posteigene Geräte auf Teilnehmerseite oder Zwang zur postalischen Genehmigung gewährleistet werden soll.

3.2.5 Zellenvermittlung im statistischen Multiplex-Betrieb (B-ISDN: ATM)

Zukünftige öffentliche Datenvermittlungsnetze sollen "breitbandig" sein, d.h. Verbindungen mit einer Übertragungsleistung >100Mbit/s unterstützen, um für menschliche Nutzer neben traditionellem Fernsprechen auch weitere Kommunikationsformen (Bildübertragung, Bewegtbildübertragung, z.B. auch in HDTV-Qualität) zu ermöglichen sowie um für Mensch-Rechner- und Rechner-Rechner-Kommunikation kurze Übertragungszeiten auch für Massendaten zu gewährleisten. Der Trend geht zu sogenannten integrierten Breitband- oder Hochgeschwindigkeits(High-Speed)-Netzen, über die Menschen und Rechner multimediale Nachrichten (Ton, Bild, Video, Daten) austauschen können (integriert: ein einziges Netz für verschiedene Kommunikationsformen).

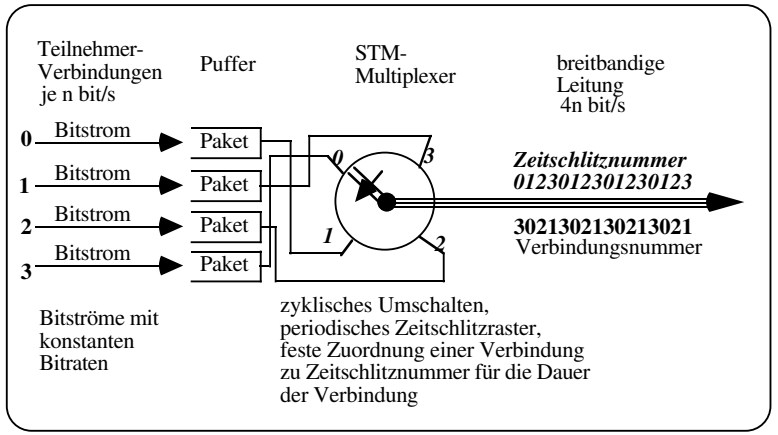

Bild 3.8.1: Zeitmultiplex im STM-Verfahren

Das schon erwähnte ISDN (Integrated Services Digital Network) wird nur als erster Schritt in die Telekommunikationslandschaft der Zukunft gesehen. Der nächste Schritt ergibt sich aus Sicht europäischer Telekom-Anbieter mit B-ISDN: Breitband-ISDN (In USA zeigt sich der Trend, nicht allein auf B-ISDN zu setzen, sondern konkurrierend weitere großflächige High-Speed-Netze anzubieten).

ISDN ist leitungsvermittelt. Zwischen zwei Teilnehmern werden Bitströme mit z.B. 64 Kbit/s je Richtung übertragen. Netzintern werden die Bitströme in kleine Pakete konstanter Länge strukturiert, die in einer Vermittlungsstelle kurzfristig zwischengespeichert werden können. Vermittlungsstellen sind miteinander über breitbandige Leitungen verbunden, die im Zeitmultiplex-Verfahren mehrere (z.B. 30) Teilnehmer-Verbindungen gleichzeitig unterstützen können. Die Zeitmultiplex-Mehrfachnutzung der breitbandigen Leitung folgt dabei dem sogenannten STM(Synchronous Transfer Mode)-Verfahren, das in Bild 3.8.1 skizziert ist. Periodenlänge und Zeitschlitzrasterung sind konstant. Mit dem Aufbau einer Teilnehmerverbindung wird für diese Verbindung ein Zeitschlitz in der Periode fest reserviert, so daß sich eine konstante Übertragungsleistung (Bitrate) ergibt.

Für die Zukunft wird ein Bedarf an kostengünstigen Teilnehmer-Verbindungen hoher Bandbreite gesehen. Dabei ist die benötigte Bandbreite je Teilnehmer-Verbindung unterschiedlich (individuell festgelegte Übertragungsleistung) und sie kann auch im Verlauf der Betriebszeit bei derselben Verbindung stark schwanken (variable Übertragungsleistung). So kann eine Video-Übertragung bei geeigneter Codierung in Phasen geringer Bildänderungen nur sehr wenig, in Phasen starker Bildände-

rungen aber sehr viel Übertragungsleistung beanspruchen. Ein anderes Beispiel wäre die Mensch-Rechner-Kommunikation innerhalb einer interaktiven Sitzung: kommunikationsarme Denkphasen wechseln sich mit Dialogphasen ab.
Ein Teilnehmer der Zukunft würde gerne beim Verbindungsaufbau vom Dienstleister die Zusicherung erhalten, daß der u.U. phasenweise benötigte Maximalbedarf gedeckt werden wird. Er würde aber sehr gerne nur den am Ende der Verbindung tatsächlich stattgefundenen Kommunikationsaufwand bezahlen.

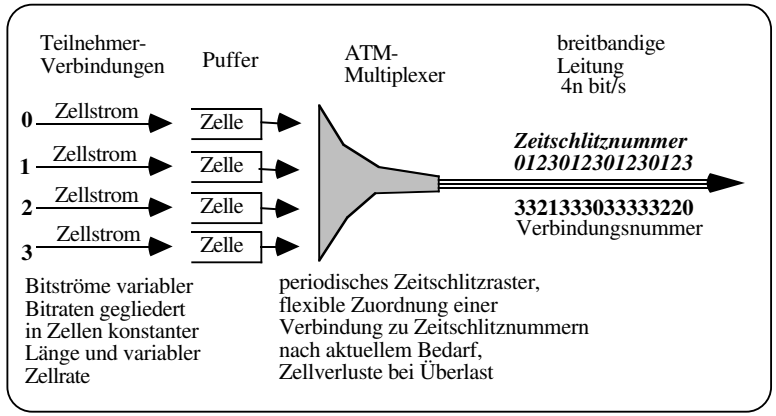

Bild 3.8.2: Zeitmultiplex im ATM-Verfahren

Auf solche (in vielen Prognosen als zukünftig dominierend angesehene) Fälle ist das vom CCITT im Jahr 1988 definierte B-ISDN auf ATM-Basis (Asynchronous Transfer Mode) ausgerichtet (siehe z.B. G. Siegmund: ATM - Die Technik des Breitband-ISDN, R.v.Decker's Verlag, 1993), für das seit 1992 bereits Komponenten prototypisch realisiert sind.
Auch im ATM-Netz sollen netzintern im Zeitmultiplex-Verfahren Leitungen zwischen Vermittlungsstellen mehrfachgenutzt werden, aber nicht mehr in einem starren Reservierungsschema. Die Zeitschlitze der Leitungen zwischen Vermittlungsstellen werden je Periode neu den Teilnehmer-Verbindungen zugeordnet, so daß Verbindungen, die momentan eine hohe Übertragungsleistung erfordern viele Zeitschlitze belegen können, während Teilnehmerverbindungen, die momentan eine geringe Leistung erfordern, wenige bis gar keine Zeitschlitze belegen. Die Belegung der Zeitschlitze erfolgt in Abhängigkeit zur zufällig momentan vorhandenen Bedarfssituation: statistisches Multiplexing.

Bild 3.8.2 skizziert das ATM-Verfahren. Der Bitstrom einer Teilnehmer-Verbindung wird in Zellen genannte Pakete gegliedert. Jede Zelle enthält neben den vom Teilnehmer bestimmten Nutzdaten zusätzlich einen Zellenkopf (Zellenlänge: 53 Oktetts/Bytes; 48 Oktetts Nutzdaten; 5 Oktetts Kontrollinformation: Flußkontrolle, Verbindungskennzeichnung aus virtueller Kanalnummer und virtueller Pfadnummer, Zelltyp, Priorität, Zellkopffehlersicherungsfeld), der er es ermöglicht, die Zellen des Zellstroms einer Übertragungsrichtung einer Verbindung ähnlich einer Paketvermittlung isoliert voneinander weiterzuleiten. Die Zellen einer Richtung einer Verbindung können in unterschiedlichen Zeitschlitzen derselben Leitung und sogar in unterschiedlichen Leitungen transportiert werden. Sie müssen aber alle denselben Pfad durch das Netz nehmen, um Reihenfolgevertauschungen zu vermeiden.

Teilnehmer treten im ATM-Netz als zufallsgesteuerte "stochastische" Lastquellen auf. Das Netz läßt im Hinblick auf die Summe der geforderten Maximalleistung der aktuellen Teilnehmer eine Überbuchung zu, geht aber davon aus, daß die momentanen Leistungsanforderungen der einzelnen Teilnehmer zueinander unabhängig sind, so daß Zellenverluste wegen Überlastung sehr unwahrscheinlich sind (Zellverlustwahrscheinlichkeit kleiner 10^{-9}).

Die Zellverlustwahrscheinlichkeit kann nur garantiert werden, wenn Kenngrößen der Lastquellen mit dem Verbindungsaufbau bekannt gemacht und während des Betriebs eingehalten werden. Wichtige Kenngrößen zur Verbindungsqualität (Quality of Service; QoS) sind:

- Laufzeit (Delay; unkritisch z.B. bei Dateitransfer, kritisch bei Sprache und Video),
- Laufzeitschwankung (Jitter; kritisch bei Sprache und Video),
- Durchsatz (Zell/Bitrate, Übertragungsleistung, Bandbreite):
  maximaler Durchsatz, mittlerer Durchsatz, Burstrate,
- maximale Zellenverlustwahrscheinlichkeit.

Beim Aufbau einer neuen Verbindung teilt der Teilnehmer dem Erbringer die geforderten QoS-Parameter mit. Der Erbringer entscheidet unter Berücksichtigung der gegenwärtigen Auslastungssituation, ob die neue Verbindung unterstützt werden kann (Admission Control oder Call Acceptance Control genannt).
Im Fall des geglückten Aufbaus sind die Teilnehmer verpflichtet, die beim Aufbau vereinbarten Kenngrößen zum Durchsatz nicht zu verletzen. Dies wird während des Betriebs der Verbindung überwacht (Policing). Sehr nahe an der Quelle (am sendenden Teilnehmer) wird der Zellstrom vom Netzerbringer daraufhin überwacht, ob die ausgehandelten Langzeitparameter momentan überschritten werden. Wenn ja, werden umgehend Gegenmaßnahmen getroffen, die vom Markieren einzelner Zellen (zum bevorzugten Ignorieren im Falle tatsächlicher Überlastsituationen) über die sofortige Eliminierung einzelner Zellen bis hin zum sofortigen Abbruch der Verbindung reichen können.

Kritisch für ATM-Netze sind geeignete Admission Control und Policing Verfahren.
Admission Control Verfahren sollen schnell sein, aber die Zahl unnötiger Ablehnungen oder ungerechtfertigter Annahmen gering halten.
Policing Verfahren sollen den Zellfluß mit sehr kurzer Ansprechzeit kontrollieren, um Überlastungen des Netzes (mit störenden Auswirkungen auf andere Verbindungen) gering zu halten. Regulierende Eingriffe sollen aber möglichst nur dann erfolgen, wenn eine Verbindung die ausgehandelten Werte wirklich überschreitet.

3.3 Datenübertragungseinrichtungen und Schnittstellen

Nachdem Aufbau und Funktion nachrichtentechnischer Einrichtungen zur Telekommunikation erläutert wurden, soll im folgenden auf den Anschluß solcher Einrichtungen an Digitalrechner eingegangen werden. Er erfolgt üblicherweise über die Ankopplung sogenannter Datenübertragungseinrichtungen an den Rechner. Sie werden wie sonstige Geräte der Rechnerperipherie über Ein/Ausgabe-Anschalt–einheiten mit einer Ein/Ausgabe-Schnittstelle des Rechners verbunden. Deshalb soll zunächst der Anschluß und Betrieb von Peripheriegeräten im allgemeinen kurz umrissen werden.

3.3.1 Peripheriegeräteanschluß

Der Anschluß eines Peripheriegeräts folgt üblicherweise dem mit Bild 3.10 gezeigten Schema. Links ist die Zentraleinheit eines Digitalrechners in der Von-Neumann-Architektur skizziert. Sie wird von Arbeitsspeicher, Prozessor(en) und E/A-Anschlußstelle(n) sowie deren interner Verbindung durch Adreß-, Daten- und Steuerleitungen geprägt. An einer E/A-Anschlußstelle wird eine Schnittstelle zum Anschluß eines Peripheriegeräts unterstützt. Eine solche E/A-Schnittstelle des Rechners ist i.a. rech-

nerbauart-spezifisch und für den räumlich sehr nahen Anschluß eines Geräts ausgelegt (z.B. < 0,5 m). Sie sieht i.a. eine größere Anzahl digitaler Signalleitungen vor, in denen nach rechnerinternen Konventionen Adreß-, Daten- und Steuerinformation mit dem Gerät ausgetauscht werden. Die Signalübertragung ist hier i.a. parallel. Über z.B. 8 gleichlaufende Daten-Ausgabeleitungen kann in einem Signalschritt ein ganzes Byte von der Zentraleinheit an das Gerät übergeben werden.

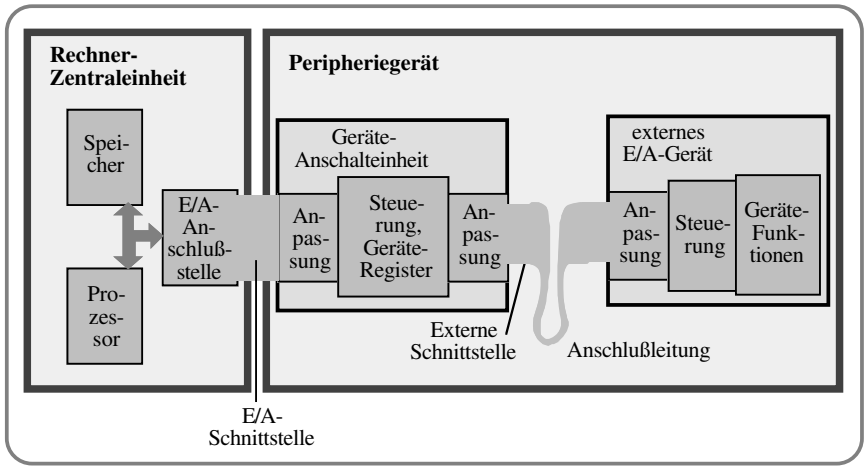

Bild 3.10: Schema zum Anschluß eines Peripheriegeräts

Die eigentlichen Peripheriegeräte sollen in etwas größerem Abstand von der Zentraleinheit plaziert werden können (z.B. in einem wenige 10 m umfassenden Bereich) und werden mit einer längeren Anschlußleitung mit dem Rechner verbunden. Ferner sind sie üblicherweise unabhängig von der speziellen Bauart des Rechners ausgelegt. Deshalb wird ein Bindeglied zwischen externem Gerät und E/A-Anschlußstelle, die Geräte-Anschalteinheit, nötig. Aus der Sicht der Zentraleinheit gehört sie zum Gerät, die der Zentraleinheit zugreifbaren Funktionen sind die Funktionen der Anschalteinheit. Örtlich befindet sie sich (z.B. als Einschubplatine ausgeführt) direkt an der Zentraleinheit. Sie ist rechnerbauart-spezifisch ausgelegt, unterstützt aber eine (u.U. geräte-spezifische) rechnerunabhängige externe Schnittstelle zum eigentlichen Gerät.

Im folgenden soll zunächst auf die Ausführung externer Schnittstellen und danach darauf eingegangen werden, wie ein Gerät vom Rechner aus angesprochen werden kann.

3.3.1.1 Externe-Schnittstellen

Die externe Schnittstelle ist i.a. nach von mehreren Herstellern anerkannten Konventionen ausgeführt. Sie beziehen sich auf die verwendeten Steckverbindungen, das Medium, seine maximale Länge sowie die für die Übertragung eingesetzten Signalpegel und Signalformen und bei paralleler Übertragung deren gegenseitige Beziehungen.

Als Medien werden oft parallele elektrische Verbindungen verwendet, die zu Kabeln zusammengefaßt sind, und es erfolgt Binärübertragung nach dem Basisbandverfahren. Der Parallelitätsgrad richtet sich nach der Gerätegeschwindigkeit. Langsame und einfache Geräte (wie z.B. Sichtgeräte) werden oft

bitseriell betrieben, während mit schnelleren und in ihrer Funktion anspruchvolleren Geräten (wie z.B. Peripherspeichereinheiten) wortweise Übertragung für Daten und Steuerung stattfindet.

Zur Unterstützung einer Schnittstelle sind rechner- und geräteseitig Anpassungseinrichtungen vorgesehen. Es handelt sich dabei im wesentlichen um Verstärker, Schwellwertschalter und Filter zur Signalformung und Abtastung sowie um Parallel / Serienwandler zur Anpassung verschiedener Parallelitätsgrade von E/A-Anschlußstelle, externer Schnittstelle und geräteinterner Verarbeitung. Wichtig sind weiterhin Maßnahmen zur galvanischen Entkopplung von Zentraleinheit und externem Gerät sowie Überspannungsschutz-Einrichtungen.

Beispiele für externe Schnittstellen sind die bitserielle 20mA-Schnittstelle einfacher zeichenweise arbeitender Geräte und die SCSI-Schnittstelle für Peripheriespeichergeräte.

3.3.1.2 Gerätezugriff

Für die Programmierung der Zentraleinheit sind einerseits die im einzelnen ansprechbaren Funktionen eines Peripheriegeräts und andererseits die verwendeten Verfahren zum Ansprechen der Funkionen von Interesse.

Die Funktionen sind allgemein:

- Lesen des Betriebszustands; der momentane Betriebszustand des Geräts wird in einem oder mehreren Status-Wörtern binär codiert der rechnerinternen Auswertung verfügbar,
- Ändern des Betriebszustands; durch Ausgabe von Adreß- oder sonstiger Steuerinformation an das Gerät wird dessen Zustand für zukünftige Funktionen vorbereitend eingestellt,
- Dateneingabe; vom Rechner aus wird eine Dateneinheit angefordert, die vom Gerät aus zur Verfügung zu stellen ist (z.B. Code des aktuell an einer Tastatur eingegebenen Zeichens),
- Datenausgabe; vom Rechner aus wird dem Gerät eine Dateneinheit zur Ausgabe zur Verfügung gestellt (z.B. Code des nächsten auf dem Bildschirm auszugebenden Zeichens).

Das Ansprechen der einzelnen Funktionen wird üblicherweise zweistufig erklärt und auch realisiert:

- Geräteregister werden von der Zentraleinheit aus geschrieben und gelesen,
- Geräteaktionen werden vom Gerät in Abhängigkeit zur Registerbelegung ausgeführt.

Die Geräte-Anschalteinheit enthält einen Satz von Speicherzellen (Geräte-Register). Sie gliedern sich in Status-, Adreß- und Datenregister. Vom Rechner aus werden diese Register schreibend oder lesend angesprochen (z.B. Datenausgabe: Datenregister beschreiben). Für die Geräteseite ist dann definiert, welche Aktionen in Abhängigkeit zur momentanen Registerbelegung oder Veränderung ausgeführt werden, und welche Auswirkungen Aktionen des Geräts auf den Registersatz haben.

Die verwendeten Verfahren sind im wesentlichen:

- Programmierte Ein/Ausgabe,
- Unterbrechungsgesteuerte Ein/Ausgabe,
- Selbständiges Gerät.

3.3.1.2.1 Programmierte Ein/Ausgabe

Bei der programmierten Ausgabe sieht der Befehlssatz des Prozessors Maschinenbefehle zum Schreiben und Lesen von Geräteregistern vor. Ein- und Ausgabevorgänge werden unter Kontrolle des Prozessors programmgesteuert abgewickelt.

Für die programmierte Ausgabe eines n Wörter umfassenden Blocks 'B : array [0..n-1] of Wort' ist im groben folgendes Programmstück vorzusehen:

```
write (Betriebsart -> Steuerregister) ;          (* Festlegung der Betriebsart *)
for I := 0 to n-1 do
    begin
    write ( B[I] -> Datenregister) ;             (* Ausgabe eines Worts anstoßen *)
    repeat                                       (* Warten bis Gerät Wortausgabe abgewickelt hat,  *)
        read (Statusregister -> Status) ;        (* und zu neuer Ausgabe bereit ist *)
        until Status = 'Wortausgabe abgewickelt' ;
    end ;
```

Für die programmierte Eingabe eines n Wörter umfassenden Blocks 'B : array [0..n-1] of Wort' ist im groben folgendes Programmstück vorzusehen:

```
for I := 0 to n-1 do
    begin
    write (Betriebsart -> Steuerregister) ;      (* Festlegung Betriebsart und Anstoß Worteingabe *)
    repeat                                       (* Warten bis Gerät Worteingabe abgewickelt hat, *)
        read (Statusregister -> Status) ;        (* und Wort in Datenregister vorliegt *)
        until  Status = 'Worteingabe abgewickelt' ;
    read (Datenregister -> B[I]) ;               (* Wort einlesen *)
    end ;
```

Als problematisch sind hierbei die aktiven Warteschleifen (repeat ...) anzusehen. Der Prozessor wird während der im Vergleich zu seiner Bearbeitungsgeschwindigkeit langsamen Gerätetätigkeit 'verschwendet'. Er kann während der Wartezeit nicht mit anderen - seine Datenverarbeitungsfähigkeiten wirklich nutzenden - Aufgaben beauftragt werden. So werden langsame Geräte zum Engpaß für ein Gesamtsystem. Auch bei sehr schnellen Geräten können sich Nachteile ergeben. Hier kann umgekehrt der Prozessor zum Engpaß werden.

3.3.1.2.2 Unterbrechungsgesteuerte Ein/Ausgabe

Bei unterbrechungsgesteuerter Ein/Ausgabe entfällt das aktive Warten. Hierzu muß der Prozessor unterbrechungsfähig sein. Die Geräteanschaltung muß den Prozessor durch ein besonderes Signal, das Unterbrechungssignal (Interrupt), in der Bearbeitung eines Programms unterbrechen können und ihn zur Verzweigung in die Bearbeitung einer besonderen Prozedur, der Unterbrechungsprozedur (Interruptroutine) veranlassen können. Die Geräteanschaltung ihrerseits muß in der Lage sein, das Statusregister selbsttätig auszuwerten, das Ende einer Aus- oder Eingabe zu erkennen und durch ein Unterbrechungssignal zu melden.

Für die unterbrechungsgesteuerte Ausgabe eines n Wörter umfassenden Blocks 'B : array [0..n-1] of Wort' sind im groben folgendes Programmstück und folgende Unterbrechungsprozedur vorzusehen:

```
write (Betriebsart -> Steuerregister) ;          (* Festlegen der Betriebsart *)
for I := 0 to n-1 do
    begin
    write ( B[I] -> Datenregister) ;             (* Ausgabe eines Worts anstoßen *)
    'Laufenden Prozeß blockieren und anderen Prozeß fortsetzen' ;
                                                 (* Blockieren des laufenden Prozesses bis er von *)
    end ;                                        (*  Unterbrechungsprozedur wieder gelöst wird *)

procedure ITR ;                                  (* Unterbrechungsprozedur *)
    begin                                        (*   wird bei Geräte-Unterbrechung aktiviert *)
    'Retten des Status des unterbrochenen Prozesses' ;
    'Fortsetzen des wartenden Prozesses' ;
    end ;
```

Für die unterbrechungsgesteuerte Eingabe eines n Wörter umfassenden Blocks 'B : array [0..n-1] of Wort' sind im groben folgendes Programmstück und folgende Unterbrechungsprozedur vorzusehen:


```
for I := 0 to n-1 do
    begin
    write (Betriebsart -> Steuerregister) ;      (* Betriebsart und Eingabe eines Worts anstoßen*)
    'Laufenden Prozeß blockieren und anderen Prozeß fortsetzen' ;
                                                 (* Blockieren des laufenden Prozesses bis er von *)
                                                 (*  Unterbrechungsprozedur wieder gelöst wird *)
    read (Datenregister -> B[I]) ;               (* Wort einlesen *)
    end ;

procedure ITR ;                                  (* Unterbrechungsprozedur *)
    begin                                        (*   wird bei Geräte-Unterbrechung aktiviert *)
    'Retten des Status des unterbrochenen Prozesses' ;
    'Fortsetzen des wartenden Prozesses' ;
    end ;
```

Die unterbrechungsgesteuerte Ein/Ausgabe ist heute weitgehend Standard, d.h. Prozessoren sind unterbrechungsfähig und Geräte-Anschalteinheiten besitzen einen Unterbrechungsausgang. Zur geregelten Abwicklung der E/A und zur vorteilhaften Nutzung der Wartezeiten wird Unterstützung durch Betriebssystemsoftware vorgesehen. Das Betriebssystem verwaltet die Betriebsmittel Prozessor und E/A-Geräte. E/A-Befehle sind privilegiert. Sie dürfen nur von speziellen Betriebssystem-Prozessen, den Geräte-Treibern, denen jeweils eine Unterbrechungsprozedur zugeordnet ist, aus aufgerufen werden. Anwenderprozesse enthalten anstelle von E/A-Befehlen Betriebssystemaufrufe, mit denen die Definition eines Ein/Ausgabe-Auftrags durch das Betriebssstem an einen Treiberprozeß zur Abwicklung weitergeleitet wird.

3.3.1.2.3 Selbständige Geräte

Wie erwähnt, kann bei sehr schnellen Geräten der Prozessor zum Engpaß bei Ein/Ausgabe-Vorgängen werden. Um dies zu vermeiden, können spezielle Prozessoren, die sogenannten E/A-Prozessoren auch selbständige Kanäle genannt, zur Unterstützung schneller E/A-Anschlußstellen vorgesehen werden. Ein E/A-Prozessor ist in seinem Befehlsvorrat beschränkt. Er ist nur auf die schnelle Abwicklung von E/A-Vorgängen ausgelegt. Er ist aber in dem Sinn ein 'richtiger' Prozessor, daß er unabhängig von den übrigen Prozessoren und simultan zu ihnen im Arbeitsspeicher stehende Programme ausführen kann. Die Programme werden Kanal-Programme genannt. Man findet E/A-Prozessoren hauptsächlich bei größeren Rechnern. Dort unterstützen sie komplexere E/A-Vorgänge schneller Geräte, wie z.B. das Positionieren des Schreib/Lesekopfes eines Plattenspeichers mit anschließendem Transfer eines Blocks und Prüflesen. Sie dienten traditionell dazu, komplexe Gerätesteuerungsfunktionen aus der Geräteanschaltung in frei programmierbarer Weise in die Zentraleinheit zu verlagern, um die Kosten der Geräteanschaltungen zu reduzieren. Mit der preiswerten Verfügbarkeit von Mikroprozessoren ist die Notwendigkeit hierzu entfallen und die Aufgaben von E/A-Prozessoren werden zunehmend geräteintern und in intelligenten Geräteanschalteinrichtungen abgewickelt. Dies soll im folgenden unter der Bezeichung 'Selbständiges Gerät' angesprochen werden.

Die Geräte-Anschalteinrichtung wird mit mehr Intelligenz versehen (z.B. per ROM programmierter Mikroprozessor mit etwas Schreib/Lesespeicher). Sie läßt zwar keine freizügige Programmierung von E/A-Vorgängen zu, unterstützt aber, daß ein ganzer Vorgang, wie z.B. Blocktransfer, durch Parameterbelegung vorab definiert wird und dann allein unter Kontrolle der Anschalteinrichtung abgewickelt wird. Diese Parameter werden ebenfalls in Geräteregistern gehalten. Wesentlich ist dabei, daß die Anschalteinheit von sich aus Blöcke wortweise aus dem Arbeitsspeicher lesen und an das externe Gerät weiterleiten kann bzw. umgekehrt vom externen Gerät ankommende Wörter in einen Bereich des Arbeitsspeichers als Block ablegen kann. Entsprechende Register sind das Adreßregister, in dem dem Gerät die Lage eines Blocks im Arbeitsspeicher mitgeteilt wird, und ein Zählregister, in dem die Anzahl der zu übertragenden Wörter dokumentiert ist (Blocklänge). Die Geräteanschaltung greift von sich aus auf den Arbeitsspeicher des Rechners zu und konkurriert somit - i.a. sogar hochprior - mit den Prozessoren des Rechners - und den übrigen selbständigen Geräteän-

schaltungen - um Speicherzugriffsszyklen (Cycle Stealing). Die Fähigkeit des direkten Arbeitsspeicherzugriffs wird oft unter dem Kürzel DMA angesprochen (Direct Memory Access).

Natürlich werden E/A-Vorgänge immer noch von einem im Rechner ausgeführten Programm aus angestoßen (Treiberprozeß) und dieses Programm muß im Hinblick auf die Beendigung eines Vorgangs auch mit der Gerätetätigkeit z.B. durch Warten synchronisiert werden. Synchronisationspunkte sind hier aber nicht mehr der Start und die Fertigmeldung des Transfers einzelner Wörter sondern dem Programmablauf erscheint der Transfer eines ganzen Blocks als Einheit. Der Treiber wartet also nach Anstoß eines Blocktransfers auf dessen Ende. Auch hier wird durch Unterbrechungsfähigkeit das aktive Warten vermieden.

Für die selbständige Ausgabe eines n Wörter umfassenden Blocks 'B : array [0..n-1] of Wort' sind im groben folgendes Programmstück und folgende Unterbrechungsprozedur vorzusehen:

```
write (Adresse von B -> Adreßregister) ;          (* Blockadresse im Arbeitsspeicher: DMA-Parameter *)
write (n-1 -> Längenregister) ;                   (* Blocklänge: DMA-Parameter *)
write (Betriebsart -> Steuerregister) ;           (* Festlegen der Betriebsart und Start *)
                (* Die Geräteanschaltung transferiert jetzt wortweise den Block zum externen Gerät *)
'Laufenden Prozeß blockieren und anderen Prozeß fortsetzen' ;
                (* Blockieren des laufenden Prozesses bis er von Unterbrechungsprozedur gelöst wird *)
read(Statusregister -> Status) ;                  (* Auftragsbesonderheiten aufgetreten ? *)

procedure ITR ;                                   (* Unterbrechungsprozedur, wird bei *)
    begin                                         (*    Geräte-Unterbrechung 'Blockende' aktiviert *)
    'Retten des Status des unterbrochenen Prozesses' ;
    'Fortsetzen des wartenden Prozesses' ;
    end ;
```

Für die selbständige Eingabe eines n Wörter umfassenden Blocks nach 'B : array [0..n-1] of Wort' sind im groben folgendes Programmstück und folgende Unterbrechungsprozedur vorzusehen:

```
write (Adresse von B -> Adreßregister) ;          (* Blockadresse im Arbeitsspeicher: DMA-Parameter *)
write (n-1 -> Längenregister) ;                   (* Blocklänge: DMA-Parameter *)
write (Betriebsart -> Steuerregister) ;           (* Festlegen der Betriebsart und Start *)
                (* Die Geräteanschaltung transferiert jetzt wortweise den Block in den Arbeitsspeicher *)
'Laufenden Prozeß blockieren und anderen Prozeß fortsetzen' ;
                (* Blockieren des laufenden Prozesses bis er von Unterbrechungsprozedur gelöst wird *)
read(Statusregister -> Status) ;                  (* Auftragsbesonderheiten aufgetreten ? *)

procedure ITR ;                                   (* Unterbrechungsprozedur, wird bei *)
    begin                                         (*    Geräte-Unterbrechung 'Blockende' aktiviert *)
    'Retten des Status des unterbrochenen Prozesses' ;
    'Fortsetzen des wartenden Prozesses' ;
    end ;
```

Variante der Blocklängenbestimmung durch vorherige Längenangabe kann sein, vorab nur eine Maximallänge anzugeben, ferner jedoch auch ein spezielles Endewort in einem entsprechenden Geräteregister zu hinterlegen. Die Anschaltung bricht den Blocktransfer dann vor Erreichen der Maximallänge ab, wenn ein dem Endewort identisches Wort übertragen wurde. Die tatsächliche Blocklänge kann anschließend vom Rechner aus einem weiteren Geräteregister (oder dem modifizierten Längenregister) ermittelt werden.

Selbständige Geräte, d.h. entsprechende Geräteanschaltungen, werden auch in der Rechner-Telekommunikation mit der zunehmenden Übertragungsleistung verwendeter Netze immer wichtiger.

3.3.2 Anschluß an öffentliche Netze

Der Anschluß eines Teilnehmers an ein öffentliches Vermittlungsnetz (und auch an fest vom öffentlichen Träger gemietete Leitungen) folgt dem in Bild 3.11 gezeigten Schema. Der teilnehmerseitige

Endpunkt seiner Anschlußleitung endet in einer sogenannten Datenübertragungseinrichtung (DÜE), die vom Netzbetreiber gemietet werden muß oder von ihm zumindest stark reglementiert wird und genehmigungspflichtig ist, um Störungen des Netzes durch unzuträgliche Signaleinspeisung zu vermeiden. Die Datenübertragungseinrichtung besitzt eine Schnittstelle für die räumlich nahe Ankopplung an den eigentlichen Teilnehmer in Form einer Datenendgerät (DEE) genannten Einrichtung. Bei der digitalen Telekommunikation ist das DEE ein Rechner oder ein Datensichtgerät.

Bei Verwendung des Telephonnetzes z.B. heißt die DÜE Modem und übernimmt die Formung der Wählimpulse, das Aufschalten von Ruhe- und Abnahmesignalen sowie die Frequenztastung der Nutzdaten.

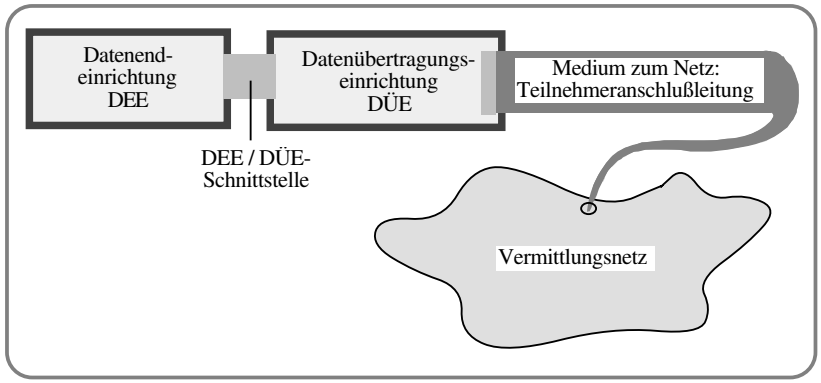

Bild 3.11: Teilnehmer an öffentlichem Netz

Der damit verträgliche Anschluß eines Rechners an ein öffentliches Netz ist in Bild 3.12 gezeigt. In dieser Sicht ist die DÜE ein externes Gerät, das über die DEE / DÜE-Schnittstelle mit einer Geräte-Anschalteinrichtung des Rechners verbunden ist. Damit wird die DÜE vom Rechner aus wie ein 'normales' Peripheriegerät angesprochen.

Die Funktionen eines Geräts des Typs DÜE können im groben sein:

- Status lesen, um zu erkennen, ob momentan eine Verbindung aufgebaut ist, ob momentan ein Verbindungsaufbauwunsch von außerhalb besteht, ob aktuell ein Eingabewort zum Einlesen vorliegt, ob die Ausgabe eines Worts abgeschlossen ist oder ob Fehlersituationen aufgetreten sind,
- Status schreiben, um eine Verbindung abzubauen oder einen Verbindungsaufbauwunsch anzunehmen,
- Adreßausgabe zur Übergabe von Wählzeichen,
- Datenausgabe zur Ausgabe eines Worts über eine bestehende Verbindung,
- Dateneingabe zum Einlesen eines angekommenen Worts.

Es empfehlen sich Unterbrechungsfähigkeiten in der DÜE-Anschalteinrichtung. Bei schnelleren und ihrerseits Blocktransfer unterstützenden Netzen wird die Anschalteinrichtung als selbständiges Gerät ausgelegt.

Zur weitergehenden Erläuterung des Anschlusses an öffentliche Netze ist die DEE / DÜE - Schnittstelle näher zu besprechen. Sie wird vom Netzbetreiber in der Regel nach CCITT-Empfehlungen in

den verwendeten Steckverbindungen, Signalpegeln und Formen und deren Bedeutung festgelegt. Es handelt sich um rechnerhersteller-unabhängige Festlegungen.

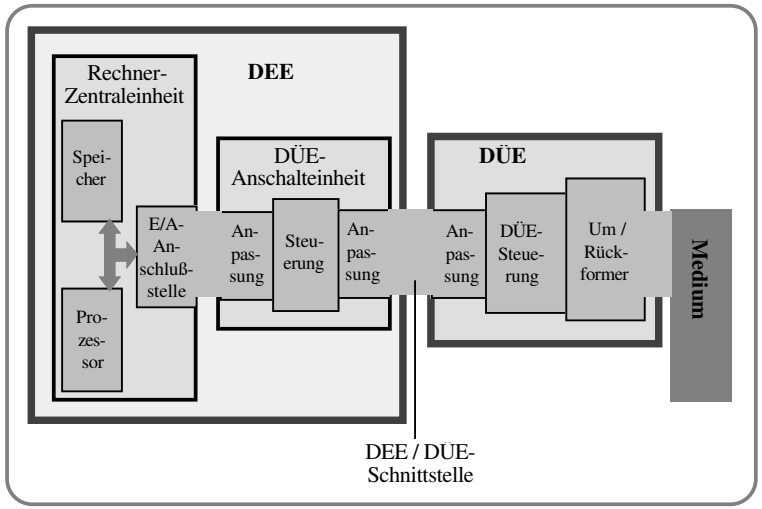

Bild 3.12: Anschluß an öffentliches Netz

Wichtig sind externe Schnittstellen nach den älteren CCITT-Empfehlungen V.24 mit V.25. Sie befassen sich mit dem Anschluß zeichenweise arbeitender DÜEs, z.B. für den Anschluß ans Telephonnetz. Im Zuge der Einführung des auf die digitale Telekommunikation zugeschnittenen DATEX-Netzes wurde ferner mit der CCITT-Empfehlung X.21 eine inzwischen sehr wichtige bitseriell arbeitende Schnittstelle genormt. Beim in naher Zukunft eingeführten ISDN ist eine feinere Struktur der Teilnehmerseite zu berücksichtigen. Hier ist die Anschaltung mehrerer und verschiedener Endgeräte an denselben Netzanschluß möglich.

3.3.2.1 Schnittstelle nach V.24, V.25

Bild 3.13 zeigt die wichtigsten Leitungen und eine mögliche Steckverbindungsausführung für eine DEE / DÜE -Schnittstelle nach V.24. Derartige Schnittstellen finden sich bei Telephon-Modems. An vielen Rechnern enstprechen auch die zum Anschluß von Datensichtgeräten vorgesehenen Schnittstellen der V.24-Empfehlung, die sich im wesentlichen mit der amerikanischen RS232-Norm deckt.

Zum Anschluß von Wählmodems, die das rechnergesteuerte Anwählen von Telephonverbindungen unterstützen, werden die V.24-Richtlinien um die Empfehlungen nach V.25 erweitert. Bild 3.14 soll einen Einblick über die dort vorgesehenen zusätzlichen Schnittstellenleitungen geben.

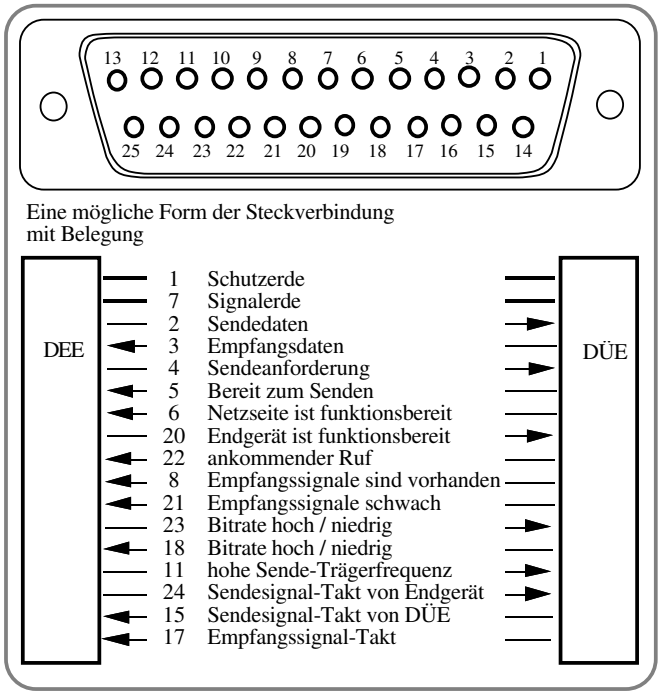

Bild 3.13: Schnittstelle nach CCITT-Empfehlung V.24

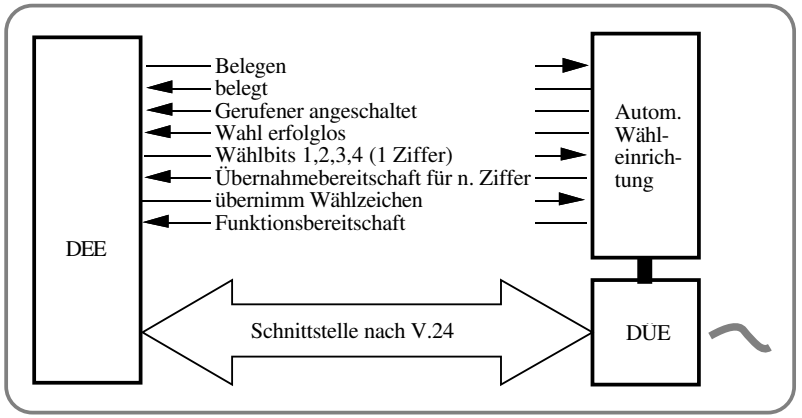

Bild 3.14: Schnittstelle nach CCITT-Empfehlung V.24, V.25

3.3.2.2 Schnittstelle nach X.21

Schnittstellen nach der neueren Empfehlung X.21 werden im Rahmen des DATEX-Netzes benutzt. Im Vergleich zu V.24 fällt hier die geringere Anzahl der Steuer- und Meldeleitungen auf. Außer Taktleitungen gibt es nur zwei Leitungen, die anzeigen, ob momentan Nutzdaten oder Steuerdaten zwischen DEE und DÜE übertragen werden. Steuer- und Meldeinformationen sind binär codiert und werden ebenso bitseriell über die Sende- bzw. Empfangsleitung ausgetauscht wie Nutzdaten. Bild 3.15 skizziert die Schnittstelle.

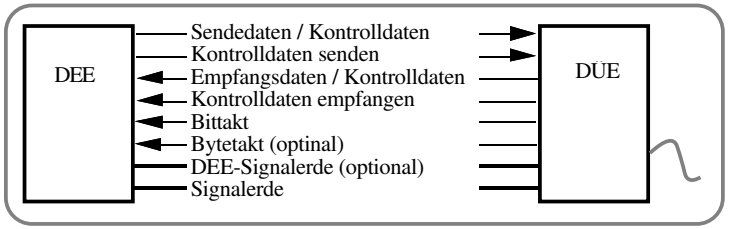

Bild 3.15: Schnittstelle nach CCITT-Empfehlung X.21

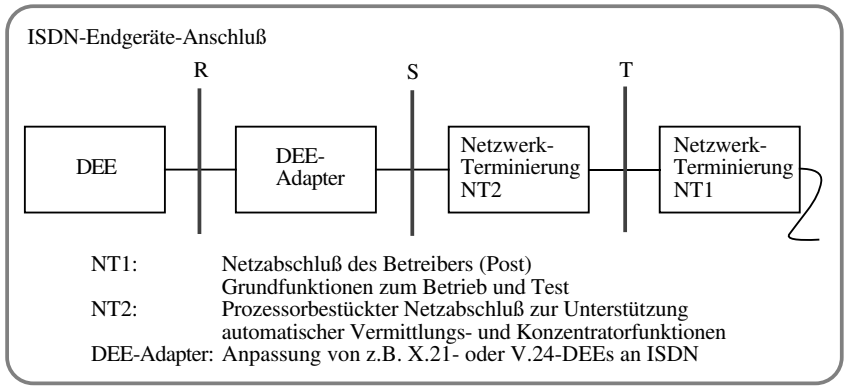

Bild 3.16: Endgeräteanschluß an ISDN

3.3.2.3 Externe Schnittstellen im ISDN

Beim zukünftigen ISDN (Integrated Services Digital Network) sollen über denselben Netzzugang die unterschiedlichsten Telekommunikationsdienste dem Dienstnehmer zugänglich werden (Telephon, Telefax, Bildschirmtext, digitale Interrechnerkommunikation etc.). Grundform dafür ist die digitale Nachrichtenübertragung, analoge Nachrichten (Sprache, Bild, Bewegtbild) werden dazu schon in den Endgeräten digitalisiert bzw. wieder zurückgewandelt (z.B. PCM: Puls Code Modulation zur digitalen Sprachübertragung).

Der bisherigen Telephonsteckdose an der Wand wird ein ISDN-Basisanschluß entsprechen, mit dem 3 Kanäle gleichzeitig zur Verfügung stehen. Zwei sogenannte B-Kanäle a 64 Kbit/sec zum Nutzdatenaustausch und ein sogenannter D-Kanal mit 16 Kbit/sec vornehmlich zum Austausch von Signalisierdaten (z.B. Wählzeichen, Abrechungsinformation) zwischen Dienstnehmer und Betreiber aber auch zwischen zwei verbundenen Dienstnehmern. Die dazu benötigte Übertragungsleistung von 192 Kbit/sec (B+B+D+Synchronisationsaufwand) kann inzwischen von einer aus dem analogen Telephonnetz verfügbaren verdrillten 2-Draht-Leitung erbracht werden. Mit einem solchen Basisanschluß können dann gleichzeitig z.B. zwei Telephonate geführt werden, während über den D-Kanal zusätzlich noch Fernwirkdienste mit geringer Übertragungsleistungsanforderung angesprochen werden können (z.B. Heizung einschalten).

Reicht dies für die Erfordernisse eines Dienstnehmers nicht aus, kann ein Primäranschluß installiert werden, der über 30 B-Kanäle a 64 Kbit/sec und einen D-Kanal mit 64 Kbit/sec verfügt.

Das Anschlußschema zur Ankopplung von Endgeräten an ISDN muß, da man von 'dummen' und einfachen Endgeräten wie Telephonapparaten über Telefax- und BTX-Geräte bis hin zu einzelnen Digitalrechnern und lokalen Rechnernetzen universell ein breites Spektrum von Geräten bedienen will, flexibel sein und darf weder einfachen Geräten unnötige Steuergeräte-Kosten aufbürden noch anspruchsvolle Geräte im Funktionsumfang einschränken.

Bild 3.16 zeigt die vorgesehene Funktionsaufteilung über drei Schnittstellen R, S und T am Beispiel eines Endgeräts, das bisher über eine X.21-Schnittstelle an DATEX angeschlossen war. Die Schnittstelle R entspricht dann den X.21-Empfehlungen und sie wird von einem DEE-X.21-Adapter angeboten. Dieser, sowie überhaupt Endgeräte mit komplexerer Funktion, sprechen eine Netzwerkterminierung des Typs 2 über die S-Schnittstelle an. Einfache Endgeräte wie Telephonapparate und Telefax-Geräte können unmittelbar an die T-Schnittstelle angeschlossen werden.

3.3.3 Anschluß an lokale Netze

Der Anschluß an lokale Netze folgt denselben Prinzipien wie der Anschluß an öffentliche. Besonderheiten sind hier, daß eine separate und über eine genormte Schnittstelle versorgte DÜE nicht vorgeschrieben ist, und daß ferner wesentlich höhere Übertragungsleistungen vom Netz her zur Verfügung stehen. Der Netzanschluß ist deshalb üblicherweise als schnelles und blockorientiert arbeitendes Gerät zu werten. Im Zusammenhang mit der Nichtreglementierung werden aus Effizienz- und Kostengründen DÜE und DÜE-Anschalteinrichtung zu einer Netzanschalteinrichtung (Netzcontroller) verschmolzen, die oft auf einer gemeinsamen Einschubplatine untergebracht sind (z.B. Ethernet-Karten, Token-Ring-Karten für Arbeitsplatzrechner).

Bild 3.17 zeigt ein lokales Netz des Typs Tokenring, bei dem als Medium verdrillte Adernpaare verwendet werden. Über eine Kombination aus Leseverstärker, 1-Bit-Verzögerung, Schalter und Sendeverstärker je Station bilden sie die Topologie eines Rings (Übertragungsleistung 4 Mbit/sec).

Bild 3.18 zeigt entsprechendes bei einem lokalen Netz des Typs Ethernet. Hier wird als Medium ein Koaxialkabel verwendet (Übertragungsleistung 10 Mbit/sec).

Die Anschalteinrichtungen sind mit eigenen (ROM-programmierten) Prozessoren und einem internen Arbeitsspeicher versehen, und wickeln neben Signalformung und Medienzugang auch Funktionen der Ebenen 2 (Sicherung) und 3 (Vermittlung) ab, so daß über die E/A-Anschlußstelle des Rechners bereits Dienstprimitive eines Schicht-3-Dienstes ausgetauscht werden. Besondere Problematik ist der Transfer von Datenblöcken zwischen Zentraleinheit und Netzanschaltung. Zur Entkopplung des i.a. isochron getakteten Blocksendens und Empfangens von Zugriffen auf den Arbeitsspeicher des Rechners werden anschaltungsinterne Pufferspeicher verwendet und es sind im Rechner besondere Maßnahmen zum bevorzugten Einlesen angekommener Datenblöcke vorzusehen, um den Empfang

neuer Datenblöcke nicht an der mangelnden Verfügbarkeit freier Pufferspeicher scheitern zu lassen (Verlust aus Überlastgründen).

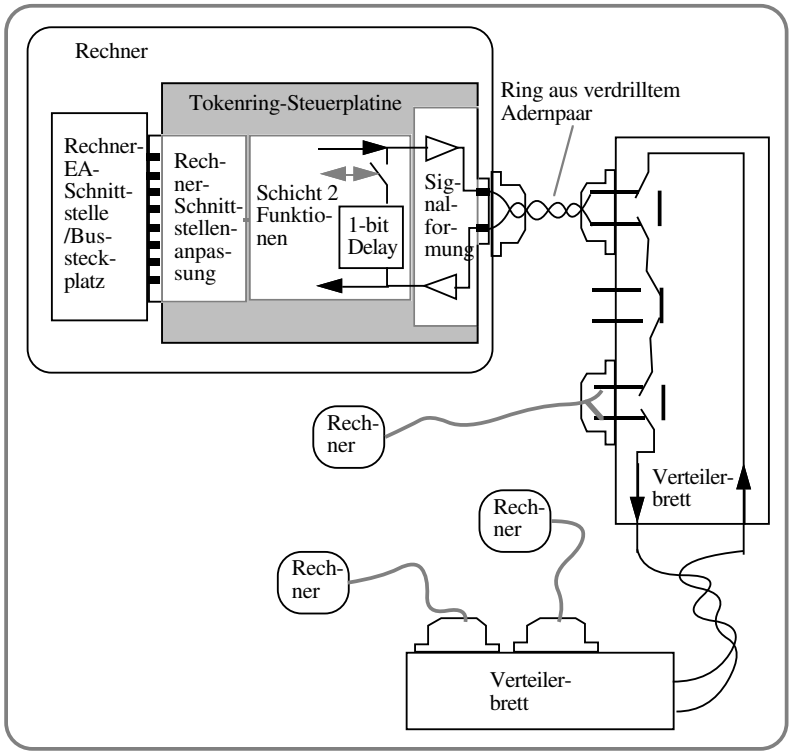

Bild 3.17: Rechner am lokalen Tokenring-Netz

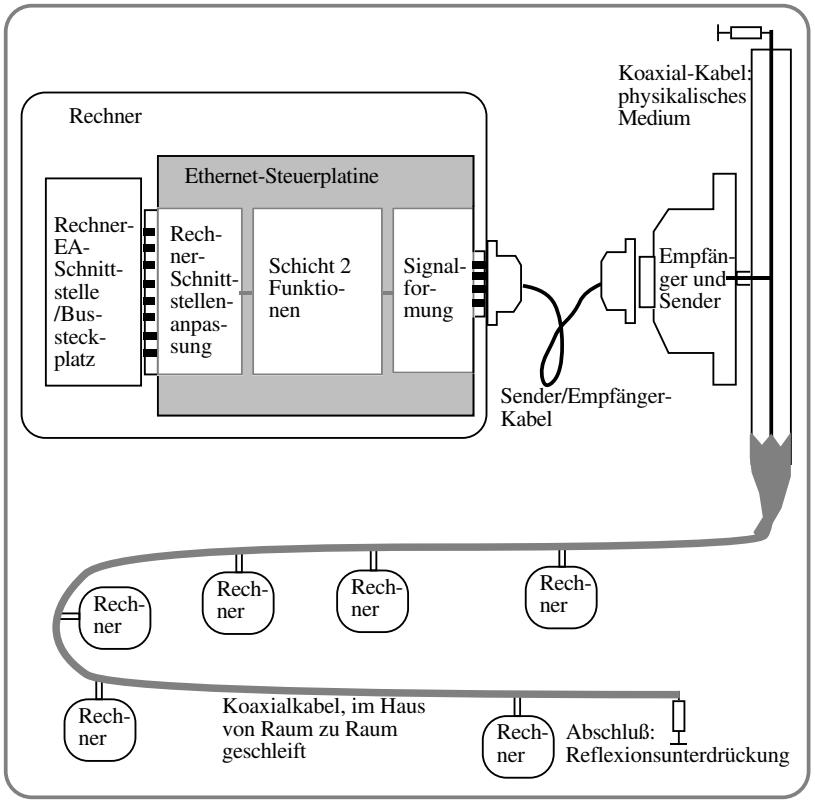

Bild 3.18: Rechner am lokalen Ethernet - Netz

4. Telekommunikationsdienste und Protokolle

Wir gehen von der eingeführten logischen Gliederung eines Telekommunikationssystems in Schichten und der Beschreibung von Schichten in Termen von Diensten und Protokollen aus. Zur verfeinerten Betrachtung sollen die zentralen Begriffe 'Dienst' und 'Protokoll' präzisiert werden. Wesentlich dafür werden Hilfsmittel zur abstrakten Beschreibung von Diensten und Protokollen sein. Über den Zusammenhang einer einzigen Schicht hinausgehende Konzepte, z.B. die Prinzipien zur Aufteilung der benötigten Gesamtfunktionalität eines Systems und die Zuweisung von Teilfunktionen zu einzelnen Schichten sollen erst Gegenstand des nachfolgenden Kapitels sein.

4.1 Telekommunikationsdienste

Die Gliederung eines Telekommunikationssystems in Schichten weist als Schnittstelle zwischen unmittelbar übereinanderliegenden Schichten die Schnittstelle eines Kommunikationsdienstes aus. Instanzen der oberhalb liegenden Schicht treten hier als Dienstnehmer des von der unterhalb liegenden Schicht erbrachten Dienstes auf.

Es ist zur Zeit noch oft üblich, den von einer Schicht erbrachten Dienst indirekt durch die Beschreibung der Modalitäten seiner Erbringung zu spezifizieren, d.h. man abstrahiert nicht von der internen Zusammensetzung der Schicht aus Instanzen und deren Verhalten (dem Kommunikationsprotokoll), sondern leitet die an der Dienstschnittstelle sichtbar werdende Funktionalität ausdrücklich daraus ab. So wurde dieser Weg auch im vorangegangenen Kapitel beschritten, um in nachrichtentechnische Grundlagen und Zusammenhänge einzuführen und eine Vorstellung von den Dienstleistungen zu vermitteln, die von physikalischen Medien, nachrichtentechnischen Kanälen und Vermittlungssystemen erbracht werden können.

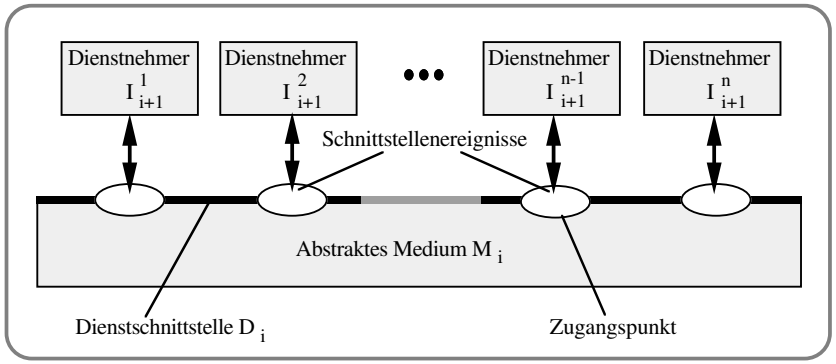

Bild 4.1: Szenario aus abstrakten Medium und Dienstnehmern

Im Gegensatz dazu sollen für die weitergehende Behandlung von Kommunikationssystemen Dienste als eigenständige Elemente der logischen Architektur aufgefaßt werden, die in sich geschlossen, ohne Berücksichtigung der Art und Weise ihrer Erbringung definiert und verstanden werden können. Das grundlegende Szenario für das Verständnis eines Dienstes besteht also, wie in Bild 4.1 gezeigt, aus Dienstnehmerinstanzen (aus der oberhalb liegenden Schicht), die als Auftraggeber und Beeinflußte von Dienstleistungen agieren, und zusätzlich nur einer einzigen, intern nicht näher aufgeschlüsselten, Komponente, dem hier sogenannten abstrakten Medium. Es besitzt eine einzige Schnittstelle, die der Dienstschnittstelle entspricht und in Dienstzugangspunkte gegliedert ist.

Zur Beschreibung eines Dienstes konzentrieren wir uns auf das Verhalten, das das abstrakte Medium an der Dienstschnittstelle den Dienstnehmern gegenüber besitzen kann. Wir können dazu so vorgehen, als ob das abstrakte Medium eine einzige elementare Instanz wäre, und verdecken dadurch seine Zusammensetzung aus örtlich verteilten Instanzen, deren Protokoll und die Nutzung von untergeordneten Diensten.

Die Vorteile dieser Vorgehensweise stammen unmittelbar aus dieser Abstraktion. Das Verständnis von speziellen Kommunikationssystemen wird wesentlich erleichert, da in die Behandlung einer Schicht i die Schichten i-1, i-2,..,0 nur noch in Form der Beschreibung des von der Schicht i-1 erbrachten Dienstes eingehen müssen. Entwurf, Implementierung und Verifikation eines Protokolls der Schicht i können sich modular auf die Zusammensetzung der Schicht i konzentrieren, weil die Umgebung, d.h. die oberhalb und unterhalb liegenden Schichten vollständig durch die Spezifikationen der Dienste an den Schnittstellen i-1 und i vertreten werden können.

4.1.1 Beschreibung der Dienstfunktionalität

Zur Betrachtung eines Telekommunikationsdienstes konzentrieren wir uns auf seine Dienstschnittstelle. Sie ist in eine Menge (örtlich verteilter) Zugangspunkte gegliedert, und an den einzelnen Zugangspunkten können im Laufe der Zeit Schnittstellenereignisse auftreten.

Die abstrakte Sicht eines Dienstes entspricht somit der Sicht auf seine Schnittstelle und kann folgendermaßen definiert werden:

- eine Menge SAP von Zugangspunktadressen, die die Zusammensetzung einer Dienstschnittstelle aus einer Menge von (i.a. örtlich zueinander entfernten) Zugangspunkten wiedergibt, und mit deren Elementen einzelne Zugangspunkte stellvertretend angesprochen werden.

- eine Menge EVT von Schnittstellenereignistypen; neben Zeitpunkt, Zugangspunktadresse (d.h. Ort) und den Werten der Datenparameter (d.h. Typverfeinerung und Nutzdaten) eines Ereignisses soll ausschließlich sein Typ als relevant erachtet werden.
- Ablauffestlegungen für das Verhalten eines entsprechenden abstrakten Mediums.

Die Vereinbarung endlicher Mengen SAP und EVT erfolgt in einfacher Weise durch Aufzählung entsprechender Elementbezeichner. Für die grobe Beschreibung von möglichen Abläufen an Dienstschnittstellen werden oft Weg/Zeit-Diagramme (Sequence Charts) verwendet. Genauer läßt sich das Verhalten in Form des Zustandsübergangsdiagramms eines indeterministischen Zustandsautomaten darstellen, wie sie üblicherweise auch für die Beschreibung von Protokollinstanzen eingesetzt wird.

4.1.1.1 Zugangspunktadressen

Die Wahl der Bezeichner für Zugangspunktadressen ist von untergeordneter Bedeutung, so daß wir in der Regel zunächst generell und ohne ausdrückliche Erwähnung von der Vereinbarung SAP = {0,1, .., n} ausgehen. Häufig zu finden ist eine hierarchische Adreßbildung, so daß eine Adresse aus einer Folge von Kennungen - etwa in der Form '<Nationalitätskennung>, <Gebietskennung>, <Ortskennung>, <Teilnehmerkennung>' - gebildet wird.

4.1.1.2 Schnittstellenereignistypen

Die Wahl der Bezeichner für Ereignistypen ist formal gesehen ohne Bedeutung, eine geeignete und auch bei verschiedenen Diensten nach gemeinsamem Schema getroffene Wahl trägt jedoch wesentlich zum erleichterten Verständnis eines Dienstes bei. Hierzu ist die Zusammensetzung eines Typbezeichners aus drei Bestandteilen üblich:

- Dienstname; Name (z.B. Sicherung, Netzwerk, Transport, ..; bzw. im englischen Link, Network, Transport, Session, Presentation, ..) des beschriebenen Dienstes, oft auch in abgekürzter Form (z.B. L, N, T, S, P, ..) oder ganz entfallend, wenn nur von einem Dienst die Rede ist,
- Dienstelementname; Name eines Dienstleistungstyps oder dessen Abkürzung (z.B. Connection-Establishment, DataTransfer, ConnectionRelease, ConnectionReset, ConnectionAbort, ..; bzw. Con, Dat, Dis, Res, Abo, ...),
- Grundtypname; Bezeichnung (dies sind Request, Indication, Response, Confirmation) bzw. Kürzel (Req, Ind, Res, Con) für einen der vier Grundtypen, aus dem Richtung (Medium --> Dienstnehmer bei Ind und Con bzw. Dienstnehmer --> Medium bei Req und Res) eines Ereignisses und seine Einordnung in die Abwicklung einer Dienstleistung (Beauftragung, Benachrichtigung des Partners, Bestätigungsübergabe, Benachrichtigung über den Abschluß der Dienstleistung) hervorgehen.

Ein Bezeichner T_ConReq steht somit z.B. für den Schnittstellenereignistyp Verbindungsaufbau-Anforderung an der Schnittstelle eines Transportdienstes.

Die verschiedenen, von einem speziellen Dienst unterstützten Schnittstellenereignistypen werden auch Primitive des Dienstes genannt. Sie werden in aufzählender Form eingeführt, wobei man aus Übersichtlichkeitsgründen die Aufzählung in Abschnitte gliedert, die den einzelnen Dienstleistungstypen gewidmet sind. Dies erlaubt es auch, auf die Bedeutung von Ereignissen entsprechender Typen in der Abwicklung einer Dienstleistung einzugehen, und oft finden sich Weg/Zeit-Diagramme zur Verdeutlichung des Ablaufs direkt am Ende eines solchen Abschnitts.

Für jeden Ereignistyp müssen ferner die vorgesehenen Datenparameter (Kontroll- und Nutzdatenparametrisierung) beschrieben werden. Jedem Ereignistyp ist i.a. eine feste Menge von Parametern mit vorgegebenen Wertebereichen zugeordnet.

Es ist zu erwähnen, daß die Wertemenge eines Parameters abstrakt bleiben kann. Die konkrete 'bitgenaue' Darstellung von Werten, die z.B. bei einer bestimmten Implementierung eines Dienstes in einer Menge von Rechnern gewählt wird, muß nicht festgelegt werden. Sie ist nicht Bestandteil der logischen Architektur sondern Implementierungssache, insbesondere kann man sich auch vorstellen, daß innerhalb derselben Implementierung desselben Dienstes auf einem Rechner A eine andere Darstellung zur Parameterübergabe als auf einem zweiten Rechner B gewählt wird, wie auch insgesamt zur Implementierung der Dienstschnittstelle je Rechner und sogar je Zugangspunkt unterschiedliche Wege beschritten werden können.

4.1.1.3 Weg/Zeit-Diagramme

Mit einem Weg/Zeit-Diagramm können einfache örtlich verteilte Abläufe innerhalb eines 2-Achsen-Koordinatensystems in eingängiger Weise graphisch dargestellt werden. Bild 4.2 zeigt zwei Beispiele.

Die vertikale Achse ist die Zeitachse (spätere Zeitpunkte liegen unterhalb vorangehender).

Die horizontale Achse ist die Ortsachse und enthält in der Regel nur zwei Stellen, je eine für die beiden betroffenen Zugangspunkte einer beispielhaft im Ablauf beschriebenen Dienstleistungsabwicklung. Die Strecke zwischen den beiden Stellen entspricht so der durch das (abstrakte) Medium überbrückten Entfernung.

Der dargestellte Ablauf besteht aus einer Menge von Ereignissen, die nach Ort und relativem Zeitpunkt als Punkte im Koordinatensystem auftreten. Jeder Punkt wird mit dem Namen des entsprechenden Ereignistyps versehen.

Um die Richtung eines Ereignisses hervorzuheben, wird anstelle eines Punktes oft ein Pfeil notiert.

Um die gemäß Dienst vorhandene Ursache-Wirkung-Beziehung zu verdeutlichen, werden innerhalb des Mediums Punkte durch Pfeile verbunden. Bei einem physikalischen Medium können sie direkt die Fortpflanzung eines Signals in Weg und Zeit versinnbildlichen und sind in entsprechend übertragenem Sinn auch für abstrakte Medien zu lesen.

Der in Bild 4.2 oben gezeigte Ablauf soll der regulären Abwicklung einer bestätigten Dienstleistung 'Verbindungsaufbau' entsprechen. Betroffen sind zwei Orte, links ist der Zugangspunkt des Initiators und rechts der des Beantworters auf der horizontalen Achse angeordnet. Der Ablauf ist einfach. Zunächst muß der Initiator ein Ereignis des Typs ConReq erzeugen, das vom Medium akzeptiert wird und nach einiger Laufzeit medieninterner Nachrichten am Ort des Beantworters zur Erzeugung eines Ereignisses des Typs ConInd durch das Medium führt. Nun ist der Beantworter im Zugzwang und sollte in Reaktion darauf ein Ereignis des Typs ConRes erzeugen, das wiederum nach einiger Verzögerung am Ort des Initiators zur Erzeugung eines Ereignisses des Typs ConCon durch das Medium führt.

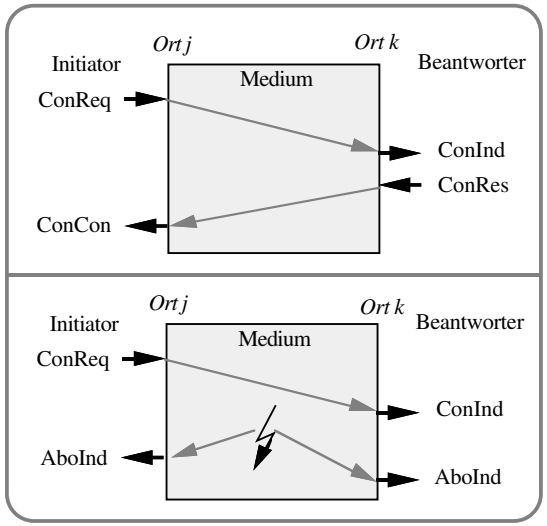

Bild 4.2: Beispiele zu Weg/Zeit-Diagrammen

Die gleichzeitige Darstellung von Alternativen im Ablauf führt bei Weg/Zeit-Diagrammen schnell zu unübersichtlichen Diagrammen. Deshalb wird jeder Alternative i.a. ein eigenes Diagramm gewidmet. Bild 4.2 zeigt im unteren Teil ein zweites Diagramm, das eine mögliche Abweichung des Ablaufs der Dienstleistung Verbindungsaufbau beschreibt. Wie oben wird mit der Anforderung ConReq begonnen und sie führt zur Anzeige ConInd. Noch bevor der Beantworter mit ConRes reagiert hat, wurde die weitere Bearbeitung der Dienstleistung jedoch von Seiten des Mediums abgebrochen und die beiden Dienstnehmer werden darüber mittels zweier Abbruchanzeigen AboInd benachrichtigt.

Die beiden Abbruchanzeigen stehen nicht in direkter Ursache-Wirkung-Beziehung zu einem auslösenden Schnittstellenereignis. Sie treten von außen gesehen spontan auf alleinige Initiative des Mediums unter von Dienstnehmern nicht ausdrücklich beeinflußbaren inneren Umständen auf. Dies wird durch den medieninternen Blitz im Weg/Zeit-Diagramm verdeutlicht.

4.1.1.4 Zustandsübergangsdiagramme

Das Verhalten eines Mediums kann im Vergleich zu Weg/Zeit-Diagrammen vollständiger durch Angabe eines Zustandsautomaten beschrieben werden. Als graphisches Hilfsmittel bietet sich dabei - zumindest bei einfacheren Verhalten - die Vereinbarung in Form eines Zustandsübergangsdiagramms an.

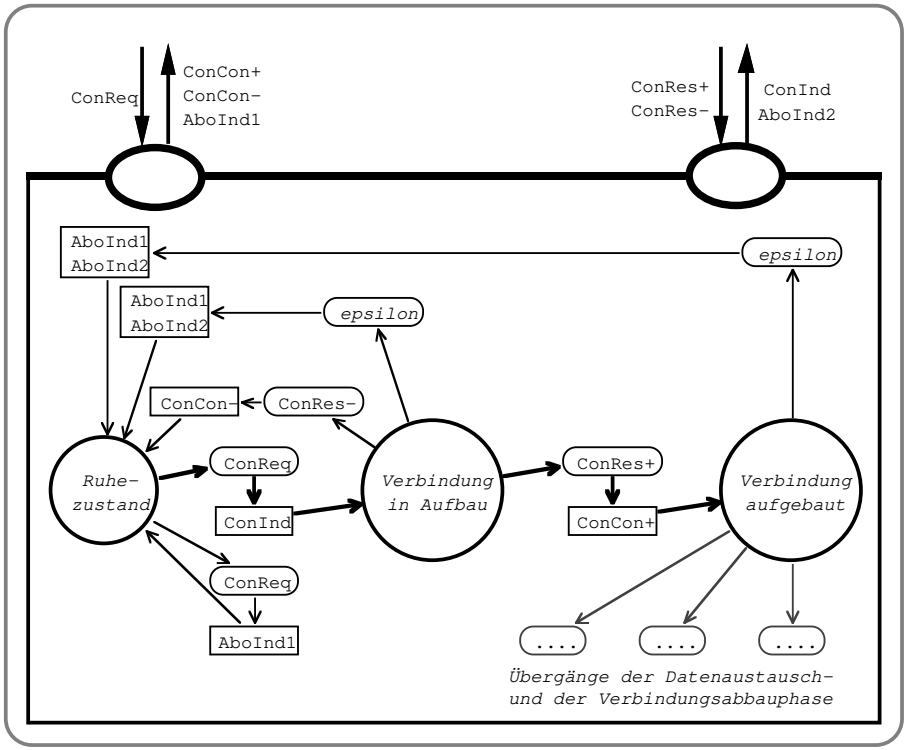

Bild 4.3: Beispiel zu Zustandsübergangsdiagramm

In der im folgenden gewählten Art der Darstellung werden Zustände als Kreise notiert. Ein Zustandsübergang wird durch eine von einem aktuellen Zustand in einen Nachfolgezustand führende Pfeilfolge dargestellt, die zuerst über ein Oval und dann über ein Rechteck führt. Das Oval enthält die Angabe eines Dienststimulustyps und besagt, daß der entsprechende Übergang stattfinden kann, wenn im aktuellen Zustand von einem Dienstnehmer ein Stimulus des Typs erzeugt wird. Das Rechteck enthält die Angabe eines oder mehrerer Dienstreaktionstypen. Die Bedeutung ist, daß das Medium mit der Ausführung des Zustandsübergangs entsprechende Reaktionen erzeugt.

Von demselben Zustand können mehrere verschiedene mit demselben Stimulustyp beschriftete Übergänge ausgehen. Hierdurch wird dem Medium die indeterministische von außen nicht beeinflußbare Auswahl aus verschiedenen als nächstes möglichen Verhaltensschritten gestattet.

Weiterhin können sogenannte Spontanübergänge notiert werden. In der graphischen Darstellung sind sie an der Beschriftung des Stimulus-Ovals mit dem Leereingabe-Symbol 'epsilon' zu erkennen. Ist zu einem aktuellen Zustand ein Spontanübergang definiert, dann kann das Medium diesen Zustand über den Spontanübergang verlassen, ohne daß von außen ein auslösender Stimulus vorhanden sein muß.

Bild 4.3 zeigt beispielhaft das Verhalten eines Mediums in der Verbindungsaufbauphase, wie es an zwei Zugangspunkten gegenüber einer Initiator- und einer Beantworter-Instanz sichtbar wird.

Am linken Zugangspunkt, dem Zugangspunkt des Initiators, können Schnittstellenereignisse folgender Typen ausgetauscht werden:

- ConReq; Stimulus, mit dem der Initiator die Dienstleistung Verbindungsaufbau anfordert.
- ConCon+; Reaktion, mit der dem Initiator der erfolgreiche Aufbau der Verbindung signalisiert wird.
- ConCon-; Reaktion, die dem Initiator mitteilt, daß sein Verbindungsaufbauwunsch den gewählten Beantworter erreicht hat, das Herstellen einer Kommunikationsverbindung von diesem aber abgelehnt wurde.
- AboInd1; Reaktion, mit der das Medium dem Initiator meldet, daß die Bearbeitung der angeforderten Dienstleistung aus medieninternen Gründen abgebrochen wurde.

Am rechten Zugangspunkt, dem des Beantworters, treten folgende Schnittstellenereignistypen auf:

- ConRes+; Stimulus, mit dem der Beantworter die Zurkenntnisnahme eines Aufbauwunschs bestätigt und gleichzeitig seine Bereitschaft zur Annahme der im Aufbau befindlichen Verbindung signalisiert.
- ConRes-; Stimulus, mit dem der Beantworter die Annahme der im Aufbau befindlichen Verbindung ablehnt.
- AboInd2; Reaktion, mit der das Medium dem Beantworter meldet, daß die Bearbeitung der angeforderten Dienstleistung aus medieninternen Gründen abgebrochen wurde.

Das Verhalten des Mediums bezogen auf ein Initiator-Beantworter-Paar und auf eine neu aufzubauende Verbindung entspricht dem gezeigten Zustandsübergangsdiagramm. Es ist über den drei Verbindungszuständen 'Ruhezustand (neu aufzubauende Verbindung existiert noch nicht)', 'Verbindung in Aufbau (der Aufbau der Verbindung ist angefordert, aber noch nicht abgeschlossen)' und 'Verbindung aufgebaut' definiert.

Anfänglich befindet sich das Medium im Ruhezustand. Hier ist es dem Initiator erlaubt, einen Stimulus des Typs ConReq abzusetzen, um den Aufbau der neuen Verbindung einzuleiten. Andere Schnittstellenereignisse sind in diesem Zustand nicht möglich, entsprechende Übergänge sind nicht vorgesehen.

Die fett gezeichneten Übergänge beschreiben das Verhalten im normalen Betriebsablauf. Im Ruhezustand erzeugt der Initiator ein ConReq-Ereignis. Es wird vom Medium angenommen und führt zum Folgezustand 'Verbindung in Aufbau' und dazu, daß dem Beantworter der Aufbauwunsch per ConInd mitgeteilt wird. Der Beantworter zeigt seine Bereitschaft zur Annahme per ConRes+. Hierdurch wechselt das Medium in den Zustand 'Verbindung aufgebaut' und signalisiert dies dem Initiator per ConCon+. Nun können, im Bild nicht näher dargestellt, Dienstleistungen durch Stimuli angefordert werden, die das Vorhandensein einer Verbindung voraussetzen.

Abweichungen vom normalen Ablauf können sich durch folgende Besonderheiten ergeben:

- Das Medium kann im Ruhezustand zwar einen Aufbauwunsch ConReq zunächst annehmen und bearbeiten, die Bearbeitung jedoch - ohne daß der Aufbauwunsch dem Beantworter ersichtlich geworden ist - aus inneren Gründen vorzeitig abbrechen und dies dem Initiator per AboInd1 mitteilen. Der Initiator hat keine Möglichkeiten zur Prophylaxe gegen diese Ablaufbesonderheit. Er muß nach Abgabe eines ConReq generell mit der Möglichkeit der Ablehnung rechnen. Das wird durch die indeterministische Auswahlmöglichkeit ausgedrückt, die dem Medium im Ruhezustand mit dem Eintreffen eines ConReq-Stimulus offensteht. Vom Ruhezustand gehen zwei mit ConReq-Stimulusangaben beschriftete Übergänge aus.
- Auch im Zustand 'Verbindung im Aufbau', also nachdem dem Beantworter der Verbindungsaufbauwunsch mitgeteilt wurde, besteht die Möglichkeit, daß das Medium von sich aus die weitere Unterstützung der aufzubauenden Verbindung ablehnt. Der entsprechende Übergang ist ein Spontanübergang. Er sieht vor, daß das Medium nun beiden Dienstnehmern den Abbruch der Unterstützung per AboInd1 und AboInd2 mitteilt und in den Ruhezustand zurückkehrt.
- Dieses Verhalten des Mediums ist ferner auch möglich, wenn der Verbindungsaufbau bereits erfolgreich abgeschlossen wurde und der Zustand 'Verbindung aufgebaut' vorliegt. Dies wird ebenfalls durch einen entsprechenden Spontanübergang beschrieben.

4.1.2 Verbindungsorientierte Dienste

Generell ergibt sich bei verbindungsorientierten Diensten, jeweils bezogen auf eine Verbindung, im groben ein Verhalten des Mediums, wie es in Bild 4.4 mit einem vergröberten Zustandsübergangsdiagramm umrissen ist. Im Ruhezustand existiert die Verbindung nicht. Mit einer Anforderung der Dienstleistung 'Verbindungsaufbau' beginnt die Aufbauphase. Sie umfaßt die Ausführung dieser Dienstleistung und stellt die Verbindung her. Der Zustand 'Verbindung aufgebaut' wird erreicht, und die Datenaustauschphase kann beginnen. In ihr werden Dienstleistungen zum Datenaustausch zwischen den verbundenen Dienstnehmern angefordert und ausgeführt. Sie endet mit dem Beginn der Abbauphase oder der Abbruchphase. Der geregelte Abbau wird dabei durch Anforderung der Dienstleistung 'Verbindungsabbau' eingeleitet und mit der Ausführung dieser Dienstleistung in der Abbauphase abgewickelt. Er findet i.a. nur auf Initiative eines Dienstnehmers statt, seine Abwicklung sieht eine Abstimmung mit den Dienstnehmer-Partnern vor, und er garantiert, daß alle vorher angeforderten Dienstleistungen vor Verbindungsabbau zuendegeführt wurden.Verbindungsabbruch dagegen bezeichnet den außerplanmäßigen und i.a. unvorhergesehenen Abbau einer Verbindung aufgrund momentaner besonderer Gegebenheiten im Medium oder bei einem der betroffenen Dienstnehmer. Hier kann nicht damit gerechnet werden, daß gegebenfalls noch anstehende andere Dienstleistungen beendet worden sind.

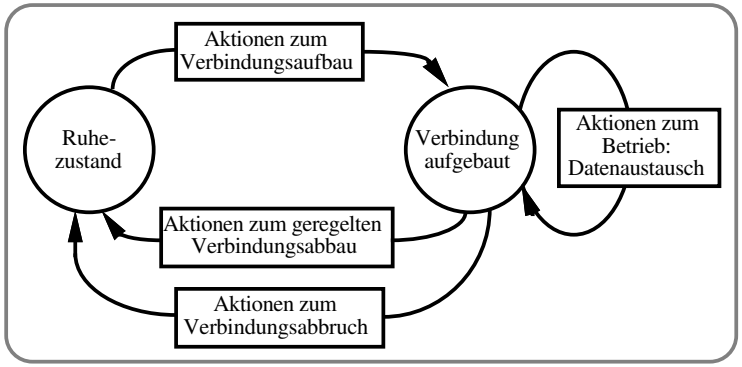

Bild 4.4: Grobes Zustandsübergangsdiagramm - Phasen

Manche Medien sind in der Lage, gleichzeitig mehrere Verbindungen zu unterstützen. Das Gesamtverhalten des Mediums gliedert sich dabei in gleichartige Teilverhalten, die eindeutig den einzelnen Verbindungen zugeordnet werden können und voneinander im wesentlichen unabhängig sind.

4.1.2.1 Adressen und Verbindungsendpunkte

Im allgemeinen Fall unterstützt ein Medium eine größere Menge von Zugangspunkten und läßt je Zugangspunkt mehrere Verbindungen gleichzeitig zu. Bild 4.5 zeigt ein entsprechendes Szenario. Darin sollen die dunklen Röhren zwischen den Zugangspunkten momentan aufgebaute Verbindungen und die heller gezeichnete Röhre eine im Aufbau befindliche Verbindung versinnbildlichen.

Die Vereinbarung der Menge unterstützter Zugangspunktadressen definiert die Zugangspunkte des Mediums. Jeder Zugangspunkt ist genau einem potentiellen Dienstnehmer zugeordnet. So identifizieren Zugangspunktadressen Dienstnehmer, und gleichzeitig im selben Zugangspunkt mündende Verbindungen sind Verbindungen, die derselbe Dienstnehmer gleichzeitig betreibt. Deshalb müssen nur entsprechende Zugangspunktadressen als Parameter von Schnittstellenereignissen vorgesehen werden, wenn - im Zuge der vom Medium unterstützten horizontalen Kommunikation - Partner identifiziert werden sollen. Dabei kann nach erfolgtem Verbindungsaufbau bei Zweipunkt-Verbindungen bei allen weiteren Schnittstellenereignissen auf die Angabe von Partneradressen verzichtet werden, weil der gewollte Partner implizit durch das andere Ende der Verbindung bestimmt wird.

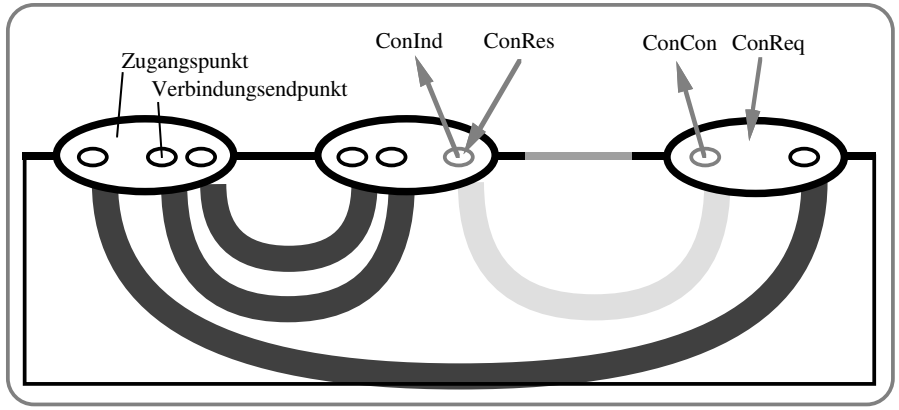

Bild 4.5: Zugangspunkte und Verbindungsendpunkte

Für die an einem Zugangspunkt stattfindende vertikale Kommunikation zwischen Medium und einem Dienstnehmer reichen Zugangspunktadressen nicht aus, wenn gleichzeitig mehrere Verbindungen bestehen. Das Medium muß einen an diesem Zugangspunkt auftretenden Stimulus der vom Dienstnehmer vorgesehenen Verbindung zuordnen können, und umgekehrt muß dem Dienstnehmer die Verbindung identifizierbar sein, von der eine Reaktion eingetroffen ist. Diese Aufgabe wird durch Verbindungsendpunkt-Identifikationen gelöst. Aus der Sicht der logischen Architektur ist dabei nur die Tatsache von Interesse, daß je Zugangspunkt für alle dort endenden Verbindungen eindeutige Endpunkt-Identifikationen vereinbart sind und somit für Medium und Dienstnehmer die Zuordnung von Schnittstellenereignissen zu Verbindungen gewährleistet ist.

Die tatsächliche Auslegung der Endpunkt-Identifikationen bezüglich Wertebereich, Darstellung und den genauen Modalitäten ihrer Vereinbarung zwischen Dienstnehmer und Medium ist ein reiner Implementierungsaspekt und kann sogar durchaus bei derselben Implementierung eines Mediums an verschiedenen Zugangspunkten verschiedenartig gelöst werden. Grob vorgezeichnet ist folgendes:

- Ein Verbindungsaufbauwunsch-Stimulus wird von einem Dienstnehmer an den ihm zugeordneten Zugangspunkt gerichtet. Mit dem Austausch dieses Stimulus wird zwischen Dienstnehmer und Medium eine Verbindungsendpunkt-Identifikation für das initiator-seitige Ende der neu aufzubauenden Verbindung vereinbart. Alle nachfolgenden Schnittstellenereignisse werden vom Erzeuger mit dieser sowohl Dienstnehmer als auch Medium dann bekannten Endpunkt-Identifikation versehen.

- Mit dem Austausch einer Verbindungsaufbauwunsch-Anzeige wird zwischen Beantworter und Medium am dortigen Zugangspunkt ebenfalls eine Identifikation für den beantworterseitigen Endpunkt der neu aufzubauenden Verbindung vereinbart.

4.1.2.2 Verbindungseigenschaften

Eine von einem abstrakten Medium unterstützte Verbindung soll für die angeschlossenen Dienstnehmer als Kanal wirken, über den sie Nachrichtenaustausch betreiben können. Im Vergleich zum nachrichtentechnischen Kanal handelt es sich dabei schon um eine höhere, anwendungsnähere Form. Verbindungseigenschaften abstrahieren aber mehr von der technischen Realisierung und berücksichtigen die anwendungsnäheren Fähigkeiten.

Es sind bereits Dienste zur Unterstützung von Mehrpartnerverbindungen in Diskussion. Von zentraler Bedeutung ist jedoch bei den gegenwärtig eingesetzten Diensten die Zwei-Punkt-Verbindung. Auf sie beziehen sich auch die weiteren Ausführungen.

Für die Dienstnehmer interessierende Verbindungseigenschaften können folgendermaßen gegliedert werden:

- Qualitätseigenschaften,
- Richtungsbetrieb,
- Rechte,
- Klasse,
- Rückstau.

4.1.2.2.1 Dienstleistungsqualitäten

Wichtig sind folgende Parameter:

- Verbindungsaufbau-Zeitdauer; maximale Zeitdauer zwischen Anforderung und Bestätigung des Verbindungsaufbaus.
- Verbindungsaufbau-Fehlerwahrscheinlichkeit; Wahrscheinlichkeit dafür, daß ein Verbindungsaufbauwunsch trotz Wohlverhalten der beteiligten Dienstnehmer allein aufgrund medieninterner Gegebenheiten nicht rechtzeitig erfolgreich ausgeführt wird.
- Durchsatz / Übertragungsleistung; bezogen auf Nutzdatenmenge in der Datenaustauschphase, z.B. als Mittelwert.
- Transferfehlerwahrscheinlichkeit für Durchsatzverletzung; Wahrscheinlichkeit dafür, daß geforderte/garantierte Übertragungsleistung nicht angeboten werden kann.
- Verzögerung / Nachrichtenlaufzeit; z.B. Mittelwerte für Datenaustausch-Dienstleistungen bei Übertragung von Nachrichten mittlerer Länge.
- Transferfehlerwahrscheinlichkeit für Nachrichtenlaufzeitverletzung; Wahrscheinlichkeit dafür, daß geforderte/garantierte Laufzeit nicht eingehalten werden kann.
- Störfehlerrate; statt Bitfehlerrate für Nutzerdaten auch oft auf Nutzdatenblöcke bezogen.
- Transferfehlerwahrscheinlichkeit für Störfehlerraten-Überschreitung; Wahrscheinlichkeit dafür, daß geforderte/garantierte Störfehlerrate überschritten wird.
- Verbindungsabbruchwahrscheinlichkeit; Wahrscheinlichkeit dafür, daß eine Verbindung aufgrund medieninterner Gegebenheiten abgebrochen wird.
- Verbindungsabbau-Zeitdauer; maximal zu akzeptierende Zeitdauer zwischen Anforderung und Bestätigung des Verbindungsabbaus.
- Verbindungsabbau-Fehlerwahrscheinlichkeit; Wahrscheinlichkeit dafür, daß ein Verbindungsabbauwunsch trotz Wohlverhalten der beteiligten Dienstnehmer nicht rechtzeitig erfolgreich ausgeführt wird

- Schutzklasse; z.B.: kein Schutz; Sicherung gegen Abhören; Erkennen von Modifikationen, Verlusten, eingefügten Nachrichten; Sicherung gegen Modifikationen; Sicherstellung der Dienstnehmeridentität;
- Priorität; Wichtigkeitsstufe der Verbindung im Vergleich zu allen anderen vom Medium unterstützten Verbindungen; sie beeinflußt die Reihenfolge, mit der das Medium bei medieninternen Engpässen oder Besonderheiten Verbindungen in der unterstützten Leistung zurückstuft oder gar abbricht.
- Kosten; unterschiedlich aufgeschlüsselte Kostenparameter.

4.1.2.2.2 Richtungsbetrieb

Zum Richtungsbetrieb wurden in Kap. 1.3.7 'Richtungsbetrieb' schon die drei grundlegenden Arten 'Simplex', '(Voll-)Duplex' und 'Halbduplex' vorgestellt. Varianten ergeben sich in einer genaueren Aufschlüsselung der Datenaustausch-Dienstleistungstypen und der Modalitäten, unter denen bestimmte Dienstnehmer Leistungen der einzelnen Typen anfordern können. Es handelt sich um einen Spezialfall der Rechte-Vergabe.

4.1.2.2.3 Rechtevergabe

Zu einem Zeitpunkt des Betriebs einer aufgebauten Verbindung ist das Medium bereit, Anforderungen nach Dienstleistungen definierter Dienstleistungstypen auszuführen. Im einfachen Spezialfall darf, sobald eine Verbindung aufgebaut ist, jeder Dienstnehmer zu jedem Zeitpunkt alle für den Zustand 'Verbindung aufgebaut' vorgesehenen Dienstleistungen anfordern. Davon abweichend können aber auch die Rechte zur Anforderung bestimmter Dienstleistungstypen den einzelnen Dienstnehmern individueller vergeben werden:

- statische Rechtevergabe mit dem Verbindungsaufbau; mit der Erzeugung der Verbindung ist festgelegt, welcher Dienstnehmer welche Dienstleistungen anfordern darf.
- dynamische Rechtevergabe während des Verbindungsbetriebs; die mit der Erzeugung der Verbindung anfänglich bestehende Rechtevergabe wird in Abhängigkeit zum Betriebsablauf modifiziert.

Die dynamische Rechtevergabe kann indirekt - abgeleitet von der Historie der bisher angeforderten und ausgeführten Dienstleistungen - vorgenommen werden. Dies erfolgt, wenn bestimmte Dienstleistungen nur sinnvoll sind, wenn bestimmte andere vorher die logischen Voraussetzungen geschaffen haben. Ferner kann - vor allem bei Exklusivrechten - eine ausdrückliche Rechtevergabe durch Bereitstellung dafür vorgesehener Dienstleistungstypen vorgesehen werden. So kann z.B. ein geregelter Halbduplex-Betrieb dadurch erreicht werden, daß das Nutzerdaten-Senderecht initial bei einem bestimmten Dienstnehmer liegt. Nur er kann Nutzerdatentransfers anfordern. Will ein anderer Partner senden, ruft er die Dienstleistung 'Senderecht-Anforderung' auf. Hierdurch wird dem Partner der Bedarf des anderen mitgeteilt, und er kann das Senderecht mit der Dienstleistung 'Senderecht-Weitergabe' abgeben.

4.1.2.2.4 Verbindungsklassen

Für die von einem bestimmten Medium unterstützten Verbindungen ist i.a. ein fester Satz von Dienstleistungstypen definiert und für die Dienstnehmer anforderbar. Zur möglichst generellen Verwendbarkeit eines abstrakten Mediums wird man dabei oft ein großes Spektrum an Dienstleistungstypen vorsehen, das sehr oft von bestimmten Dienstnehmern gar nicht in vollem Umfang benötigt wird. Ob in einer Verbindung nur eine Teilmenge der definierten Dienstleistungstypen verwendet werden soll und welche, ist für die beteiligten Dienstnehmer (schließlich müssen sie auf entsprechende Reaktionen des Dienstes vorbereitet sein) und für das Medium selbst (es muß Betriebsmittel zur Ausführung der Dienstleistungen bereitstellen oder einplanen) von Interesse. Den Bemühungen zu einer allgemeinen Verwendbarkeit entgegengesetzt ergibt sich also ein Bedarf nach spezialisierten, im Funktionsangebot eingeschränkten Verbindungstypen. Hierzu wird i.a. ein fester Satz an

Dienstleistungstypen-Teilmengen, sogenannte Verbindungsklassen, definiert, und mit dem Verbindungsaufbau wird die benötigte Klasse festgelegt.

4.1.2.2.5 Rückstau

Datenaustausch findet am häufigsten in der Form unbestätigter Dienstleistungen statt. Ein sendender Dienstnehmer erzeugt eine Datentransfer-Anforderung, die mit den vom ihm bestimmten Nutzdaten parametrisiert ist, das Medium überträgt die Nutzdaten und liefert sie am anderen Ende der Verbindung mit einer Datentransfer-Anzeige aus. Hierdurch sind Senden und Empfangen von Nutzdaten nur schwach synchronisiert. Potentiell kann ein Dienstnehmer durch allzu schnelles Senden Medium und Partner überfüttern. Ist er schneller als das Medium übertragen kann, müssen seine Nutzdaten-Pakete im Medium senderseitig gepuffert werden. Sind er und Medium schneller als der Empfänger besteht im Medium Speicherungsbedarf auf der Empfängerseite. Der Pufferbedarf wirkt sich medienintern auf die Kosten zur Unterstützung der Verbindung aus. Auch den Dienstnehmern ersichtlich wird dies, wenn das Medium aufgrund von Betriebsmittelengpässen Dienstleistungen nicht mehr erfolgreich abwickeln kann oder sogar Verbindungen abbrechen muß.

Zur Vermeidung solcher Situationen kann natürlich auf der Ebene der Dienstnehmer eine ausdrückliche Synchronisation über gegengerichtete Datenaustausch-Dienstleistungen zur Empfangsbestätigung vorgesehen werden. Sie liegt aber allein in der Verantwortung der Dienstnehmer.

Um bereits mit medieninternen Mitteln solche Situationen zu verhindern, werden deshalb Rückstau-Maßnahmen als Teilfunktion der medieninternen Flußkontrolle vorgesehen. So kann eine maximale Pufferungskapazität je Richtung für eine Verbindung vereinbart sein. Ist die Verbindung aufgebaut, kann ein Sender Datenaustausch-Anforderungen abgeben, bis diese Kapazität erschöpft ist. Ist sie erschöpft, wird das Medium keine weiteren Datenaustausch-Anforderungen mehr akzeptieren, bis durch Auslieferung von Nutzdaten beim Empfänger wieder freie Pufferkapazität verfügbar ist.

4.1.2.3 Verbindungsaufbau

Der Aufbau einer Verbindung wird - wie bereits mehrfach erwähnt - durch eine Dienstleistung des Typs 'Verbindungsaufbau (Connect bzw. Con)' vorgenommen, und die Abwicklung dieser Dienstleistung bildet die Verbindungsaufbauphase. Es handelt sich in der Regel um eine bestätigte Dienstleistung, sodaß sich die vier Schnittstellenereignistypen ConnectRequest, ConnectConfirm (initiatorseitig) und Connect Indication, ConnectResponse (beantworterseitig) ergeben (Abkürzungen: ConReq, ConCon, ConInd, ConRes). In Kap. 4.1.3 Bild 4.2 und Kap. 4.1.4 Bild 4.3 hat die Abwicklung eines Verbindungsaufbaus bereits als Beispiel gedient, so daß hier auf die Erläuterung der generellen Abfolge verzichtet werden soll. Kap. 4.1.4 Bild 4.3 führte neben den Ablaufbesonderheiten und den damit verbundenen Ereignistypen ferner bereits die oft zu findende Möglichkeit ein, daß der Beantworter die Gelegenheit zur geregelten Ablehnung eines Aufbauwunschs erhält.

Im folgenden sollen deshalb nur noch die einzelnen Dienstprimitive in ihrer Parametrisierung näher ausgeführt werden.

4.1.2.3.1 Parametertypen

Folgende Parametertypen sind beim Verbindungsaufbau relevant:

- Initiatoradresse,
- Beantworteradresse,
- Verbindungseigenschaften,
- Anzeige Annahme / Ablehnung,
- Kurze Nutzdaten,

- Ablehnungsgrund.

Die Initiator- und Beantworter-Adreßangaben werden jeweils als Zugangspunktadressen ausgeführt.

Die Parameter zur Anforderung von bzw. Benachrichtigung über bestimmte Verbindungseigenschaften entsprechen den Erläuterungen von Kap. 4.2.2 'Verbindungseigenschaften'. Sie werden oft gruppiert in Klassenangabe und Optionen zur Angabe des Funktionsumfangs sowie in 'Quality of Service'-Parameter zur Angabe der Leistungs-, Störungs-, Schutz- und Kosteneigenschaften.

In der hier beschriebenen Form des Verbindungsaufbaus kann die Bestätigung auch negativ sein und dem Initiator die Ablehnung durch den Beantworter signalisieren.

Der Parameter 'Kurze Nutzdaten (Short User Data)' kann ein beliebiges aber i.a. stark längenbeschränktes Bitmuster zur freien Verwendung durch die Dienstnehmer tragen. Hierdurch ist es möglich, bereits mit dem Verbindungsaufbau Nutzinformation zu übermitteln.

In Richtung 'Initiator->Beantworter' werden die kurzen Nutzdaten i.a. zu Darstellung von Informationen verwendet, die über den vorgesehenen Zweck und die Einbettung der Nutzung der aufzubauenden Verbindung Aufschluß geben (z.B. Kennung des auf nächsthöherer Ebene ablaufenden Protokolls) sowie zum Nachweis der Identität des Initiators dienen (z.B. auf Dienstnehmer-Ebene vereinbarte Nutzerkennungen und Paßwörter). Der Beantworter kann diese Informationen unmittelbar nach Erhalt der Aufbauwunsch-Anzeige auswerten und die Auswertung in seine Entscheidung über Annahme oder Ablehnung des Aufbauwunschs einfließen lassen.

In Richtung 'Beantworter->Initiator' können die Nutzdaten bei Ablehnung den Ablehnungsgrund und bei Annahme nähere Modalitäten zur Nutzung der Verbindung codieren.

Diese Verwendung der Nutzdaten und die gewählte Form der Darstellung ist nicht vom Medium reglementiert und fällt in den Verantwortungsbereich der nächsthöheren Schicht.

Der Parameter 'Ablehnungsgrund' tritt deshalb auch nur bei der Anzeige eines vom Medium selbst spontan veranlaßten Verbindungsabbruchs auf und gibt den Dienstnehmern Aufschluß über die Ursache des Abbruchs.

4.1.2.3.2 Primitive

Die Dienstprimitive des bestätigten Dienstleistungstyps Verbindungsaufbau (Kürzel: Con) sind im wesentlichen folgendermaßen parametrisiert. Nicht ausdrücklich erwähnt sind die beiden Parameter, die nur im Zuge der vertikalen Kommunikation zur Identifikation des Orts eines Ereignisses und des betroffenen Verbindungsendpunkts benötigt werden, d.h. Zugangspunktadresse des Orts des Ereignisses und Verbindungsendpunkt-Identifikation.

- ConReq (Initiatoradresse, Adresse des gewünschten Beantworters, vom Initiator gewünschte Verbindungseigenschaften, vom Initiator belegte kurze Nutzdaten),
- ConInd (Initiatoradresse, Adresse des Beantworters, vom Initiator gewünschte aber gegebenenfalls vom Medium 'zurückgestutzte' Verbindungseigenschaften, vom Initiator belegte kurze Nutzdaten),
- ConRes (Adresse des Beantworters, vom Beantworter her unterstützbare Verbindungseigenschaften entstanden aus Zurückstutzen der Angaben des ConInd, vom Beantworter belegte kurze Nutzdaten),
- ConCon (Adresse des Beantworters, Verbindungseigenschaften des ConRes gegebenfalls vom Medium abermals zurückgestutzt, vom Beantworter belegte kurze Nutzdaten).

Zur erwähnen ist die besondere Behandlung der Parameter zu den Verbindungseigenschaften. Über sie wird im Verlauf des Verbindungsaufbaus zwischen den drei Parteien Initiator, Medium und Beantworter verhandelt (Negotiation). Und zwar kann der Initiator mit ConReq seinen Vorstellungen entsprechend Wünsche zu den Verbindungseigenschaften anmelden. Das Medium teilt diese per

ConInd dem Beantworter mit, hat sie aber zuvor daraufhin geprüft, ob es selbst in der momentanen Betriebssituation in der Lage ist, eine entsprechende neue Verbindung zu unterstützten. Ist das nicht der Fall, werden zuvor Abstriche gemacht. Nach ConInd prüft der Beantworter, ob er überhaupt und wenn ja mit welchen Eigenschaften eine Verbindung zum Initiator betreiben will. Auch er kann weitere Abstriche an den Verbindungseigenschaften machen, die er anschließend mit ConRes dem Medium mitteilt. Nun ist das Medium wieder an der Reihe und kann die dem Initiator per ConCon gemeldeten und für die Verbindung verbindlichen Eigenschaften nochmals mit Abstrichen versehen.

4.1.2.4 Datenaustausch

Für den Datenaustausch über räumlich weite Entfernungen werden vornehmlich unbestätigte Dienstleistungen benutzt, weil sich hiermit eine sehr lose Synchronisation zwischen Sender und Empfänger ergibt. Sie entkoppelt Sender und Empfänger in ihrem Fortschreiten weitgehend, so daß bei den beteiligten Dienstnehmern Wartezeiten oft vermieden werden können und die Parallelarbeit gefördert wird. So bieten transportorientierte Dienste in der Regel nur eine und zwar eine unbestätigte Datenaustausch-Dienstleistung an und zwar in Form des Dienstleistungstyps 'Datentransfer' mit den Primitiven Datentransfer-Anforderung (DataRequest) und Datentransfer-Anzeige (DataIndication). Bei anwendungsorientierten Diensten sind dagegen mehrere verschiedene Dienstleistungstypen und auch bestätigte Dienstleistungstypen zum Transfer verschiedener Nutzdaten-Arten vorgesehen, wobei auch hier für den reinen Nutzdatentransfer in der Regel nur der unbestätigte Dienstleistungstyp 'Datentransfer' vorgesehen ist. Andere ebenfalls mit Datentransfer befaßte Dienstleistungen übertragen zusätzlich Information, die in der speziell vorliegenden Art der Kooperationsbeziehung zwischen den Dienstnehmern genauer eingeordnet werden kann.

4.1.2.4.1 Unbestätigter Datenaustausch

Einziger Parametertyp der Primitive DataRequest und DataIndication sind Nutzdaten, i.a. Bitmuster beliebigen Aufbaus und (u.U. innerhalb einer vorgegebenen Obergrenze) beliebiger Länge, die vom Medium vom Ort der Anforderung zum Ort der Anzeige transportiert werden.

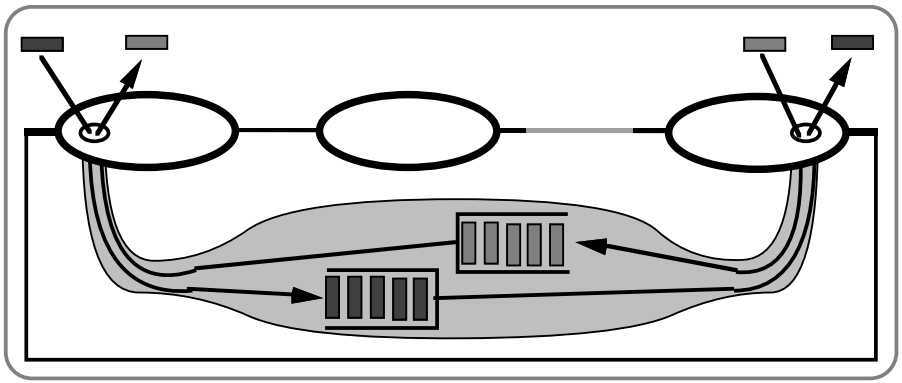

Bild 4.6: Datenaustausch über Verbindung

Bild 4.6 zeigt eine abstrakte Sicht auf eine aufgebaute Zweipunkt-Verbindung im Duplex-Betrieb. Sobald die Verbindung aufgebaut ist, können an beiden Endpunkten Nutzdaten (dargestellt durch die kleinen Rechtecke) per DataReq übergeben werden. Der Betrieb beider Richtungen erscheint nach außen als voneinander unabhängig, es bestehen keine den Dienstnehmern sichtbaren Querbeziehungen. Medienintern können je Richtung solche Nutzdaten-Pakete über eine gewisse (und i.a. nicht vorbestimmbare Zeit) zwischengespeichert werden (Pufferung an den beiden Endpunkten und Laufzeit über untergeordnete Dienste).

Im Idealfall erfolgt die Pufferung und Auslieferung nach der FIFO-Disziplin ohne daß Reihenfolgevertauschungen, Duplikationen oder Verluste möglich sind. Die interne Pufferungskapazität ist jedoch in allen Fällen endlich. Das Medium wird aus Realisierbarkeitsgründen deshalb häufig eine Flußbeschränkung durch Rückstau vorsehen, die u.U. durch einen Parameter beim Verbindungsaufbau von den Dienstnehmern beeinflußt werden kann. Bei nicht verlust-freien Verbindungen kann anstelle von Rückstau auch das Vergessen anhängiger Datenpakete zur Einhaltung der endlichen Kapazität vorkommen.

Zur garantierten Verlustfreiheit ist folgendes anzumerken. Natürlich kann in einem echten Telekommunikationssystem generell für eine individuelle Nachricht nicht garantiert werden, daß sie beim Empfänger ankommt. Trotzdem kann für eine Verbindung die Eigenschaft der Verlustfreiheit zugesichert werden. Und zwar führt hier der Fall des medienintern nicht behebbaren Verlustes zum erbringer-initiierten Abbruch der Verbindung. Dienstnehmern, denen ein ordnungsgemäßer Abbau einer Verbindung geglückt ist, kann so garantiert werden, daß alle im Verlauf des Betriebs angelieferten Datenpakete auch beim Empfänger ausgeliefert wurden. Kommt zur Garantie der Verlustfreiheit die FIFO-Eigenschaft hinzu, kann ferner auch während des Verbindungsbetriebs aus der Ankunft eines Datenpakets auf den korrekten Transfer aller vorher in derselben Richtung gesendeten geschlossen werden.

Weiterhin anzumerken ist, daß die Dienstleistung 'Datentransfer' auch in verschiedenen Prioritätsstufen angeboten werden kann. So kann es daneben z.B. auch noch den prinzipiell gleichwirkenden Dienstleistungstyp 'Hochpriorer Datentransfer' (mit den Primitiven ExpeditedDataRequest und ExpeditedDataIndication) geben. Hier hat man sich dann je Richtung 2 voneinander unabhängige FIFO-Schlangen vorzustellen, eine für den normal priorisierten, eine für den hochprioren Transfer.

4.1.2.4.2 Bestätigter Datenaustausch

Der bestätigte Datenaustausch findet sich nur bei anwendungsorientierten Diensten, bei denen die Kooperationsbeziehung zwischen den Dienstnehmern aus Sicht des Dienstes feiner aufgeschlüsselte Rollen als nur Datenpaketsender und Datenpaketempfänger vorsieht. Im folgenden wird ein Beispiel aus dem Kommunikationssteuerungsdienst besprochen. Und zwar gibt es hier eine in der Datenaustauschphase anforderbare Dienstleistung 'Vereinbarung eines untergeordneten Wiederaufsetzpunkts (SyncMinor)' mit den 4 Primitiven:

- SyncMinReq (Typ, Laufnummer, Nutzerdaten),
- SyncMinInd (Typ, Laufnummer, Nutzdaten),
- SyncMinRes (Laufnummer, Nutzdaten),
- SyncMinCon (Laufnummer, Nutzdaten).

Mit dieser Dienstleistung sollen speziell Nutzdaten ausgetauscht werden, die sich auf die Vereinbarung eines Wiederaufsetzpunkts zwischen den Dienstnehmern beziehen.

4.1.2.5 Verbindungsabbau

Der geregelte Verbindungsabbau wird von einem dazu berechtigten Dienstnehmer aus der Datenaustauschphase heraus durch Anforderung der Dienstleistung 'Verbindungsabbau (Disconnect bzw. Dis

oder aber auch Release bzw. Rel)' eingeleitet. Üblicherweise steht Disconnect für einen entsprechenden unbestätigten Dienstleistungstyp (Abbau durch Abbruch), Release für einen bestätigten. Transportorientierte Dienste sehen dabei i.a. Disconnect zum Abbau vor, während bei anwendungsorientierten Release verwendet wird. Die bestätigte Form Release öffnet die Möglichkeit, daß das Medium den Abbau bis zur Beendigung aller anstehenden Dienstleistungen verzögert, und daß der zweite Partner dem Abbauwunsch des anderen widersprechen kann. Die Primitive sind:

- RelReq (Kurze Nutzdaten),
- RelInd (Kurze Nutzdaten),
- RelRes (Anzeige +/-, Nutzdaten),
- RelCon (Anzeige +/-, Nutzdaten).

Im Zusammenhang mit dem Abbau (d.h. seiner Anforderung und seiner Annahme bzw. Ablehnung) können weiterhin kurze Nutzdaten zwischen den Dienstnehmern ausgetauscht werden.

4.1.2.6 Verbindungsabbruch

Generell muß bei einer echten Telekommunikationsverbindung jederzeit mit dem außerplanmäßigen Abbruch der Verbindung gerechnet werden. Zu unterscheiden sind zwei entsprechende Dienstleistungstypen, der Erbringerabbruch (Provider-Abort bzw. PAbo) und der Nutzerabbruch (UserAbort bzw. UAbo).

Der Erbringerabbruch ist keine Dienstleistung in dem engen Sinn, daß ein Dienstnehmer ihn angefordert hat. So ist hierfür auch kein Stimulustyp sondern nur eine Reaktion zur Anzeige vorgesehen:

- PAboInd (Ablehnungsgrund).

Kann das Medium eine Verbindung aus internen Gründen nicht mehr weiter unterstützen, teilt es dies allen Dienstnehmern, die momentan schon von der Existenz oder vorgesehenen Existenz der Verbindung wissen, durch eine PAboInd-Anzeige mit. Der Erbringer-Abbruch kann in jedem Zustand der Verbindung unbeachtet der Tatsache auftreten, daß u.U. noch Dienstleistungen nicht zu Ende abgewickelt wurden. Müssen z.B. in der Datenaustauschphase beide Endpunkte vom Abbruch unterrichtet werden, so werden an jedem Endpunkt je eine Anzeige erzeugt. Die Gleichzeitigkeit der beiden Anzeigen oder eine bestimmte zeitliche Reihenfolge werden nicht garantiert.

Der Nutzerabbruch ist eine echte Dienstleistung, die in unbestätigter Form und nur auf Anforderung eines Dienstnehmers ausgeführt wird:

- UAboReq (Nutzdaten),
- UAboInd (Nutzdaten).

Sie gestattet es einem Dienstnehmer kurzfristig den Abbruch anzufordern. Mit dem Zeitpunkt der Anforderung gilt die Verbindung für den Anforderer sofort als abgebrochen. Er darf selbst keine Anforderungen an diese Verbindung mehr stellen und wird umgekehrt keinerlei Anzeigen mehr erhalten. Dem anderen Dienstnehmer wird der Abbruch in möglichst kurzer (aber dennoch unvorhersehbarer) Zeitdauer per UAboInd mitgeteilt. Manche Dienste gestatten mit dem Abbruch noch den Austausch (i.a. kurzer) Nutzdaten (Abbruchgrund-Information).

Auch der Nutzerabbruch kann in jedem Zustand einer Verbindung und zwar von beiden betroffenen Dienstnehmern angefordert werden. Es können insbesondere auch beide Dienstnehmer annähernd gleichzeitig den Abbruch anfordern oder ein Nutzerabbruchwunsch kann mit einem Erbringerabbruch zusammentreffen. In solchen Fällen verhält sich das Medium so, daß überflüssige (und nach Abgabe eines UAboReq am selben Ort auch unzulässige) Abbruchanzeigen unterdrückt werden.

Als Besonderheit ist anzumerken, daß bei transportorientierten Diensten, die die unbestätigte Form des Verbindungsabbaus Disconnect vorsehen, auf ein gesondertes Dienstelement UAbort verzichtet wird, weil Disconnect dieselbe Wirkung besitzt.

4.1.3 Datagramm Dienste

Datagramm-Dienste (Connectionless Services) unterstützen dem Transport von Briefen vergleichbare Datenaustauschdienstleistungen. Jede Dienstleistung wird vom Medium für sich gesondert ohne den Dienstnehmern ersichtliche Querbeziehungen zu anderen Dienstleistungen erbracht. Insbesondere werden auch keine Garantien zur Reihenfolgetreue gegeben. Es handelt sich also - in Bezug auf die den Dienstnehmern angebotenen Funktionalität - um sehr einfache Dienstleistungen. Grundtyp ist die unbestätigte Datagramm-Dienstleistung zur Zwei-Parteien-Kommunikation, die im folgenden umrissen werden soll.

Die Primitive des Dienstleistungstyps 'Datagramm-Transfer (UnitdataTransfer bzw. UnitData)' sind:

- UnitDataReq (Absenderadresse, Empfängeradresse, geforderte Dienstleistungsqualitäten, Nutzdaten),
- UnitDataInd (Absenderadresse, Empfängeradresse, geforderte Dienstleistungsqualitäten, Nutzdaten).

Die Adressen sind dabei als Zugangspunktadressen ausgeführt.

Als Dienstleistungsqualitäten sind relevant Laufzeit, Schutzklasse, Fehlerwahrscheinlichkeit, Kostengrößen und u.U. eine Prioritätsangabe). Da es sich um eine unbestätigte Dienstleistung handelt, werden die Qualitätsangaben nicht verhandelt. Der Sender legt seine Anforderungen nieder und das Medium versucht diesen nach seinen Möglichkeiten gerecht zu werden.

Die Nutzerdaten müssen nun (wie z.B. auch in Briefen üblich) in Regie des Absenders eingebrachte Informationen enthalten, die es dem Empfänger erlauben, sie in eine Kooperationsbeziehung einzuordnen (z.B. Bezug, Datum, Laufnummer).

4.2 Telekommunikationsprotokolle

Betrachtet man ein abstraktes Medium genauer, so setzt es sich intern aus einzelnen Instanzen zusammen, die ihrerseits über ein untergeordnetes Medium kommunizieren. Bild 4.7 zeigt ein entsprechendes Szenario.

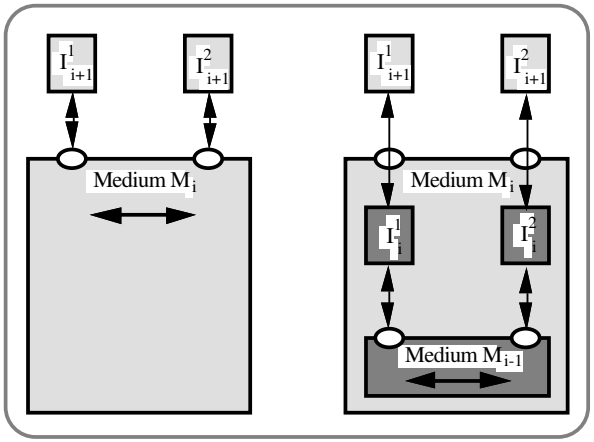

Bild 4.7: Medien und Protokoll

Im groben sollen die Instanzen einer Schicht in Kooperation die an der Schichtschnittstelle geforderten Dienstleistungen erbringen. Wenn die Funktionalität des unterliegenden Mediums als gegeben angenommen wird, soll die Menge der Instanzen die zwischen zu erbringendem und zur Verfügung stehendem Dienst bestehende Differenz in Funktionalität und Leistungsfähigkeit überbrücken.

Bei verfeinerter Betrachtung dienen die Instanzen diesem Ziel durch geregelte Kooperation. Für jede einzelne Instanz liegen Verhaltensfestlegungen vor, nach denen sie agieren muß, um unter Kommunikation über das zur Verfügung stehende Medium die geregelte Kooperation zu gewährleisten. Die Verhaltensfestlegungen mit den Richtlinien zur Nutzung des unterliegenden Mediums bilden ein Telekommunikationsprotokoll.

4.2.1 Beschreibung der Protokollfunktionalität

Ein Telekommunikationsprotokoll wird mit der Beschreibung des Verhaltens entsprechend agierender Instanzen festgelegt. Die Beschreibung wird i.a. in zwei Teile gegliedert. Ablauffestlegungen beschreiben das Instanzenverhalten unter Abstraktion von dem tatsächlichen Aufbau der Nutzdaten, die die Instanzen über den untergeordneten Dienst austauschen, Formatfestlegungen legen die konkret zu wählende Syntax der Nutzdaten fest.

4.2.1.1 Ablauffestlegungen

Ablauffestlegungen werden sehr häufig und anschaulich durch Zustandsdiagramme von Mealy-Automaten getroffen, wie sie ebenfalls schon zur Beschreibung des Verhaltens von Medien verwendet wurden. Diese Form der Instanzenbeschreibung soll durch nachfolgendes Beispiel verdeutlicht werden.

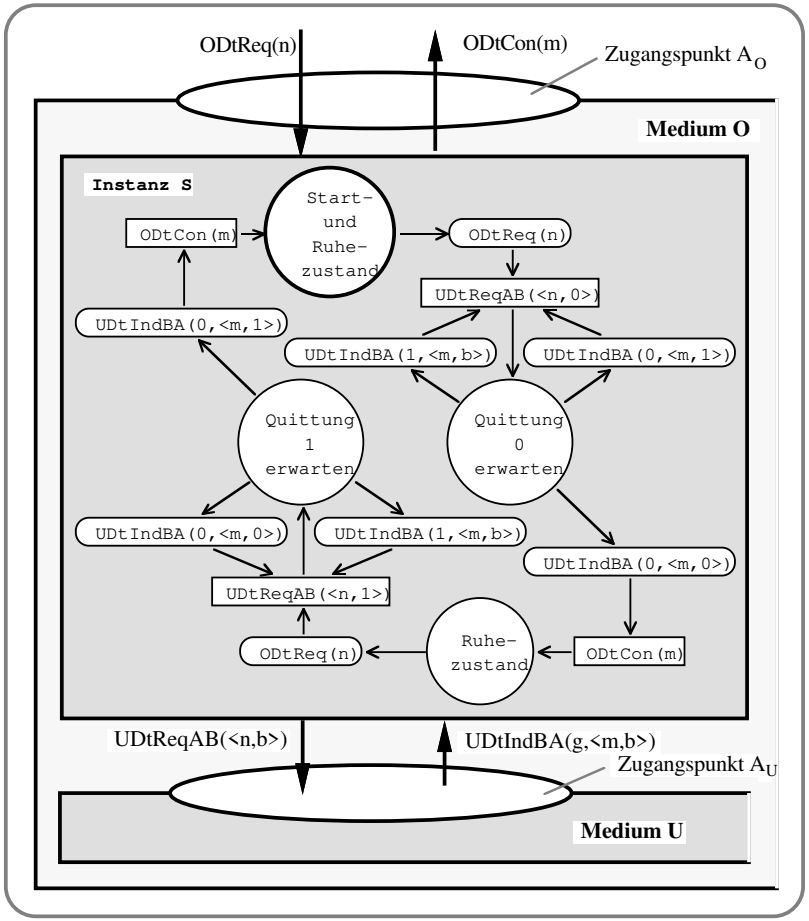

Bild 4.8: Beispielinstanz S

Bild 4.8 zeigt das Zustandsdiagramm einer Instanz S. Sie unterstützt den Dienstzugangspunkt A_O des Mediums O und betreibt als Dienstnehmer den Zugangspunkt A_U des Mediums U.

Eingaben des Automatenmodells für S sind:

- ODtReq(n); Anforderung einer O-Dienstleistung 'bestätigte Datenübertragung'. n steht für eine Parametrisierung mit nicht näher reglementierten Nutzdaten.
- UDtIndBA(g,n); Anzeige einer U-Dienstleistung 'unbestätigte Datenübertragung von einem zweiten Zugangspunkt B_U nach A_U'. n steht für eine Parametrisierung mit Nutzdaten. g ist ein Kontrollparameter mit der Bedeutung 'g=0': bei der Datenübertragung wurden keine Verfälschungen erkannt, und 'g=1': die Nutzdaten wurden verfälscht.

Ausgaben des Automatenmodells für S sind:

- ODtCon(n); Bestätigung der O-Dienstleistung 'bestätigte Datenübertragung' (n: mit Bestätigung übertragene Nutzdaten).
- UDtReqAB(n); Anforderung einer U-Dienstleistung 'unbestätigte Datenübertragung von A_U nach einem zweiten Zugangspunkt B_U' (n: Nutzdaten).

Die mit U-Dienstleistungen übertragenen Nutzdaten bilden die Protokolldateneinheiten (PDUs) des Protokolls und müssen einem festgelegten Format entsprechen. Hier, bei den Ablauffestlegungen interessiert aber zunächst nur ihr logischer Inhalt. Dies ist bei diesem einfachen Protokoll in beiden Richtungen ein Paar <n,b> aus einer nicht näher reglementierten Komponente n (die Nutzdatenkomponente einer PDU) und einem Kontrollparameter b, der die Werte 0 oder 1 annehmen kann. Im strengen Sinn bildet z.B. jede mögliche Wertkombination einer Anzeige UDtIndBA(g,<m,b>) ein von anderen zu unterscheidendes Eingabezeichen des Automaten.

Der Automat enthält die 4 folgenden Zustände:

- Start- und Ruhezustand; in diesem Zustand befindet sich die Instanz S unmittelbar nach der Systeminitialisierung. Er wird zur Bearbeitung eines ODt-Dienstleistung verlassen. Der Übergang wird durch ein ODtReq-Ereignis mit beliebigem Nutzdatenparameter n aktiviert, unter Erzeugung eines UDtReqAB-Ereignisses (Nutzdatenparameter: <n, Sequenzzahl 0>) wird der Folgezustand 'Quittung 0 erwarten' betreten.
- 'Quittung 0 erwarten'; dieser Zustand wird verlassen, wenn eine UDtIndBA-Anzeige eintrifft. Drei Übergänge sind in Unterscheidung dreier Parameterkombinationsklassen vorgesehen. Wenn die Verfälschungsanzeige g gesetzt ist, wird unbeachtet der Nutzdaten rückgesetzt (Wiederholung der letzten UDtReqAB-Erzeugung). Wenn die Verfälschungsanzeige nicht gesetzt ist, wird ebenfalls rückgesetzt, wenn der PDU-Kontrollparameter 'Sequenzzahl' nicht den hier erwarteten Wert 0 besitzt, ansonsten wird die Bearbeitung der Dienstleistung ODt unter Erzeugung einer Bestätigung ODtCon abgeschlossen. ODtCon liefert die mit der letzten PDU <m,b> empfangenen PDU-Nutzdaten m an den O-Dienstnehmer aus. Mit dem Abschluß wird der Zustand 'Ruhe–zustand' eingenommen.
- 'Ruhezustand'; dieser Zustand ist mit dem Zustand 'Start- und Ruhezustand' vergleichbar. Auch hier wird auf den nächsten O-Auftrag gewartet, und er wird wie dort behandelt mit der Besonderheit, daß nun die Bedeutung der Sequenzzahlen 0 und 1 vertauscht ist.
- 'Quittung 1 erwarten'; entspricht 'Quittung 0 erwarten' bei vertauschter Bedeutung der Sequenzzahlen 0 und 1.

Es soll noch auf die im Beispiel verwendeten Datenvariablen (n,m,b,g) etwas näher eingegangen werden. Sie verfeinern den grob nur aus den 4 vorgenannten Hauptzuständen bestehenden Zustandsraum des Automaten S. In jedem Hauptzustand können in n und m beliebige Nutzdaten und in b und g verschiedene Anzeigenkombinationen gespeichert sein. Würde man auf die Variablen verzichten und für alle Kombinationen einen eigenen Hauptzustand vorsehen, wäre das Modell sehr unübersichtlich. Deshalb sind derartige Erweiterungen des Automatenkonzepts üblich und werden auch in den beiden genormten Protokollbeschreibungssprachen SDL und ESTELLE vorgesehen, wie sie von den beiden Normungsgremien CCITT und ISO für die zukünftige Normung von Protokollen eingesetzt werden sollen.

```
process  S ;                        ! Instanz S als erweiterter Automat
  const                             ! in programmiersprachlicher Notation
    maxl = 256 ;                    ! maximale Nutzdatenlaenge
  type
    usd = record                    ! Datentyp: Nutzdaten
          lng : integer ;
          octetts : array [0..maxl-1] of char ;
          end ;
    pdu = record                    ! Datentyp: Protokolldateneinheiten
          b : (0..1) ;              ! Kontrollbit
          d : usd ;                 ! eingepackte Nutzdaten
          end ;
  interactions   in  ODtReq ( n : usd ) ;
                     UDtIndBA ( g : boolean ; p : pdu ) ;
                 out ODtCon ( n : usd ) ;
                     UDtReqAB ( p : pdu ) ;
  var                               ! Zustandsspeicher per Variablen
    Majorstate : ( idle, wait ) ;       ! Kontrollzustandskomponente
    Buffer : usd ;                  ! Nebenzustandskomponente: Puffer
    Seq : (0..1) ;                  ! Nebenzustandskomponente: Sequenzzaehler
  initialize                        ! Initialisierungsklause: Startzustand
    Majorstate:=idle ;
    Seq:=0 ;
  transitions                       ! Transitionsklassen in Transitionsklausen
    when ODtReq(n) from idle to wait
      begin                         ! ODtReq annehmen und PDU senden
      var h : pdu ;
      Buffer:=n ; h.b:=Seq ; h.d:=Buffer ; out UDtReqAB(h) ;
      end ;
    when UDtIndBA(g,p) provided g=true or p.b<>Seq from wait to wait
      begin                         ! Negative Quittung annehmen und PDU wiederholen
      var h : pdu ;
      h.b:=Seq ; h.d:=Buffer; out UDtReqAB(h) ;
      end ;
    when UDtIndBA(g,p) provided g=false and p.b=Seq from wait to idle
      begin                         ! Positive Quittung annehmen und ODtCon an Nutzer
      if Seq=0 then Seq:=1 else Seq:=0 ; out ODtCon(p.d) ;
      end ;
  end S.
```

Bild 4.8.1: Textuelle Spezifikation der Beispielinstanz S

Bild 4.8.1 zeigt dazu eine textuelle Spezifikation der Instanz S aus Bild 4.8. Die Notation orientiert sich an Estelle und SDL und verwendet (wie Estelle) PASCAL-Sprachelemente.
Das Ein/Ausgabe-Alphabet von S wird durch das 'interactions'-Konstrukt eingeführt. Datenparameter erlauben es, Klassen von Eingaben/Ausgaben (z.B. alle möglichen und sich in den Nutzdaten unterscheidenden ODtReq-Eingaben) zusammengefaßt einzuführen.
Die Zustandsmenge von S entspricht dem Kreuzprodukt der Wertemengen der Datentypen der Variablen von S (i.e. Majorstate, Buffer, Seq). Im Vergleich zu Bild 4.8 wurde die Anzahl der Hauptzustände halbiert, so fallen nun 'Quittung 0 erwarten' und 'Quittung 1 erwarten' im Hauptzustand 'wait' zusammen. Die Unterscheidung, ob 0 oder 1 erwartet wird, wird mit der Nebenzustandskomponente Seq getroffen. Die Nebenzustandkomponente Buffer dient der Pufferung des zuletzt gesendeten und unter Umständen zu wiederholenden Nutzdatums.
Die Initialisierungsklause beschreibt in Form von Zuweisungen den Startzustand.
Die Transitionsklausen führen die Transitionen von S in Klassen strukturiert ein. Jede Transitionsklause ('when...end;') beschreibt eine Transitionsklasse. Das 'when'-Konstrukt weist auf die benötigte Eingabe hin, das 'from'-Konstrukt auf den momentanen Hauptzustand und das 'to'-Konstrukt auf den Folge-Hauptzustand. Zwischen 'begin' und 'end' steht eine Anweisungsfolge (hier Zuweisungen und Ausgabe-Anweisungen), deren Ausführung im Momentanzustand gestartet wird und nach Terminierung den Folgezustand bestimmt. Im Anweisungteil von Transitionsklausen werden im Beispiel ferner temporäre Variablen 'h' deklariert, deren Lebensdauer jeweils nur der Ausführungszeit einer Transition entspricht.
In der Sicht des Automatenmodells werden die Transitionen - obwohl sequentiell als Anweisungsfolge notiert - atomar ausgeführt, so daß die Variablen 'h' nicht im Zustand berücksichtigt werden müssen, und so daß ferner - wenn mehrere miteinander über wechselseitige Ein/Ausgaben kommunizierende Automaten zu einem System gekoppelt werden - man von der Annahme ausgeht, daß gleichzeitige Transitionen nicht vorkommen.

4.2.1.2 PDU-Format-Festlegungen

Zu unterscheiden sind die Formatfestlegungen für PDUs von Protokollen der Darstellungsschicht und darunter liegender Schichten von den Formatfestlegungen für oberhalb der Darstellungsschicht angesiedelte Protokolle.

4.2.1.2.1 Bitmuster-PDU-Formate

Bei Protokollen der Darstellungsschicht und darunter wird je PDU-Typ genau eine verbindliche Bitmuster- oder Oktett-Folgen-Repräsentation angegeben. Erst oberhalb des Darstellungsdienstes können verschiedene Darstellungskontexte von Dienstnehmern gewählt werden.

Die Formatfestlegung erfolgt traditionell durch verbale Beschreibung in Verbindung mit graphischen Skizzen.

Im vorgehend mit Bild 4.8 gezeigten einfachen Beispiel müssen nur zwei PDU-Typen unterschieden werden, von A_U nach B_U übertragene U-Nutzdaten und von B_U nach A_U übertragene; und sogar diese beiden Typen können, da sie gleichen Aufbau besitzen, in einem Typ zusammengefaßt werden. Die Vereinbarung der konkreten Darstellung für diesen PDU-Typ <n,b> mit den beiden logischen Komponenten n und b könnte z.B. folgendermaßen aussehen:

- Die PDU besteht aus einer Folge von lng Bytes bzw. Oktetten (lng>3, lng<1027).
- 1.Oktett: Codierung der Komponente b im niederwertigsten Bit, übrige sieben Bits beliebig.
- 2. und 3. Oktett: Codierung der n-Länge lng-3 mit dem 1. Oktett als höher- und dem 2. als niederwertiges Byte in der üblichen Binärcodierung von Betragszahlen.
- Folgeoktetts; Darstellung eines O-Nutzdatums n in derselben Form, in der n an den Dienst O übergeben wurde.

Diese verbale Festlegung in natürlicher Sprache könnte zusätzlich durch eine geeignete graphische Darstellung verdeutlicht werden, wie sie in Bild 4.9 gezeigt ist.

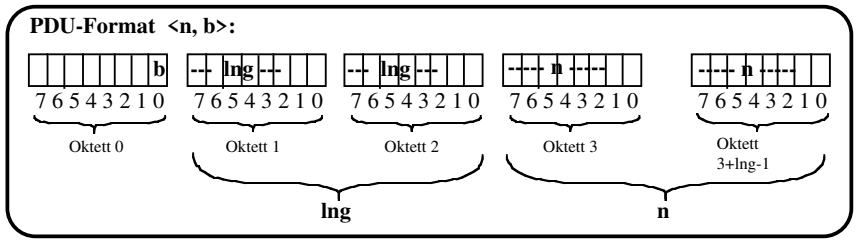

Bild 4.9: PDU-Format - Beispiel

4.2.1.2.2 Anwendungsprotokoll-PDU-Formate

Bei Protokollen, die oberhalb der Darstellungsschicht angesiedelt sind, wird die logische Zusammensetzung einer PDU, d.h. ihre abstrakte Syntax, von ihrer tatsächlichen Bitmuster-Darstellung, d.h. ihrer konkreten Transfersyntax separiert. Für jeden PDU-Typ ist eine einzige verbindliche abstrakte Syntax festgelegt; es können aber - in Abhängigkeit zu unterschiedlichen Darstellungsvereinbarungen - dafür mehrere konkrete Darstellungsweisen in Gebrauch kommen.

Zur Vereinbarung einer abstrakten Syntax wird die Syntaxbeschreibungssprache ASN.1 (Abstract Syntax Notation Number 1) vorgeschlagen und auch in der Regel verwendet, und zur Festlegung der Transfersyntax bezieht man sich üblicherweise auf die im Zusammenhang mit ASN.1 gegebenen Standard-Codierregeln. Beide Konzepte sollen hier nicht im Detail eingeführt werden, es soll aber anhand eines einfachen Beispiels ein Eindruck hierzu vermittelt werden. Es ist in Bild 4.10 gezeigt.

Der zu vereinbarende PDU-Typ 'PDUTyp1' sieht - wie im Beispiel nach Bild 4.9 - zwei logische Komponenten n und b vor. In der abstrakten Syntax ist dazu die feste Abfolge mit b als erster und n als zweiter Komponente vereinbart.

Man kann die ASN.1-Typvereinbarung mit der Vereinbarung eines PASCAL-Datentyps vergleichen (eine dem Beispiel in etwa entsprechende Vereinbarung befindet sich im oberen Teil von Bild 4.10). Wie in PASCAL gibt es Schlüsselwörter für standardmäßig definierte Basistypen und Konstruktionsprinzipien, mit denen über definierten Elementen neue zusammengesetzte Typen anwendungsabhängig vereinbart werden können. ASN.1 bietet im Vergleich zu PASCAL hierbei allerdings einen größeren Spielraum, insbesondere sind mehr unterschiedliche Konstruktionsprinzipien vorhanden (z.B. Folge mit variabler Länge / optionalen Elementen / Elementen unterschiedlicher Typen; Mengen mit Elementen unterschiedlicher Typen; Zusammensetzungen mit alternativem Aufbau).

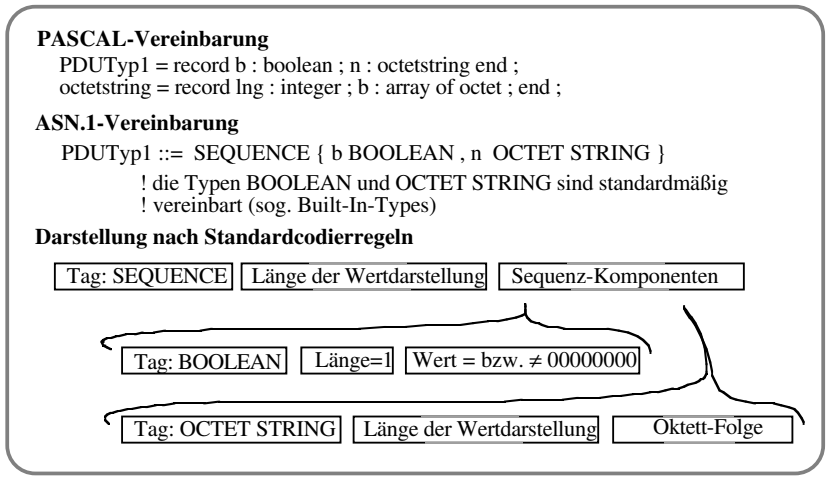

Bild 4.10: Abstrakte PDU-Syntax und Transfersyntax

Die Wertdarstellung nach den Standardcodierregeln folgt wenigen einfachen Prinzipien. Alle Wertdarstellungen sind in Oktetts gerastert. Grundprinzip ist ferner, daß jede Wertdarstellung durch eine Kennung (engl. Tag) eingeleitet wird. Kennungen sind für die Standardtypen und die Konstruktionsprinzipien vereinbart. Man kann aber auch für selbst definierte Typen eigene Kennungen wählen oder das Unterdrücken von Kennfeldern vereinbaren. Für das Tag-Oktett bestehen Konventionen, wie es durch Folgeoktette ergänzt werden kann, so daß die Anzahl unterschiedlicher Typcodes nicht begrenzt ist. Der Typkennung folgt die Codierung der Oktett-Länge der eigentlichen Wertdarstellung. Sie kann ebenfalls ein oder mehrere Oktette lang sein, so daß keine Längenbeschränkung gegeben ist. Bei zusammengesetzten Typen besteht die Wertdarstellung aus der Folge der Darstellungen der Komponenten.

4.3 Einfaches Beispiel

In den beiden vorangegangenen Unterkapiteln wurden Modellvorstellungen und entsprechende Beschreibungsmittel jeweils für Dienste und Protokolle getrennt voneinander eingeführt. Im folgenden soll nun der Zusammenhang zwischen Diensten und Protokollen an einem Beispiel verdeutlicht werden. Wir greifen hierbei auf die in Bild 4.8 in ihren Ablauffestlegungen und mit den Bildern 4.9 bzw. 4.10 in ihren Formatfestlegungen beschriebene Instanz S zurück. S soll zusammen mit einer weiteren Instanz E unter Nutzung eines abstrakten Mediums U einen dem abstrakten Medium O entsprechenden Dienst erbringen. Bild 4.11 gibt dazu einen Überblick über die Zusammensetzung des vom Beispiel-Protokoll betroffenen abstrakten Mediums O.

Das System erstreckt sich über die beiden Orte A und B. Die beiden Instanzen S und E sind mit den Zugangspunkten A_O und A_U bzw. B_O und B_U verbunden. Die Instanzen kommunizieren über das Medium U und realisieren in Kooperation das Medium O.

Mit den Bildern 4.8 und 4.9 bzw. 4.10 ist die Instanz S definiert. Zur Vervollständigung des Beispiels werden also noch Beschreibungen zu folgenden Komponenten erforderlich:

- Dienst O; der Telekommunikationsdienst, der vom Protokoll erbracht werden soll, als Zielvorgabe (Zieldienst).
- Dienst U; der per Medium U den Instanzen S und E zur Verfügung stehende Telekommunikationsdienst als Randbedingung des Protokolls (Basisdienst).
- Instanz E; die Formatfestlegungen sind bereits mit Bild 4.9 bzw. 4.10 getroffen, es fehlen noch die Ablauffestlegungen.

Mit diesen zusätzlichen Angaben ist das Beispielprotokoll dann - zumindest in funktioneller Hinsicht - vollständig beschrieben. Zusätzlich müßten - hier nicht näher ausgeführt - noch die geforderten bzw. vorhandenen Qualitätsparameter der Dienste O und U aufgeführt werden.

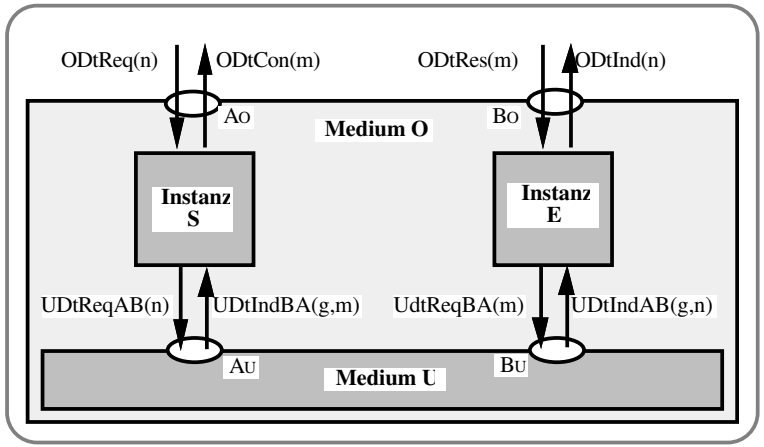

Bild 4.11: Übersicht über Beispiel

4.3.1 Zieldienst O

Bild 4.12 zeigt den Zieldienst O. Es wird an den beiden Zugangspunkten A_O und B_O nur ein Typ von Dienstleistungen, ein bestätigter Datentransfer angeboten. Der Initiator einer Dienstleistung befindet sich immer am Zugangspunkt A_O, der Beantworter an B_O.

Das Automatenmodell zeigt die Abwicklung dieser Dienstleistung, wie sie Dienstnehmern an der O-Dienstschnittstelle sichtbar wird. Verfälschungen, Verluste oder andere Störungsauswirkungen werden bei diesem (darin für echte Telekommunikationsdienste unrealistischen) Beispiel voll verdeckt.

Eine weitere Besonderheit ist, daß zu einem Zeitpunkt sich höchstens eine Dienstleistung in Bearbeitung befinden kann. ODtReq-Ereignisse werden nur im Ruhezustand angenommen.

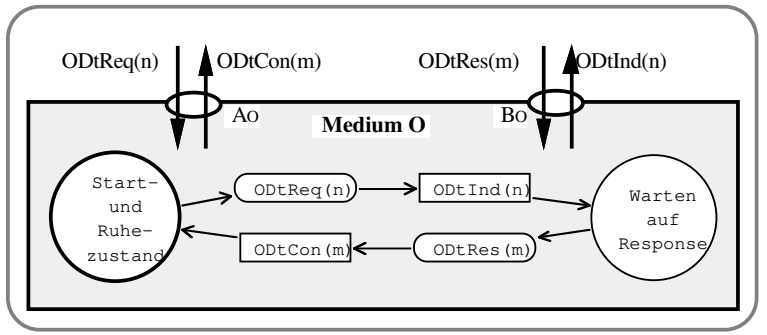

Bild 4.12: O-Dienst

4.3.2 Basisdienst U

Bild 4.13 zeigt den Dienst des Mediums U. Er unterstützt zwischen zwei Zugangspunkten A_U und B_U den einen Dienstleistungstyp 'unbestätigter Datentransfer', der in beide Richtungen möglich ist. Der Richtungsbetrieb muß aber startend mit der Richtung AB alternieren. Nach einer Übertragung in eine Richtung wird nur eine Anforderung zur Übertragung in Gegenrichtung akzeptiert.

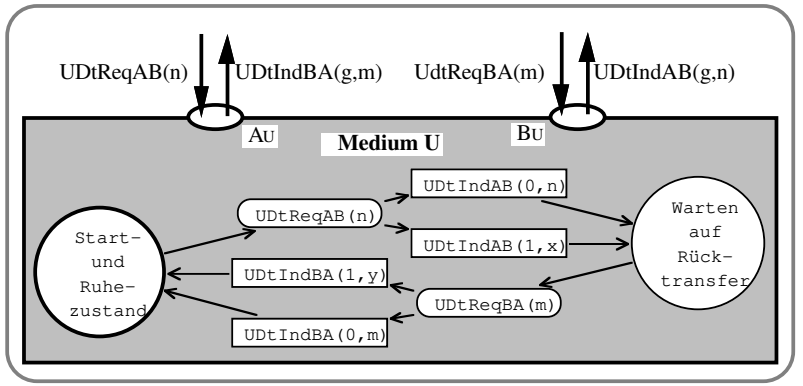

Bild 4.13: U-Dienst

Der U-Dienst kann Auswirkungen medieninterner Störungen nicht ganz verdecken. Er garantiert zwar, daß Verdopplung oder Verlust nicht auftreten. Verfälschung kann jedoch stattfinden. Dies ist an den Übergängen mit der Ausgabe 'UDtIndAB(1,x)' bzw. 'UDtIndBA(1,y)' zu ersehen, die unbeeinflußbar von den Dienstnehmern indeterministisch anstelle der korrekten Anzeigen aufreten können. Verfälschungen werden dem empfangenden Dienstnehmer generell angezeigt durch den Wert des Kontrollparameters g der DtInd-Ereignisse. Dies ist eine unrealistische Vereinfachung, bei echten Te-

lekommunikationsdiensten besteht immer eine - wenn auch durch Redundanzerhöhung bei der Kanalcodierung beliebig verringerbare - Wahrscheinlichkeit >0 für das Auftreten unentdeckter Verfälschungen. Weiterhin ist es unrealistisch, daß der Dienst immer betriebsbereit ist, es sind keine Zustände vorgesehen, in denen keine Dienstleistung mehr unterstützt wird.

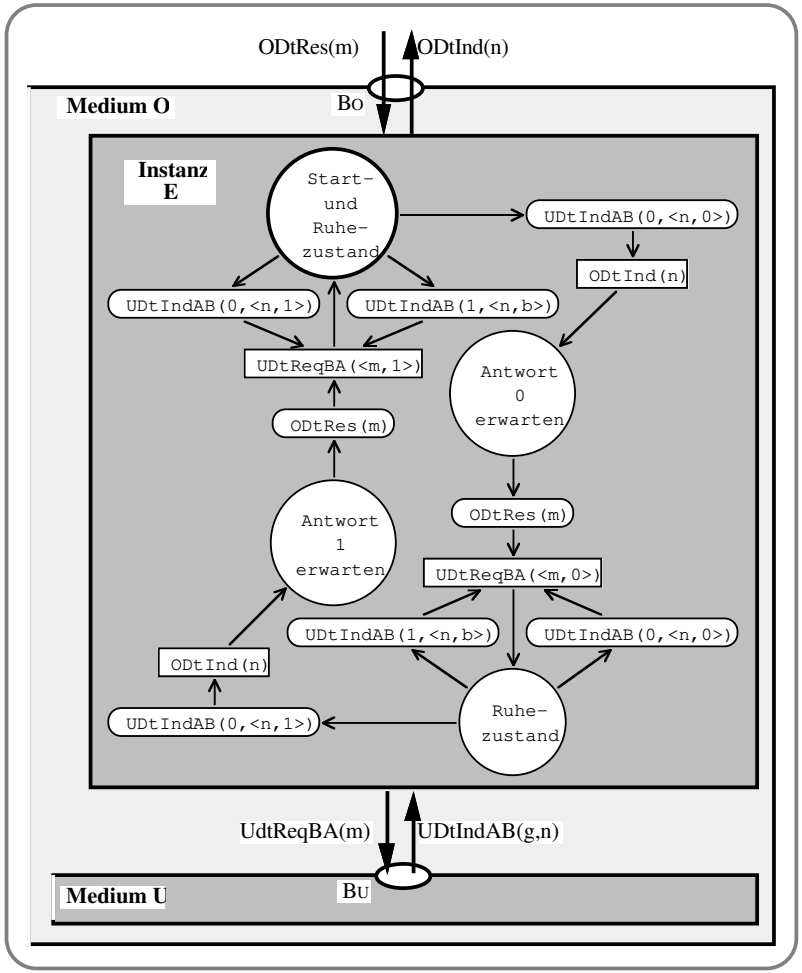

Bild 4.14: Instanz E

4.3.3 Instanz E

Bild 4.13 zeigt die Ablauffestlegungen der zweiten Instanz E. Sie wirkt als Pendant zur Instanz S. Unter Verweis auf die dortige Bilderläuterung soll hier von einer genaueren Erklärung abgesehen werden. Auch hier werden, gesteuert durch Verfälschungsanzeige und Sequenzzahl, entsprechende Rücksetzmaßnahmen vorgesehen. Sie stellen sicher, daß ODtInd-Ereignisse nur nach korrektem Empfang einer Nachricht erzeugt werden. Ferner erlauben sie, daß die Partnerinstanz S die Wiederholung einer Antwort anfordern kann.

Interessant bei diesem Protokoll ist die Doppelrolle, die die zwischen 0 und 1 variierende Sequenzzahl b von PDUs besitzt. Eine Sequenzzahl, die einen im momentanen Zustand nicht erwarteten Wert besitzt, wirkt als negative Quittung, also als Anforderung einer Wiederholung. Durch geschicktes Alternieren der Quittungskennung wirkt sie aber auch als Sequenzzahl und erlaubt es einem Empfänger, den wiederholten Empfang einer bereits korrekt empfangenen PDU zu erkennen.

Um dies besser zu verstehen, sollte man einen Ablauf nachvollziehen, bei dem S zunächst eine PDU mit erwarteter Sequenzzahl 0 sendet. Sie wird korrekt übertragen, und führt bei E zur Erzeugung eines ODtInd-Ereignisses, zum Erwarten des ODtRes und zum anschließenden Senden der Antwort-PDU mit Sequenzzahl 0. Die Antwort werde verfälscht. Nun setzt S zurück und reagiert mit der Wiederholung der Anforderung-PDU mit Sequenzzahl 0. E erwartet inzwischen die Sequenzzahl 1, sie lehnt von ihrem Standpunkt aus die PDU mit negativer Quittungssequenzzahl 0 ab und ignoriert die PDU im weiteren. S empfängt die PDU mit der für S positiven Sequenzzahl 0 und schaltet weiter. Dieser Ablauf ist mit einem nachfolgenden Besonderheits-freien Ablauf in Bild 4.15 als Weg/Zeit-Diagramm skizziert.

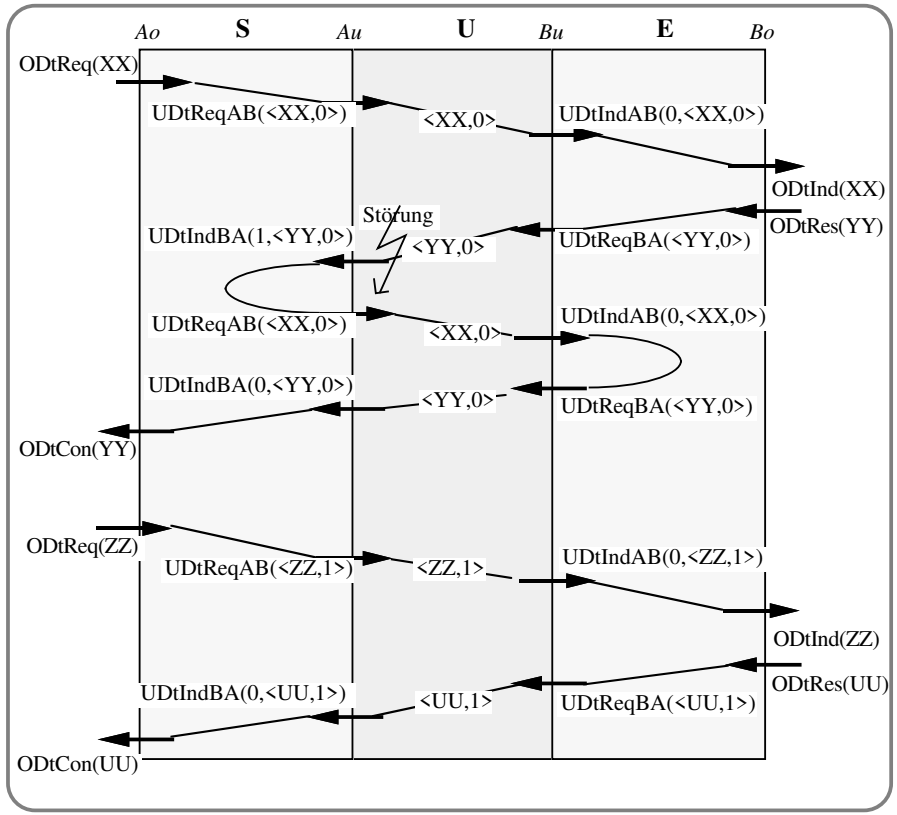

Bild 4.15: Protokollabläufe im Weg/Zeit-Diagramm

Dieses sehr einfache Protokoll wurde bereits in den frühen 60er Jahren zur Abwicklung gesicherter Datenübertragungen über unsichere Halbduplex-Verbindungen entworfen. Es wird in Anspielung auf die Sequenzzahlen Alternating Bit Protokoll genannt und heute nur noch als einfaches Protokollbeispiel verwendet. In seiner Originalform geht es von einem weniger unrealistischen Basisdienst aus, der auch mit dem unentdeckten Verlust von Nachrichten behaftet sein kann. In unserem Beispiel wurde dies zur Vereinfachung des Protokolls ausgeschlossen.

4.3.4 Schlußbetrachtung

Nach der vollständigen Vorstellung des Beispiels sollen folgende wichtigen Fragestellungen kurz angesprochen werden:

- Korrektheit,
- Leistungsfähigkeit,

- Realitätsnähe.

4.3.4.1 Protokollkorrektheit

Ein Protokoll wird als korrekt aufgefaßt, wenn nachgewiesen werden kann, daß das System aus Instanzen und unterliegendem Medium - wenn seine Komponenten (im Beispiel S, E und U) beschreibungskonform agieren - sich an der Schnittstelle des zu erbringenden Dienstes (im Beispiel O) konform zur Beschreibung dieses Dienstes verhält. Hierzu werden i.a. alle möglichen Abläufe eines Systemmodells untersucht und mit der Zieldienstbeschreibung verglichen. Dies ist i.a. eine sehr aufwendige, jedoch für jedes neu entworfene Protokoll durchzuführende Maßnahme.

In unserem Beispiel kann die Korrektheit - aufgrund der Realitätsferne des Dienstes O - nur unter (realitätsfernen) zusätzlichen Annahmen über die Güte des Basisdienstes U nachgewiesen werden: U entdeckt in der Tat alle Verfälschungen und zeigt sie an, U ist wirklich immer betriebsbereit, U wird zu jedem Zeitpunkt in Zukunft immer auch unverfälschte Übertragungen in beide Richtungen durchführen. Mit diesen Annahmen kann nachgewiesen werden, daß jede per ODtReq zulässig angeforderte Dienstleistung in endlicher Zeit erfolgreich und der Dienstbeschreibung entsprechend abgewickelt wird.

Eine Möglichkeit diesen Nachweis zu führen soll im folgenden kurz umrissen werden. Bild 4.16 zeigt dazu zunächst ein vereinfachtes Modell des Protokollsystems. Es besteht ebenfalls aus den drei Komponenten S, E und U, die den Dienst O erbringen sollen. Im Vergleich zu den Bildern 4.8, 4.12, 4.13 und 4.14 wurden die Komponenten vereinfacht. Und zwar werden die zu übertragenden Nutzdaten des O-Dienstes nicht berücksichtigt. Damit werden auch die Nutzdaten der PDUs vernachlässigt. Die Kontrollparameter bleiben - da funktionsrelevant - erhalten; so wird z.B. aus UDtIndAB(0, <n, 0>) in der Vereinfachung UDtIndAB0, aus UDtIndAB(0, <n, 1>) wird UDtIndAB1, und aus UDtIndAB (1, <n, b>) wird UDtIndABg.

Geändert hat sich ferner das Verhaltensbeschreibungsmittel. Anstelle von Mealy-Automaten werden sehr einfache Transitionssysteme verwendet, bei denen mit einem Zustandsübergang entweder nur eine einzige Eingabe oder nur eine einzige Ausgabe stattfinden kann. Die Instanz S ist z.B. im Zustand 0 bereit eine ODtReq-Eingabe anzunehmen. Im nachfolgenden Zustand 1 möchte sie die Ausgabe UDtReqAB0 senden.

Zur Kopplung zwischen den Komponenten werden in solchen Analysemodellellen häufig puffernde Kanäle eingesetzt. Eine andere Möglichkeit bildet die Anwendung des Kopplungskonzepts 'Rendez-Vous'. Diese Möglichkeit ist im Beispiel zugrundegelegt. Eine sendewillige Komponente kann nur dann senden, wenn der Empfänger momentan in einem für das zu sendende Zeichen empfangsbereiten Zustand ist. Zum Beispiel seien Instanz S in Zustand 1 und Basisdienst U in Zustand 0. Dann kann im nächsten Zeitschritt S die Sendetransition nach 2 und U eine Empfangstransition nach 1 oder 2 ausführen.

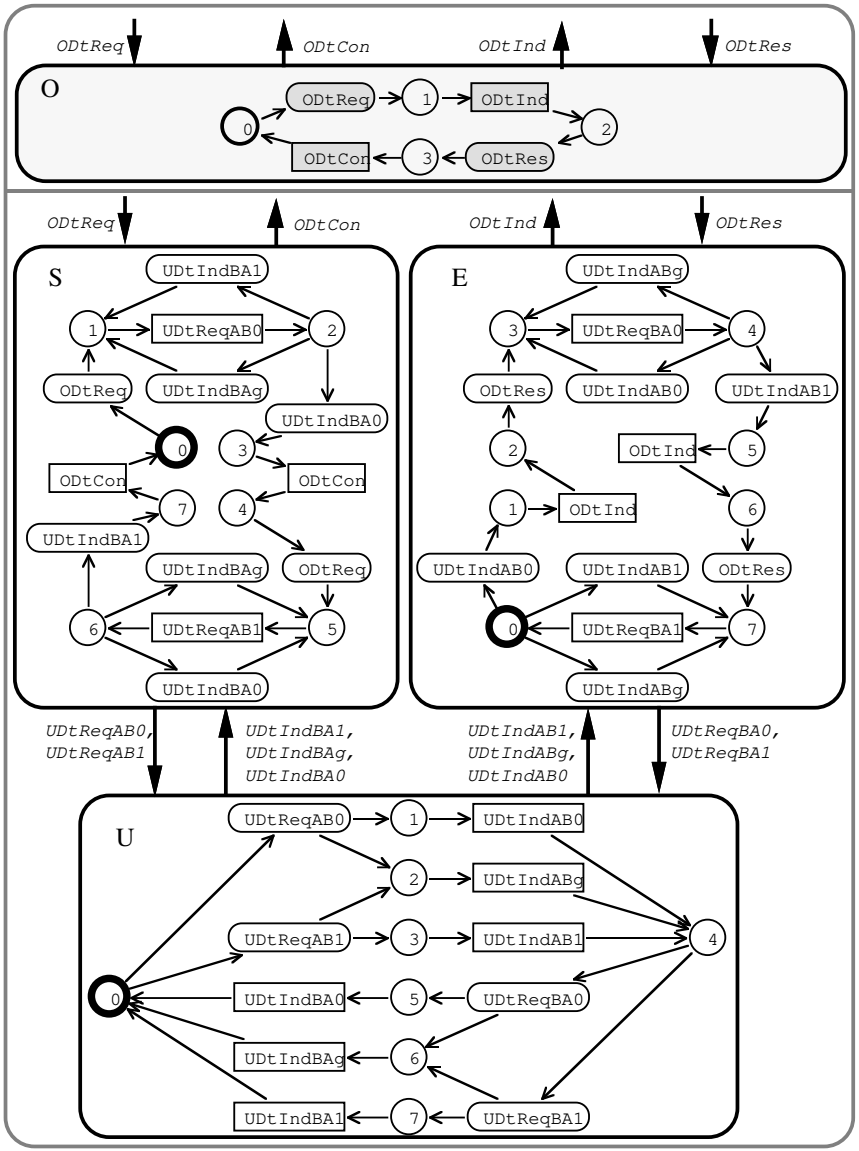

Bild 4.16 Protokollsystem zur Korrektheitsanalyse

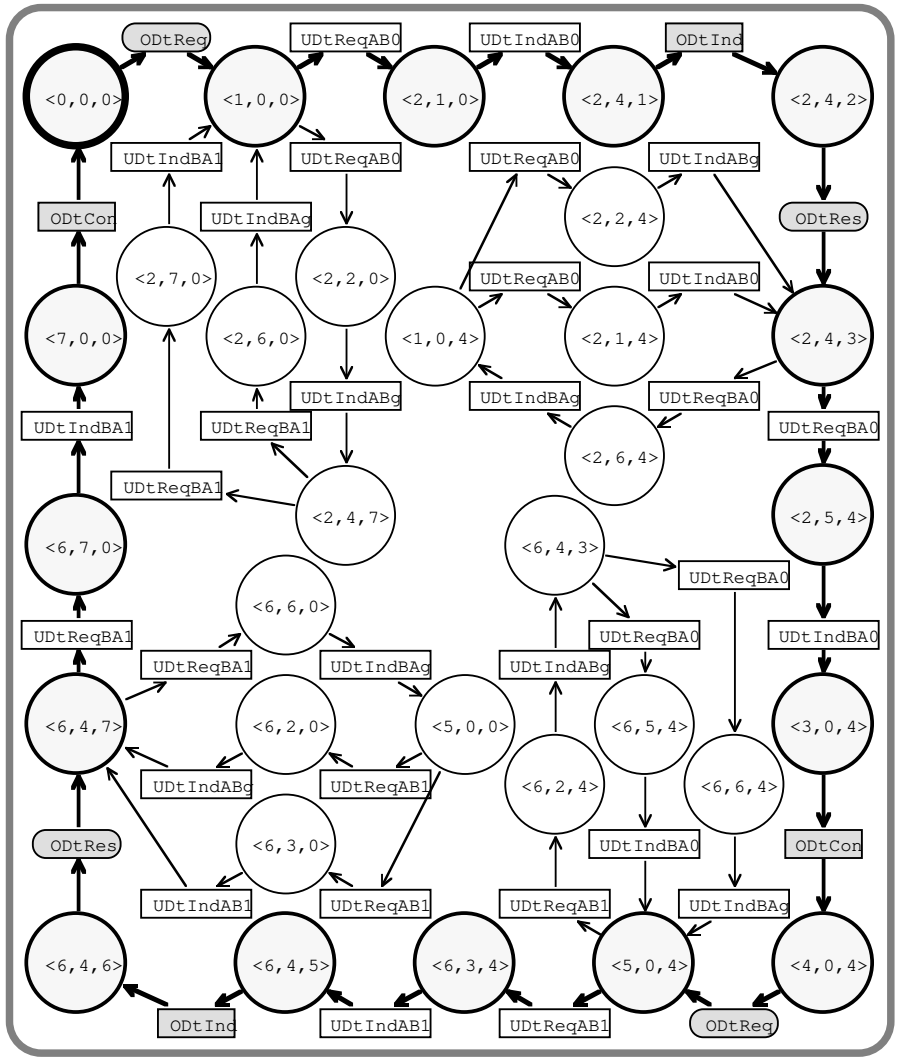

Bild 4.17: Graph der erreichbaren Globalzustände

Wir untersuchen nun beginnend mit dem globalen Startzustand <0,0,0> (Komponente S ist in Zustand 0, Komponente U ist in Zustand 0, ebenfalls Komponente E) die möglichen globalen Zustandsübergänge. Es gibt hier nur eine Möglichkeit: An A_O sendet ein Dienstnehmer ODtReq. Folgezustand ist <1,0,0>. Von <1,0,0> ausgehend gibt es zwei Möglichkeiten: a) S sendet UDtReqAB0, und U empfängt es gleichzeitig mit einem Übergang nach 1; b) S sendet ebenfalls UDtReqAB0, und U empfängt es gleichzeitig mit einem Übergang nach 2. Mögliche Folgezustände von <1,0,0> sind also

<2,1,0> und <2,2,0>. Nun werden die Folgezustände dieser beiden Zustände ermittelt und so weiter. Das Ergebnis dieser Analyse der erreichbaren Globalzustände kann in einem Graphen, dem sogenannten Erreichbarkeitsgraphen, dokumentiert werden. Bild 4.17 zeigt ihn für unser Beispiel.

Wir erkennen im Erreichbarkeitsgraphen den fett gedruckten Hauptzyklus des Protokolls, der abgelaufen wird, wenn keine Störungen in U auftreten. Konzentrieren wir uns auf die Ein- und Ausgaben an der Zieldienstschnittstelle, d.h. auf die (schraffiert grundierten) Ereignistypen ODtReq, ODtInd, ODtRes und ODtCon, dann wird eine gewisse Äquivalenz mit der Zieldienstspezifikation in Bild 4.16 oben ersichtlich. Die dort festgelegte Abfolge der Ereignisse wird eingehalten. Nimmt man an, daß alle Transitionen in endlicher Zeit durchgeführt werden, und betrachtet man nur den Hauptzyklus im Erreichbarkeitsgraphen, dann wird jeder Stimulus nach endlich vielen Transitionen, also nach endlicher Zeit, durch eine Reaktion beantwortet.

Es gibt jedoch Verzweigungen aus dem Hauptzyklus. Vier Seitenschleifen sind zu erkennen. Sie werden betreten, wenn der Basisdienst eine Nachricht verfälscht. Verfälscht er die Nachrichten immer wieder, wird eine solche Seitenschleife nicht verlassen. Es handelt sich dann um eine Quasi-Terminierung. Obwohl Basisdienst und Instanzen fortgesetzt tätig sind, entsteht kein Fortschritt in der Abwicklung der Zieldienstfunktion. An der Zieldienstschnittstelle treten keine Ereignisse mehr auf. Die im vorigen Abschnitt erwähnte Lebendigkeit ist nicht mehr gegeben, kann aber mit der zusätzlichen Annahme, daß der Basisdienst immer wieder Nachrichten auch korrekt überträgt, aufrechterhalten werden.

Korrektheitsaussagen zu Nachrichtenverlust, Reihenfolgevertauschung, Verdopplung und Verfälschung des Zieldienstes können nicht direkt aus diesem Erreichbarkeitsgraphen abgelesen werden, da im zugrundegelegten Modell von den Nachrichten abstrahiert wurde. Aber wir können in einfacher Weise anhand des Erreichbarkeitsgraphen folgende Argumentation z.B. für den Zustand <1,0,0> führen. Die Nachricht m, die mit einem ODtReq an die Instanz S übergeben wurde, wird von S per UDtReqAB0 solange gesendet, bis ein UDtIndAB0 bei E auftritt. Dann wurde sie (per Annahme über U: alle Verfälschungen werden erkannt) korrekt übertragen. E liefert genau m an der Zieldienstschnittstelle per ODtInd aus, Verlust und Verfälschung sind also ausgeschlossen. Ferner erfolgt erst jetzt die relevante Übertragung in Gegenrichtung, so daß Reihenfolgenvertauschungen nicht stattfinden können. Die Verdopplung muß etwas differenzierter betrachtet werden. Denn in Zustand <2,4,3> geht zwar E (in Zustand 3) schon davon aus, daß m an den Dienstnehmer ausgeliefert ist, aber S (in Zustand 2) muß noch damit rechnen, daß m noch nicht korrekt übertragen wurde und liefert u.U. Wiederholungen von m per UDtReqAB0. Wie am Erreichbarkeitsgraphen zu sehen ist, geschieht dies aber nur innerhalb der von <2,4,3> ausgehenden Seitenschleife, in der der Hauptzyklus lediglich verzögert wird und keine Ereignisse der O-Schnittstelle auftreten.

Bild 4.17 zeigt unter diesen Überlegungen den Erreichbarkeitsgraphen eines korrekten Protokolls. Es erbringt den gewünschten Zieldienst in lebendiger Weise. Innerhalb des Entwurfsprozesses eines Protokolls treten dagegen üblicherweise fehlerhafte Zwischenergebnisse auf, die zum großen Teil mit sehr einfachen Kriterien anhand des Erreichbarkeitsgraphen entdeckt werden können. Solche auf Fehler hinweisende Kriterien sind:

- Verzögerung; es existieren Schleifen, die (wie in Bild 4.17 die Seitenschleifen) endlos durchlaufen werden können, ohne daß Zieldienst-Ereignisse auftreten, es führen jedoch auch Wege aus diesen Schleifen heraus in den Hauptzyklus zurück,
- Terminierung; das Protokollsystem erreicht einen Globalzustand, von dem überhaupt keine Übergänge mehr ausgehen,
- Quasi-Terminierung; im Graphen erscheinen Seitenschleifen ohne Zieldienstereignis-Vorkommen (wie bei der Verzögerung), aber es führen keine Wege wieder aus den Schleifen heraus,
- Toter Ereignistyp; ein spezifizierter Ereignistyp tritt im Erreichbarkeitsgraphen gar nicht auf, weil es keinen global erreichbaren Zustand gibt, in dem ein entsprechendes Ereignis erzeugt werden kann,

- Toter Instanzenzustand; ein spezifizierter Instanzenzustand s_i kann unter keinen Umständen eingenommen werden, im Erreichbarkeitsgraphen gibt es keinen Globalzustand der Form $<s_i,x,x>$.

Verzögerung muß nicht unbedingt auf einen Fehler hinweisen. Sie tritt in der Regel bei Protokollen auf, bei denen die Wiederholung von PDUs nicht der Anzahl nach beschränkt wird. Terminierung und Quasi-Terminierung sind dagegen generell unerwünscht, da ein Medium nicht plötzlich die Unterstützung der Dienstnehmer einstellen darf. Die beiden letztgenannten Effekte weisen auf ein grobes Fehlverständnis des Protokollentwerfers hin. Es gibt Instanzenzustände oder Ereignistypen, die bei Systemablauf nie auftreten können, die der Protokollentwerfer aber als notwendig erachtete.

4.3.4.2 Protokoll-Leistungsfähigkeit

Um die Leistungsfähigkeit eines Protokolls zu prüfen, wird untersucht, welche Dienstleistungsqualitäten für die Leistungen des Zieldienstes (im Beispiel O) erreicht werden, wenn die Instanzen protokollgerecht agieren und die Dienstleistungen des benutzten Mediums (im Beispiel U) mit gegebenen Qualitätsparametern zur Verfügung stehen. Im folgenden soll ein (allerdings sehr kurzer) Einblick in Methoden zur Leistungsanalyse anhand unseres Beispielprotokolls 'Alternating Bit' gegeben werden.

gegeben

n	Blocklänge der PDUs (= O-Nutzdatenlänge + 1)
p_{bitf}	Bitfehlerwahrscheinlichkeit
t_{L_U}	Nachrichtenverzögerungszeit in U
f	Bitrate von U

gesucht

t_{T_O}	Transferzeit ODtReq --> ODtInd

Hilfsgrößen

t_{T_U}	Blocktransferzeit in U $= t_{L_U} + n / f$
p_{bitk}	Wahrscheinlichkeit für korrekte Übertragung eines Bits durch U $= 1 - p_{bitf}$
p_{blockk}	Wahrscheinlichkeit für korrekte Übertragung eines Blocks durch U $= (p_{bitk})^n$
p_{blockf}	Wahrscheinlichkeit für Verfälschung eines Blocks bei Übertragung durch U $= 1 - p_{blockk}$

Wahrscheinlichkeitsfunktion für t_{T_O}

$$p(t_{T_O} = (1 + 2*j)*t_{T_U}) = (p_{blockf})^j * p_{blockk} \quad \text{für } j = 0, 1, ..$$
$$= (p_{blockf})^j * (1 - p_{blockf}) \quad \text{geometrische Verteilung}$$

$E(t_{T_O})$ Erwartungswert für Transferzeit ODtReq --> ODtInd

$$= (1 + (p_{blockf} / (1 - p_{blockf})) * 2) * t_{T_U}$$
$$= (1 + ((1 - (1 - p_{bitf})^n) / 1 - (p_{bitf})^n) * 2) * (t_{L_U} + n / f)$$

Bild 4.18: Einfache Leistungsbetrachtung zum Alternating-Bit-Protokoll

Bild 4.18 enthält einen einfachen Ansatz zur groben Abschätzung der Transferzeit des Zieldienstes, die zwischen zusammengehörigen ODtReq- und ODtInd-Ereignissen verstreicht. Es wurden folgende vereinfachende Annahmen getroffen. Die Bitlänge der O-Nutzdaten und damit auch der PDUs ist konstant. Der U-Dienst verfälscht einzelne Bits bei der Übertragung unabhängig voneinander mit gleicher Wahrscheinlichkeit. Die Wahrscheinlichkeit für die korrekte Übertragung einer PDU ist damit ebenfalls konstant und ergibt sich als Potenz der Wahrscheinlichkeit für die korrekte Übertragung eines Bits. Ferner sei die Nachrichtenverzögerungszeit von U konstant und in beide Richtungen gleich. Die Bearbeitungszeiten der Protokollinstanzen S und E seien vernachlässigbar klein.

Der Ansatz ist durch die Annahme der gegenseitigen Unabhängigkeit der Bitstörungen und der konstanten U-Übertragungszeit sehr einfach. Dennoch wird nicht von Störungen und ihren Auswirkungen auf die Leistung abstrahiert. Dies ist für Leistungsbetrachtungen von Telekommunikationsprotokollen typisch, weil hier mit dem Auftreten von Störungen während des praktischen Betriebs gerechnet werden muß, und die im Protokoll vorgesehenen Maßnahmen zur Störungsbehandlung großen Einfluß auf die Qualitätsparameter von Dienstleistungen besitzen.

Ein ODtReq-ODtInd-Transfer kann unter den obigen Annahmen $(1+2j)t_{T_U}$ Zeiteinheiten mit j = 0,1,.. in Anspruch nehmen. Falls die erste Übertragung in Richtung A->B nicht verfälscht wird, ergibt sich eine Dauer von t_{T_U}. Wird die erste Übertragung gestört, folgen eine Quittung und eine Wiederholung, etc. Die Wahrscheinlichkeit dafür, daß eine O-Übertragung $(1+2j)t_{T_U}$ Zeiteinheiten in Anspruch nimmt, ergibt sich aus dem Produkt der Wahrscheinlichkeiten für die gestörte Übertragung von j PDUs in Richtung AB und einer ungestörten Übertragung in dieselbe Richtung. Die Übertragung der Quittung-PDUs in Gegenrichtung muß nicht beachtet werden, da eine Verfälschung der Quittungen keinen Einfluß auf den Folgeablauf besitzt. Es ergibt sich damit die Wahrscheinlichkeitsfunktion einer geometrischen Verteilung 'p(X=i) = $p^i*(1-p)$ für i=0,1,..' mit dem Erwartungswert E(X)=$p/(1-p)$. Der Mittelwert der ODtReq-ODtInd-Übertragungszeit entspricht also der in Bild 4.18 unten angegebenen Funktion über den Parametern n, p_{bitf}, t_{L_U} und f.

Als Faktoren gehen in diese Funktion die U-Blocktransferzeit t_{T_U} und ein nur noch von n und p_{bitf} aber von diesen sehr stark abhängiger Wert ein. Bild 4.19 zeigt ein Diagramm der mittleren O-Übertragungszeit für unterschiedliche Bitfehlerwahrscheinlichkeiten und PDU-Längen (Die Parameter t_{L_U}=10µs, dies entspricht ca. 1 km bei 1/3 Lichtgeschwindigkeit, und f=10 kHz wurden fest gewählt). Wie zu sehen ist, darf bei größeren Bitfehlerwahrscheinlichkeiten des benutzten Mediums die O-Nutzdatenlänge nicht zu groß gewählt werden, wenn akzeptable O-Übertragungszeiten erreicht werden sollen.

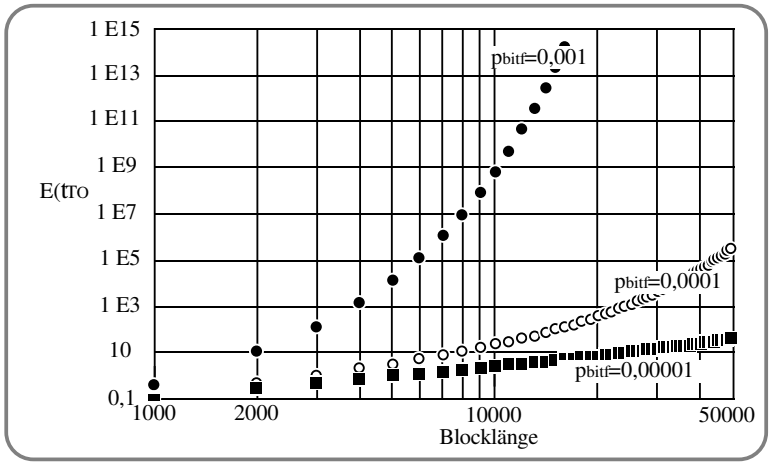

Bild 4.19: Mittlere Ubertragungslaufzeit

Eine längere PDU überträgt natürlich mehr O-Nutzdaten und darf von daher gesehen auch eine längere Übertragungszeit in Anspruch nehmen. Um dies einzubeziehen, nehmen wir an, es seien über den O-Dienst (bei einer Bitfehlerwahrscheinlichkeit von 10^{-5} des U-Dienstes) 1 Mbit Daten zu übertragen. Diese O-Daten werden segmentiert, um kürzere PDU-Blöcke zu erzielen. Bild 4.20 zeigt (bei Vernachlässigung der Zeiten für zwischen Segmenten liegende ODtRes-ODtCon-Transfers) die mittelere Übertragungszeit in Abhängigkeit von der PDU-Länge. Auch hier ergibt sich ein überproportionales Anwachsen der Übertragungszeit mit der Blocklänge.

Eine Folgerung könnte sein, die Segmentlängen möglichst kurz zu wählen. Bei sehr kleinen PDUs dominieren dann aber die (nicht modellierten) je PDU anfallenden Bearbeitungszeiten der Protokollinstanzen S und E sowie die Nachrichtenverzögerungszeit in U und der zusätzlich je PDU anfallende Kontrollparameter-Anteil. Bild 4.21 zeigt dazu die Kurve aus Bild 4.20 bei kleinen Segmentlängen. Wie man sieht, verschiebt sich die optimale Segmentlänge bei wachsender Nachrichtenverzögerungszeit von U (wachsender Entfernung zwischen A und B) in Richtung auf höhere Segmentlängen.

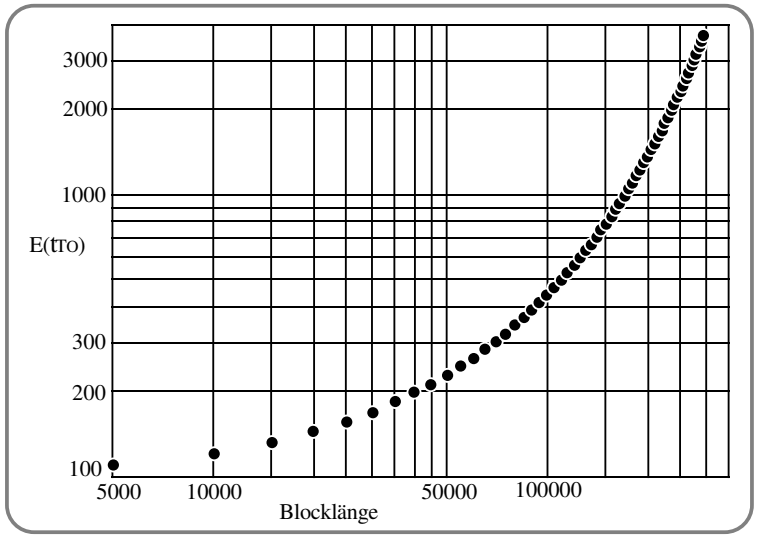

Bild 4.20: Zeit zur Ubertragung von 1 MBit O-Nutzdaten

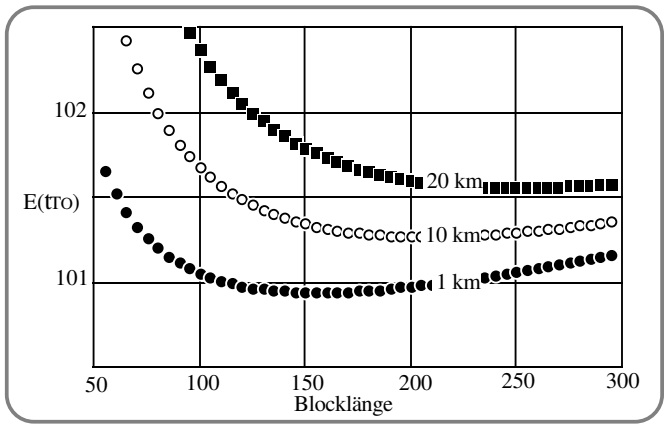

Bild 4.21: Zeit zur Übertragung von 1 MBit O-Nutzdaten

Beim betrachteten Alternating Bit Protokoll zeigen die Kurven im Bereich des Optimums einen ziemlich 'gutmütigen' flachen Verlauf. Auch war die grobe Leistungsbetrachtung noch mit einfachen Mitteln durchzuführen. Dies ändert sich aber mit einer verfeinerten Betrachtung, z.B. wenn nicht mehr von konstanten PDU-Laufzeiten in U ausgegangen werden kann, oder die Störung einzelner Bits in U nicht mehr unabhängig voneinander erfolgt.

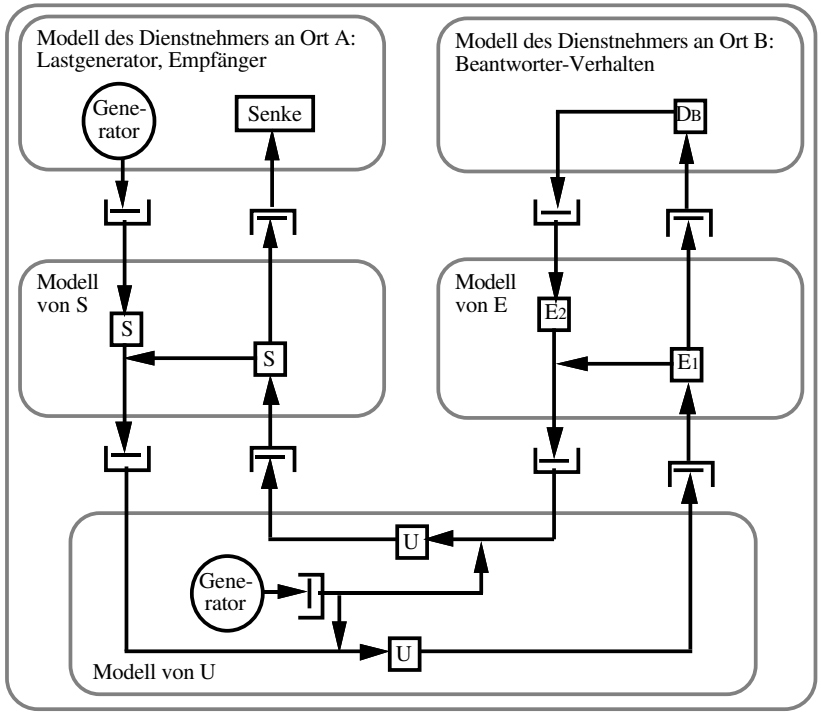

Bild 4.22: Skizze eines Warteschlangen-Bedienstationen-Netzes

Zur detaillierteren Leistungsanalyse werden auf der Warteschlangen-Theorie aufbauende komplexere analytische Berechnungen, Simulationsmodell-Läufe am Rechner und / oder Vermessungen realisierter Systeme durchgeführt. Im folgenden soll kurz darauf eingegangen werden.

Bild 4.22 zeigt dazu die Skizze eines Netzes aus Warteschlangen und Bedienstationen, das zur Leistungsbewertung unseres Protokollbeispiels 'Alternating Bit Protokoll' herangezogen werden könnte. Die verschiedenen Bedienstationen D_B, S_1, S_2, E_1, E_2, U_1und U_2 modellieren das Ablaufverhalten der Komponenten des Protokollsystems. Eine Bedienstation entnimmt dabei zunächst einen Auftrag aus einer Warteschlange, dann benötigt sie eine gewisse Bedienzeit zur Bearbeitung des Auftrags, und erzeugt ihrerseits Folgeaufträge, um dann wiederum den nächsten Auftrag - falls schon vorhanden - aus einer Eingabe-Warteschlange zu entnehmen. Warteschlangen dienen als Puffer anstehender Aufträge, Generatoren erzeugen neue Aufträge und Senken entfernen Aufträge aus dem System.

Das Modell des Dienstnehmers an Ort A umfaßt einen Lastgenerator, d.h. eine Quelle für ODtReq-Aufträge, ferner eine Senke zur Entgegennahme der Fertigmeldungen ODtCon. Das Modell des Dienstnehmers an Ort B besteht aus einer Bedienstation D_B, sie nimmt anstehende ODtInd-Aufträge entgegen und erzeugt nach einer Dienstnehmer-Bearbeitungszeit die entsprechenden ODtRes-Aufträge. Die Modell der beiden Protokollinstanzen S und E bestehen jeweils aus zwei Bedienstationen.

Eine modelliert die Bearbeitung vom Basisdienst kommender UDtInd-Aufträge, sie erzeugt (nach Protokollvorschrift) entweder-OReaktionen oder Aufträge zur negativen Quittungsübertragung an U. Die zweite Bedienstation eines Instanzenmodells nimmt O-Stimuli entgegen und überführt sie U-Aufträge. Das Modell von U sieht für jede Transferrichtung eine Bearbeitungsstation vor. Zusätzlich ist ein Generator enthalten, der Aufträge zur Störung der nächsten anstehenden Übertragung an die beiden Transfer-Bedienstationen liefert.

Zur Aufschlüsselung der relevanten Modellparameter muß unterschieden werden, ob das Leistungsverhalten des 'Protokolls an sich' modelliert werden soll - hierbei wird dann von Implementierungsaspekten abstrahiert, man nimmt eine idealisierte Implementierung der Protokollinstanzen, Dienstschnittstellen und Dienstnehmer an -, oder ob das Protokoll in Verbindung mit einer bestimmten Art und Weise der Implementierung bewertet werden soll.

Im ersten Fall, der reinen Protokollbewertung, besitzen die Bedienstationen der Protokollinstanzen vernachlässigbare Bedienzeiten. Wichtig sind hier nur die der Protokollvorschrift entsprechende Erzeugung und die Weitergabe von Folgeaufträgen. Genauer spezifiziert werden müssen das benutzte Medium U und die Systemlast an der O-Schnittstelle. Im Modell des Basisdienstes U muß der vom Generator erzeugte Störauftragsstrom in seiner Wahrscheinlichkeitsverteilung angegeben werden. Ferner sind die Bedienzeiten der Transfer-Stationen U_1 und U_2 relevant. Sie sind auch von der Bitlänge der zu übertragenden Daten abhängig, wenn eine PDU-Übertragung insgesamt als ein einziger Auftrag behandelt wird. Alternative dazu ist es, die Übertragung einzelner Bits in separate Aufträge abzubilden. Die Systemlast an der O-Schnittstelle wird durch den vom Generator erzeugten OReq-Auftragsstrom, d.h. durch die Zeitperiode zwischen zwei aufeinanderfolgenden Aufträgen, die Bitlänge der Nutzdatenparameter und die Bitlänge der Bestätigungsnutzdaten bestimmt. Die Bedienzeiten von D_B und der ODtCon-Senke werden vernachlässigt. Üblicherweise werden zur Bewertung eines Protokolls verschiedene Lastprofile herangezogen, z.B. ein die Anforderungen des vermuteten Normalbetriebs modellierendes Profil und verschiedene Profile zur Untersuchung der Protokolleigenschaften in Überlastsituationen.

Im zweiten Fall, der für die Praxis wichtigen Bewertung des Verbunds aus Protokoll und Implementierung, muß das Modell wesentlich verfeinert werden. Wichtig wird jetzt auch der Aufwand, der in einzelnen Rechnerknoten bezüglich Speicher- und Prozessorzeitbedarf durch die Abwicklung der Protokollfunktionen und die Realisierung der Dienstzugangspunkte entsteht. Dieser Aufwand ist dabei üblicherweise nicht allein von den im Protokollsystem betrachteten Aufträgen, bzw. deren Parametern, abhängig. Prozessor und Speicher sind als konkurrierend genutzte Betriebsmittel zu betrachten, die ferner auch noch von außerhalb des Modells liegenden Aufträgen (z.B. sonstige Rechenprozesse auf einem Universalrechner, Instanzen anderer Schichten) beansprucht werden. Um hier aussagekräftige Ergebnisse zu erzielen, ist oft eine sehr detaillierte Modellierung der speziell gegebenen Basismaschine (Rechner-Hardware, Betriebssystem) erforderlich.

Zur Analyse komplexer Leistungsmodelle kann Simulation herangezogen werden. Das Warteschlangen-Bedienstationen-Netz wird in einer auf dem Rechner ausführbaren Form definiert (z.B. durch ein Programm oder durch entsprechende Parametrisierung eines Simulationsprogramms) und ausgeführt. Die Ausführung wird beobachtet (durch Erzeugung von Statistiken und / oder Journaldateien über die Historie relevanter Ereignisse) und die Beobachtungsergebnisse werden ausgewertet. Problematik beim simulativen Vorgehen ist, daß, um verläßliche Ergebnisse zu erzielen, bei komplexen Modellen sehr lange Rechenzeiten benötigt werden.

Die analytische Vorgehensweise zielt darauf ab, direkt aus der Modellbeschreibung mit mathematischen Mitteln Formeln für die interessierenden Systemgrößen (z.B. Mittelwert und Streuung der Verweilzeit von Zieldienstaufträgen im System, Nutzungsgrad des unterliegenden Mediums) abzuleiten. Hier liegt die Problematik darin, daß sowohl exakte Lösungsverfahren als auch Algorithmen für numerische Lösungen nicht bei beliebigen Systemen verwendet werden können (Exakte Lösungen

stehen für bestimmte Systemklassen, z.B. sich wechselseitig synchronisierende Komponenten, nicht zur Verfügung, numerische Verfahren sind ab einer bestimmten Systemgröße aus Rechenzeitgründen nicht mehr anwendbar). Wichtig ist es deshalb, das eigentlich zu untersuchende System unter vereinfachenden Annahmen in ein behandelbares System zu überführen.

4.3.4.3 Realitätsnähe des Beispiels

Bei der Erläuterung der Dienste U und O wurde auf ihre Realitätsferne hingewiesen. Um das Beispiel in Richtung auf praxisgerechte Dienste und Protokolle zu erweitern, wären folgende Modifikationen notwendig:

- Es ist sowohl bei U als auch bei O zu berücksichtigen, daß die Verfälschungsfreiheit sowie das Erkennen aller Verfälschungen nie garantiert werden kann. Allerdings geht man bei der Prüfung der Fragestellung der Korrektheit eines Protokolls auch bei realen Protokollen davon aus, daß durch Codiermaßnahmen alle aufgetretenen Verfälschungen erkennbar sind. Protokolle können i.a. nur unter dieser Annahme als korrekt bewiesen werden.
- Sowohl U als auch O können über gewisse Zeitperioden unbekannter Länge (auch ∞) in jedem Zustand ausfallen. Zum Erkennen von langen Ausfällen von U müssen in den Instanzen S und E zusätzliche Maßnahmen getroffen werden. Lange U-Ausfälle führen dazu, daß der Dienst O ebenfalls nicht mehr unterstützt werden kann. Die Instanzen S und E sollten das ihren Dienstnehmern mit einer Abbruch-Anzeige mitteilen.
- Aufgrund kurzfristiger Ausfälle kann ein dem physikalischen Medium naher Dienst, wie U, in der Regel Nachrichten verlieren. Im Beispielprotokoll führt der Verlust einer Nachricht zum Stillstand des Systems. Um dies zu vermeiden, sind durch Zeitüberwachungsmaßnahmen initiierte Nachrichtenwiederholungen bei den Instanzen vorzusehen. Die Erkennung langer Ausfälle kann davon ausgehend dadurch stattfinden, daß die Anzahl aufeinanderfolgender erfolgloser Wiederholungen mitgezählt wird. Bei Erreichen eines vorher festgelegten Limits wird von einer langen Unterbrechung ausgegangen, die den O-Dienstnehmern nicht mehr verborgen werden kann.

Diese Modifikationen, insbesondere die Berücksichtigung von Nachrichtenverlusten und die durch Zeitüberwachung gesteuerte Wiederholung von Nachrichten führen auch zu einem stark veränderten Leistungsverhalten.

5. Logische Architektur von Telekommunikationssystemen

Telekommunikationssysteme besitzen einen hierarchisch strukturierten schichtenweisen logischen Aufbau. Die beiden zentralen Begriffe zur Beschreibung einer Schicht, Telekommunikationsdienst und Telekommunikationsprotokoll, wurden schon im allgemeinen vorgestellt. Um die Grundzüge der Gesamtarchitektur eines Telekommunikationssystems weiter zu präzisieren, soll nun näher auf die Aufteilung der Systemfunktion in Teilfunktionen und ihre Zuordnung zu Schichten und Protokollinstanzen eingegangen werden.

Interessant ist, welche Funktionalität die Dienste einer Diensthierarchie besitzen, und welche Qualitätsanforderungen an sie gestellt werden. Dabei sollen die Dienste sich schrittweise ausgehend von physikalischen Medien demjenigen abstrakten Medium nähern, das eine Menge von Anwendungsprozessen zur wechselseitigen Kommunikation benötigt. Mit den Ausführungen hierzu soll der Entwurf des ISO / OSI-Modells nachvollzogen werden.

5.1 Grobe Aufgabenaufteilung der Schichten

Die hierarchische Gliederung eines Systems in aufeinanderliegende Schichten soll einerseits allgemeinen Richtlinien zum Erzielen einer klaren, übersichtlichen und eingängigen Systemstruktur folgen:

- Jeder Schicht soll eine logisch in sich geschlossene Funktionalität zugeordnet werden.
- Die Schnittstellen zwischen zwei Schichten sollen überschaubar und wenigen Konzepten folgend ausgelegt sein.
- Die nach oben angebotene Funktionalität einer Schicht soll abstrahierend von der schichtinternen Realisierung definiert werden können.

Diesen Prinzipien kommt die bereits erklärte Sichtweise von Diensten, Protokollen und insbesondere das Konzept der Dienstschnittstelle bereits entgegen. Darüberhinaus soll ein Einfachheitsprinzip gelten: Sowenig Schichten wie möglich.

Die Vorteile der hierarchisch geschichteten Struktur, das Vorhandensein isoliert betrachtbarer Subsysteme, bei entsprechender Implementierung die Wiederverwendbarkeit von Subsystemen in neuen Projekten und die enge Eingrenzbarkeit von Änderungen bei Anpassungen an veränderte Problemstellungen oder technische Randbedingungen, hängen weiterhin davon ab, daß eine hierfür geeignete hierarchische Funktionsaufteilung gefunden wird. Mit steigender Schicht soll die Abhängigkeit von technischen Randbedingungen (von benutzten Rechnern, von den Datenübertragungseinrichtungen und der Netztopologie) strikt abnehmen, während umgekehrt das Maß an Spezialisierung der Dienste auf bestimmte Anwendungsklassen in Richtung zu tieferen Schichten hin abnehmen soll. Diese beiden Aspekte in der Auslegung von Schichten nennen wir Transportorientierung und Anwendungsorientierung.

Gegebene Eckpunkte der Schichtenhierarchie sind zuunterst die unzuverlässige Übertragung digitaler Daten über physikalische Medien und zuoberst Dienstleistungen eines Anwendungsdienstes, die direkt die Kommunikation zwischen den Prozessen eines bestimmten Anwendungstyps unterstützen.

Während die unterste Schicht ausschließlich von technischen Gegebenheiten geprägt wird, sollte ein Anwendungsdienst in seiner Auslegung nur von der Art der zu unterstützenden Anwendung abhängen. Ganz unten in der Hierarchie befindet sich also eine völlig vom physikalischen Medium geprägte transportorientierte Schicht, ganz oben eine völlig auf die Anwendungstypen zugeschnittene anwendungsorientierte Schicht.

Nun gibt es eine Vielfalt von physikalischen Medien (die in Zukunft bedingt durch den technischen Fortschritt auch weiter wachsen wird). Und andererseits gibt es ebenfalls eine Vielfalt von Anwendungstypen (die ebenfalls mit sinkenden Rechnerkosten und steigender Rechnernutzung weiter wachsen wird). Zwischen beiden Spektren bestehen nur indirekte Abhängigkeiten (über die Kosten bzw. benötigte und zur Verfügung stellbare Datenübertragungsleistung). Deshalb ist es sinnvoll, transportorientierte und anwendungsorientierte Schichten strikt zu trennen.

Es wird eine Trennungslinie definiert, die die unteren transportorientierten Schichten von den oberen anwendungsorientierten Schichten separiert. Diese Trennungslinie sollte - wenn wir das Gesamtsystem schon vertikal in Schichten gliedern - an einer Schichtgrenze liegen und so mit einem Dienst zusammenfallen. Er wird Transportdienst genannt, und alle unterhalb der Transportdienst-Schnittstelle liegenden Schichten bilden das sogenannte Transportsystem.

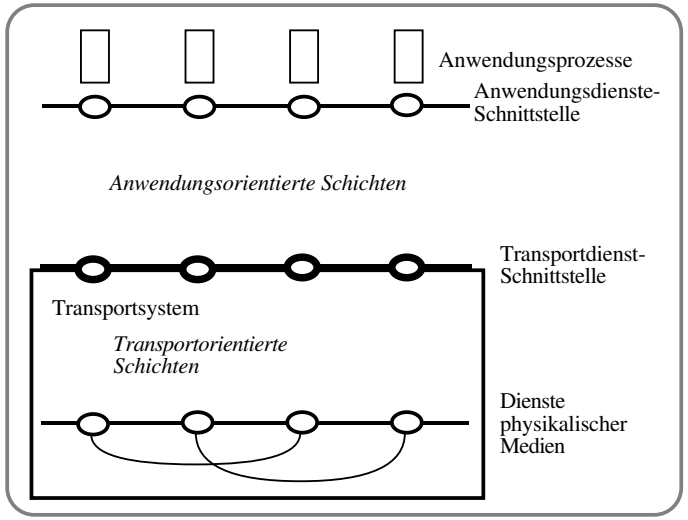

Bild 5.1: Grobe Gliederung in 2 Schichtklassen

Bild 5.1 gibt eine Darstellung der Ergebnisse der bisherigen Strukturüberlegungen. Man beachte auch die Singular- und Plural-Wahl bei der Beschriftung. Idealerweise gibt es einen einzigen Transportdienst als Schnittstelle zu einem einzigen Transportsystem. Dagegen wird mit zahlreichen physikalischen Medien und von diesen direkt angebotenen Diensten sowie mit zahlreichen Anwendungen, Anwendungstypen und dafür benötigten Anwendungsdiensten zu rechnen sein.

5.2 Transportsystem

Das Transportsystem soll unabhängig von der Art der zu unterstützenden Anwendungen sein. Mit dem Transportdienst soll ein universell standardmäßig für alle denkbaren Anwendungstypen einsetzbarer Kommunikationsdienst zur Verfügung gestellt werden. Er soll also noch keine anwendungsbezogene Aspekte widerspiegeln.

Andererseits soll das Niveau des Transportdienstes so hoch sein, daß alle technischen Aspekte der Datenübertragung über bestimmte physikalische Medien bzw. Netze mit bestimmter Topologie verdeckt werden, so daß die anwendungsorientierten Schichten frei von Transportaspekten ausgelegt werden können.

5.2.1 Transportdienst

Direkte Dienstnehmer des Transportsystems sind die Instanzen der nächsthöheren, der ersten anwendungsorientierten Schicht. Allerdings ist zu berücksichtigen, daß diese Instanzen letztendlich für Anwendungsprozesse tätig werden. Abstrahieren wir zunächst von diesem Zwischenbau zwischen Transportsystem und Anwendungsprozessen, so hat der Transportdienst den Datenaustausch zwischen Anwendungsprozessen zu unterstützen: Die Endpunkte der vom Transportdienst erbrachten Kommunikationsleistungen sind die Anwendungsprozesse selbst. Dies nennt man den Ende-zu-Ende-Aspekt des Transportdienstes. Er hat Konsequenzen für die Gliederung der Transportdienstschnittstelle in Zugangspunkte, bzw. für die Vergabe von Zugangspunktadressen. Für jeden Anwendungsprozeß (besser sogar für jede individuelle Rolle in der ein Anwendungsprozeß auftreten kann) soll ein eigener dafür exklusiv reservierter Dienstzugangspunkt existieren. Die Adresse dieses Zugangspunkts ist dann sozusagen die private Telefonnummer des Anwendungsprozesses.

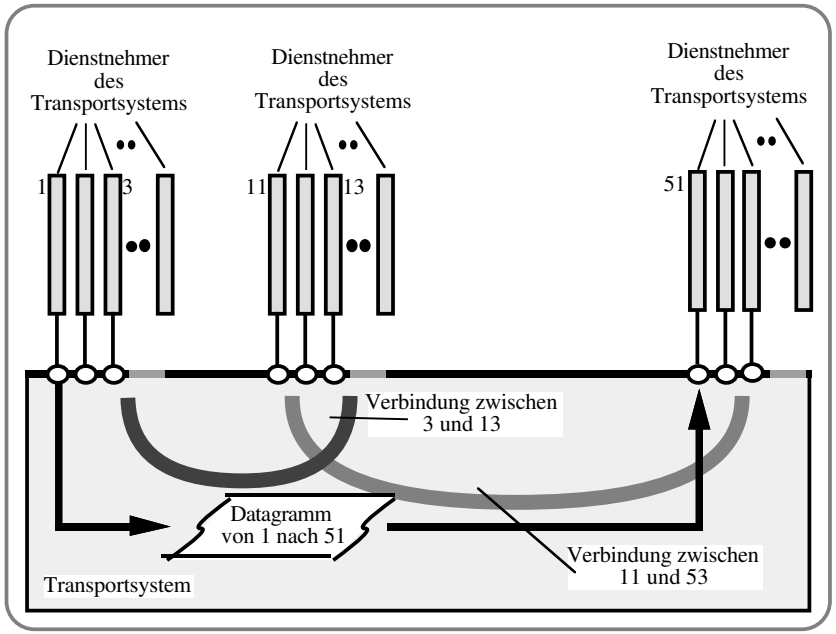

Bild 5.2: Transportsystem und Nutzung

Diese Ende-zu-Ende-Adressierung ist eine wichtige Funktionalität des Transportdienstes. Sie erlaubt es, daß Anwendungsprozesse, die die Transportadresse eines gewünschten Partner-Anwenderprozesses kennen, diesen gegenüber dem Transportsystem adressieren können. Dies ist noch unabhängig vom Typ einer Anwendung, aber bereits so nah an den Kommunikationsanforderungen von Anwendungen im allgemeinen, daß bei allen später zu besprechenden anwendungsorientierten Diensten nur noch eine 1:1-Abbildung zwischen Transportadresse und Zugangspunktadresse des anwendungsorientierten Dienstes besteht (Die Fragestellung 'Wie findet man die Telefonnummer eines gewünschten Partners' geht über die reine Kommunikation zwischen den Partnern hinaus. Man braucht dazu spezielle Anwendungen, die im Rechnernetz z.B. den Zugriff auf 'weiße oder gelbe Telefonbücher' erlauben. Der Zugriff auf entfernte Verzeichnisse ist dann wiederum Sache eines eigenen Anwendungsdienst-Typs 'Verzeichnisdienst / Directory Service').

Eine weitere von Anwendungen unabhängig von ihrem Anwendungstyp gewünschte Fähigkeit ist es, die Kommunikation selbst stattfinden zu lassen. Dazu müssen binäre Daten als Bitmuster variabler Länge ausgetauscht werden können. Die binären Daten sind in Aufbau und Bedeutung aus der Sicht des Transportsystems nicht weiter reglementiert, um die Anwendungsunabhängigkeit nicht zu stören.

Als Grundform des Datenaustauschs wird zur Zeit die Zwei-Partner-Kommunikation (1 Sender, 1 Empfänger) per unbestätigten Datentransfer-Dienstleistungen als elementar angesehen. Davon ausgehend gibt es zwei Varianten des Transportdienstes, den verbindungsorientierten und den Datagramm-Transportdienst (Es sind aber bereits Transportdienste in Diskussion, die Mehrparteien-Kommunikation unterstützen).

Der Datagramm-Transportdienst unterstützt nur den einen Dienstleistungstyp Datagramm-Transfer.

Der verbindungsorientierte Transportdienst sieht Dienstleistungen zum Auf- und Abbau eines Transportkontextes, der Transportverbindung, zwischen zwei Anwenderprozessen vor. Innerhalb einer bestehenden Verbindung können die Datentransfer-Dienstleistungen in Anspruch genommen werden.

Bild 5.2 skizziert ein nach diesen Festlegungen grob in Dienstnehmer und Transportsystem gegliedertes Telekommunikationssystem mit vom Transportdienst für Anwendungsprozesse angebotenen Dienstleistungen.

Zur weitergehenden Charakterisierung des Transportdienstes ist der Qualitätsaspekt zentral.

Physikalische Medien, bzw. die darüber gebildeten nachrichtentechnischen Kanäle, besitzen Leistungseigenschaften, die unmittelbar von ihrem physikalischen Aufbau und physikalischen (Stör-) Einflüssen darauf abhängen, und es ist zu erwarten, daß die Qualität der Transportverbindungen oder Transport-Datagramm-Übertragung auch davon mitgeprägt wird. Besteht z.B. zwischen zwei Orten A und B nur eine einzige langsame physikalische Verbindungsmöglichkeit, so ist eine Transportverbindung zwischen einem Anwenderprozeß in A zu einem Prozeß in B nicht möglich, die eine höhere Übertragungsleistung aufweist. Es bestehen also bei manchen Qualitätseigenschaften Abhängigkeiten zur konkreten technischen Auslegung des Transportsystems.

Ein anwendungsunabhängiger Transportdienst sollte andererseits in der Lage sein, Dienstleistungen in beliebiger Qualität anzubieten, wenn er universell verschiedenste Anwendungen unterstützen will (z.B. langsame Elektronische Post zu günstigen Gebühren, Video-Konferenzen mit überaus hohen Anforderungen an Übertragungsleistung und geringe Verzögerungszeiten). Es ist zu erwarten, daß viele Anwendungen nur dann sinnvoll realisierbar (zufriedenstellende Erfüllung des Systemzwecks, vertretbare Kosten) sind, wenn der Transportdienst den Anwendungsprozessen Dienstleistungen mit einem von der Anwendung frei bestimmten Qualitätsprofil zur Verfügung stellen kann.

Der Konflikt zwischen Anwendungsanforderungen an Technikunabhängigkeit und Technikabhängigkeit ist offensichtlich. Im Transportdienst-Konzept wird zur Lösung die Qualitätsverhandlung eingesetzt:

- Ein Anwenderprozeß, der eine Dienstleistung anfordert (insbesondere Verbindungsaufbau), übergibt neben Parameteren zur näheren Beschreibung der Funktionalität (z.B. Partneradresse) auch Parameter zur Beschreibung der gewünschten Qualitätseigenschaften (sogenannte QoS-Parameter, Quality of Service). Er bringt dadurch gegenüber dem Transportdienst seine Qualitätsanforderungen (und zwar in einer technikunabhängigen Weise) zum Ausdruck. Der Transportdiensterbringer muß nun (unter Kenntnis der speziellen Leistungsfähigkeit der zur Verfügung stehenden Mittel und unter Kenntnis der momentanen Betriebssituation) entscheiden, ob er diese Anforderungen erfüllen kann, oder ob er Abstriche in Richtung sinkender Qualität vornehmen muß. Die u.U. durch Vermindern veränderten Qualitätsparameter werden der Anwendung wieder mitgeteilt. Der Anwendungsprozeß kann nun entscheiden, ob er die Verbindung mit der u.U. gegenüber seinen ursprünglichen Anforderungen reduzierten Qualität nutzen oder weil nicht verwertbar sofort abbauen will.

5.2.2 Aufbau des Transportsystems

Mit der Zielvorgabe der Transportdienst-Erbringung und der Realisierungsanforderung, nachrichtentechnische Kanäle zu verwenden, liegen die Randbedingungen für die interne Architektur des Transportsystems fest. Die oberhalb der nachrichtentechnischen Kanäle liegenden Schichten des Transportsystems haben die 'funktionellen und qualitativen Unterschiede' zwischen dem Dienstleistungsangebot der verfügbaren Menge nachrichtentechnischer Kanäle und dem gewünschten Transportdienst zu füllen, wobei die genannten Richtlinien für wohlstrukturierte Systeme eingehalten werden sollen.

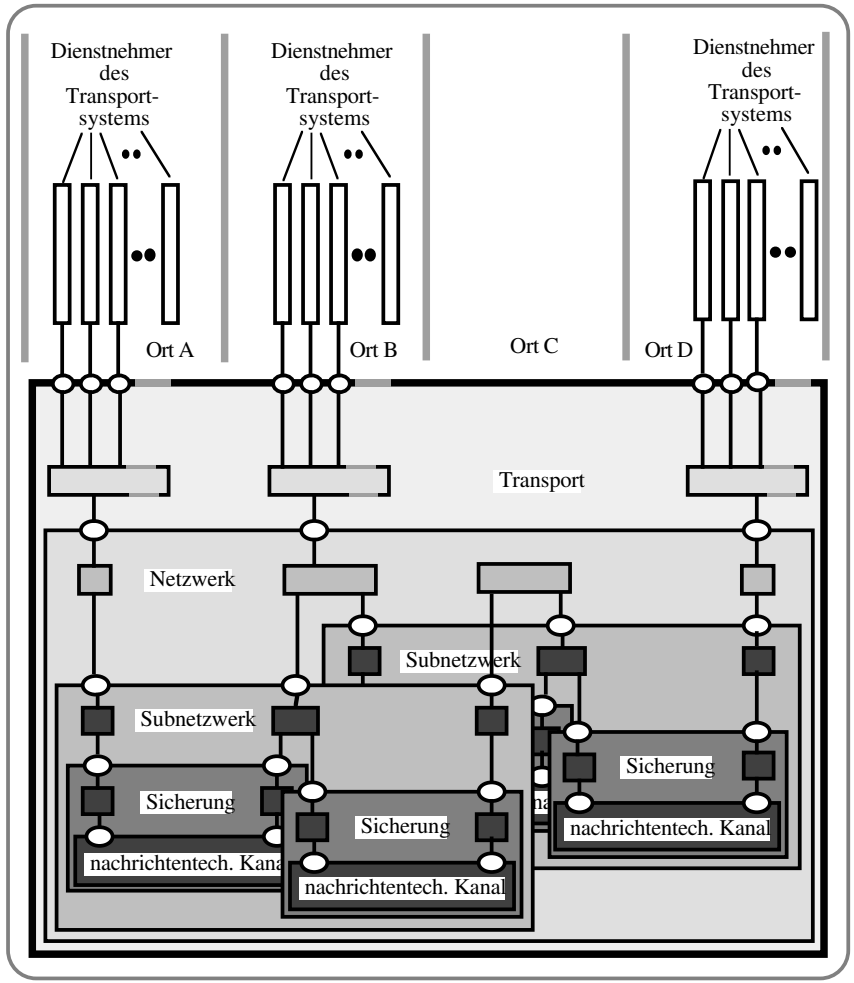

Bild 5.3: Transportsystem

Die unterste betrachtete Schicht, sie enthält die nachrichtentechnischen Kanäle, wird mit der Schichtnummer 1 (in ISO/OSI-Terminologie: Physical Layer, 1-Layer) belegt. Im Vergleich zum geforderten Transportdienst bestehen folgende wesentlichen Unterschiede:

- Unzuverlässigkeit; die Wahrscheinlichkeit des Auftretens von Störungen überschreitet i.a. die von den Dienstnehmern des Transportsystems gewünschte Qualität wesentlich.
- Flexibilität der Dienstqualitäten; die Qualitäten der angebotenen Dienstleistungen hängen direkt von den Eigenschaften des physikalischen Mediums ab und können von den Dienstnehmern durch Parametrisierung kaum beeinflußt werden.

- Nachrichtenpaket-Längen; die Längen der übertragbaren Nachrichtenpakete können oft nur in sehr engen - von der Realisierung des nachrichtentechnischen Kanals bestimmten - Grenzen variieren.
- Synchronisation; der erfolgreiche Betrieb eines nachrichtentechnischen Kanals ist oft nur möglich, wenn die Dienstleistungsaufträge unter Einhaltung realisierungsbedingter zeitlicher Randbedingungen abgegeben werden. So besitzt auch ein nachrichtentechnischer Kanal i.a. keine FIFO-Pufferungseigenschaften, und ein Empfänger muß ankommende Nachrichten unverzüglich entgegennehmen um Verlust zu vermeiden.
- Konnektivität; ein nachrichtentechnischer Kanal verbindet i.a. nur eine Teilmenge der Knoten. Handelt es sich z.B. nur um Kanäle über Zwei-Punkt-Leitungen, so bietet jeder nachrichtentechnische Kanal nur Dienstleistungen zum Nachrichtentransfer zwischen zwei statisch festgelegten Orten an. Austausch zwischen zwei beliebigen Orten ist i.a. nur unter geeigneter Verwendung mehrerer Kanäle möglich.
- Adressierung; die Organisation der Dienstschnittstelle eines Kanals und insbesondere die Zu–ordnung von Zugangspunktadressen ist i.a. unabhängig von der der anderen Kanäle. Damit existiert kein einheitlicher, alle Knoten des Rechnernetzes umfassender Adreßraum, und erst recht keiner, in dem alle potentiellen Dienstnehmer des Transportsystems (i.a. mehrere je Knoten) erfaßt sind.
- Mehrfachnutzung; auch wenn z.B. die beiden Orte zweier Dienstnehmer des Transportsystems durch einen nachrichtentechnischen Kanal direkt verbunden sind, besitzt der Kanal noch nicht die Fähigkeit, mehrere verschiedene Verbindungen gleichzeitig zu unterstützen, wie das vom Transportdienst erwartet wird.

Im Sinne einer möglichst geringen Schichtanzahl sollte die darüberliegende Schicht möglichst viele Unterschiede bewältigen. Im Sinne einer Abgrenzung unabhängiger und wiederverwendbarer Subsysteme sollten aber nur solche Aufgaben der nächsten Schicht (Schichtnummer 2, Sicherungsschicht, Link Layer) zugeordnet werden, die in einem engen in sich geschlossenen Zusammenhang lösbar sind. Die Schicht 1 gliedert sich in mehrere voneinander unabhängige Kanäle, und es ist deshalb sinnvoll, in der Schicht 2 ebenfalls voneinander unabhängige Medien vorzusehen, die jeweils einem einzigen nachrichtentechnischen Kanal zugeordnet sind. In diesem engen Kontext können an Aufgaben im wesentlichen nur bewältigt werden:

- die Koordination des Zugangs zum Kanal,
- die Hebung der Zuverlässigkeit.

In der Sicherungsschicht wird je nachrichtentechnischem Kanal separat ein eigenständiges abstraktes Medium 'gesicherter Kanal' gebildet. Die Dienstleistungen eines gesicherten Kanals stimmen deshalb bezüglich Konnektivität, Adressierung und Mehrfachnutzung i.a. mit dem des nachrichtentechnischen Kanals noch weitgehend überein. Die wichtigen Unterschiede sind:

- Unzuverlässigkeit; im Vergleich zum nachrichtentechnischen Kanal besitzt der gesicherte Kanal - durch Kanalcodierung und ablauforganisatorische Maßnahmen erreicht - i.a. Datenaustausch mit wesentlich verrringerter Fehlerwahrscheinlichkeit an.
- Synchronisation; aus der im gesicherten Kanal gegebenen Sicherung gegen Verlust und Reihenfolgevertauschung folgen i.a. im Vergleich zum nachrichtentechnischen Kanal gelockerte Synchronisationsbedingungen. So besitzt der gesicherte Kanal oft schon FIFO-Pufferungseigenschaften.
- Zugangssteuerung; beim nachrichtentechnischen Kanal agieren die verschiedenen am selben Medium befindlichen Umformer und Rückformer noch in einer voneinander unabhängigen Weise. So können gleichzeitig von verschiedenen Orten aus Sendesignale auf dasselbe physikalische Medium gegeben werden, so daß die Signale sich im Medium überlagern und von den Empfängern nicht erkannt werden können. In der Schicht 2 sind deshalb Funktionen zum geregelten Zugriff auf nachrichtentechnische Kanäle, d.h. zur zeitweisen Zuteilung an einzelne Sender, vorzusehen.

In der nächsten Schicht muß nun der enge Zusammenhang einzelner physikalischer Medien verlassen werden. Die Menge aller gesicherten Kanäle ist nun im Verbund zu betrachten. Sie bilden ein Netz,

woraus sich auch der Name dieser Schicht ergibt (Netzwerk-Schicht, Network Layer). In diesem Gesamtzusammenhang könnten jetzt tatsächlich alle für den Transportdienst definierten Aufgaben bewältigt werden, und der von der Schicht 3 angebotene Dienst könnte schon der gewünschte Transportdienst sein. Dann wäre diese Schicht allerdings sehr 'dick'. Bereits die Problematik der freizügigen Konnektivität im Netz ist ein sehr großer (und die derzeit komplexesten Protokolle erfordender) Aufgabenbereich, so daß nicht von der Konnektivitätsherstellung berührte Ende-zu-Ende-Aufgaben in der Netzwerkschicht noch ausgeklammert bleiben.

Ein Netz faßt die Menge aller verfügbaren gesicherten Kanäle zusammen und kann so Dienstleistungen zum Nachrichtenaustausch zwischen beliebigen Knoten des Rechnernetzes unterstützen. Hauptunterschiede zur Sicherungsschicht sind:

- Flexibilität der Dienstqualitäten; durch Wahl der für eine Dienstleistung zu verwendenden gesicherten Kanäle und der Modalitäten ihrer Nutzung kann bereits eine - allerdings im Vergleich zu der vom Transport-Dienstnehmer gewünschten noch sehr eingeschränkte - Flexibilität erzielt werden.
- Konnektivität und Adressierung; zentrale Eigenschaft eines Netzes ist es, daß Nachrichtenaustausch zwischen beliebigen Knoten angefordert werden kann. Jedem Knoten des Rechnernetzes ist mindestens ein Netz-Zugangspunkt zugeordnet, und die Zugangspunktadressen bilden einen einheitlichen Adreßraum, so daß der Netz-Dienst den Nachrichtenaustausch zwischen beliebigen Orten als einheitliche Dienstleistung anbieten kann. Im Vergleich zur Zugangspunktgliederung der Transportschnittstelle ergibt sich allerdings oft noch eine wesentlich gröbere Aufgliederung. Zwar können im selben Rechnerknoten auch bereits mehrere Netz-Zugangspunkte pro Rechner vorgesehen sein, doch ist die Anzahl der Zugangspunkte noch an der Knotenmenge und nicht an der Menge der Transport-Dienstnehmer orientiert und folgt somit realisierungstechnischen Randbedingungen.

Die verbleibende Menge der Aufgaben, das sind alle unter Ende-zu-Ende-Aspekten lösbaren, wird der nächsthöheren Schicht 4 und damit auch der obersten Schicht des Transportsystems zugeordnet. Sie wird Transportschicht (Transport Layer) genannt. Als wesentliche Aufgaben verbleiben hier die Unterstützung der Ende-zu-Ende-Adressierung und der möglichst kostengünstige Einsatz von Netzwerk-Dienstleistungen zur Realisierung der Transportdienstleistungen in einer von den Transport-Dienstnehmern gewünschten Qualität.

Bild 5.3 gibt verdeutlicht die vorstehenden Festlegungen zur Struktur von Transportsystemen anhand der Skizze eines konkreten Transportsystems. Es wurde eine häufig zu findende Variante der Netzwerkschicht gewählt. Das Netz setzt nicht unmittelbar auf gesicherten Kanälen auf sondern wird zweistufig über Subnetze gebildet. Die Trennung eines Netzes in Subnetze erfolgt in der Praxis aus organisatorischen und Kostengründen. So können z.B. in einem geographisch verteilten Netz von der Post zur Verfügung gestellte Netze für weite Strecken benutzt werden, während Strecken innerhalb eines Betriebsgeländes durch ein oder mehrere private lokale Netze überbrückt werden.

Abschließend soll anhand der in Bild 5.3 gezeigten Architektur noch auf die Unterscheidung der Knoten, d.h. einzelnen Rechnersysteme, eines Rechnernetzes in Endsystem / Endknoten / Wirtsrechner und Transitsystem / Transitknoten / Vermittlungsrechner hingewiesen werden. Nach Bild 5.3 ist an Ort C ein Knoten vorhanden. Hier gibt es Zugangspunkte zu physikalischen Medien, nachrichtentechnischen Kanälen und Subnetzwerken. Allerdings besitzt der Netz-Dienst hier keinen Zugangspunkt, in der Transportschicht ist keine Instanz vorgesehen, und es stehen auch keine Transport-Dienstzugangspunkte für Transport-Dienstnehmer zur Verfügung. Damit bleibt die Funktion des Knotens an Ort C im Transportsystem und dort bereits schon durch den Netz-Dienst verdeckt. Es handelt sich in der ISO/OSI-Terminologie um ein Transitsystem, während die drei anderen Knoten, da sie Transport-Dienstnehmer also Anwendungsinstanzen tragen, als Endsysteme zu bezeichnen sind.

5.3 Anwendungsorientierte Schichten

In den voranstehenden Bildern 5.2 und 5.3 werden die Dienstnehmer des Transportsystems noch als atomare Einheiten dargestellt, die miteinander in der sehr allgemeinen Kooperationsbeziehung 'Austausch binärer Nachrichten' stehen.

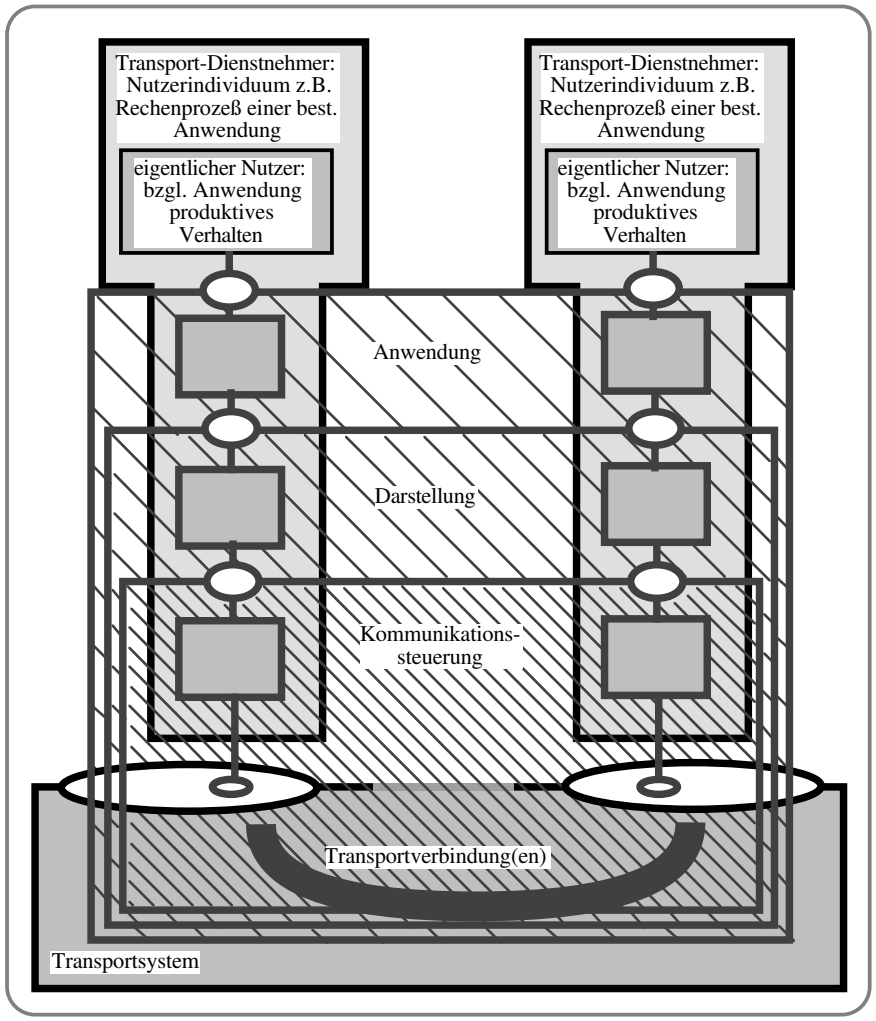

Bild 5.4: Anwendungsorientierte Schichten

Betrachtet man eine bestimmte Anwendung, z.B. ein Flugbuchungssystem, so gibt es einen speziellen für die Anwendung insgesamt definierten Zweck. Die Kooperationsbeziehung der Anwendungsinstanzen ist auf diesen Zweck zugeschnitten. Aus der Sicht dieser Kooperationsbeziehung wird nicht mehr der Austausch allgemeiner Binärnachrichten als Kommunikationsdienstleistung benötigt. Hier will man vielmehr über im Typ auf die Anwendung abgestimmte Kommunikationsdienstleistungen verfügen, mit denen spezielle Informationen übertragen werden können, um z.B. eine Abfrage- und Buchungssitzung mit einem entfernten Partner einzurichten sowie Buchungsdaten, Buchungsanweisungen und Bestätigungen auszutauschen.

Derartige Kommunikationsdienstleistungen werden natürlich vom sehr allgemein gehaltenen Transportdienst nicht zur Verfügung gestellt. Sie können nur innerhalb eines Dienstes angeboten werden, der auf den speziell gegebenen Typ der Anwendung abgestimmt ist. Ein solcher Kommunikationsdienst wird Anwendungsdienst genannt und gilt als von der Schicht 7, der sogenannten Anwendungsschicht (Application Layer) erbracht.

Wenn aus der Differenz zwischen Anwendungsdiensten und Transportdienst für verschiedendste Anwendungen gemeinsame Aufgaben herausgelöst werden können, ist es sinnvoll, diesen eigene zwischen Schicht 4 und 7 liegende Schichten zuzuordnen. Im ISO/OSI-Modell wurden dazu zwei Aufgabenklassen identifiziert:

- die Problematik der Quellencodierung, d.h. die Darstellung von Information durch binäre Nachrichten,
- die Steuerung von - mehrere Austauschvorgänge umfassenden - Kommunikationsabläufen.

Da es sich um logisch nicht zusammengehörige Problembereiche handelt, werden zwei Schichten eingeführt:

- die Schicht 5 (Kommunikationssteuerungsschicht, Session Layer),
- die Schicht 6 (Darstellungsschicht, Presentation Layer).

Die Unterordnung der Kommunikationssteuerung unter die Darstellung erfolgt, weil hier Datenmengen-Aspekte zu berücksichtigen sind. Es empfehlen sich abhängig vom Umfang auszutauschender Daten (gemessen an ihrer Bitmuster-Darstellung) unterschiedliche Kommunikationssteuerungsstrategien.

Bild 5.4 zeigt diese Hierarchie der anwendungsorientierten Schichten am Beispiel einer aus zwei Instanzen bestehenden Anwendung.

Logisch wird ein Dienstnehmer des Transportsystems in vier Instanzen aufgegliedert. Sie kommunizieren vertikal miteinander über die Zugangspunkte der an den zwischenliegenden Schichtgrenzen angebotenen Kommunikationsdienste, um letztendlich die Kommunikationsfähigkeiten des Transportsystems im Sinne der Anwendung zu nutzen. Dabei ist zunächst eine Zweigliederung in die Abwicklung der eigentlichen aus Anwendungssicht notwendigen Aktionen und in die Abwicklung der Kommunikation zu berücksichtigen. Ersteres wird als vollständig in der obersten Instanz abgehandelt angenommen. Letzeres kann als durch Instanzen anwendungsorientierter Kommunikationsdienst-Schichten realisiert verstanden werden.

Die oberste Instanz ist die eigentliche Nutzerinstanz. Hier wird die im Sinne der Anwendung produktive Arbeit (z.B. mit dem menschlichen Bediener über eine Datenstation kommunizieren, Tabellen, Belegungspläne und Abrechnungen erstellen) verrichtet und zwar in Kooperation mit den anderen Nutzerinstanzen derselben Anwendung unter Verwendung der Dienstleistungen des Anwendungsdienstes.

Alle Kommunikationsaspekte der Anwendung werden im Konzept des speziellen Anwendungsdienstes abgehandelt. Zu seiner Definition ist - dies ist typisch für einen Anwendungsdienst - zunächst eine anwendungsbezogene Sicht der Nutzerobjekte und Operationen vereinbart, und zwar um Implementierungsunabhängigkeit zu erreichen eine abstrakte Sicht. Zweitens werden die verschiedenen

Dienstleistungstypen - mit Primitiven und deren Parametern - eingeführt und in ihrer Bedeutung für die Nutzer anhand dieser abstrakten Sicht erklärt. So kann z.B. im sogenannten virtuellen Flugbuchungsmodell, die abstrakte Sicht eines Flugbuchungssystems und seiner auf Nutzerinstanzebene relevanten Komponenten erklärt sein, und eine Dienstleistung des Typs 'Übertrage den Wunsch zum Öffnen einer Sitzung einer Reiseagentur mit Fluggesellschaft zur Abfrage und Buchung' wird demgegenüber verständlich.

In der Bezugsetzung von Dienstleistungstypen zur abstrakten Sicht der Anwendung ist die Bedeutung der einzelnen Dienstleistungen des Anwendungsdienstes vollständig erfaßt, und es wird vorausgesetzt, daß die Nutzerinstanzen - und zwar in eigener Verantwortung - nur entsprechend dieser Festlegung die Dienstleistungen verwenden.

Unterhalb der Schnittstelle des Anwendungsdienstes tritt eine Bezugnahme auf die für die Anwendung gegebene spezielle Bedeutung angeforderter Dienstleistungstypen nicht mehr auf. Die drei in Bild 5.4 gezeigten anwendungsorientierten Schichten befassen sich intern nur noch mit der reinen Kommunikation und interpretieren Dienstleistungsprimitive und Parameter nicht mehr bezüglich des eigentlichen Anwendungszwecks. Die Funktion der gezeigten drei Instanzen besteht allein darin, das Schnittstellenereignisgeschehen an der Anwendungsdienstschnittstelle auf ein geeignetes Schnittstellenereignisgeschehen an der Transportdienstschnittstelle abzubilden.

Der an der Schnittstelle der Schicht 5 angebotene Kommunikationssteuerungsdienst besitzt gegenüber dem Transportdienst folgende zusätzliche Funktionalität:

- Persistenz von Kommunikationsbeziehungen; während Transportverbindungen aus transportsysteminternen Gründen jederzeit abgebrochen werden können, wollen Nutzerinstanzen Kommunikationsbeziehungen untereinander für eine allein von ihnen bestimmte Zeitdauer unterhalten. Der Abbruch von Transportverbindungen kann zwar nach wie vor noch dazu führen, daß für eine unbestimmte Zeitdauer kein Nachrichtenaustausch zwischen den Nutzerinstanzen mehr möglich ist. Aber - unter der Annahme, daß derartige Unterbrechungen der Kommunikationsfähigkeit zeitlich beschränkt sind - soll eine Fortsetzung einer vorherigen Beziehung möglich sein.
- Gegenseitige Konsistenz der Teilnehmersichten auf eine Kommunikationsbeziehung; eine Nutzerinstanz A, die eine Beziehung zu einer Nutzerinstanz B unterhält, muß zu einem Zeitpunkt des Betriebs, allein aus der Kenntnis der Historie der bisher am A-Zugangspunkt aufgetretenen Ereignisse, auf die Sicht der Nutzerinstanz B schließen können, die B zu einem sowohl A als auch B bekannten vorangegangenen Zeitpunkt hatte. Hierbei müssen solche Zeitpunkte zur Erzeugung konsistenter Sichten über die bisher stattgefundene Kommunikation - sogenannte Synchronisationspunkte - zwischen den Nutzern nach Bedarf vereinbart werden können. Entsprechende Dienstleistungen zur Vereinbarung solcher Punkte sind im Kommunikationssteuerungsdienst vorgesehen. Synchronisationspunkte können zum wohldefinierten Rücksetzen in einen früheren Stand einer Kommunikationsbeziehung und nach Unterbrechungen zum wohldefinierten Wiederaufsetzen dienen und betreffen somit auch die Persistenz.
- Gliederung des Nachrichtenaustauschs; spezielle Anwendungen möchten ihren Nachrichtenaustausch i.a. nicht allein in Nachrichtenpakete gliedern, wie dies vom Transportdienst unterstützt wird. Allgemein wäre - ausgehend von Nachrichtenpaketen - eine über eine beliebige Anzahl von Stufen bildbare hierarchische Gliederung wünschenswert. Der Kommunikationssteuerungsdienst kommt dem bereits durch die direkte Unterstützung einer dreistufigen Gliederung in Nachrichtenpakete, Dialogeinheiten und Aktivitäten entgegen. Diese Gliederung erfolgt im Einklang mit der zeitlichen Strukturierung von Nutzerbeziehungen durch Synchronisationspunkte.
- Ablauforganisation des Nachrichtenaustauschs; der vom Transportdienst angebotene Vollduplex-Nachrichtenaustausch ist, wenn Persistenz, Konsistenz und logische Gliederung vorhanden sein sollen, direkt nicht verwendbar. Nutzerinstanzen muß eine gegenseitige Reglementierung des Nachrichtenaustauschs im Sinne strengerer Ablauffestlegungen möglich sein. Hierzu werden Dienstleistungen zur Festlegung und Modifikation von Rechten, wie z.B. dem Senderecht, notwendig.

Der an der Schnittstelle der Schicht 6 angebotene Darstellungsdienst entspricht im wesentlichen dem Kommunikationssteuerungsdienst, d.h. es finden sich für die einzelnen Dienstleistungstypen des Kommunikationssteuerungsdienstes gleichartige Pendants des Darstellungsdienstes. Die Funktion der Darstellungsinstanzen besteht hier in einer einfachen 1:1-Abbildung von Ereignissen, einem Weiterreichen von Anforderungen und Anzeigen, zwischen den Schnittstellen der Schichten 5 und 6. Unterschiedlich - und hierin liegt die zentrale Bedeutung des Darstellungsdienstes - ist die Sicht der Nutzdaten, und zwar bezüglich ihrer Darstellung (Syntax). Während der Kommunikationssteuerungsdienst - wie der Transportdienst und alle untergeordneten - für den Transfer beliebiger Bitmuster als Nutzdaten konzipiert ist, soll der Darstellungsdienst mit seinen Transferdienstleistungen anwendungsspezifisch typisierte und zusammengesetzte Daten übertragen. Zu unterscheiden sind:

- Konkrete Syntaxen von Daten bei den Nutzerinstanzen; dies ist ein reiner Implementierungsaspekt der Realisierung der Nutzerinstanzen. Von der Anwendung her relevante und per Kommunikation auszutauschende Daten können innerhalb einer Nutzerinstanz in einer für diese Instanz besonderen, und unabhängig von der Darstellung gleichartiger Daten in anderen Nutzerinstanzen derselben Anwendung gewählten Form gespeichert sein. So können z.B. bei Implementierung einer Flugbuchungsnutzerinstanz in PASCAL per Programm in den Abschnitten TYPE und VAR spezielle Datentypen und Variablen zur Darstellung einer Buchungsanforderung gewählt werden. Der verwendete PASCAL-Compiler und das zugeordnete PASCAL-Laufzeitsystem legen die Binärdarstellung der Buchung im Speicher fest. Es ist nicht im Sinne offener Systeme, eine solche nur lokal definierte Binärdarstellung unmittelbar über das Transportsystem zu übertragen. Damit wird die Nutzerinstanz auf der anderen Seite - sie muß die als Bitmuster ankommenden Daten interpretieren - von der Auslegung der anderen Nutzerinstanz und sogar von dem dort zur Implementierung gerade verwendeten Programmiersystem abhängig.
- Abstrakte Syntax; erster Schritt zur Behandlung der Darstellungsproblematik ist, sich von der konkreten, implementierungsbestimmten Wertdarstellung der Daten zu lösen. Der Anwendungsdienst ist bereits über einem abstrakten Modell der anwendungsrelevanten Objekte und Operationen definiert, das auch benötigte Datentypen in ihrer Semantik vereinbart, und die Dienstleistungen des Anwendungsdienstes stehen in direktem Bezug dazu. Die Instanzen der Schicht 7 benötigen deshalb sinnvollerweise vom Dienst der Schicht 6 Kommunikationsoperationen zum Austausch typisierter Daten. Dabei spielen Operationen der Datentypen und deren Semantik keine Rolle mehr. Da die Daten über eine Repräsentation ausgetauscht werden müssen, ist ihre Wertdarstellung, die Syntax, wichtig. Um Implementierungsunabhängigkeit zu erreichen, geht man dabei von einer abstrakten, nicht bis hin zur Codierung per Bitmuster festgelegten Syntax aus. Diese abstrakte Syntax ist für alle von einer bestimmten Anwendung berührten Orte gleich.
- (Konkrete) Transfersyntax; natürlich muß die abstrakte Syntax der Daten doch zu einer konkreten Syntax bis zur Bitmusterdarstellung verfeinert vereinbart sein, wenn Daten ausgetauscht werden sollen. Eine solche Syntax wird Transfersyntax genannt und ist, wie die abstrakte Syntax für alle in der momentanen Betriebsphase von der Anwendung betroffenen Orte verbindlich.
- Codierregeln; die Differenz zwischen abstrakter und Transfersyntax wird durch Codierregeln überbrückt. Ein solcher Satz von Codierregeln legt fest, wie die benutzten Standarddatentypen binär dargestellt werden, und wie darüber Werte von durch Zusammensetzung (d.h. durch Mengen und Folgenbildung, vgl. PASCAL: RECORD, SET OF, FILE OF) definierten Datentypen in ein Bitmuster abzubilden sind.
- Darstellungskontexte; für eine bestimmte Anwendung wird a priori eine nichtleere Menge von Darstellungskontexten vereinbart sein (dies erfolgt i.a. beim Entwurf der Anwendung). Ein Kontext ist ein Verbund aus Kontextidentifikation (ein Name zur Bezugnahme auf den Kontext), und einer Menge von Datentypen, die per Typidentifikator (ein Name zur Bezugnahme auf einen abstrakten Datentyp), abstrakter Syntax und anzuwendende Codierregeln erfolgt. Der Darstellungsdienst bietet nun Dienstleistungen an, mit denen aktuell in einer Betriebsphase zugrundezulegende Kontextmengen zwischen den beteiligten Dienstnehmern durch Austausch von Kontextidentifikator-Listen ausgehandelt werden können. Die angebotenen Datenaustausch-Dienstleistungen beziehen sich dann in einer Betriebsphase auf die aktuelle Kontextmenge. Ein Dienstnehmer darf den Transfer eines abstrakten Datums veranlassen, wenn dessen Typ in der aktuellen Kontextmenge definiert ist. In der logischen Architektur werden die Codierung und Decodierung des Datums in Entsprechung zum Kontext innerhalb der Schicht 6 vorgenommen.

- Syntaxvereinbarungssprache und Standardcodierregeln; beim Entwurf einer bestimmten Anwendung sind -wie erwähnt - die vorgesehenen Darstellungskontexte zu konzipieren. Sie müssen exakt dokumentiert werden, da sie in den Entwurf der Anwendungsinstanzen wesentlich einfließen. Zu diesem Zweck wurde im Rahmen der ISO/OSI-Arbeiten, begleitend zur Normung des Darstellungsdienstes mit seinen Dienstleistungstypen, eine ASN.1 (abstract Syntax Notation No. 1) genannte Syntaxvereinbarungssprache und ein Satz von darauf bezugnehmenden Codierregeln, die Standardcodierregeln, entwickelt. Wie aus der Namensgebung (...No.1, Standard...) hervorgeht, handelt es sich hier um zusätzliche Festlegungen, die nicht zwingend zum Verständnis des Konzepts des Darstellungsdienstes notwendig sind, insbesondere sind Alternativen denkbar.

6. Protokollmechanismen

Es ist charakteristisch für einen echten Telekommunikationsdienst - egal welcher Schicht er zugeordnet wird -, daß nicht alle seine Erbringung beeinflussenden Störungen in ihren Auswirkungen verdeckt werden können. Zueinander örtlich entfernte Instanzen können ihre Kooperation nur auf einer losen Kopplung aufbauen. Der Nachrichtenaustausch zwischen ihnen kann mit unvorhersehbaren Verzögerungen, Verfälschungen, Verlusten, Phantomnachrichten und Reihenfolgevertauschungen behaftet sein. Auch Ausfälle eines Partners werden aufgrund seiner Entfernung nicht sofort und auch nicht nach einer vorhersagbaren maximalen Zeitspanne bemerkt. So können zum selben Zeitpunkt des Betriebs eines Rechnernetzes an unterschiedlichen Orten verschiedene Sichten des stattgefundenen Nachrichtenaustauschs bestehen.

Aus diesen Gründen kann die Aufgabe einer Schicht nicht durch eine einzige sequentiell nach einem einzigen Programm agierende Instanz gelöst werden, die z.B. zur Durchführung einzelner Aktivitäten von Ort zu Ort wandert, oder aber an einem Ort fest lokalisiert ist und von dort aus den Gesamtablauf zentral steuert.

Stattdessen muß dem Prinzip der dezentralen Kontrolle gefolgt werden. An jedem Ort befindet sich eine autonome Instanz, die nach einem ihr zugeordneten, in sich geschlossenen Programm nebenläufig zu den anderen Instanzen agiert, wobei sie ihre einzelnen Aktionen nur von der bei ihr momentan bestehenden Sicht des Gesamtablaufs abhängig macht. Damit der Verbund miteinander kommunizierender aber autonom agierender Instanzen eine gemeinsame Aufgabe lösen kann, sind die einzelnen Verhaltensvorschriften der Instanzen zueinander im Zusammenhang zu sehen und im Zusammenhang miteinander als Protokoll zu entwerfen.

Durch die lose Kopplung der Instanzen bedingt sind in den Protokollen in der Regel Mechanismen zum Erkennen von Übertragungs- und Ablaufbesonderheiten sowie zur Erzeugung konsistenter Teilsichten zwischen Partnern vorhanden. Sie werden in folgenden Abschnitten 'Fehlererkennung' und 'Fehlerbehebung' vorgestellt und machen oft den größten Teil der funktionellen Komplexität eines Telekommunikationsprotokolls (jedenfalls eines Transferprotokolls) aus.

Zusätzlich vorzustellen sind Maßnahmen, die hauptsächlich qualitative Dienstmerkmale beeinflussen. Hier sind die Aufgaben der Längenanpassung zu übertragender Nachrichten, der Flußkontrolle, der Verknüpfung mehrerer Medien oder Subnetze zu einem Netz, der Durchsatzerhöhung und der Kostenreduktion zu nennen. Vorab werden Mechansimen zur Regelung des Zugangs unter Vermeidung oder Auflösung von Zugriffskonflikten bei der geteilten Nutzung physikalischer Medien erörtert.

Die folgende Aufzählung von Protokollmechanismen ist nicht vollständig. Behandelt werden vorrangig die in gegenwärtigen transportorientierten Protokollen gängigen Verfahren.

6.1 Zuteilung geteilter Medien

Im einfachsten Fall wird ein physikalisches Medium von genau zwei Dienstnehmern zur Übertragung in einer einzigen Richtung genutzt. Hier sind - vielleicht abgesehen von bestimmten Ruhephasen zwi-

schen zwei aufeinanderfolgenden Sendevorgängen - keine besonderen Maßnahmen erforderlich, um den Zugang des Senders zum Medium zu steuern.

Bereits im Fall eines ebenfalls nur zwei Dienstnehmer verbindenden Mediums, das in beide Richtungen im Halbduplex-Betrieb zur Übertragung genutzt wird, stellt sich das Zuteilungsproblem. Kollidieren zwei zeitlich überlappende gegengerichtete Übertragungen, dann können beide Übertragungen durch gegenseitige Überlagerung der Signale gestört werden. Deshalb muß ein Halb-Duplex-Medium einem einzigen Sender für die Dauer einer Übertragung exklusiv zugeteilt werden.

Solche Kollisionen müssen weiterhin natürlich auch bei Medien vermieden werden, die mehr als zwei Dienstnehmer verbinden, z.B. Sammelleitungen (bei lokalen Netzen) oder Funkkanäle (Satellitenverbindungen). Hierzu muß der Zugang von Sendern zum Medium geregelt werden.

Bild 6.1 zeigt eine Übersicht über die verschiedenen Verfahren. Bei der geteilten Nutzung eines Mediums unter Frequenzmultiplex wird sehr häufig eine feste Aufteilung in Teilbänder vorgenommen, die vorab und bleibend einzelnen Sendern fest zugeordnet sind. Auf Frequenzmultiplex-Verfahren soll im folgenden nicht weiter eingegangen werden.

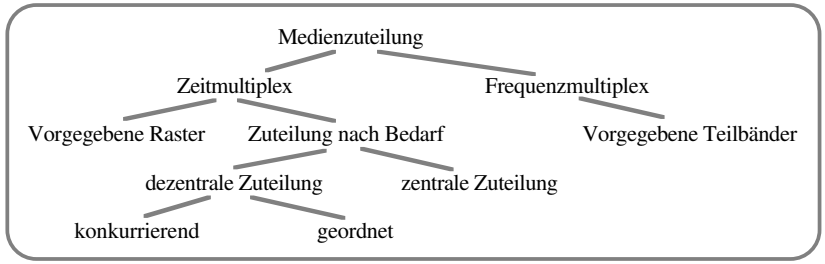

Bild 6.1: Medienzuteilung

Die geteilte Nutzung im Zeitmultiplex basiert darauf, daß das Medium einzelnen Sendern für eine gewisse Zeitspanne exklusiv zugeteilt wird. Bei einfachen Verfahren wird die Zeitachse entsprechend einem periodisch wiederkehrenden festen Schema in Zeitschlitze gegliedert, und einem Dienstnehmer wird statisch des Nutzungsrecht für einen bestimmten Zeitschlitz zugeordnet. Hat ein Dienstnehmer zum Zeitpunkt des Beginns seines Zeitschlitzes nichts zu senden, bleibt der Zeitschlitz ungenutzt.

Bei variierendem Bedarf der Dienstnehmer werden deshalb Verfahren interessant, bei denen die Zuordnung von Zeitschlitzen (fester, gerasterter oder in Grenzen beliebiger Länge) an Sender bedarfsgesteuert erfolgt.

Verfahren/Protokolle zur Zuteilung geteilter Medien werden oft MAC-Verfahren (Medium Access Control) genannt.

6.1.1 Zentrale Zuteilung nach Bedarf

Im hier einfacheren Fall der zentralen Zuteilung gibt es einen hervorgehobenen Dienstnehmer, die Leitstation. Die anderen heißen Unterstationen. Die Leitstation hat die Aufgabe, für gewisse Zeitdauern einzelnen Stationen bedarfsgesteuert das Wort zu geben.

Ein wichtiger Mechanismus dazu ist die Abfrage (Polling). Die Leitstation besitzt anfänglich das Senderecht. Sie verwendet es, wenn sie selbst Nachrichten senden will. Außerdem sollte sie in fairer Weise nacheinander den Unterstationen die Möglichkeit zum Absenden von Nachrichten geben. Dazu richtet sie eine Anfrage an eine Unterstation, die damit das Senderecht (für eine oft per Konvention limitierte Zeitdauer) erhält, um Nachrichten zu senden und anschließend das Senderecht an die Leitstation zurückzugeben.

Problematisch ist die zentrale Rolle der Leitstation. Einerseits verursacht die Abstimmung mit der Leitstation Kommunikationsaufwand, andererseits ist das Medium für alle Teilnehmer gesperrt, wenn die Leitstation ausfällt.

Unter diesem Aspekt werden dezentrale Verfahren interessant. Hier gibt es keinen Dienstnehmer mit hervorgehobener Funktion. Die Zuteilungsmechanismen sind bei allen Dienstnehmern gleichartig vorhanden.

6.1.2 Dezentrale Zuteilung unter Konkurrenz

Die wichtigsten Verfahren zur Zuteilung geteilter Medien unter Konkurrenz sind:

- Aloha, (zuerst in Hawai in einem Terminal-Funknetz angewendet), in den Varianten:
  - non-slotted (pure) / slotted (Slot: Schlitz, Zeitschlitz),
- CSMA, (Carrier Sense Multiple Access), in den Varianten:
  - non-slotted / slotted,
  - 1-persistent / p-persistent / non-persistent,
- CSMA/CD, (Carrier Sense Multiple Access / Collision Detection), in Varianten entsprechend den CSMA-Varianten.

Die dezentralen Zuteilungsverfahren unter Konkurrenz haben den Vorteil der leichten Ein- und Ausgliederung von Stationen. Ein Nachteil ist, daß auch bei Abwesenheit von Störungen nicht garantiert werden kann, daß ein bestimmter Nutzdaten-Übertragungsauftrag innerhalb einer vorabbestimmbaren maximalen Bearbeitungszeitdauer beendet wird. Mit diesem Nachteil wird das Verfahren für zeitkritische Anwendungen (z.B. Sprachübertragung bei Gesprächen, Lenkung technischer Prozesse) problematisch.

Aloha: Das einfache Verfahren basiert auf der unreglementierten Konkurrenz aller Dienstnehmer. Wer etwas zu senden hat, sendet sein Paket sofort auf das Medium. Überlagert sich die Übertragung des Pakets mit der eines anderen Pakets, werden beide Pakete verfälscht und werden von den Empfängern ignoriert (-> Sicherung gegen Verfälschung, Rückführung der Verfälschung auf Paketverlust). Mechanismen zur Sicherung gegen Verlust (später erläutert) veranlassen den Sender eines verlorengegangenen Paktes dann etwas später zur Wiederholung des Sendens.

Slotted Aloha: Es gibt einen Takt mit konstanter Periode, die der Übertragungszeit für ein Paket auf dem Medium entspricht (Paketlänge ist konstant). Wer etwas zu senden hat, wartet bis zum nächsten Start eines Slots, und sendet dann sein Paket. Ansonsten entspricht slotted Aloha dem pure Aloha Verfahren.

Non persistent CSMA (non-slotted / slotted): Wer etwas zu senden hat, hört vorher am Medium, ob gerade ein anderes Paket übertragen wird (Mithören vor dem Senden, Listen before transmit). Er sendet nur, wenn das Medium frei zu sein scheint. Man beachte, daß auch wenn ein Sender momentan keine Signale einer anderen Übertragung detektiert, trotzdem ein anderes Paket schon auf dem Medium sein kann (Medienlänge und endliche Ausbreitungsgeschwindigkeit der Signale). In der 'slotted'-Variante wird das Senden zusätzlich noch bis zum Start der nächsten Slots verzögert.

Wenn ein Sender das Medium als belegt erkannte, berechnet er eine Wartezeit. Nach Verstreichen der Wartezeit, beginnt er seinen Sendeversuch erneut. Bei erkanntem Paketverlust wird die Wiederholung des Sendens ebenfalls erst nach einer Wartezeit begonnen.

Die Wartezeiten werden nach einem sogenannten Backoff-Algorithmus berechnet.

1-persistent CSMA (non-slotted / slotted): Wer etwas zu senden hat, hört ebenfalls vorher am Medium, allerdings kontinuierlich solange, bis das Medium frei zu sein scheint (bei 'slotted' wartet er zusätzlich noch bis zum Beginn des nächsten Slots).

p-persistent CSMA (non-slotted / slotted): Wer etwas zu senden hat, hört ebenfalls vorher am Medium kontinuierlich solange, bis das Medium frei zu sein scheint. Dann wird eine Zufallszahl z aus [0..1] ausgewählt. Falls $z \leq p$ wird gesendet (bei 'slotted' wird noch der Beginn des nächsten Slots erwartet). Falls $z > p$ wird eine Verzögerungszeit (konstante Zeitdauer) abgewartet und (ohne vorher zu senden) erneut mit dem kontinuierlichen Mithören begonnen.

CSMA/CD: Bis zum Start des Sendens ergeben sich dieselben Abläufe wie bei den entsprechenden CSMA-Varianten. Während des Sendens hört ein Sender jedoch ebenfalls am Medium mit. Detektiert er, daß seine gesendeten Signale sich mit anderen Signalen überlagern (Kollision nach Sendebeginn), wird das Senden sofort abgebrochen und es wird ein kurzes Abbruch-Signal gesendet (engl. Jam-Signal). Nach dem Senden des Abbruch-Signals wird der nächste Versuch erst nach Verlauf einer Wartezeit begonnen, die entsprechend einem Backoff-Algorithmus berechnet wird.

Backoff-Algorithmen: Man unterscheidet zwischen dynamischen und statischen Verfahren. Statische Verfahren ermitteln die Wartezeit als Zufallszahl aus einer Verteilung mit festem Mittelwert. Bei dynamischen Verfahren wird eine Verteilung gewählt, deren Mittelwert von der Anzahl der vorangegangenen nicht-erfolgreichen Versuche abhängt.

Zur Berechnung der Wartezeit wird häufig (z.B. bei Ethernet, das ein 1-persistent CSMA/CD-Verfahren ist) der 'Truncated binary exponential Backoff-Algorithmus' eingesetzt. Es werden nur ein erster Übertragungsversuch eines Pakets sowie eine maximale Anzahl von Wiederholungen erlaubt (typisch 15). Die Wartezeit berechnet sich aus einem Produkt $i * t_b$ (i: zufällig gewählte natürliche Zahl aus $\{0, .., 2^{k-1}\}$, mit k=min(10, Anzahl bisher aufgetretener Sendeversuche); t_b: Backoff-Zeitkonstante, oft das Doppelte der maximalen Übertragungsdauer eines Pakets).

Einige Leistungsaspekte

Bei allen diesen Verfahren kann die Übertragung eines Pakets durch Kollisionen gestört werden. Wenn viele Sender viel zu senden haben, scheint das Chaos unvermeidlich.

Es interessieren die Zieldienst-Leistungskenngrößen:
- Durchsatz,
- Zeitdauer für die erfolgreiche Übertragung eines Pakets (kurz Paketverzögerung).

Beeinflussend sind die Kenngrößen des Mediums:
- Medium-Länge,
- Signalausbreitungsgeschwindigkeit auf dem Medium,
- Sende / Empfangsbitrate,

und die Kenngrößen des Medium-Betriebs:
- Anzahl der Stationen,
- Paketlänge,
- angebotene Last.

Bei pure Aloha ergeben sich oft schlechte Leistungseigenschaften. Maßnahmen zur Verbesserung zielen ab auf:
- die Verringerung der Verletzlichkeitszeitdauer eines Pakets,

- die Verringerung der Belegungszeitdauer des Mediums durch kollidierte Pakete,
- die Drosselung der Sendeversuche bei hoher angebotener Last.

Bei pure Aloha ergibt sich eine Verletzlichkeitszeitdauer vom Zweifachen der maximalen Übertragungszeit eines Pakets (einmal für das untersuchte Paket, einmal für ein Störpaket). Bei slotted Aloha halbiert sich die Verletzlichkeitszeitdauer wegen der getakteten Paketstarts. Bei CSMA reduziert sich die Verletzlichkeitszeitdauer weiter, nach spätestens einer Zeitdauer von 'Medium-Länge / Signalausbreitungsgeschwindigkeit' können alle konkurrierenden Stationen die Belegung des Mediums erkennen.

Bei Aloha und CSMA werden kollisionsbehaftete Pakete in voller Länge übertragen. Bei CSMA/CD erkennt ein Sender die Kollision seines Pakets nach spätestens einer Zeitdauer vom Zweifachen von 'Medium-Länge / Signalausbreitungsgeschwindigkeit' und bricht dann die Übertragung ab.

Verbesserungen durch CSMA gegenüber Aloha und durch CSMA/CD gegenüber CSMA hängen vom Verhältnis 'Signallaufzeit / Paketlaufzeit' ab (z.B. bei 500m Mediumlänge, Kupfer, 10Mbit/s Bitrate, Paketlänge 5000bit: Signallaufzeit / Paketlaufzeit $\approx 5\ \mu s / 500\ \mu sec = 0.01$). Bei kleinem Verhältnis sind die Pakete lang im Vergleich zum Medium (ein Paket von 5000bit bei 10Mbit/s hat auf Kupfer ungefähr die Länge von 50 km). CS- und CD-Maßnahmen können sich dann besser auswirken als bei im Vergleich zum Medium kurzen Paketen.

Zur Drosselung der Sendeversuche dienen die Backoff-Wartezeiten (die bei allen Verfahren angewendet werden können). Ohne diese Drosselung wird mit steigender angebotener Last zwar die Anzahl erfolgreich übertragener Pakete zunächst auch steigen. Dann wird jedoch ein Punkt erreicht, ab dem eine weitere Steigerung des Lastangebots zur Durchsatzverringerung führt (Überlastverhalten mit Leistungsverminderung).

Die Abhängigkeiten zwischen beeinflussenden Größen und Zieldienst-Durchsatz und -Paketverzögerung sollen hier nicht im Detail behandelt werden (siehe z.B. Hammond, J., O'Reilly, P.; Performance Analysis of Local Computer Networks). Abschließend sollen lediglich grob einige typischen Werte zum maximal erreichbaren Durchsatz (im Verhältnis zur Bitrate des Mediums) genannt werden (Verhältnis Signallaufzeit / Paketlaufzeit ≈ 0.01): Aloha: 18%, slotted Aloha: 36%, 1-persistent CSMA: 55%, non-persistent CSMA: 82%, CSMA/CD über 90%. Die mittlere Paketverzögerung wächst mit dem Durchsatz und liegt bei maximalem Durchsatz in der Größenordnung des 20fachen der Paketlaufzeit.

Anmerkung

Bei Satelliten-Funkkanälen können CSMA-Verfahren im übrigen nicht eingesetzt werden. Ein Aufwärts-Kanal (Richtung Bodenstationen->Satellit) ist zwar wie ein Ethernet-Kabel ein geteiltes Medium, aber die Bodenstationen können sich gegenseitig nicht empfangen.

6.1.3 Dezentrale Zuteilung durch zirkulierendes Recht

Einer beliebigen Station soll - unter der Annahme, daß Störeinflüsse von außen auf das Medium ausgeschlossen sind - garantiert werden, daß sie eine Nutzdaten-Übertragung innerhalb einer feststehenden Frist erfolgreich abwickeln kann. Dazu sind reine Konkurrenzverfahren nicht in der Lage. Neben der Beschränkung der Zuteilungsdauer für einzelne Stationen muß den Zuteilungen eine feste Ordnung aufgeprägt sein, so daß z.B. feststeht, daß einer sendewilligen Station im schlimmsten Fall spätestens mit Ablauf der Zeitdauer (Anzahl Stationen)*(Zeitdauerkonstante) das Senderecht eingeräumt wird. Hierzu können Verfahren zur zyklischen Weitergabe des Senderechts (Token) verwendet werden.

Über den Stationen am geteilten Medium ist eine zyklische Ordnung definiert (logischer Ring). Ein Senderecht (Freitoken, Leernachricht) zirkuliert auf dem Ring. Wenn nun eine Station sendewillig ist, leitet sie das Senderecht nicht weiter, sondern gibt stattdessen ihre Nachricht (u.U. auch eine Folge von Nachrichten) auf den Ring. Anschließend gibt sie das Senderecht wieder auf den Ring. Derartige Verfahren sind in drei Varianten üblich:

- Multiple Token,
- Single Token,
- Single Packet.

Bei Multiple Token generiert die sendende Station unmittelbar nach dem Ende ihrer Nachricht wieder ein Freitoken. Bei Single Token wartet die sendende Station, bis der Kopf ihrer Nachricht wieder bei ihr eingetroffen ist. Erst dann generiert sie wieder ein Freitoken. Bei Single Packet wartet die sendende Station sogar, bis das Ende ihrer Nachricht wieder bei ihr eingetroffen ist, bevor das Freitoken erzeugt wird. Im lokalen Netz (z.B. Leitungslänge 500m, aber Paketlänge 50km) ist der Unterschied zwischen Multiple Token, Single Token und Single Packet verschwindend.

Die Verfahren werden sowohl bei echten Sammelleitungen (Token Bus Verfahren) als auch bei Ringen (Token Ring Verfahren) eingesetzt.

Ringe entstehen aus gerichteten Zweipunkt-Leitungen in Ring-Topologie und jede Leitung ist einem Sender exklusiv zugeordnet. Also dürfte kein Bedarf für eine Zugangskontrolle bestehen. Allerdings werden eingehende und abgehende Leitung in einer Station auf einfache Weise (auf Schicht 1) gekoppelt (üblich: 1-Bit-Verzögerung), so daß der Ring insgesamt ein geteiltes Medium bildet (Will eine Station senden, trennt sie die Überbrückung zwischen ankommender und abgehender Leitung auf und sendet ihre Nachricht auf die abgehende Leitung; ankommende Bits gehen jetzt verloren).

Beim Ring ist durch die Topologie eine zyklische Ordnung der Stationen in der Art 'Station A liegt genau vor Station B' definiert. Bei einer Sammelleitung kann sie per Konvention eingeführt werden.

Zusätzlich müssen Maßnahmen getroffen werden, um Besonderheiten - wie kein oder mehrere Senderechte im Ring - zu erkennen und beheben.

6.2 Fehlererkennung

Eine Instanz muß beim Empfang einer PDU ihre Sicht der aktuellen Betriebssituation analysieren, um zu erkennen, ob die PDU als dem geplanten Systemablauf entsprechend behandelt werden kann, oder ob mit ihrem Eintreffen eine störungsbedingte Besonderheit vorliegt. Da neben dem Austausch von PDUs keine andere Möglichkeit zur Kommunikation mit den Partnerinstanzen besteht, können in die Analyse nur die Inhalte der aktuell empfangenen PDU sowie der - früher empfangene PDUs reflektierende - Instanzenzustand eingehen.

Störungsbedingte Besonderheiten können gegliedert werden in:

- Nachrichtenverfälschung,
- Zusammenhang-Besonderheiten.

Nachrichtenverfälschung ist eine direkte Störauswirkung und bezieht sich auf den sehr engen Zusammenhang einer einzigen PDU.

Weiterhin ist der Austausch einer PDU in der Regel in einem logischen Zusammenhang zu sehen, der mehrere PDU-Übertragungen zueinander in Bezug setzt, bestimmte Nachrichtentypen unterscheidet und für den ungestörten Betrieb geplante Austausch-Abfolgen vorsieht. Hier sind folgende grundlegenden Besonderheiten relevant:

- Nachrichtenverlust,
- Ungewollte Duplizierung einer Nachricht,
- Eintreffen einer Phantom-Nachricht,
- fehlerhafte Einordnung einer PDU in den momentan verfolgten Zusammenhang, insbesondere
- Abweichungen der Empfangsreihenfolge von Nachrichten.

Duplizierung, Phantom-Nachrichten und fehlerhafte Einordnung können dabei nicht nur direkte Auswirkungen von Störungen des Mediums sondern auch Auswirkungen dessen sein, daß Instanzen momentan aufgrund früherer Übertragungsbesonderheiten oder durch Ausfall von Rechnern bedingt zueinander inkonsistente Sichten des Betriebszustands besitzen.

Verfahren/Protokolle zur Sicherung der Übertragung auf einem physikalischen Medium gegen Verfälschung und Reihenfolgebesonderheiten werden of LLC-Verfahren genannt (Logical Link Control).

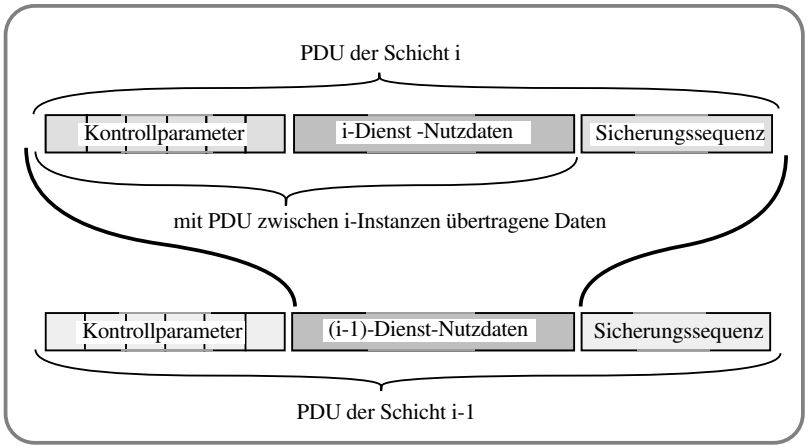

Bild 6.2: PDU - Aufbau

Generelle Maßnahme zur Erkennung der vorgenannten Fehlerarten durch eine Instanz sind, daß PDUs mit gegenüber dem normalen Betriebsablauf zusätzlichen Daten versehen werden:

- Zur Erkennung von Verfälschungen wird eine PDU in redundanter Codierung dargestellt.
- Zur Erkennung von Zusammenhang-Besonderheiten werden Kontrollparameter vorgesehen, in denen die sendende Instanz zur Kontrolle durch die empfangende Instanz Information über ihre Sicht des Betriebszustands hinterlegt.

Bild 6.2 zeigt den daraus folgenden grundlegenden Aufbau einer PDU aus den Komponenten Kontrolldaten, Zieldienst-Nutzdaten und Sicherungssequenz. Sowohl Verfälschungen der Kontrollparameter als auch solche der Nutzdaten werden in der Regel anhand der Sicherungssequenz erkennbar. Ferner weist Bild 6.2 darauf hin, daß bei Schichtenbildung PDUs die Nutzdaten des nächstunteren Dienstes und somit auch die Nutzdaten der in der nächstunteren Schicht verwendeten PDUs bilden, wo sie wiederum von einer Sicherungssequenz mitgeschützt werden können. Zu dieser mehrfachen Sicherung gegen Verfälschungen ist allerdings anzumerken, daß Verfälschungssicherung standardmäßig nur in den transportorientierten Schichten und dort teilweise optional vorgesehen ist.

Natürlich besteht bei vielen Protokollen oft auch der Bedarf nach PDU-Typen, in denen keine Nutzdaten vorkommen (hierzu streiche man in Bild 6.2 die Nutzdatenkomponente). Wenn nämlich von einer Instanz ein Betriebszustand erkannt wurde, in dem kein Nutzdatenaustausch sinnvoll ist aber Kontrolliformation an die Partnerinstanz übertragen werden muß. Man bemüht sich jedoch ein Protokoll so zu entwerfen, daß im laufenden Betrieb die Häufigkeit solcher reinen Kontroll-PDUs gering gehalten werden kann, da nach Möglichkeit Kontrollparameter zusammen mit Nutzdaten in derselben PDU übertragen werden (engl. 'Piggy Backing').

6.2.1 Fehlererkennende Codes

Die zur Erkennung von Verfälschungen zu sichernden Daten sind, wie in Bild 6.2 gezeigt, Folgen von binären Wertdarstellungen von Kontrollparametern und binäre Nutzdaten. Das bei einer Übertragung zu sichernde Datum ist also eine Bitfolge, deren Länge - bei Nutzdaten in bestimmten Grenzen variabler Länge - ebenfalls in Grenzen variabel ist.

In nachrichtentechnischer Sicht treten also als Wörter des Quellencodes beliebige Bitfolgen in Grenzen variabler Länge auf. Zur Übertragung wird ein Kanalcode gewählt, d.h. jedem möglichen Wort des Quellencodes wird ein zulässiges Wort des Kanalcodes umkehrbar eindeutig zugeordnet. Ein Kanalcodewort ist dabei länger als sein zugeordnetes Quellencodewort, die Kanalcodierung fügt Redundanz hinzu. Im Bildbereich der Kanalcodierung gibt es unzulässige Elemente, denen kein Quellcodewort zugeordnet ist. Beim Empfang eines Kanalcodeworts wird geprüft, ob es sich um ein unzulässiges Kanalcodewort handelt. Wenn ja, wird von einer Verfälschung bei der Übertragung ausgegangen, wenn nein, wird das zugeordnete Quellencodewort durch Umkehrabbildung ermittelt.

Es kann allerdings nicht ausgeschlossen werden, daß eine Verfälschung zufälligerweise ein zulässiges Kanalcodewort in ein anderes zulässiges überführt. In solchen Fällen wird der Übertragungsfehler nicht erkannt. Die Qualität einer Kanalcodierung hängt nun davon ab, mit welcher Wahrscheinlichkeit solche unentdeckbaren Verfälschungen noch auftreten können, und kann durch Längenbegrenzung der Quellencodewörter und Berücksichtigung der Kanalstörcharakteristik bei der Auslegung der Kanalcodierung günstig beeinflußt werden.

ISO/OSI-Protokolle sehen zur Kanalcodierung den sogenannten Cyclic Redundancy Check (CRC) vor. Hier entsteht ein Kanalcodewort aus dem Quellencodewort durch Anfügen einer zusätzlichen Bitfolge fester Länge, der Sicherungssequenz (FCS: Frame Check Sequence), wie es in Bild 6.2 gezeigt ist. Vorteil des Verfahrens ist, daß auch bei einfacher Realisierbarkeit sehr gute Fehlererkennungswahrscheinlichkeiten erreicht werden.

Das Verfahren verwendet Operationen der Modulo-2-Binärarithmetik, bei der keine Überträge von einer Stelle zur nächsthöheren weitergegeben werden.

Folgendes einfache Beispiel soll das Verfahren verdeutlichen. Feste Parameter der Kanalcodierung seien die Länge der Sicherungsfolge L=4 und der Divisor D=1 1001_2 (D ist L+1 stellig). Die Operation / ist die Division in Modulo-2-Binärarithmetik, + die Addition. Zu übertragen sei das Quellencodewort Q=11 0011_2 durch das Kanalcodewort K, das aus Q durch Anfügen der Sicherungsfolge S entsteht. K' ist das empfangene Kanalcodewort, Q' das beim Empfang ermittelte Quellencodewort, F=00 1100 0001_2 ein Übertragungsfehlerbitmuster.

- Senden:
  1) H:=Q*2^L; H=11 0011 0000 ! Linksschieben des Quellencodeworts um L Bitstellen
  2) S:=Rest(H/P); S=1001

```
N.R.: 11 0011 0000 // 1 1001 = 10 0001
      11 001
           1 0000
```



```
           1 1001
           0 1001 = Rest
```

  3) K:=H+S; K=11 0011 1001
- Übertragen (fehlerfrei): K'=K
- Empfangen:
  Q':=K'/2^L, Fehler:=Rest(K'/P)≠0;

```
N.R.: 11 0011 1001 // 1 1001 = 10 0001
      11 001
           1 1001
           1 1001
           0 0000 = Rest
```

- Übertragen (fehlerhaft): K'=K+F; K'=11 1111 1000
- Empfangen:
  Q':=K'/2^L, Fehler:=Rest(K'/P)≠0;

```
N.R.: 11 1111 1000 // 1 1001 = 10 1000
      11 001
         1101 1
         1100 1
            1 0000 = Rest ≠0!
```

Es werden in diesem Verfahren alle Verfälschungen mit einem Fehler F erkannt, der nicht durch P ohne Rest teilbar ist. Für einen zusammenhängenden Bündelfehler der Länge L+1 kann z.B. gezeigt werden, daß er nur mit der Wahrscheinlichkeit $1/2^{(L-1)}$ durch P teilbar ist.

In Normen werden die folgenden Prüfmuster vorgeschlagen:
- CRC-12: 1 1000 0000 1111_2,
- CRC-16: 1 1000 0000 0000 0011_2,
- CRC-CCITT: 1 0001 0000 0010 0001_2.

Abschließend soll angemerkt werden, daß in modernen Protokollen die Verfälschungssicherung dem eigentlichen Protokoll unterlagert ist. In einer untergeordneten und in sich geschlossenen Hilfsfunktion wird jede gesendete PDU mit der FCS versehen, und beim Eintreffen einer PDU auf Empfangsseite wird ebenfalls in einer untergeordneten, in sich geschlossenen Hilfsfunktion die FCS unter Prüfung der Nachricht auf Verfälschung entfernt. Wurde eine Verfälschung erkannt, wird der Empfang der PDU nicht weitergegeben. Durch Ignorieren der verfälschten Nachricht wird die Besonderheit der Verfälschung auf den Verlust der Nachricht abgebildet.

6.2.2 Sequenzzahl

Zur Erkennung von Zusammenhang-Besonderheiten, die sich auf eine festgelegte Reihenfolge der in einer Richtung im Verlauf der Zeit übertragenen Nachrichten beziehen, können die Nachrichten (z.B. mit 0 beginnend) anhand eines zusätzlichen Kontrollparameters 'Sequenzzahl' einfach durchnumeriert werden. Damit werden für den Empfänger Abweichungen der Empfangsreihenfolge von der Sendereihenfolge wie Überholen und Vertauschen oder Duplikation erkennbar.

Auch auf den Verlust von Nachrichten kann geschlossen werden, wenn die Differenz der Sequenzzahlen der zuletzt empfangenen Nachricht zu der einer noch ausstehenden so groß ist, daß eine Verzögerung unwahrscheinlich ist.

Bei intensivem PDU-Austausch können sehr schnell sehr hohe Sequenzzahlen erreicht werden. Dies erschwert die Darstellung der Sequenzzahl in der PDU und die Buchführung über empfangene und ausstehende PDUs beim Empfänger. Deshalb wird in der Regel eine Fenstertechnik angewendet. Die Sequenzzahlen werden in einer Modulo k - Arithmetik weitergeschaltet (üblich sind Modulo 8 und

Modulo 128), so daß sich die Zahlen im Wertebereich {0..(k-1)} periodisch wiederholen. Dies Verfahren ist allerdings nur anwendbar, wenn aufgrund anderer zusätzlicher Maßnahmen (Flußkontrolle, Quittierung, Neuanforderung) sichergestellt ist, daß zu erkennende Reihenfolgeabweichungen sich nur im aktuellen Fenster befinden.

Das im letzten Kapitel vorgestellte Alternating Bit Protokoll verwendet bereits einen derartigen Fenstermechanismus mit dem Parameter k=2. Über eine Sequenzzahlendifferenz von (k-1)=1 hinausgehende Reihenfolgeabweichungen werden durch positive Quittierung bzw. Neuanforderung und Wiederholung ausgeschlossen.

6.2.3 Zeitstempel

Im privaten und geschäftlichen Briefverkehr werden Sequenzzahlen sehr selten benutzt. Ihre Rolle nehmen hier Hinweise auf vorangegangene Briefe und die Angabe des Datums ein. Auch bei der Rechner-Rechner-Kommunikation können zum Erkennen von Reihenfolgen-Besonderheiten PDUs mit einem Kontrollparameter 'Zeitstempel' versehen werden, der die Sendezeit enthält. Hierdurch können nicht nur Reihenfolge-Besonderheiten erkannt werden, die sich auf die in einer Richtung zwischen zwei Parteien ausgetauschten Nachrichten beziehen, sondern es können auch bei einer Mehr-Parteien-Kooperationsbeziehung Nachrichten verschiedener Absender zueinander in Bezug gesetzt werden.

Problematisch bei Zeitstempeln ist die Zeitauflösung und die Ganggenauigkeit der Uhren. Im Vergleich zur Datum-Angabe auf Briefen (dort kann man davon ausgehen, daß nur ein Brief pro Tag und Absender innerhalb derselben Kooperationsbeziehung geschrieben wird und der Transport ebenfalls mindestens einen Tag in Anspruch nimmt), kann ein Rechner sehr schnell sehr viele Nachrichten erzeugen und absenden, und es steht oft auch ein sehr schnelles Medium zum Transport zur Verfügung. Deshalb kann z.B. eine Zeitauflösung im Milisekundenbereich notwendig werden. Bei dieser Auflösung wird die Ganggenauigkeit der verschiedenen zueinander entfernten Uhren zum Problem. Der Uhrengleichlauf und Abgleich führt zu hohem technischen Aufwand.

Eine technisch einfachere Variante, die sich konzeptuell wieder der Sequenzzahl annähert, ergibt sich durch logische Uhren, in denen nicht mehr die reale Zeit sondern Ordnungsnummern geführt werden. Die Fortschaltung der logischen Uhren bei den verschiedenen Parteien erfolgt nachrichtengesteuert, so daß eigens hierfür besondere Protokollmaßnahmen, auf die hier im einzelnen nicht näher eingegangen werden soll, erforderlich werden.

6.2.4 Quittieren

Reihenfolgen-Besonderheiten werden beim Empfänger durch Sequenzzahlen oder Zeitstempel erkennbar, Verfälschungen durch den Test des Kanalcodeworts. In den meisten Kooperationsbeziehungen ist die Tatsache, ob Besonderheiten aufgetreten sind, aber nicht nur für den Empfänger sondern auch für den Absender relevant. Dazu muß ein PDU-Austausch in Gegenrichtung, die Übertragung von positiven Quittungen (Bestätigungen), negativen Zurückweisungen oder Anforderungen ausstehender Nachrichten stattfinden.

Im groben gibt es die folgenden beiden Verfahren:

- **Stop and Go - Protokolle**; hier muß jede Nachricht durch eine Quittung in Gegenrichtung beantwortet werden. Der Sender einer Nachricht wartet auf das Eintreffen der Quittung. Die Quittung kann positiv oder negativ sein. Im positiven Fall fährt der Sender mit der nächsten Nachricht fort. Im negativen Fall wiederholt er die letzte Nachricht (vergleiche Alternating Bit Protokoll).

- **Sliding Window Protokolle**; hier kann ein Sender eine Folge mehrerer Nachrichten senden, ohne daß Bestätigungen vorliegen.

  Bei positiven Quittungen können unterschieden werden:
  - Einzelbestätigung; die Quittung bestätigt den erfolgreichen Empfang einer bestimmten Nachricht.
  - Sammelbestätigung; die Quittung bestätigt, daß eine Menge von Nachrichten (oft die Folge der Nachrichten bis zu einer angegebenen Sequenzzahl) erfolgreich übertragen wurden.

  Bei negativen Quittungen können unterschieden werden:
  - Einzelzurückweisung (Selective Reject); die Quittung teilt mit, daß eine bestimmte Nachricht noch nicht erfolgreich empfangen werden konnte und fordert damit die Wiederholung dieser Nachricht an.
  - Sammelzurückweisung; die Quittung teilt mit, daß einige Nachrichten aus einer Menge von Nachrichten (oft aus der Folge der Nachrichten bis zur angegebenen Sequenzzahl) bis jetzt nicht erfolgreich empfangen werden konnten und fordert die Wiederholung der Nachrichten aus der Menge an.

Varianten des Quittierens ergeben sich durch zusätzliche Konventionen. Z.B. kann vereinbart sein, vorrangig nur positive Quittungen auszutauschen. Ausstehende positive Quittungen wertet der Sender von Nachrichten als Aufforderungen zur Wiederholung.

Häufig findet der Austausch von Nutzdaten nicht nur in eine Richtung statt. Dann wird die Quittungsinformation oft den Nachrichten in Gegenrichtung beigepackt (Piggy Backing).

6.2.5 Zeitüberwachung

Der Verlust einer Nachricht kann dem Empfänger vermutbar werden, wenn es sich nicht um die letzte Nachricht einer Folge handelt und spätere Nachrichten schon eingetroffen sind. In diesem Fall kann er entsprechend quittieren und so die Besonderheit auch dem Absender mitteilen.

Kommen in einer Folge aber keine späteren Nachrichten mehr beim Empfänger an, z.B. weil das Medium längerfristig gestört ist oder keine anschließenden Nachrichten mehr abzusenden waren, kann bereits der Empfänger nicht auf den Verlust schließen. Weiterhin könnte der Empfänger zwar den Verlust vermutet und eine Quittung 'Anforderung' gesendet haben, aber diese Quittung könnte verlorengegangen sein. In diesen Fällen kann ein Verlust einer Nachricht auftreten, ohne daß der Sender vom Verlust erfährt, und dies kann u.U. zum Absterben der Aktivitäten der Schicht führen, weil der Sender auf eine Quittung wartet und der Empfänger auf Nachrichten. Da bei modernen Protokollen - wie bereits erwähnt - Verfälschung auf den Verlust zurückgeführt wird, müssen in solchen Protokollen unbedingt besondere Maßnahmen zur Vermeidung des Absterbens der Aktivität getroffen werden. Um sie anzustoßen, muß schon einem Sender der Verlust seiner Nachricht vermutbar werden.

Hierzu werden in der Regel Zeitüberwachungsmaßnahmen (Time Out Mechanisms) vorgesehen. Sendet ein Sender eine Nachrichten-PDU ab, startet er einen Kurzzeitwecker mit einer Laufzeit, innerhalb der mit hoher Wahrscheinlichkeit die Nachrichten-PDU zum Empfänger übertragen werden kann, vom Empfänger bearbeitet werden kann, der Empfänger eine Quittung-PDU erzeugen kann, die Quittung-PDU in Gegenrichtung übertragen werden kann und vom Sender erkannt werden kann. Dann wartet der Sender bis entweder die Quittung eintrifft (in diesem Fall stellt er den Wecker ab), oder bis die Laufzeit des Weckers abgelaufen ist. Im letzten Fall geht er davon aus, daß entweder Nachrichten-PDU oder Quittung-PDU verlorengegangen sind.

Ferner kann es sich natürlich auch nicht nur um einen bloßen Nachrichten-Verlust sondern darum handeln, daß Medium oder Partnerinstanz langfristig nicht betriebsbereit sind. Die Unterscheidung zwischen diesen beiden Fallklassen wird i.a. dadurch getroffen, daß von schwerwiegenden

langfristigen Störungen nur ausgegangen wird, wenn mehrere Nachrichten-Übertragungen in Folge zu Zeitüberschreitungen geführt haben.

Zeitüberwachungsmaßnahmen finden sich - wie erwähnt - in vielen modernen Protokollen. Sehr wichtig für die Leistungsfähigkeit eines Protokolls ist dabei die Wahl der Laufzeit der Wecker. Sie müssen in einem realen System unter Berücksichtigung der besonderen Stör- und Leistungseigenschaften des Mediums und der implementierungsabhängigen Reaktionszeiten der Instanzen sorgfältig gewählt werden. Sind die Zeiten zu kurz, kann dadurch jeder Protokollfortschritt verhindert werden, weil alle Übertragungen als verlustig angenommen werden. Sind die Zeiten bei doch recht häufigem Aufreten von Verlusten zu lang, ergeben sich lange Phasen der Untätigkeit von Instanzen und der Nicht-Nutzung des Mediums.

6.3 Fehlerbehebung

Bei der vorangegangen Erläuterung der wichtigsten Fehlererkennungsmechanismen wurden zum Teil schon Maßnahmen angesprochen, die - an die Fehlererkennung anschließend - zur Behebung eingesetzt werden (z.B. Ignorieren). In diesem Abschnitt sollen nun die wichtigen Behebungsmechanismen vorgestellt werden.

Für die Behebung von Verfälschungen werden im groben zwei Verfahren eingesetzt:
- FEC-Verfahren (Forward Error Correction),
- ARQ-Verfahren (Automated Repeat Request).

FEC-Verfahren basieren auf fehlerkorrigierenden Codes und werden in der Rechner-Rechner-Kommunikation selten eingesetzt. Sie sind in den Fällen interessant, in denen von der Anwendung des Zieldienstes her Nachrichtenlaufzeiten nicht toleriert werden können, die ein Mehrfaches der Nachrichtenlaufzeit des zur Verfügung stehenden Mediums betragen. In allen anderen Fällen verwendet man ARQ-Verfahren. Verfälschte Nachrichten werden wiederholt. Häufigste Variante dabei ist es, verfälschte Nachrichten durch Ignorieren auf den Verlust zurückzuführen, der dann durch Wiederholung der Übertragung behoben wird.

6.3.1 Fehlerkorrigierende Codes

Ähnlich der Erkennung von Nachrichtenverfälschungen durch fehlererkennende Codes kann im Kanalcode auch Redundanz vorgesehen werden, die die Korrektur von Verfälschungen unterstützt. Auch dies ist eine dem eigentlichen Protokoll unterlegte Maßnahme, die in in sich geschlossenen, untergeordneten Funktionen angesiedelt sein kann, und nicht in den Protokollablauf insgesamt eingreift. Da fehlerkorrigierende Codes in der Rechner-Rechner-Kommunikation zur Zeit nur in Spezialfällen eine Rolle spielen, soll hier nur das Grundprinzip kurz erwähnt werden.

Auch hier gliedert sich die Bildmenge des Kanalcodes in zulässige und unzulässige Codeworte. Eine Teilmenge der unzulässigen Codeworte sind die korrigierbaren. Sie zerfällt in Klassen, die jeweils genau einem zulässigen Codewort zugeordnet sind. Wird ein unzulässiges Kanalcodewort empfangen, das korrigierbar ist, wird der dem zugeordneten zulässigen Kanalcodewort entsprechende Quellencode ausgeliefert.

6.3.2 Ignorieren

Das Ignorieren einer empfangenen Nachricht behebt eigentlich keinen Fehler sondern überführt eine mit dem Nachrichtenempfang erkannte Besonderheit in die Besonderheit des Nachrichtenverlusts und trägt so zur Vereinfachung eines Protokolls bei.

Üblicherweise werden als verfälscht erkannte Nachrichten - wie schon oben mehrfach angesprochen - ignoriert. Weiterhin bietet sich das Ignorieren auch für Fälle an, in denen als unverfälscht erkannte PDUs empfangen werden, deren Kontrollparameterbelegung keine sinnvolle Einordnung in den momentan aus der Sicht der empfangenden Instanz bestehenden Zusammenhang zuläßt, kann also als Maßnahme zur Behandlung von Zusammenhang-Besonderheiten verwendet werden.

6.3.3 Wiederholung

Das Wiederholen des Sendens einer PDU bildet die wichtigste Maßnahme zur Behandlung von Reihenfolge-Besonderheiten (und - bei Ignorieren verfälschter Nachrichten - von Verfälschungen). Es wird in der Regel durch Zeitüberwachungsmaßnahmen oder Quittungen zur Anforderung von Nachrichten eingeleitet. Denkbar - aber aufgrund des hohen Übertragungsaufwands kaum eingesetzt - sind auch Protokolle, bei denen eine Instanz eine Nachricht solange wiederholt sendet, bis eine positive Quittung dafür eingetroffen ist.

Unterschieden werden kann, ob ein Wiederholungsmechanismus immer nur die zuletzt gesendete Nachricht betrifft, ob unter Umständen auch mehrere Nachrichten wiederholt werden, oder ob auch länger zurückliegende Nachrichten gezielt zur Widerholung ausgewählt werden können (vgl. Quittieren).

Praktisch eingesetzte Protokolle zählen die Anzahl der Wiederholungen einer Nachricht mit und brechen die Übertragungsversuche nach Erreichen eines Limits ab. Ein verbindungsorientiertes Sicherungsprotokoll wird so dann z.B. einen Verbindungsabbruch durchführen, weil davon ausgegangen wird, daß die gegenwärtige Verbindung aufgrund einer längerfristigen Medium-Störung nicht mehr unterstützt werden kann.

6.3.4 Rücksetzen

Bei der Wiederholung der zuletzt gesendeten Nachricht handelt es sich schon um einen Spezialfall des Rücksetzens. Eine Instanz, die in ihrer Sicht schon den Zustand 'Nachricht gesendet' erreicht hat, setzt - durch Zeitüberschreitung oder Anforderung veranlaßt - in den Zustand zurück, den sie unmittelbar vor dem Absenden der Nachricht eingenommen hat. Damit wird sie im nächsten Schritt dieselbe Nachricht wiederholt senden.

Wenn mit mehreren bereits ausgetauschten Nachrichten eine gewisse Übereinstimmung in der Sicht der betroffenen Instanzen hergestellt wurde, die für die Einordnung späterer Nachrichten von Interesse ist, und wenn dann schwerwiegende Besonderheiten (i.a. längerfristige Ausfälle von Instanzen oder Medien) auftreten, ist bei vielen Protokollen das Rücksetzen in einen u.U. mehrere Nachrichtenaustauschphasen zurückliegenden Betriebszustand vorgesehen.

Zum Rücksetzen setzt die Instanz, die die Besonderheiten erkannt hat, unter Absenden einer PDU, die vom Partner ebenfalls das Rücksetzen verlangt, in einen früheren lokalen Betriebszustand zurück. Haben beide Instanzen rückgesetzt, wurde gleichermaßen ein Stück der stattgefundenen Nachrichtenaustausch-Historie vergessen, und es soll ein globaler Betriebszustand hergestellt sein, bei dem die konsistente Sicht der Partner auf den Globalzustand gewährleistet ist.

6.4 Längenanpassung

Die eigentliche Nutzerinstanz einer verteilten Anwendung möchte komfortablerweise über den untergeordneten Kommunikationsdienst Nutzdaten in nur von der Anwendung und im Einzelfall bestimmter Länge austauschen. Dagegen steht (neben der Schnittstellen-Realisierungsproblematik bei sehr langen, mehrere MByte umfassenden Nutzdaten), daß die PDUs unterer Schichten (in die ja die Nutzdaten 'eingepackt' werden) aus Gründen einer ausgewogenen Dienstleistungsqualität in ihrer Länge nur in engeren Grenzen differieren sollen.

Werden die PDUs in unteren Schichten zu kurz, entsteht unnötiger Verpackungs- und Verwaltungsaufwand, der sich negativ auf den Durchsatz auswirkt. Werden sie so lang, daß sich Verfälschungen und damit i.a. Verluste häufen, wirken sich die Zeitverzögerungen, die bis zur Erkennung von Verlusten verstreichen, und der Aufwand zu Neuanforderung und Wiederholung negativ auf den Durchsatz aus. In den unteren Schichten müssen deshalb die unterstützten PDU-Längen sehr sorgfältig auf die Eigenschaften des verwendeten Mediums abgestimmt werden und sind in engen Grenzen vorgegeben.

Damit kann für die Instanzen einer Schicht die Aufgabe der Längenanpassung bestehen:
- Blocken,
- Fragmentieren.

Beim Blocken werden mehrere Zieldienst-Nutzdaten-Einheiten zu einem Nutzdaten-Block zusammengefaßt und in einer einzigen PDU übertragen. Die empfangende Instanz teilt den Block zur Auslieferung an ihren Dienstnehmer wieder in die ursprünglichen Einheiten auf.

Beim Fragmentieren geht es umgekehrt darum, lange Zieldienst-Nutzdaten in einer Folge kleinerer Fragmente anhand mehrerer PDUs zu übertragen. Hier fällt der empfangenden Instanz die Aufgabe des Wiederzusammensetzens zu.

6.5 Flußkontrolle

Bild 6.3 zeigt ein Szenario zur Erläuterung von Flußkontroll-Maßnahmen. In einer Phase des Betriebs einer Zieldienst-Verbindung habe ein Dienstnehmer (Sender) einen (längeren) Strom von Nutzdaten-Paketen an den anderen Dienstnehmer (Empfänger) zu übertragen. Er übergibt die Pakete nacheinander per Zieldienst-Datentransfer-Requests an den Zieldienst. Diese Requests werden von der P-Instanz (Sendeseite) entgegengenommen und bearbeitet. Sie betreibt eine Basisdienst-Verbindung zur P-Instanz (Empfangsseite). Sie erzeugt einen Strom von PDUs, die die Nutzdaten des Senders enthalten, und übergibt die PDUs nacheinander per Basisdienst-Datentransfer-Requests an den Basisdienst. Der Basisdienst liefert die PDUs an die P-Instanz (Empfangsseite) aus. Die packt die Nutzdaten aus und liefert die Nutzdaten nacheinander per Zieldienst-Datentransfer-Indication an den Empfänger aus.

Ist der Sender langsamer als die P-Instanz (Sendeseite), die langsamer als der Basisdienst, der langsamer als die P-Instanz (Empfangsseite), und die wiederum langsamer als der Empfänger, entstehen keine Probleme. Jede Instanz kann so schnell agieren, wie sie kann. Maßnahmen zur Steuerung des Flusses (der Anzahl produzierter / weitergegebener Pakete pro Zeiteinheit) werden nicht benötigt. Sind Produzenten schneller als Konsumenten, kann es jedoch zu Problemen kommen (z.B. Betriebsmittel / Puffer - Überlauf, Basisdienst-Überlastung, siehe 'Überlastkontrolle'). Um ihnen zu begegnen, werden Möglichkeiten benötigt, mit deren Hilfe der Fluß gesteuert werden kann.

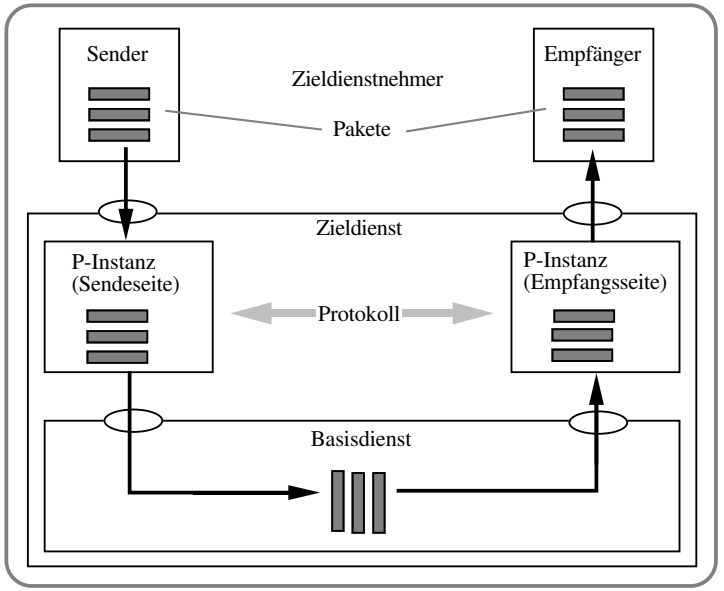

Bild 6.3: Flußkontroll-Szenario

Grob kann man unterscheiden:
- Verlust beim Empfänger,
- Flußbeschränkung beim Sender,
- Flußabstimmung zwischen Sender und Empfänger.

Verlust beim Empfänger findet einseitig bei einem Konsumenten statt. Alle Pakete, die bei ihm eintreffen, aber momentan weder bearbeitet noch gepuffert werden können, werden ignoriert, d.h. gehen verloren. Diese Maßnahme sollte in der Regel nur als selten anzuwendende Notmaßnahme vorgesehen werden. Die Erzeugung und der Transfer korrekt übertragener aber empfangsseitig ignorierter Nachrichten verursachen unnötigen Aufwand. Weiterhin ist zu beachten, daß nach einem Verlust oft andere Mechanismen (Verlusterkennung, Wiederholung) wirksam werden und die Erzeugung weiterer Nachrichten zur Folge haben.

Flußbeschränkung beim Sender findet einseitig bei einem Produzenten statt. Er mißt den Fluß, den er erzeugt (z.B. mittels Uhr und Mitzählen in [bit/s] oder [Pakete/s]). Es ist eine obere Grenze festgelegt, deren Überschreitung er falls nötig durch Verzögern der Produktion des nächsten Pakets vermeidet.

Eine Variante dazu ist das 'Lecker Eimer'-Verfahren. Der Grenzwert ist hier als Mittelwert gedacht, der kurzfristig überschritten werden darf. Ein Zähler wird mit jedem Erzeugen eines Pakets inkrementiert (Einfüllen in den Eimer). In konstanten Zeitintervallen wird der Zähler um einen konstanten Wert dekrementiert (konstanter Abfluß aus dem 'Leck' des Eimers). Übersteigt der Zähler einen Grenzwert, wird die Produktion verzögert (Vermeidung des Eimer-Überlaufs).

Flußabstimmung ist ein zweiseitiges Verfahren, in welchem ein Produzent und ein Konsument in Wechselwirkung stehen.

Wichtigster Mechanismus zur Flußkontrolle per Abstimmung ist die Kredit-Vergabe. Hier wird zunächst initial dem Sender vom Empfänger aus ein Kredit (eine positive Zahl mit der Einheit [bit] oder [Pakete]) eingeräumt. Der Sender führt Buch über seinen Kredit. Mit jeder Produktion vermindert der Sender seinen Kredit. Würde er seinen Kredit überziehen, verzögert er die Produktion. Der Empfänger führt ebenfalls Buch über den dem Sender eingeräumten Kredit und sendet dem Sender ab und zu eine Krediterhöhungsmitteilung zu.

Mit der Kreditzahl 1 ergibt sich der Spezialfall des 'Stop and Go'. Ein Produzent sendet ein Paket und wartet auf eine Quittung für dieses Paket. Dann sendet er das nächste. Eine Kreditzahl größer 1 wird bei den Sliding Window Protokollen verwendet. Hier harmonieren Sequenzzahlen in Fenstertechnik und Kredite gut zusammen, die die Fenstergröße nicht übersteigen.

Flußbeschränkung beim Sender und Flußabstimmung verzögern die Produktion zu schneller Sender, sie bewirken Rückstau beim Sender. Dies kann als Pendant zum Verlust beim Empfänger angesehen werden.

Bezugnehmend auf das Szenario in Bild 6.3 soll nun skizziert werden, welche Maßnahmen üblicherweise wo eingesetzt werden. In jeder konsumierenden Instanz (P-Instanz-Sendeseite, Basisdienst, P-Instanz-Empfangsseite, Empfänger) kann Verlust beim Empfänger als Notmaßnahme in der Ausnahmesituation des momentanen Betriebsmittelmangels vorgesehen werden. Flußkontrolle per Abstimmung über Kredite wird zwischen Protokoll-Partnern eingesetzt (zwischen Sender und Empfänger, zwischen P-Instanz-Sendeseite und P-Instanz-Empfangsseite). Die Kredithöhe richtet sich nach der Pufferungskapazität des Mediums und der der empfangenden Partnerinstanz. Vertikal (zwischen Sender und P-Instanz-Sendeseite, zwischen P-Instanz-Sendeseite und Basisdienst, zwischen Basisdienst und P-Instanz-Empfangsseite, zwischen P-Instanz-Empfangsseite und Empfänger) werden häufig in der logischen Architektur keine Maßnahmen vorgesehen. Es handelt sich um lokale Schnittstellen, deren Auslegung Implementierungssache ist. In Implementierungen verwendet man häufig Stop and Go, jedem logisch relevanten Ereignis folgt ein lokales Quitttungsereignis, das auch die Bereitschaft zur Annahme des nächsten Ereignisses signalisiert. Eine andere Möglichkeit ist es, die beim Konsumenten zur Pufferung benötigten Betriebsmittel bereits vom Produzenten reservieren zu lassen.

6.6 Weiterleiten

Bei der Weiterleitung (Routing) stellt sich das grundlegende Problem, daß ein Zieldienst (ein Netzdienst) direkte Transfers zwischen beliebigen Paaren einer Menge von Orten unterstützt, während das unterliegende Medium in separate Medien gegliedert ist, so daß hier nur direkte Transfers zwischen einer sehr eingegrenzten Menge von Ortspaaren, d.h. zwischen bestimmten Teilstrecken, möglich sind.

Weiterleitung ist ein Problem, für dessen umfassende Lösung verschiedenartigste und oft sehr komplexe Mechanismen verwendet weden, die hier im einzelnen nicht aufgeführt werden sollen (Protokolle mit Routing-Funktionen werden oft als komplexeste Protokolle überhaupt gewertet). Es soll im folgenden nur ein Einblick in Problem und mögliche Lösungen gegeben werden.

Die Abwicklung einer Zieldienst-Dienstleistung erfolgt in einer i.a. wesentlich mehr als zwei Instanzen betreffenden Kooperationsbeziehung. Um ein Nutzdatum von einem Ort A an einen Ort B zu übertragen, muß ein Weg über zur Verfügung stehende Teilstrecken gefunden werden. Es wird an

jedem dazwischenliegenden Teilstrecken-Endpunkt eine Instanz (Transitinstanz) berührt, die die Weiterleitung des Nutzdatums übernehmen muß.

Das Weiterleiten an sich ist einfach, es besteht aus dem Auspacken eines Nutzdatums aus einer empfangenen PDU, dem Einpacken in eine neue und dem Absenden der neuen auf der nächsten Teilstrecke. Die Weiterleitungsinformation kann dabei auf zwei Arten zur Verfügung gestellt werden:

- quellengesteuerte Weiterleitung,
- transitknotengesteuerte Weiterleitung.

Bei der quellengesteuerten Weiterleitung legt die quellenseitige Instanz den Weg der Nachricht durch das Netz fest und hinterlegt entsprechende Pfadinformation in der Nachricht. Ein Transitknoten leitet die Nachricht unter Auswertung der Pfadinformation weiter.

Bei der transitknotengesteuerten Weiterleitung enthält eine PDU keine Pfadinformation sondern nur die Adresse des Ziels. Jede Transitinstanz liest die Zielangabe einer angekommenen PDU, bestimmt selbst davon abhängig die nächste Teilstrecke und leitet die Nachricht weiter.

Zur Ermittlung der nächsten Teilstrecke können Wegetabellen verwendet werden. Üblicherweise werden dezentrale Wegetabellen, eine Tabelle je Transitinstanz am Ort der Instanz, vorgesehen. Hierdurch wird im Vergleich zu einer einzigen, für das ganze Netz verbindlichen Tabelle Kommunikationsaufwand (zum Zugriff auf die möglicherweise entfernte Tabelle) eingespart.

Die eigentliche Problematik ergibt sich dadurch, daß die Tabellen sinnvollerweise nicht statisch sein können, sondern in ihren Inhalten der aktuellen Situation des Netzes während des Betriebs angepaßt werden müssen (adaptives Routing). Nur so kann auf den Ausfall und die Wiedereingliederung von Teilstrecken und Knoten oder auf momentane Überlastung von Teilstrecken im Sinne einer Optimierung des Betriebs eingegangen werden. Die Abwicklung der Aktualisierung benötigt ihrerseits Kommunikationsaufwand. Ferner stellen sich hier Fragen nach den Auswirkungen zueinander inkonsistenter Tabellen. U.U. kann ein Netz sich in mehrere disjunkte Teilnetze spalten, oder ein Nutzdatum wird andauernd weitergereicht, ohne je sein Ziel zu erreichen.

Zwei wichtige Varianten der Weiterleitung unterscheiden sich darin, ob der Weg eines Nutzdatums individuell für dieses Nutzdatum gewählt wird (Speichervermittlung im engeren Sinn), oder ob in einer vorbereitenden Phase für eine nachfolgende Menge von Nutzdaten-Übertragungen ein Weg ermittelt und festgelegt wird (Leitungsvermittlung, z.B. bei der Abwicklung des Verbindungsaufbaus einer Zieldienst-Verbindung).

Bei sehr großen Netzen wird oft hierarchisches Routing verwendet. Das Netz ist in Subnetze gegliedert. Subnetze sind organisatorisch in sich geschlossene Einheiten, die dortigen Weiterleitungsfunktionen (auch Tabellen) beschränken sich auf das Subnetz. Das Gesamtnetz wird eine Schicht höher in der Inter-Netz-Schicht gebildet (Diese höhere Schicht liegt nach ISO/OSI ebenfalls in der Schicht 3). So wie Instanzen eines Subnetzes über Teilstrecken des Subnetzes weiterleiten, leiten Internetz-Instanzen über Subnetze weiter.

Eine einfache spezielle Lösung für Weiterleitungsprobleme ist das sogenannte Fluten. Eine Instanz, die sich nicht am Zielort eines Nutzdatums befindet, sendet das Datum auf allen ihr zugänglichen Teilstrecken weiter.

Das Fluten ist i.a. aufgrund des hohen Kommunikationsaufwands nicht oder nur unter starken Einschränkungen realisierbar. In dem speziellen Fall, in dem die verfügbaren Teilstrecken die Topologie eines unidirektionalen Rings bilden, ist es die natürliche Lösung des Problems. Ein Datum wird von Station zu Station weitergereicht bis es am Zielort eintrifft.

Bei einfachen, über einem einzigen Sammelkanal gebildeten Netzen wird Weiterleitung nicht erforderlich. Die Netzwerk-Funktion besteht hier in der empfangsseitigen Selektion einer per Sammelkanal überall empfangbaren Nachricht.

6.7 Überlastkontrolle

Überlastkontrolle beschäftigt sich damit, die Überlastung eines Dienstes zu vermeiden. Überlastung (auch Stau genannt) tritt ein, wenn bei steigender angebotener Last die tatsächlich übernommene Last, d.h. die Leistung, abnimmt.

Beispiele für Überlastverhalten sind:
- geteiltes Medium mit Zuteilung unter Konkurrenz,
- gesicherte Zweipunkt-Verbindung mit endlicher Pufferkapazität der beiden Sicherungsinstanzen, die durch Verlust eingehalten wird.

Zur Unterscheidung zwischen den Begriffen Überlastkontrolle und Flußkontrolle beachte man, daß sich Flußkontrolle auf Maßnahmen bezieht, die separat den Fluß einzelner Ströme beeinflussen, während Überlast-Kontrolle global für ein Gesamtnetz zu sehen ist.

Im groben können zwei Strategien zur Überlastkontrolle unterschieden werden:
- Ressourcen-Vorabreservierung zur Überlastvermeidung,
- Überlasterkennung und -bekämpfung.

Bei der Vorabreservierung wird man z.B. beim Aufbau einer Verbindung den Bedarf an zur Unterstützung notwendigen Ressourcen (Pufferkapazität in den Knoten, Übertragungsleistung der Verbindungen) vorab bestimmen und reservieren. Gelingt dies nicht, wird man den Aufbau der neuen Verbindung ablehnen.

Bei der Überlasterkennung und -bekämpfung läßt man in optimistischer Weise 'überbuchen' zu. Überlasterkennung erfolgt durch Beobachtung (Monitoring) des Netzverhaltens. Zur Bekämpfung bieten sich dann folgende Maßnahmen an:
- Nachrichten-Vernichtung,
- Nachrichtenstrom-Drosselung.

Ein einfacher Mechanismus zur Nachrichtenvernichtung ist es z.B. eine endliche Menge von Puffern je Station (besser je Ausgabeleitung einer Station) vorzusehen. Ist der Puffervorrat momentan erschöpft, werden neu ankommende Nachrichten ignoriert.

Zur Strom-Drosselung kann verwendet werden:
- Einsatz von Flußkontroll-Mechanismen, um die angebotene / übernommene Last zu beschränken, in den Varianten 'bei Verbindungsaufbau vorher berechnete Flußkontroll-Parameter' und 'adaptive Veränderung der Flußkontroll-Parameter über Drosselungsnachrichten an die Quelle',
- zirkulierende Berechtigungsmarken, die verbraucht werden, wenn eine Station eine Nachricht sendet und die generiert werden, wenn eine Station eine Nachricht empfängt.
- Einsatz adaptiver Routing-Verfahren bei Netzen, um lokale Engpässe zu umgehen,
- Veränderung anderer Protokoll-Parameter untergeordneter Schichten, um für das gegenwärtige Benutzungsprofil ein günstigeres Leistungsverhalten zu erzielen (z.B. Paketlänge, Time-Out Konstanten).

Überlastkontrolle wird oft im Zusammenhang mit der allgemeinen Verwaltung von Netzen / Transportsystemen gesehen (Network Management). Zur Aufrechterhaltung eines geordneten und effizienten Gesamtbetriebs werden in Niveau und Zeitdauer breite Spektren abdeckende Maßnahmen vorgesehen (z.B. von der langfristigen Beobachtung des Nutzerverhaltens abgeleitete Investitionsmaßnahmen / Netzumkonfigurierungen, Änderung von Routingtabellen um kurzfristige Störung einer Teilstrecke zu umgehen).

6.8 Übertragungsleistungsanpassung

Das Problem der 'Übertragungsleistungsanpassung' entsteht dadurch, daß die an der Zieldienst-Schnittstelle von Übertragungsaufträgen geforderte Übertragungsgeschwindigkeit zu der Geschwindigkeit differiert, die unter direkter Verwendung von Dienstleistungen des Basisdienstes erzielt werden kann. Mechanismen basieren auf dem Multiplexen:
- Bündeln von Basisdienstleistungen zur Leistungssteigerung,
- Mehrfachnutzen von Basisdienstleistungen zur Kostenreduktion.

Zur Erläuterung wird angenommen, daß Ziel- und Basisdienst verbindungsorientiert sind. Es bestünden mehrere Basisdienst-Verbindungen zwischen zwei Instanzen und beide Instanzen unterstützen mehrere gleichlaufende Zieldienst-Verbindungen.

Im einfachen Normalfall wird zur Unterstützung einer Zieldienst-Verbindung genau eine und immer dieselbe Basisdienst-Verbindung benutzt.

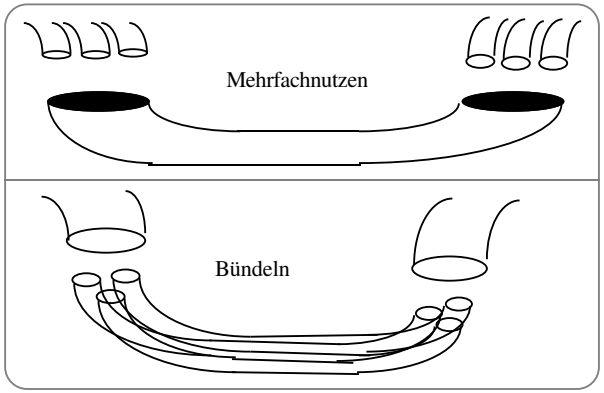

Bild 6.4: Bündeln und Mehrfachnutzen

Beim Bündeln soll eine Zieldienst-Verbindung mit hoher Leistungsfähigkeit unterstützt werden, während nur Basisdienst-Verbindungen mit geringer Leistungsfähigkeit zur Verfügung stehen. Es werden mehrere gleichlaufende Basisdienst-Verbindungen zur Unterstützung derselben Zieldienstverbindung eingesetzt.

Beim Mehrfachnutzen sollen mehrere gleichlaufende Zieldienst-Verbindungen mit geringer Übertragungsleistung (und geringen Kosten) unterstützt werden, während nur (teure) sehr leistungsfähige Basisdienst-Verbindungen zur Verfügung stehen. Hier wird dieselbe Basisdienst-Verbindung zur Unterstützung mehrerer Zieldienst-Verbindungen eingesetzt.

Bild 6.4 soll Bündeln und Mehrfachnutzen veranschaulichen.

6.9 Schutz und Sicherheit

Der Aufgabenkomplex 'Schutz und Sicherheit' wird für große Rechnernetze in dem Maße immer wichtiger, in dem die unterstützten Anwendungen sensibler werden. Im Jahr 1989 wurde das

ISO/OSI-Referenzmodell deshalb um eine Sicherheitsarchitektur ergänzt. Ein kurzer Abriß soll zur groben Übersicht dienen.
Die ISO/OSI-Sicherheitsarchitektur identifiziert 5 Sicherheitsdienste in unterschiedlicher Ausprägung:

- Authentifikation:
  - ° Ursprungsauthentifikation soll bei Datagrammen sicherstellen, daß Absenderangabe und Herkunft des Datagramms identisch sind.
  - ° Partnerauthetifikation soll sicherstellen, daß die im Rahmen einer Verbindung / Assoziation kommunizierenden Instanzen sich nicht gegenseitig in ihrer Identität täuschen.
- Zugriffskontrolle:
  Betriebsmittel des Kommunikationssystems dürfen nur von dafür Berechtigten manipuliert werden.
- Vertraulichkeit:
  Passive Mithörangriffe auf die stattfindende Kommunikation sind zu verhindern.
  - ° Schutz der übertragenen Nachrichten in einer Verbindung,
  - ° Schutz von Datagrammen,
  - ° Selektiver Schutz einzelner Komponenten von Nachrichten,
  - ° Schutz gegen Verkehrsfluß-Analysen.
- Integrität:
  Aktive Angriffe auf den Nachrichtenaustausch sind zu verhindern:
  - ° Datagramm-Integrität sichert daß empfangene Nachrichten so gesendet wurden.
  - ° Datagramm-Feldintegrität bezieht sich auf die unverfälschte Übermittlung von Datagramm-Komponenten.
  - ° Verbindungsintegrität sichert, daß alle empfangenen Nachrichten in derselben Anzahl und Reihenfolge und mit demselben Inhalt gesendet wurden (keine Verfälschungen, Einspielungen, Wiedereinspielungen und Verluste).
  - ° Selektiver Schutz einzelner Komponenten von Verbindungsnachrichten.
- Verbindlichkeit:
  Rechtsverbindlicher Informationsaustausch:
  - ° Verbindlichkeit mit Ursprungsbeweis stellt sicher, daß ein Absender das Absenden nicht abstreiten kann.
  - ° Verbindlichkeit mit Empfangsbeweis stellt sicher, daß ein Empfänger den Empfang nicht abstreiten kann.

Die zur Erbringung dieser Funktionalität eingesetzten Mechanismen sind zahlreich. Es sollen nur die wichtigsten angerissen werden:

- Verschlüsselung
  Symmetrische Verfahren verwenden einen (geheimen) Schlüssel zur Chiffrierung bzw. Dechiffrierung bei der Quelle bzw. Senke eines Transfers und werden traditionell zum effizienten Vertraulichkeitsschutz angewendet.
  Verschlüsselte Prüfsummen können Verfälschungen erkennbar werden lassen.
  Asymmetrische Verfahren mit öffentlichem Chiffrierschlüssel und geheimem Dechiffrierschlüssel erlauben ebenfalls Vertraulichkeitsschutz und gestatten, daß der Chiffrierschlüssel nicht in vertraulicher Weise kommuniziert werden muß.
  Asymmetrische Verfahren mit geheimem Chiffrierschlüssel und öffentlichem Dechriffrierschlüssel können die Grundlage zum Authentizitätsschutz und für Ursprungsbeweise bilden.
- Verschlüsselungsverfahren allein genügen nicht zur Erbringung der Schutzfunktion. Sicherheitsprotokolle (z.B. Authentifikationsprotokolle, Schlüsselverteilprotokolle) kombinieren initiale Schlüsselverteilungen, Verschlüsselungsverfahren und Ablaufkonventionen um unter gegebenen Randbedingungen einer Menge von Teilnehmern geschützte Kommunikation zu gestatten bzw. ihnen die dafür benötigten Voraussetzungen (Einträge in Verzeichnisse, Schlüsselkenntnisse) zu schaffen.
- Zugriffskontrollmechanismen orientieren sich an den Mechanismen modernerer Mehr-Nutzer-Betriebssysteme (z.B. Zugriffskontroll-Listen, Capabilities, befristete Tickets).
- Verkehrstopfungsmechanismen bringen irrelevante Nachrichten in Umlauf um Verkehrsflußanalysen zu täuschen.

- Physikalisch gegen Eingriffe geschützte Subsysteme und von vertrauenswürdiger Seite betriebene Subsysteme erbringen Grundfunktionen (z.B. Chipkarten, aber auch Verzeichnisdienste, Authentizitätsdienste und Notariatsdienste).

Die Zuordnung der verschiedenen Mechanismen in die 7 ISO/OSI-Schichten ist nicht zwangsläufig. Sehr anwendungsnahe Funktionen wie z.B. Verbindlichkeit wird man natürlich der Schicht 7 zuordnen. Einfachere, transferbezogene Funktionen wie z.B. Vertraulichkeitsschutz durch Verschlüsselung werden vorzugsweise mediumnah angesiedelt, können unter Umständen aber auch in Schicht 7 erbracht werden.

6.10 Verwaltung

Unter dem Stichwort Netzwerk-Management werden die Aktivitäten zur Steuerung, Koordination und Überwachung der Betriebsmittel eines Datennetzes und oft auch im weiteren Sinn auch eines Rechnernetzes insgesamt zusammengefaßt. Sie zielen auf die kontinuierliche Verfügbarkeit eines ausfallsicheren Netzes mit bestmöglicher Qualität ab.

Zur Standardisierung wurde das 'ISO/OSI Management Framework' entwickelt. Es gliedert die Verwaltungsaufgaben in fünf Bereiche:

- Fehlerverwaltung
  Erkennen, Analyse und Beheben von Fehlern und Fehlerzuständen.
- Abrechnungsverwaltung
  Quoten und Abrechung.
- Konfigurationsverwaltung
  Dokumentation und Veränderung der Netz-Konfiguration.
- Leistungsverwaltung
  Messen von Leistungsparametern, Leistungsanalyse, Leistungsvorhersage und Maßnahmen zur Leistungsteigerung.
- Sicherheitsverwaltung
  Verwaltung der Sicherheitsdienste, Beobachtung und Dokumentation sicherheitsrelevanter Ereignisse im Netz.

Es handelt sich um umfangreiche und Schichten-übergreifend zu lösende Aufgaben. Sie können i.allg. nicht durch einzelne Protokollmechanismen gelöst werden, sondern werden von Management-Systemen behandelt, die aus Anwendungsprozessen bestehen und in der Regel auch mit Menschen, den Netzverwaltern, in Dialog stehen. Deshalb ist der vorliegende Abschnitt im Kapitel 'Protokollmechanismen' eigentlich am falschen Ort. Nachdem wir hier aber z.B. Mechanismen zur Flußkontrolle, zur Zeitüberwachung und zur Wiederholung kennengelernt haben, scheint es sinnvoll, abschließend auch einen Blick darauf zu werfen, daß Parameter und Betriebsmittel der Mechanismen (wie z.B. Kreditmaxima, Zeitkonstanten und Puffervorrat zum Speichern der gegf. zu wiederholenden Nachrichten) auch während des laufenden Betriebs von Mangement-Eingriffen beeinflußt werden können.

Zur Erfüllung der Verwaltungsaufgaben müssen Möglichkeiten geschaffen werden, die Betriebsmittel an ihrem jeweiligen Ort zu beobachten und zu manipulieren. Dies sind im wesentlichen lokale Funktionen deren genaue Auslegung von der tatsächlichen Implementierung abhängt und nicht im Blickfeld der Standardisierung offener Systeme liegt.
Von Interesse für die Kommunikation offener Systeme ist der Austausch von Management-Informationen zwischen Stationen. Es werden Zustandsmeldungen, Anfragen und Eingriffsaufträge kommuniziert und die Darstellung der Informationen in Management-Nachrichten, die abstrakte Semantik der Management-Nachrichten sowie Ablauffestlegungen sind von Interesse. Dies führt zu Management-

Kommunikationsdiensten, Management-Kommunikationsprotokollen und einer abstrakten Sicht auf die verwalteten Betriebsmittel, der Management Informationsbasis, die in Managed Objects - also abstrakte Betriebsmittel - strukturiert ist.
Der Nutzer, d.h. der Verwalter eines Netzes, möchte natürlich in komfortabler Weise ein Netz rechnergestützt kontrollien und dazu Werkzeuge zur interaktiven und / oder automatischen Netzkontrolle einsetzen. Dies sind z.B. Monitorprogramme zu Status- und Ereignisverfolgung, 'regelnde' und alarmgebende Programme, Programme zur Journalauswertung, Programme zur interaktiven Beeinflussung. Die Werkzeuge sind in der Sicht des ISO/OSI-Modells Anwendungen (gehören also in unserer Sicht zur 'Schicht 8'). Die vorgenannten Management-Kommunikationsdienste dienen der Kommunikation zwischen den Prozessen einer solchen Anwendung.

Im ISO/OSI Architekturvorschlag können die Prozesse einer Magement-Anwendung bezüglich einer Kommunikationsbeziehung in zwei Rollen auftreten (es ist möglich, daß derselbe Prozeß bezüglich unterschiedlicher Assoziationen unterschiedliche Rollen einnimmt):

- Manager-Prozeß, der übergeordnete, steuernde Partner; er sendet Anforderungen / Requests.
- Agenten-Prozeß, der reagierende Partner; er reagiert auf Anforderungen mit Antwort-Nachrichten, kann aber auch von sich aus Ereignismeldungen liefern. Er greift lokal auf die ihm zugeordneten Betriebsmittel zu.

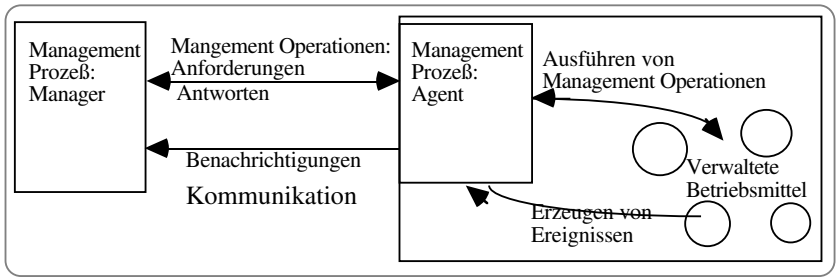

Bild 6.5: Management Prozesse

Bild 6.5 zeigt diese Rollenaufteilung und verdeutlicht die Einordnung von Kommunikation, Betriebsmitteln und Betriebsmittelmanipulationen.

Der IS/OSI-Vorschlag präzisiert die Architektur weitergehend durch Einteilung der Management-Prozesse nach ihrem Zweck in eine Dreier-Hierarchie:

- Prozesse zur Verwaltung einzelner Instanzen des Kommunikationssystems (N-Layer-Operation); Ein solcher Prozeß ist z.B. der Netzwerk-Instanz eines bestimmten Netzknotens zugeordnet und führt lokal Eingriffe in diese Instanz aus (z.B. Kreditparameter, Pufferreservierung).
- Prozesse zur integralen Verwaltung einer Schicht (N-Layer-Management); das Netzwerk-Schicht-Management enthält z.B. die Funktionen zur Prüfung der insgesamt verfügbaren Schicht-2-Verbindungen, zur Administration der Routing-Tabellen des Gesamtnetzes, zur Beobachtung des Gesamt-Datennetzes.
- Prozesse zur integralen Verwaltung des Gesamtsystems (Systems Management); das Systems Management ist die hierarchisch höchste Gruppe und erlaubt es, koordiniert in mehrere Schichten einzugreifen.

Das Management Informationsmodell führt das Schema der Management Informationsbasis ein. Es ist ein objektorientiertes Schema, die Objekte entsprechen den verwalteten Betriebsmitteln und fassen deren Manipulations- und Beobachtungsoperationen sowie ihre Informationsattribute zusammen. Wesentlich ist weiterhin, daß ein hierarchisches Namensschema zur Objektidentifikation / Adressierung verwendet wird.

Vor diesem Hintergrung der Gliederung eines Management Systems in die Dreier-Hierarchie, der beiden Rollen einer Kommunikationsbeziehung zwischen zwei Mangement Prozessen und der Informationsbasis können nun verschiedene Management-Kommunikationsdienste definiert werden, die durch Mangement-Kommunikationsprotokolle zu erbringen sind.
So ist z.B. CMIS (Common Management Information Service) ein Schicht-7-Kommunikationsdienst, der Dienstleistungstypen zum Austausch von Ereignisbenachrichtigungen, von Operationsaufrufen und Antworten, von Objekt-Erzeug- und Löschaufträgen sowie von Aktionsaufträgen dient.
CMIP (Common Management Information Protocol) als zugeordnetes Protokoll regelt die Erbringung dieser Dienstleistungen. Es ist ein Schicht-7-Protokoll, seine Festlegungen betreffen im wesentlichen die Syntax der auszutauschenden PDUs.

7. Dienste und Protokolle - Beispiele

Im letzten Kapitel wurden die grundlegenden Konzepte zur logischen Architektur von Telekommunikationssystemen eingeführt. Sie entsprachen bereits den wichtigsten Festlegungen des ISO/OSI-Modells. Im vorliegenden Kapitel soll nun näher auf die wichtigsten zur Zeit in Standardisierung und Gebrauch befindlichen Dienste und Protokolle eingegangen werden.

Diese Darstellung ist allerdings, um sie im Umfang zu beschränken, weder vollständig noch detailliert. Insbesondere die Protokolle werden nur sehr grob angesprochen (hier empfiehlt es sich immer bei Bedarf die entsprechende Norm hinzuzuziehen).

Basis- und Zieldienst einer Schicht können als Definition der Aufgaben des Protokolls angesehen werden. Den Instanzen einer Schicht 2 stehen zur Kommunikation z.B. nur die Dienstleistungen eines nachrichtentechnischen Kanals zur Verfügung, und das entsprechende Protokoll soll gewährleisten, daß die Instanzen einen bezüglich Übertragungsfehlern wesentlich zuverlässigeren Dienst erbringen, wie er einem gesicherten Kanal entspricht.

Dies bestimmt bereits im groben die Eigenschaften eines geeigneten Protokolls, es muß z.B. unbedingt Mechanismen zur Sicherung gegen Verfälschungen enthalten. Welche Sicherungsmechanismen im einzelnen zu wählen sind, und wie sie mit anderen ebenfalls benötigten Mechanismen (z.B. Flußkontrolle und Sequenzzahlen) sinnvoll und effizienzsteigernd zu kombinieren sind, also die genaue Auslegung des Protokolls, hängt von weiteren Randbedingungen ab, wie den Leistungscharakteristiken der Dienste und den bei der Realisierung verfügbaren Betriebsmitteln und deren Kosten.

Die wichtigen in der Praxis genutzten Protokolle zeichnen sich dadurch aus, daß verschiedene Mechanismen im Sinne einer Effizienzsteigerung kombiniert sind, und sehen darüberhinaus Varianten zur Berücksichtigung spezieller Randbedingungen vor. Sie sind hierdurch i.a. sehr komplex und sollen nicht im Detail vorgestellt werden.

Im folgenden wird deshalb nur eine kurze Übersicht darüber gegeben, welche Protokollmechanismen in welchen Schichten vornehmlich Anwendung finden, und es werden einige wichtige Protokolle dem Namen nach erwähnt und grob eingeordnet.

7.1 Dienstleistungen der Sicherungsschicht

Die Sicherungsschicht bietet verbindungsorientierte und / oder datagrammartige Dienstleistungen zum Nachrichtenaustausch an.

Die Datagramm-Dienstleistung ist unbestätigt:

- UnitDataReq (Absenderadresse, Empfängeradresse, geforderte Dienstleistungsqualitäten, Nutzdaten),
- UnitDataInd (Absenderadresse, Empfängeradresse, Dienstleistungsqualitäten, Nutzdaten).

Auslieferungsreihenfolgentreue, Verhinderung von Verlust von Nachrichten und Abwesenheit von Phantomnachrichten und Duplikaten werden nicht garantiert.

Die verbindungsorientierten Dienstleistungen sind:

- bestätigter Verbindungsaufbau (Connect) mit Aushandlung der Verbindungseigenschaften,
- unbestätigter Verbindungsabbau - gleichbedeutend mit Verbindungsabbruch - (Disconnect),
- unbestätigter Transfer von Nutzdaten (DataTransfer),
- unbestätigter Transfer höherpriorer Nutzdaten (ExpeditedDataTransfer),
- bestätigtes Rücksetzen einer Verbindung (Reset),
- Fehleranzeige (ErrorReport).

Zu den Dienstleistungen Connect, Disconnect, DataTransfer und ExpeditedDataTransfer vergleiche man die allgemeinen Ausführungen zu verbindungsorientierten Diensten in Kap. 3.

Das Rücksetzen überträgt auf dem Weg vom Initiator zum Beantworter der Dienstleistung zwei Parameter, die Adresse des Initiators und ein Datum zur Codierung des Rücksetzgrunds. Es löscht alle momentan anhängigen Dienstleistungen.

Die Fehleranzeige benachrichtigt die Dienstnehmer über das Auftreten besonderer Fehlersituationen, die mit dem Verlust von Nachrichten einhergehen können.

7.2 Protokolle der Sicherungsschicht

Ein Protokoll der Sicherungsschicht hat im wesentlichen die Verdeckung von Übertragungsfehlern zu leisten. Hierzu werden Codiermaßnahmen (Verfälschungserkennung), Sequenzzahlen (Reihenfolgebesonderheiten, Verluste), Zeitüberwachung (Verluste), Wiederholen und Rücksetzen eingesetzt. Weiterhin empfehlen sich Flußkontrollmaßnahmen, um Überlastungen des nachrichtentechnischen Kanals und insbesondere Überlastungen des empfangenden Partners durch Rückstau bei der Übernahme von Zieldienst-Aufträgen vorzubeugen.

Handelt es sich bei dem verwendeten nachrichtentechnischen Kanal um ein geteiltes Medium muß ferner noch die Aufgabe der Zuteilung gelöst werden. Maßnahmen hierzu können direkt in das eigentliche Sicherungsprotokoll integriert sein, oder in einer Subschicht der Sicherungsschicht untergeordnet abgehandelt werden.

Weitverkehrsnetze

Wichtigstes Protokoll ist das HDLC-Protokoll (High Level Data Link Control, ISO IS4335). Es ist zur Verwendung bei Zwei-Punkt-Medien geeignet, sieht aber auch die Verwendung von Mehr-Punkt-Medien vor, wobei dort dann unter dem Aspekt der Zuteilung zwischen einer Leitstation und Unterstationen unterschieden wird.

Die übrigen wichtigen Sicherungsprotokolle orientieren sich eng an HDLC und können als Varianten bezeichnet werden. Zu nennen ist LAP-B (Link Access Procedure - Balanced), das vom CCITT im Rahmen der X.25-Empfehlung (öffentliche Datenpaketvermittlungsnetze) für die Sicherungsschicht vorgesehen ist. Ferner besitzt auch das LLC-Protokoll (Logical Link Control), das in der IEEE 802-Norm und bei ISO/OSI als Normvorläufer (ISO DIS 8802/2) für lokale Netze enthalten ist, große Ähnlichkeit.

Ältere aus der 'Datenfernverarbeitungsära' stammende Protokolle der Sicherungsschicht sind die sogenannten zeichenorientierten Protokolle. Es handelt sich um Stop and Go Protokolle. Zur Verfälschungserkennung werden Zeichenparitätsbits sowie optional zusätzlich ein Block-Querparitätszeichen eingesetzt (Kreuzsicherung). Zeichenorientiert heißt, daß nicht Bitmuster sondern Zeichenfolgen (z.B. aus dem ISO 7-bit Code) übertragen werden. Einige Steuerzeichen des Codes

haben dabei direkten Kontrollparameter- und Formatierungscharakter für die PDUs des Protokolls (z.B. SYN, ENQ, SOH, STX, EOT, EOB, ETX).

Lokale Netze

Lokale Netze entsprechen üblicherweise der IEEE 802 - Norm (auch ISO/OSI 8802), die die Schichten 0 bis 3 abdeckt.

In den Schichten 0 und 1 werden dazu das einzusetzende physikalische Medium und Signalformen festgelegt. Dabei werden verschiedene Medien alternativ vorgeschlagen: ein gemeinsames Koaxialkabel unter Basisband- oder Breitbandübertragung, gekoppelte Koaxialkabel, Glasfaser-Strecken, eine gemeinsame verdrillte Zweidraht-Leitung oder ein Ring aus Zwei-Punkt-Zweidraht-Leitungen.

Die Schicht 2 wird in zwei Schichten gegliedert. Der Zugang zum Medium und die Zuteilung des geteilten Mediums sind Aufgaben der unteren, MAC (Medium Access Control) genannten Teilschicht. Sie ist spezifisch entsprechend dem eingesetzten physikalischen Medium und seiner Topologie ausgelegt. Es werden grob drei Zugangsverfahren unterschieden: CSMA/CD (Carrier Sense Multiple Access / Collision Detection), Token Ring und Token Bus. CSMA/CD ist hierbei ein Konkurrenzverfahren mit Mithören, Token Ring und Token Bus sind Verfahren mit zirkulierendem Senderecht.

Oberhalb der MAC-Schicht soll für alle Alternativen einheitlich das HDLC-artige Sicherungsprotokoll LLC eingesetzt werden.

7.3 Dienstleistungen der Netzwerkschicht

Die Dienstleistungen der Netzwerkschicht sind mit den Dienstleistungen der Sicherungsschicht vergleichbar. Der wesentliche Unterschied liegt an der Menge unterstützter Zugangspunkte.

Auch hier werden ein nur den Typ der Datagramm-Dienstleistung vorsehender Datagramm-Dienst und ein verbindungsorientierter Dienst vorgeschlagen (CONS: Connection-Oriented Network Service, CLNS: Connectionless Network Service).

Die verbindungsorientierten Dienstleistungen umfassen die vom Sicherungsdienst bekannten Dienstleistungstypen Connect, Disconnect, DataTransfer, ExpeditedDataTransfer und Reset. Zusätzlich ist der Typ der Datenempfangsbestätigung (DataAcknowledge) definiert:

- DataAcknowledgeReq (keine Parameter),
- DataAcknowledgeInd (keine Parameter).

Und zwar kann, wenn laut ausgehandelten Verbindungseigenschaften diese Dienstleistung unterstützt werden soll, eine Datentransfer-Anforderung (DataTransferReq) mit einem Parameter zur Bestätigungsanforderung versehen werden. Der empfangende Dienstnehmer soll dann den Empfang ausdrücklich per DataAcknowledgeReq bestätigen, das beim Absender zur Anzeige DataAcknowledgeInd führen wird.

7.4 Protokolle der Netzwerkschicht

Hauptfunktion eines Netzwerkprotokolls ist die Weiterleitung. Aus einer Menge von Teilstrecken oder Subnetzen soll ein einheitliches Netz entstehen. Nachrangig werden i.a. Flußkontrollmechanismen und solche zur Behebung von Reihenfolge-Besonderheiten eingesetzt. Weiterhin ist oft davon

auszugehen, daß einer Vielzahl gleichzeitig angeforderter Netzwerk-Verbindungen nur eine eng begrenzte Anzahl von Verbindungen des Sicherungsdienstes gegenübersteht. Hierzu wird ein Multiplex-Mechanismus in der Form des Mehrfachnutzens von Verbindungen des Sicherungsdienstes erforderlich.

Wichtigstes Beispiel für ein Netzwerkprotokoll sind die Schicht 3 - Festlegungen der X.25-Empfehlung des CCITT (öffentliche Datenpaketvermittlungsnetze), die für alle vorgenannten Funktionen Mechanismen enthalten.

Sehr oft ist ein Netz aus organisatorischen Gründen aus - auch unterschiedlichen - Subnetzen zusammengesetzt. Innerhalb jedes Subnetzes ist dann eine eigene Netzwerkschicht ausgebildet, die die subnetz-interne Weiterleitung gewährleistet. Das Gesamtnetz wird über den Subnetzen durch Aufsetzen einer (ebenfalls der Schicht 3 zuzuordnenden) Internetz-Schicht gebildet. Wird bei allen Subnetzen ein einheitliches Adreßschema verfolgt, und folgt die Zwischennetz-Wegfindung einfachen Prinzipien (z.B. Übergang zwischen Subnetzen nur an wenigen sog. Gateway-Stationen) so kann die Internetz-Schicht funktionell sehr einfach ausfallen und die entsprechenden Protokollkonventionen konzentrieren sich auf die Funktionalität der Gateway-Instanzen. Ein Beispiel dafür ist die im Zusammenhang mit der X.25-Empfehlung stehende CCITT-Empfehlung X.75, nach der der Übergang zwischen verschiedenen nationalen Datenpaketvermittlungsnetzen ausgelegt wird.

Auf die Netzwerkfunktionen zur Kopplung von lokalen Subnetzen soll im folgenden etwas ausführlicher eingegangen werden.

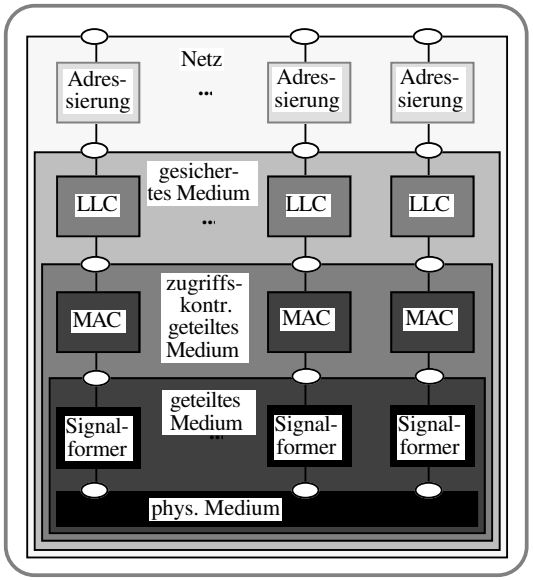

Bild 7.1: Lokales Netz

7.4.1 Lokale Netze

Bild 7.1 zeigt ein einfaches lokales Netz, das auf der Verwendung eines einzigen physikalischen Mediums basiert, und wie es z.B. einem einfachen Ethernet-Netz nach der Norm IEEE 802 entspricht (Medium: Koaxialkabel, Zugangsverfahren CSMA/CD).

Schicht 0 wird vom physikalischen Medium gebildet. Auf Schicht 1 befinden sich Sender und Empfänger, mit denen dem Medium Signale aufgeprägt bzw. Signale vom Medium abgetastet werden können. Hierdurch wird ein geteilter nachrichtentechnischer Kanal realisiert. Die untere Subschicht der Schicht 2 übernimmt die Zugangskontrolle (MAC) und stellt einen zugriffskontrollierten geteilten Kanal zur Verfügung. Die obere Subschicht von Schicht 2 dient der Realisierung gesicherter Kanäle zwischen beliebigen, vom physikalischen Medium berührten Orten (LLC).

Die Netzwerkschicht ist von den zu realisierenden Funktionen her gesehen im wesentlichen leer. Das Netz bietet unmittelbar die Dienstleistungen des gesicherten Mediums nach oben hin an. Logisch sind der Netzwerkschicht die Adressierkonventionen für Netzzugangspunkte zuzuordnen. Das Adressierschema ist verträglich zu dem auf LLC-Ebene verwendeten.

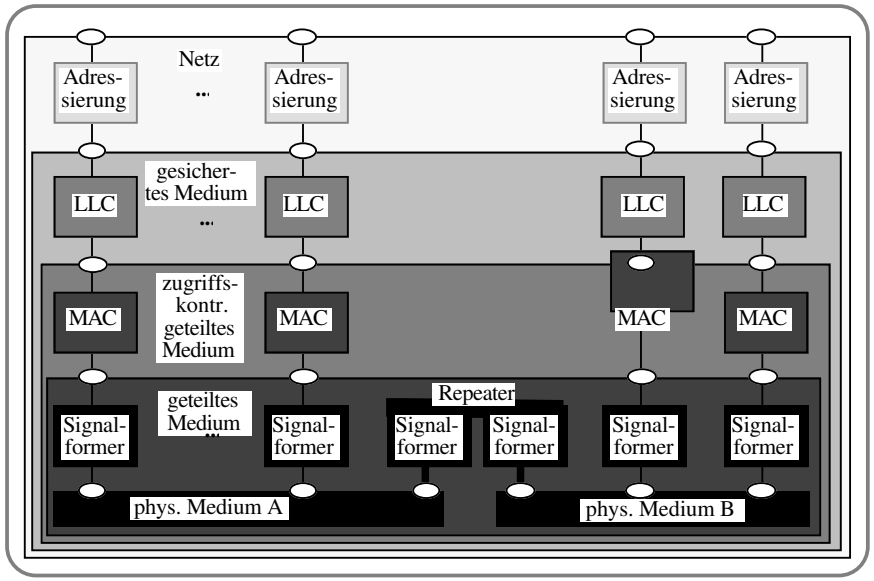

Bild 7.2: Repeater

7.4.1.1 Kopplung physikalischer Medien

Oft ergibt sich bei lokalen Netzen die Problematik, daß aufgrund der Längenbeschränkung für physikalische Medien (Dämpfung, Signal/Störabstand, Laufzeiten) bei größerer räumlicher Ausdehnung ein Medium allein nicht mehr verwendet werden kann.

Zur einfachen Lösung dieses Problems werden dazu Koppler angeboten, die gleichartige physikalische Medien unter Signalverstärkung koppeln (sog. Repeater). Bild 7.2 zeigt eine entsprechende Architektur. Der Repeater detektiert die auf einem Medium ankommenden Signale und gibt sie verstärkt, aber unverzüglich an das andere Medium weiter und umgekehrt. Protokollen der Schichten 2 und höher bleibt dies verdeckt. Und dieser Abschnitt hätte deshalb strenggenommen nicht in die Beschreibung der Netzwerkschicht aufgenommen werden dürfen.

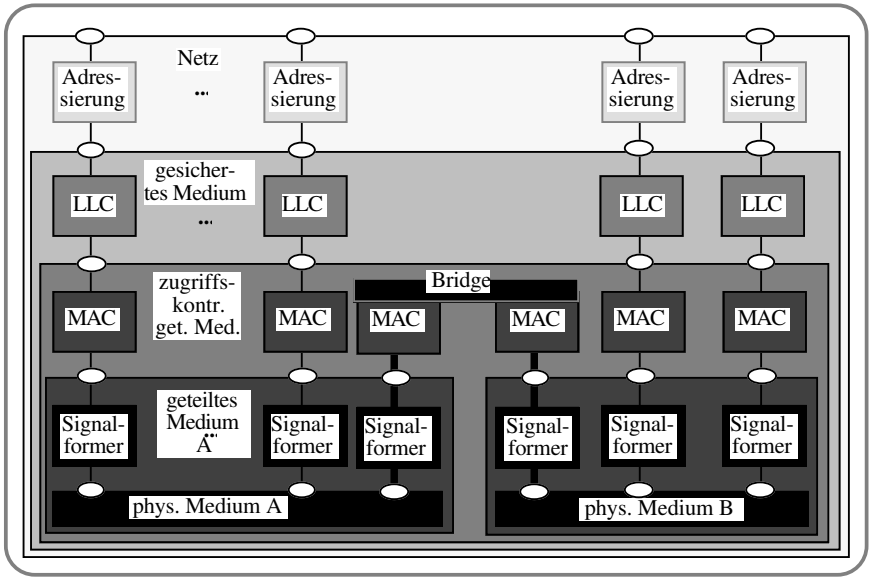

Bild 7.3: Bridge

7.4.1.2 Kopplung zugriffskontrollierter Medien

Bei Vergrößerung eines Netzes nimmt i.a. auch der Nachrichtenverkehr zu. Die einfache Repeater-Lösung besitzt hier den Nachteil, daß jeder Nachrichtentransfer alle physikalischen Medien belastet. Besser wäre es, nur die Nachrichten von einem Medium zu dem anderen weiterzuleiten, die tatsächlich dort empfangen werden sollen (selektive Weiterleitung). Hierzu kann die in Bild 7.3 gezeigte Lösung der Überbrückung zugangskontrollierter Medien verwendet werden.

Da nicht mehr alle Nachrichten weitergeleitet werden sollen, muß jedes physikalische Medium durch ein eigenes Zugangsverfahren kontrolliert werden. Damit können nun auch unterschiedliche physikalische Medien mit unterschiedlichen Zugangsverfahren gekoppelt werden. Es gibt einen gemeinsamen, der Brücke bekannten Adreßraum (hier ist ein Bezug zur Schicht 3), so daß die Brücke die Nachrichten unter Selektion weitergeben kann. Der LLC-Schicht gegenüber bleibt die Brücke verborgen (wenn man davon absieht, daß Zeitüberwachungskonstanten u.U. verlängert werden müssen).

Es werden ebenfalls bereits Brücken angeboten, die die Voraussetzung des gemeinsamen Adreßraums dadurch lockern, daß sie einfache Adreßabbildungen bei der Weitergabe unterstützen.

Quellengesteuerte Brücke

Bild 7.3a zeigt einen bei Tokenring-Netzen oft verwendeten Brückentyp, die quellengesteuerte Brücke. Die Netzadresse einer Empfängerstation ist eine Folge von Identifikationen, die individuell für den Weg zwischen Quellstation und Empfängerstation die Weiterleitung durch die Brücken steuert. Jedes Teilnetz hat eigene Stationsadressen (dieselbe Stationsadresse kann in unterschiedlichen Teilnetzen verwendet werden). Die Brücken enthalten keine Weiterleitungstabellen.

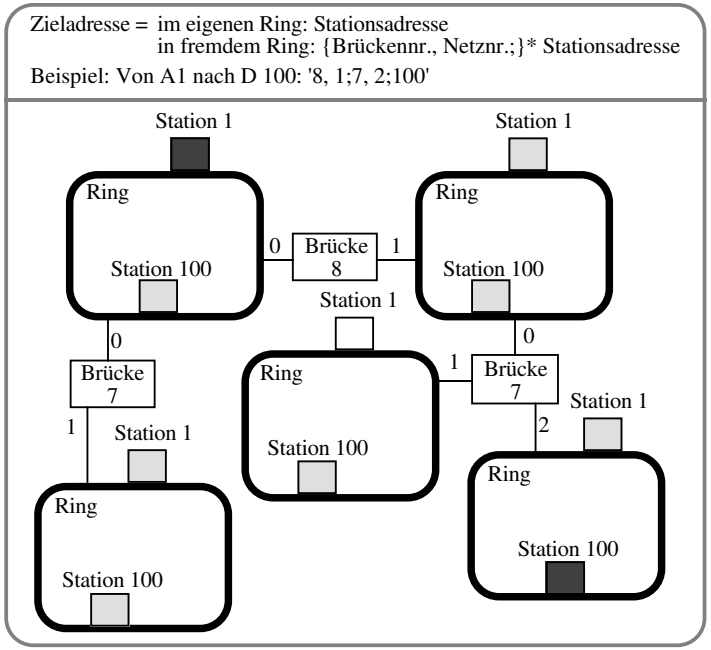

Bild 7.3a: Source-Routing Brücken

Transparente Brücke

Bild 7.3b zeigt einen bei Ethernet-Netzen oft verwendeten Brückentyp, die transparente Brücke. Die in den einzelnen Teilnetzen verwendeten Adreßmengen sind disjunkt (einheitlicher Adreßraum über alle Teilnetze). Die Zielstation wird durch ihre (eindeutige) Stationsadresse adressiert. Die Quelle muß deshalb nicht wissen, ob sie über Brücken oder direkt erreicht wird (deshalb transparente Brücke). Die Brücke allerdings muß nun eine Weiterleitungstabelle enthalten, die durch Netzmanagementmaßnahmen gepflegt werden muß.

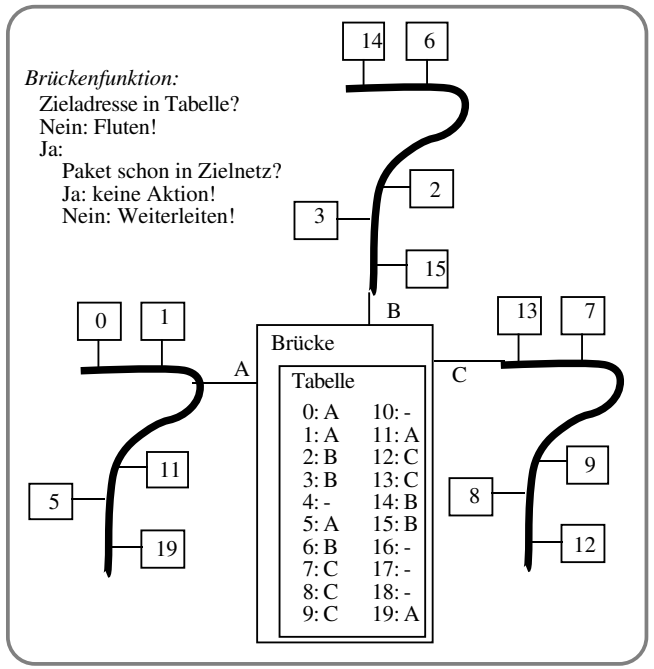

Bild 7.3b: Transparente Brücke

7.4.2 Subnetz-Kopplung

Die flexibelste, aber auch aufwendigste Lösung ist es, separierte eigenständige Netze als Subnetze zu einem Gesamtnetz zu verbinden. Hier können Subnetze dann durchaus auch nichtlokal sein, und das Gesamtnetz kann geographisch weiträumig verteilt sein. Diese Lösung ist in Bild 7.4 gezeigt. Das Gesamtnetz wird nun durch eine - an allen Orten Instanzen vorsehende - Internetz-Schicht gebildet. Die Internetz-Instanzen sind die Dienstnehmer der einzelnen Subnetze und einzelne Instanzen fungieren als Verbindungsglieder zwischen Subnetzen.

Eine entsprechender, auch im nichtmilitärischen Bereich vielbeachteter Standard ist der IP-Anteil von TCP/IP (Transmission Control Protocol / Internet Protocol), wie es vom amerikanischen DoD (Department of Defense) unter MIL-STD-1777 festgelegt ist und innerhalb des großen amerikanischen DARPA-Netzes (zivil im Internet) verwendet wird.

IP sieht die Erbringung eines Netzdienstes vor, der nur einen Dienstleistungstyp, die unbestätigte Datagramm-Übertragung unterstützt. Die Internet-Schicht enthält Mechanismen zur Weiterleitung zwischen Netzen über Gateway-Instanzen, zur Nutzdatenlängen-Anpassung (Fragmentieren), und zur Unterstützung sehr vieler Netz-Zugangspunktadressen je Ort. Sicherung gegen Verfälschung erfolgt nur bezüglich der Kontrollparameter von Internet-PDUs.

Zusätzlich ist schichtintern das IMCP (Internet Control Message Protocol) vorgesehen. Es dient der Meldung von Ablaufbesonderheiten (nicht angekommene Datagramme, Überlastprobleme) und der Erkundung des momentanen Netzzustands (Test ob zwischen zwei Orten ein Weg existiert; Test der Nachrichtenlaufzeit über einen Weg). Die IMCP-Mechanismen setzen selbst wieder auf dem Datagram-Dienst auf.

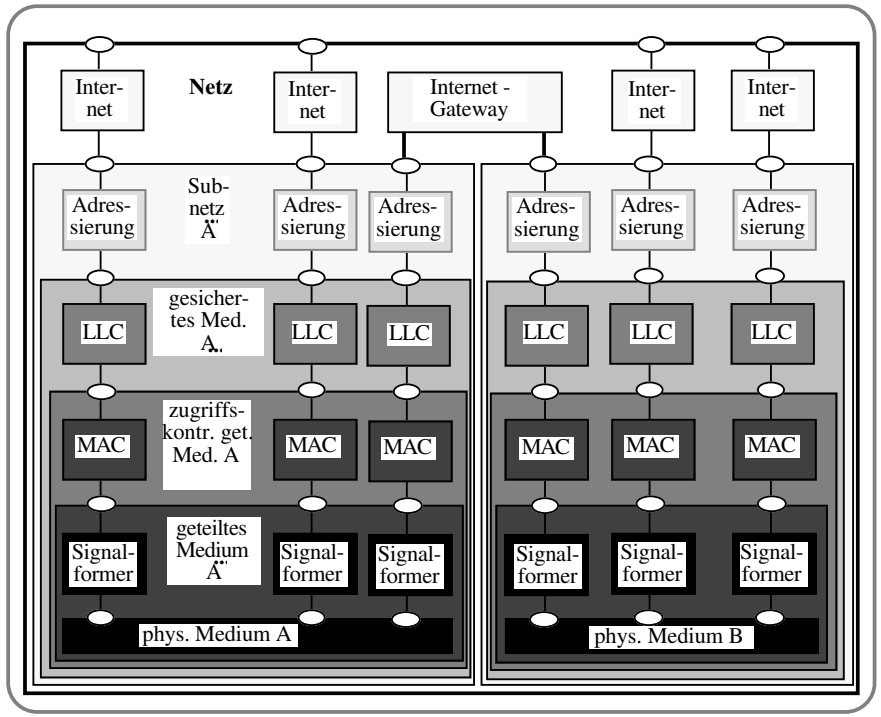

Bild 7.4: Netz-Netz-Kopplung

Die Wegetabellen der Instanzen können statisch oder dynamisch sein. Sie können Ausweichwege enthalten. Zur Aktualisierung dynamischer Tabellen können durch IMCP-Nutzung gewonnene Informationen verwendet werden, zur Aktualisierung selbst werden aber keine detaillierten Mechanismen vorgeschrieben.

7.5 Dienstleistungen der Transportschicht

Auch bei der Transportschicht wird zwischen Datagramm-Betrieb und verbindungsorientiertem Betrieb unterschieden (COTS: Connection Orienten Transport Service, CLTS: Connectionless Transport Service), wobei der Datagramm-Betrieb wiederum nur über einer Dienstleistung des Typs UnitData-Transfer definiert ist.

Der verbindungsorientierte Transportdienst ist funktionell mit dem verbindungsorientierten Netzwerkdienst vergleichbar, wenn man von der unterschiedlichen Bedeutung der Zugangspunkte abstrahiert. So sind an Dienstleistungstypen der bestätigte Aufbau von Zwei-Punkt-Verbindungen (Connect), ihr unbestätigter Abbau bzw. Abbruch (Disconnect), der normale (DataTransfer) sowie der beschleunigte (ExpeditedDataTransfer) Datentransfer definiert. Im Unterschied zum Netzwerkdienst ist die Möglichkeit der Empfangsbestätigung und des Rücksetzens nicht vorgesehen.

Nach vielen Prognosen gehen die Anforderungen zukünftiger Anwendungen über das von gegenwärtig dominierenden Transportdiensten gegebene Funktions- und Leistungsangebot hinaus:

- effiziente Mehr-Partner-Kommunikation,
- hohe Bitraten,
- variable Bitraten,
- kleine garantierte Laufzeiten und Laufzeitverzerrungen

werden benötigt. Neuere Hochgeschwindigkeitsnetze (z.B. B-ISDN/ATM, FDDI-Netze) kommen diesen Anforderungen bereits entgegen. Zu erwarten sind erweiterte Transportdienst-Konzepte. Sie werden funktionell die Mehrpartner-Kommunikation und in qualitativer Hinsicht die Aushandlung eines erweiterten Spektrums an Dienstleistungsqualitäten ermöglichen.

7.6 Protokolle der Transportschicht

Hauptfunktion eines Transportprotokolls ist es, mit den vom Netzwerk her zur Verfügung stehenden Mitteln und unter deren kostengünstiger Verwendung von Anwendungen her benötigte Übertragungsdienstleistungen mit von den Anwendungen vorgegebenen Qualitätsparametern zu erbringen.

Für jede im verteilten System insgesamt von anderen zu unterscheidende Anwendungsinstanz muß eine Transportdienstadresse, also je Ort i.a. sehr viele Zugangspunkte, unterstützt werden, so daß zur Bearbeitung von verschiedene Transportzugangspunkte betreffenden Aufträgen u.U. am selben Netzzugangspunkt Netzwerk-Aufträge abzusetzen sind (Adreßabbildung).

Unter Umständen erlaubt der Netzdienst je Netzzugangspunkt nur den Betrieb einer sehr beschränkten Anzahl von Verbindungen. Dann muß dieselbe Netzwerkverbindung unter funktionellen Aspekten mehrfachgenutzt werden.

Wenn der Transportdienst verbindungsorientiert ist, und der Netzwerkdienst nur Datagramm-Transfer gestattet oder bei Netzwerkverbindungen Reihenfolge-Besonderheiten häufig auftreten können, müssen Maßnahmen zur Reihenfolgesicherung getroffen werden.

Wenn die Verfälschungsfehlerwahrscheinlichkeit der Netzwerkdienstleistungen zu groß ist, müssen in der Transportschicht zusätzliche Verfälschungssicherungsmaßnahmen vorgesehen werden.

Zur Leistungsanpassung und Kostenreduktion sind u.U. Multiplex-Mechanismen (Bündeln, Mehrfachnutzen) erforderlich. Weiterhin können Flußkontroll-Mechanismen vorgesehen sein.

Schließlich kann es vorkommen, daß ein Netzdienst verfügbar ist, der nur beschränkte Nutzdaten-Längen unterstützt. Hierzu muß auf Fragmentierungsmechanismen zurückgegriffen werden.

Wichtige Beispiele für Transportprotokolle sind die ISO/OSI-Transportprotokolle (ISO IS8073) und der TCP-Teil von TCP/IP (DoD MIL-STD-1778).

ISO/OSI Transportprotokolle

Bei ISO/OSI werden 5 Varianten (Klasse 0 .. Klasse 4) als Transportprotokolle vorgesehen.

Klasse 0, die einfachste Variante bildet jede Transportverbindung 1:1 auf eine entsprechende Netzwerkverbindung ab. Der Netzwerkdienst wird als qualitativ unmittelbar zufriedenstellend vorausgesetzt.

Klasse 1 geht davon aus, daß der Netzwerkdienst Reihenfolgevertauschungen möglich macht. Hierzu werden Reihenfolge-Sicherungsmaßnahmen in der Transportschicht erforderlich. Ferner werden Nutzdaten bei Bedarf fragmentiert.

Klasse 2 befaßt sich damit, daß eine Leistungsanpassungs- und Überlastproblematik bestehen kann. Das Bündeln von Netzwerkverbindungen und Flußkontrolle bilden hier wesentliche Protokollbestandteile.

Klasse 3 ist ein Verbund der Mechanismen der Klassen 1 und 2.

Klasse 4 stellt die geringsten Anforderungen an das Netzwerk. Es sind Fragmentieren, Bündeln, Mehrfachnutzen, Flußkontrolle, Reihenfolgesicherung und eine eigene Verfälschungssicherung vorgesehen.

TCP

Das TCP-Protokoll setzt auf dem durch IP erbrachten Datagramm-Netzwerk-Dienst auf. Es erbringt auf dieser Basis einen verbindungsorientierten Dienst. Dabei wird angenommen, daß der Netzwerkdienst mit Reihenfolge-Besonderheiten und Nachrichtenverfälschungen behaftet sein kann, so daß entsprechende Sicherungsmaßnahmen vorgesehen sind (Verfälschungssicherung, Sequenzzahlen in Fenstertechnik).

Hochleistungstransportprotokolle

Zur effizienten Nutzung von Hochleistungsnetzen und zur Unterstützung erweiterter Transportdienste werden Hochleistungstransportprotokolle vorgeschlagen (z.B. XTP: Xpress Transfer Protokoll). Die Protokolle sind für die schnelle Bearbeitung in den Transportinstanzen zugeschnitten. Optionale Protokollelemente werden in Abhängigkeit zu den aktuellen Anforderungen bestimmter Anwendungen eingesetzt und helfen Bearbeitungs- und Kommunikationsoverhead zu vermeiden.

Neben der Entwicklung spezieller Hochleistungsprotokollen besteht die Tendenz, vorhandene Protokolle (z.B. TCP im Internet) entsprechend zu erweitern.

7.7 Dienstleistungen der Kommunikationssteuerungsschicht

Der Kommunikationssteuerungsdienst unterstützt eine vergleichsweise große Anzahl von Dienstleistungstypen.Sie sollen können in folgende Gruppen gegliedert werden:

- Verbindungshandhabung,
- Datentransfer,
- Rechtevergabe,
- Synchronisation des Datenaustauschs,
- Aktivitätsgliederung,
- Ausnahme-Benachrichtigung.

Zum Auf- und Abbau von Zwei-Punkt-Verbindungen sind folgende Dienstleistungstypen vorgesehen:

- Bestätigter Verbindungsaufbau (Connect); neben den üblichen Parametern (Initiator- und Beantworter-Adresse, Verbindungseigenschaften, kurze Nutzdaten und Annahme/Ablehnungsanzeige sind als weitere Parameter vorgesehen:
  - Verbindungskennung; eine Kennung zur Identifikation der Verbindung bei den Dienstnehmern,
  - Anforderungen; Angaben, aus denen die bei Betrieb der Verbindung anforderbaren Dienstleistungstypen hervorgehen (als Liste von Dienstleistungstyp-Gruppen-Angaben),
  - Folgenummer; Startwert der bei den Synchronisationsdienstleistungen zu verwendenden Folgenummern,
  - initiale Rechtezuordnung; Liste, aus der hervorgeht, welcher Dienstnehmer anfänglich welche Rechte besitzt.
- Bestätigter, geregelter Verbindungsabbau (Release); Anforderung und Bestätigung können kurze Nutzdaten übertragen, die Bestätigung kann auch eine Ablehnung durch den Beantworter darstellen,
- Unbestätigter Nutzer-Abbruch (U-Abort); die Anforderung kann kurze Nutzdaten übertragen,
- Erbringerabbruchanzeige (PAbort); Anzeige, mit der das abstrakte Kommunikationssteuerungsmedium den Abbruch einer Verbindung mit Abbruchgrund nennt.

Zum Datentransfer steht ein Dienstleistungstyp zum sogenannten normalen Datentransfer (DataTransfer) zur Verfügung, der per Rechtevergabe als geregelter Halbduplex- oder als Vollduplex-Betrieb ausgeführt werden kann, sowie weitere Dienstleistungstypen, die nicht den Rechte-Konventionen unterliegen und für die Übertragung besonderer Informationen zum Betriebsablauf zwischen den Dienstnehmern gedacht sind. Alle Dienstleistungstypen sind unbestätigt:

- (Normaler) Datentransfer (DataTransfer); Austausch eines Nutzdatums als beliebiges Bitmuster beliebiger Länge,
- Vorrangdatentransfer (ExpeditedDataTransfer); hochpriore Übertragung kurzer Nutzdaten, um Informationen über besondere, aber nicht näher reglementierte Ereignisse unabhängig von dem bezüglich des normalen Datenaustauschs bestehenden Verbindungszustand austauschen zu können,
- Typendatentransfer (TypedDataTransfer); ebenfalls nicht den Konventionen des normalen Datenaustauschs unterliegender Austausch von Daten, die z.B. Information über besondere Randbedingungen des aktuellen Datenaustauschs tragen können,
- Fähigkeitsdatentransfer (CapabilityDataTransfer); bei Bindung des normalen Transfers an Aktivitäten wird hierdurch der aktivitätenunabhängige Austausch kurzer Nutzdaten möglich, z.B. um die Modalitäten aufzunehmender Aktivitäten abzustimmen.

Die Rechtevergabe unterscheidet zwischen den Rechten 'Datensenderecht', 'Hauptsynchronisationspunkt / Aktivitäten -Vereinbarungsrecht', 'Nebensynchronisationspunkt-Vereinbarungsrecht', Verbindungsabbaurecht'. Mit dem Verbindungsaufbau wird eine initiale Rechtevergabe festgelegt. Sie kann unter Verwendung folgender drei unbestätigter Dienstleistungstypen während des Betriebs verändert werden:

- Rechtewunsch (PleaseToken); hiermit teilt ein Dienstnehmer dem anderen mit, daß er eine gewisse Menge von Rechten erhalten will (Parameter: Rechtekennungsmenge),
- Rechteübergabe (GiveToken); hiermit übergibt ein Dienstnehmer dem anderen eine Menge von Rechten (Parameter: Rechtekennungsmenge und Nutzdaten),
- Kontrollabgabe (ControlGive); hiermit übergibt ein Dienstnehmer alle verfügbaren Rechte an den anderen Partner.

Die Dienstleistungen zur Synchronisation sollen das konsistente Wiederaufsetzen beider Dienstnehmer in frühere Datenaustauschzustände ermöglichen. Ausgehend von der mit dem Verbindungsaufbau gewählten initialen Folgenummer werden Synchronisationspunkte mit einer Folgenummer identifiziert. Sind bei einem der beiden Partner oder im Kommunikationssystem Probleme (Verluste) aufgetreten, können die Partner mit Resynchronisation (Rücksetzen auf den Zustand, der zum Zeitpunkt eines früher vereinbarten Synchronisationspunkts herrschte) reagieren. Natürlich sollte von den Partnern dazu ein möglichst junger Synchronisationspunkt gewählt werden, um unnötige Wiederholun-

gen zu vermeiden.
Da Synchronisationspunkte bei den Partnern auch Kosten verursachen (z.B. Erstellen einer Sicherungskopie des gegenwärtigen Zustands) und die Kosten auch abhängig sind von der Persistenz der Sicherungen, wird zwischen zwei Typen von Synchronisationspunkten unterschieden: Haupt- und Nebensynchronisationspunkte.
Bei geglückter Vereinbarung eines Hauptsynchronisationspunkts sollen beide Partner ihren Zustand persistent gesichert haben, damit Rücksetzversuche auf den zuletzt vereinbarten Hauptsynchronisationspunkt unbedingt erfolgreich sind. Gleichzeitig können die Partner alle vorherigen Synchronisationspunkte vergessen, um den Speicher zu entlasten.
Bei Vereinbarung eines Nebensynchronisationspunkts sollten beide Partner ebenfalls ihren momentanen Zustand sichern, aber die Sicherung muß nicht unbedingt fehlertolerant sein. Deshalb kann es sein, daß der Versuch, auf den jüngsten Nebensynchronisationspunkt zurückzusetzen, scheitert und es sollten somit auch die vorherigen Synchronisationspunkte (eben unbedingt der zuletzt vereinbarte Hauptsynchronisationspunkt und nach Möglichkeit die danach vereinbarten Nebensynchronisationspunkte) noch verfügbar sein.
Somit können je nach den Erfordernissen einer speziellen Anwendung unterschiedliche Sicherungsstrategien verfolgt werden.
Der durch zwei aufeinanderfolgende Hauptsynchronisationspunkte begrenzte Nutzdatenaustausch wird oft Dialogeinheit genannt. Alle Dienstleistungen zur Synchronisation sind bestätigt (die Nebensynchronisationspunktvereinbarung kann auch unbestätigt abgewickelt werden) und neben der Folgenummer des in Bezug genommenen Synchronisationspunkts mit Nutzdaten parametrisierbar:

- Nebensynchronisationspunktvereinbarung (SyncMinor),
- Hauptsynchronisationspunktvereinbarung (SyncMajor),
- Resynchronisation (Resync); nach Abwicklung der Dienstleistung sollen beide Dienstnehmer in den durch den identifizierten Wiederaufsetzpunkt bezeichneten früheren Zustand zurückgesetzt haben, zusätzlich können Rechtevergabe-Angaben als Parameter mit der Dienstleistung ausgetauscht werden, um gleichzeitig den nach dem Wiederaufsetzen gegebenen Stand der Rechtevergabe zu bestimmen.
  Empfängt ein Partner den Wunsch auf einen bei ihm zerstörten Nebensynchronisationspunkt zurückzusetzen, reagiert er seinerseits mit dem Wunsch auf den nächsten früheren, bei ihm vorhandenen zurückzusetzen.

Die Gliederung des Nutzdatenaustauschs durch Synchronisationspunkte reicht für viele Anwendungen bereits zur Kommunikationssteuerung aus. Oft wird es aber sinnvoll, mehrere Dialogeinheiten in größere Einheiten, sogenannte Aktivitäten zusammenzufassen. Z.B. sollte der Transfer einer sehr langen Datei in mehrere Dialogeinheiten gegliedert werden. Der Transfer der gesamten Datei bildet dann eine Aktivität, und Dienstleistungen sind interessant, mit denen eine solche Aktivität ausdrücklich begonnen, unterbrochen, wiederaufgenommen und beendet werden kann. Dabei ist es insbesondere wichtig, daß auch nach Abbruch einer Kommunikationssteuerungsverbindung (z.B. weil die unterliegende Transportverbindung abgebrochen wurde oder weil einer der beiden Partner lokale Probleme hatte) und Aufbau einer neuen früher vereinbarte Aktivitäten fortsetzbar sind:

- Unbestätigte Aktivitätsbeginnbenachrichtigung (ActivityStart); es werden eine Aktivitätsidentifikation und Nutzdaten übertragen.
- Unbestätigte Aktivitätswiederaufnahmebenachrichtigung (ActivityResume); an Parametern wird eine Aktivitätsidentifikation zur Bezugnahme auf eine vorher unterbrochene Aktivität und eine weitere für die vorgesehene Wiederaufnahme benötigt. Die Angabe der alten Aktivitätsidentifikation wird durch Nennung der Verbindungsidentifikation der Verbindung detailliert, in der die alte Aktivität begonnen wurde. Ferner werden die Folgenummer zur Kennzeichnung des Wiederaufsetzpunkts sowie Nutzdaten übertragen.
- Bestätigte Aktivitätsunterbrechungsvereinbarung (ActivityInterrupt); hiermit können die Dienstnehmer auf Anstoß eines der beiden die Unterbrechung der laufenden Aktivität auf unbestimmte Zeit vereinbaren. Als Parameter ist die Angabe des Abbruchgrunds vorgesehen.

- Bestätigte Aktivitätsstornierung (ActivityDiscard); hiermit können die Dienstnehmer auf Anstoß eines der beiden Abbruch und Löschen der laufenden Aktivität vereinbaren. Als Parameter ist die Angabe des Abbruchgrunds vorgesehen.
- Bestätigte Aktivitätsbeendigung (ActivityEnd); hiermit wird das reguläre Ende der aktuellen Aktivität zwischen den beiden Dienstnehmern vereinbart.

Während das Auftreten von Ausnahmesituationen beim Betrieb von Transportverbindungen nach dem Konzept des Transportdienstes auf den Abbruch einer Transportverbindung abzubilden ist (PAbort, falls die Ausnahmesituation innerhalb des Mediums vom Medium gegenüber den Dienstnehmern nicht verdeckt werden kann; UAbort, falls die Ausnahmesituation eines Dienstnehmers ihn außer Stand setzt, die Verbindung weiter zu betreiben), sind beim Kommunikationssteuerungsdienst Dienstleistungstypen vorgesehen, mit denen das Medium bzw. ein Dienstnehmer die beiden Dienstnehmer bzw. den anderen Partner über das Auftreten der Ausnahmesituation unterrichten kann, sodaß den Dienstnehmern die Wahl für geeignete Reaktionen darauf (doch Abbruch der Verbindung, Resynchronisation) offen bleibt:

- Ausnahmeanzeige des Mediums (PExceptionReport); im Parameter der Anzeige ist der Grund zur Kennzeichnung der Ausnahmesituation codiert.
- Unbestätigte Ausnahmebenachrichtigung (UExceptionReport); im ersten Parameter der Anzeige ist der Grund zur Kennzeichnung der Ausnahmesituation codiert, ein zweiter erlaubt die Übertragung zusätzlicher Nutzdaten z.B. zur genaueren Kennzeichnung.

7.8 Protokolle der Kommunikationssteuerungsschicht

Zur Erbringung des Kommunikationssteuerungsdienstes auf der Basis eines Transportdienstes kommt üblicherweise das ISO/OSI Session Protokoll (ISO IS8327) zum Einsatz, das ebenfalls der CCITT-Empfehlung X.225 entspricht. Das Protokoll ist im Vergleich zu den bisher angesprochenen transportorientierten Protokollen sehr einfach. Es unterscheidet zwar zwischen einer Vielzahl unterschiedlicher PDU-Typen, die Ablauffestlegungen sind jedoch, verglichen z.B. mit den Festlegungen zur Sequenzzahl-Numerierung, Nachrichten-Widerholung und Flußkontrolle beim HDLC-Protokoll, sehr einfach.

Die Einfachheit der Ablauffestlegungen resultiert daraus, daß die ganze Problematik der detaillierten Behandlung von Übertragungsfehlern und der Übertragungsqualitätsanpassung entfällt. Sie wird innerhalb des Transportsystems abgehandelt. Dort wird bereits alles unternommen, um Übertragungsfehler zu beheben, so daß oberhalb des Transportsystems hierzu keine Maßnahmen mehr sinnvoll sind. Alle vom Transportsystem nicht verdeckbaren Übertragungsfehler werden durch schwerwiegende, längerfristige Ausfälle von Medien oder Knoten verursacht und vom Transportdienst an die Kommunikationssteuerungsinstanzen gemeldet. Die Kommunikationssteuerungsinstanzen reagieren darauf nicht durch Behebungsversuche, sondern durch den Abbruch der Abwicklung betroffener Dienstleistungen unter Weitermelden der Besonderheit an die Dienstnehmer des Kommunikationssteuerungsdienstes. Auch wird zur Unterstützung einer Kommunikationssteuerungsverbindung nur eine einzige Transportverbindung genutzt, die für die Dauer der Kommunikationssteuerungsverbindung exklusiv zur Verfügung stehen muß.

Im groben sieht das Kommunikationssteuerungsprotokoll für jeden bestätigten Dienstleistungstyp zwei PDU-Typen vor, einen zur Übertragung der Dienstleistungsanforderung vom Initiator zum Beantworter, einen zur Übertragung der Antwort. Für unbestätigte Dienstleistungen ist entsprechend nur ein PDU-Typ je Dienstleistungstyp erforderlich. Die Ablauffestlegungen schreiben davon ausgehend vor, daß eine Instanz Anforderungen und Antworten in entsprechende PDUs abbildet und eingehende PDUs in entsprechende Anzeigen und Bestätigungen.

Es soll noch darauf hingewiesen werden, daß der Einfachheit des Protokolls eine im Vergleich zu transportorientierten Diensten wesentlich differenziertere Kooperationsbeziehung der Dienstnehmer gegenübersteht, die den Dienstnehmern umfangreichere Ablauffestlegungen auferlegt. So obliegt z.B. die Vergabe und Respektierung von Folgenummern für Synchronisationspunkte nicht den Protokollinstanzen sondern den Dienstnehmern, die dies im Einklang mit der zwischen ihnen bestehenden Kooperationsbeziehung bewerkstelligen müssen.

7.9 Dienstleistungen der Darstellungsschicht

Die Darstellungsschicht unterstützt ebenfalls **alle** Dienstleistungstypen, die auch bereits von der Kommunikationssteuerungsschicht angeboten werden. Sie werden im folgenden deshalb nicht mehr im einzelnen aufgeführt.

Unterschiede gibt es nur in Bezug auf die Darstellung austzutauschender Nutzdaten gemäß Darstellungskontexten. Normale Nutzdaten, Typendaten, beschleunigte Daten und Fähigkeitsdaten sowie alle mit anderen Dienstleistungen zusätzlich übertragene Nutzdaten gelten als an einem Dienstzugangspunkt des Darstellungsdienstes gemäß einem Darstellungskontext übergeben.

Zur Bestimmung der aktuell gültigen Darstellungskontexte und Veränderung während des Betriebs einer Darstellungsverbindung sind im Vergleich zum Kommunikationssteuerungsdienst zusätzliche Parameter bei der Verbindungsaufbaudienstleistung und zusätzliche Dienstleistungstypen vorgesehen:

- Verbindungsaufbau; zusätzliche Parameter sind:
  - Anzeige ob mehrere Darstellungskontexte oder nur einer benutzt werden,
  - Liste der Namen der zu benutzenden Kontexte,
  - Name eines hervorgehobenen Darstellungskontexts, des Defaultkontexts, der bei einigen Datenübertragungsdienstleistungen standardmäßig angewendet werden soll,
  - Darstellungsnforderungen; Angaben, aus denen die bei Betrieb der Verbindung anforderbaren Dienstleistungstypen zur Kontexthandhabung hervorgehen,
- Bestätigte Kontextänderungsvereinbarung (AlterContext); Parameter der Benachrichtigung (AlterContextReq --> AlterContextInd) sind:
  - Liste der Namen der nach Initiatorwunsch zukünftig zusätzlich zu benutzenden Kontexte,
  - Liste der Namen der nach Initiatorwunsch zukünftig nicht mehr benötigten Kontexte,
  - zusätzliche Nutzdaten.

  Parameter der Bestätigung (AlterContextRes --> AlterContextCon) sind:
  - Liste der Namen der auch nach Beantworterwunsch zukünftig zusätzlich zu benutzenden Kontexte,
  - Liste der Namen der auch nach Beantworterwunsch zukünftig nicht mehr benötigten Kontexte,
  - zusätzliche Nutzdaten.

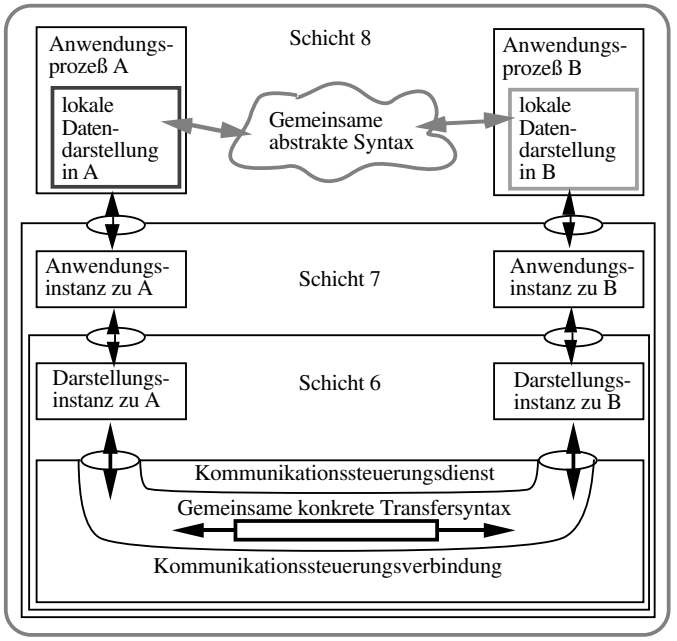

Bild 7.5: Szenario zur Datendarstellung

Weitergehend enthält das Konzept des Darstellungsdienstes Festlegungen, aus denen hervorgeht, wie sich Rücksetzmaßnahmen auf die aktuell unterstützten Kontexte auswirken. Hierauf soll hier im einzelnen nicht näher eingegangen werden.

Bild 7.5 zeigt ein Szenario zur Darstellungsproblematik. I.a. gibt es zwei lokale Datendarstellungen (die in den beiden Anwendungsprozessen verwendet werden). Konkret gibt es außerdem noch eine gemeinsame Darstellung, die beim Transfer von Daten in AnwendungsPDUs verwendet wird. Alle drei sind verträglich zur gemeinsamen abstrakten Syntax (die z.B. in ASN.1 definiert werden kann).

Bild 7.6 soll verdeutlichen, daß zur Laufzeit bei einer Übertragung eines Datums zwei Wandlungen der Darstellung vorzunehmen sind (z.B. Lokal A --> Transfer --> Lokal B). Diese Wandlungen können nicht direkt standardisiert werden (da die lokale Darstellung Implementierungssache ist und nicht in Normen angesprochen werden soll). Indirekt werden sie durch Bezugnahme auf die Codierung 'Abstrakte Syntax --> Transfersyntax' festgelegt (z.B. Standardcodierregeln zu ASN.1).

Ein (z.B. in einem internationalen Normdokument definierter) Darstellungskontext legt somit die Wandlung indirekt fest. Durch Austausch der Kontextidentifikation (zur Laufzeit) weiß ein Partner, welche abstrakten Datentypen in welcher Codierung der andere auszutauschen in der Lage ist.

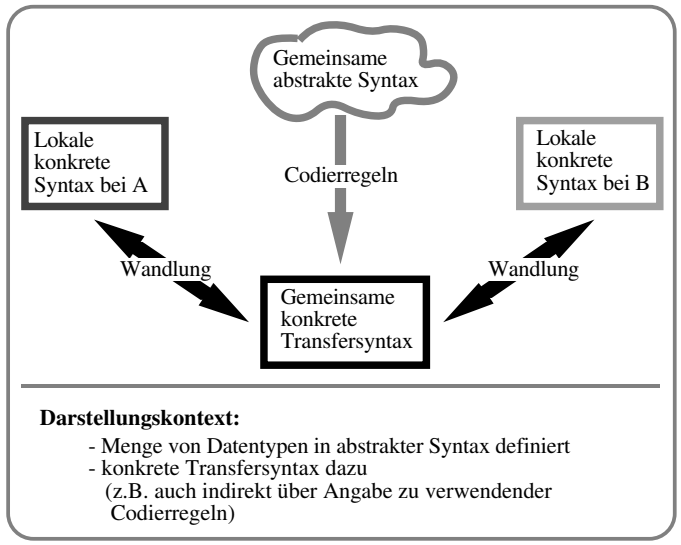

Bild 7.6: Darstellung, Codierregeln und Wandlung

7.10 Protokolle der Darstellungsschicht

Zur Erbringung des Darstellungsdienstes wird von ISO/OSI das Presentation-Protokoll (ISO IS8823) vorgeschlagen. Es handelt sich, wie das Session-Protokoll ebenfalls um ein sehr einfaches Protokoll. Die Dienstleistungen des Kommunikationssteuerungsdienstes werden in unmittelbar entsprechender Form ebenfalls vom Darstellungsdienst unterstützt. Hier reduziert sich die Funktion von Darstellungsinstanzen auf das Durchreichen von Dienstnehmeraufträgen an den Kommunikationssteuerungsdienst. Die wenigen weiteren - darstellungsbezogenen - Dienstleistungstypen werden durch Übertragung entsprechender Darstellung-PDU-Typen ebenfalls in einfacher Weise, vergleichbar der Abwicklung von Kommunikationssteuerungsdienstleistungen, erbracht. Die Einfachheit des Darstellungsprotokolls wird nicht zuletzt auch dadurch erreicht, daß eine Darstellungsverbindung 1:1 auf eine Kommunikationssteuerungsverbindung abgebildet wird. Auch Auf- und Abbau einer Darstellungsverbindung gehen direkt im Auf- und Abbau der zu benutzenden Kommunikationssteuerungsverbindung auf.

7.11 Anwendungsdienste

Ein Anwendungsdienst ist natürlich - wie bereits erwähnt - in Art und Umfang der Dienstleistungstypen auf eine spezielle Anwendung oder mindestens auf einen genauer umrissenen Typ von Anwendungen zugeschnitten, und die Dienstleistungstypen werden nicht nur bezüglich ihrer 'mechanischen' Datenübertragungseigenschaften definiert, sondern es wird auch ihre Bedeutung für die Anwendung

durch Inbezugsetzung zum Anwendungsmodell erklärt. Deshalb muß eine allgemeine Beschreibung von Anwendungsdiensten recht oberflächlich bleiben.

Die Normung spezieller Anwendungen befindet sich ebenfalls noch in Anfängen. Konkretisiert haben sich in diesem Bereich zur Zeit - von einigen Ausnahmen abgesehen - erst Bestrebungen zur Normung weiter gefaßter Anwendungsgrundtypen, wie z.B. FTAM (File Transfer Access and Management), Directory, Authentisierung, Autorisierung, E-Mail, JTM,.., welche die Basis zur Behandlung enger gefaßter spezieller Anwendungen bilden sollen. Hierzu wird im folgenden als Beispiel übersichtsweise auf den FTAM-Dienst eingegangen.

Ferner hat sich herauskristallisiert, daß auch zwischen verschiedenen speziellen Anwendungsdiensten Gemeinsamkeiten in den benötigten Dienstleistungstypen bestehen. Es gibt eine Menge von Dienstleistungstypen, die unabhängig vom Typ der Anwendung einem gemeinsamen Grundmuster entsprechen. Sie werden Common Application Service Elements (CASE) genannt und sind als generische Typen zu betrachten, die innerhalb eines speziellen Anwendungsdienstes dann in auf diesen Typ zugeschnittenen Ausprägungen vorkommen. Dienstleistungstypen eines speziellen Anwendungsdienstes, die nicht zu den CASE zählen, werden SASE genannt (Specific Application Service Elements). Hauptsächlich auf sie muß in der Beschreibung eines speziellen Anwendungsdienstes eingegangen werden.

7.11.1 Dateitransfer - FTAM

Der im Rahmen des Arbeiten um das ISO/OSI-Modell in Normung befindliche Anwendungsdienst FTAM stellt Dienstleistungstypen bereit, mit deren Hilfe zwei Nutzerinstanzen diejenige Kommunikation miteinander abwickeln können, die für den Zugriff einer Instanz auf ein der anderen Instanz direkt zugängliches Dateisystem benötigt werden.

Die FTAM-Norm hat eine für Standards der Anwendungsschicht 'klassische' Struktur. Sie besteht aus vier Teilen:

- Übersicht,
- Virtual Filestore Model,
- FTAM-Dienst,
- FTAM-Protokoll.

Zum leichteren Verständnis berücksichtige man beim Lesen der folgenden Erläuterungen Bild 7.7. Es zeigt ein Szenario zu FTAM. Zwei Anwendungsprozesse führen produktive Dateioperationen aus (z.B. ein 'Server' betreibt ein Dateisystem an Ort 'rechts', ein 'Client' ermöglicht einem menschlichen Nutzer an Ort 'links' per Sichtgerät die entfernten Dateien zu manipulieren. Der von FTAM angebotene Dienst ist ein Telekommunikationsdienst und sieht Dienstleistungstypen zum Übertragen von Wünschen und Ergebnissen für/von Dateizugriffsoperationsaufrufen vor.

Virtual Filestore Model

Die Semantik der Dienstleistungstypen bezieht sich auf die Datei. Um hier Bauartunabhängigkeit zu erreichen, enthält der Standard auch die Definition des Virtual Filestore Model, einer abstrakten Sicht von Dateien und Dateiorganisationen. Die in der Dienstspezifikation zugrundegelegte Semantik der Dienstleistungstypen bezieht sich auf dieses generelle Schema. Die im realen Betrieb auftretenden Schnittstellenereignisse sind von den Anwendungsinstanzen unter Abbildung der realen Dateistruktur in eine Ausprägung des Schemas zu interpretieren.

In der abstrakten Sicht ist eine Datei ein Verbund aus Dateiattributen und Inhalt in der Zugriffsstruktur eines geordneten Baums. Die Zugriffsstruktur sieht als Sonderfälle die flache und die unstrukturierte Form vor. Jedem Knoten der Zugriffsstruktur können ein Name sowie eine Dateneinheit zugeordnet

sein. Die Dateneinheiten bilden den Inhalt der Datei. Jeweils ein Knoten vertritt eine Zugriffseinheit (FADU: File Access Data Unit), welche die Dateneinheit des Knotens und die Dateneinheiten aller seiner Nachfolger umfaßt. Die Identifikation eines Knotens kann über seinen Namen oder über seine Stellung im Baum erfolgen. Die allgemeine Struktur kann für verschiedene Dateitypen einer Anwendung anhand vordefinierter Constraint Sets eingeschränkt werden. Es wird empfohlen, Constraint Sets und Beschreibungen der abstrakten Syntax der Inhalte verschiedener Dateitypen in einer Dokumenttyp-Bibliothek zu führen.

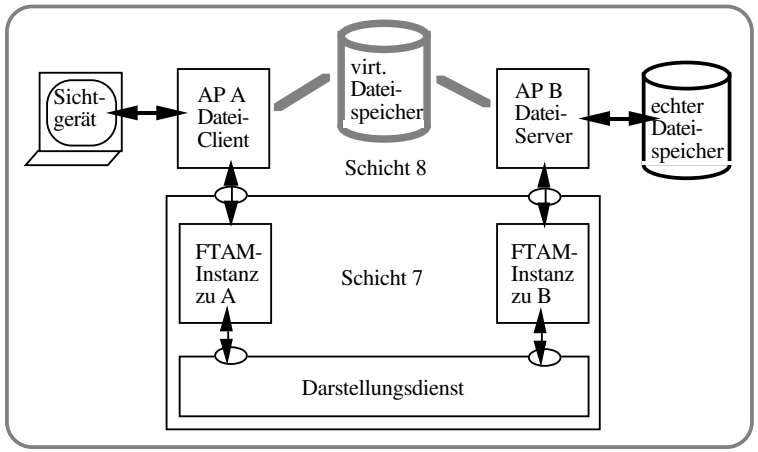

Bild 7.7: Szenario zu FTAM

An Operationen, die eine Datei insgesamt betreffen, sieht das Datei-Modell Erzeugung, Löschen und Auswahl (per Name) einer Datei in der Menge der von einer angesprochenen Nutzerinstanz zugreifbaren Dateien sowie Öffnen, Schließen und Attributzugriffe vor. Zum Zugriff auf den Inhalt sind die Operationen Lokalisieren, Lesen, Schreiben, Ersetzen und Löschen für FADUs definiert. Bei der Übertragung von Dateiinhalten muß hierbei nicht unbedingt die volle Zugriffsstruktur zugrundegelegt werden. Sie kann entsprechend einer daraus abgeleiteten Transferstruktur im Rahmen eines zwischen den Anwendungsinstanzen abgestimmten Zugriffskontextes erfolgen.

FTAM-Dienst

Der FTAM-Dienst stellt nun die zur Übertragung von Anforderungen und Ergebnissen von Operationen des Datei-Modells benötigten Dienstleistungen zur Verfügung. Sie sind zeitlich in Klammerstrukturen (Regimes) einzubetten. Das äußerste Regime bindet eine Darstellungsverbindung zwischen zwei Anwendungsinstanzen an die FTAM-Nutzung. Auf zweiter Stufe stehen Selektions-Regimes. Innerhalb eines Selektions-Regimes ist die zu erreichende Datei ausgewählt und kann als ganzes manipuliert werden. Inhaltszugriffe verlangen die Einbettung in ein auf dritter Stufe stehendes Open-Regime, in das Lokalisierungsoperationen und Transfer-Regimes eingelagert werden können. Zur Übertragung großer Datenmengen innerhalb eines Transfer-Regimes können sog. Bulk Data - Mechanismen verwendet werden, die eine beiden Anwendungsinstanzen ersichtliche Fragmentierung des Datenaustauschs ermöglichen.

FTAM-Protokoll

Das FTAM-Protokoll wird in zwei Varianten vorgesehen. Eine Variante erbringt unter CCR-Einbindung einen 'Reliable FTAM Service'. Die andere nur einen 'User correctable FTAM Service'. Interessant ist ferner noch die Verwendung der (via Presentation durchgereichten) Session-Dienstleistungstypen.

7.11.2 Gemeinsame Anwendungsdienstleistungstypen - CASE

Die gemeinsamen Dienstleistungstypen werden grob in folgende Klassen geteilt:

- Dienstleistungstypen zur Assoziationskontrolle (Association Control Service Elements ACSE),
- Dienstleistungstypen zur Transaktionskontrolle (Commitment, Concurrency, and Recovery CCR).
- Dienstleistungstypen zum 'Reliable Transfer' (RTSE: Reliable Transfer Service Elements),
- Dienstleistungstypen zu 'Remote Operations' (ROSE: Remote Operation Service Elements).

Die ersten beiden Klassen sind schon länger definiert. Die letzten beiden Klassen wurden wesentlich von der E-Mail-System-Normung (CCITT: X.400ff in '84- und '88- Version, bzw. deren Aufarbeitung durch ISO/OSI) beeinflußt.

7.11.2.1 Assoziationskontrolle - ACSE

Zur Thematik der Assoziationskontrolle soll zunächst der Begriff der Anwendungsassoziation (Application Association) erklärt werden. Wenn zwei Nutzerinstanzen miteinander kommunizieren wollen, müssen sie hierzu eine Darstellungsverbindung verfügbar haben, und sie werden die Darstellungsverbindung im Rahmen ihrer speziellen Kooperationsbeziehung mittelbar über die Anwendungsschicht nutzen. Man kann sagen, daß die Darstellungsverbindung für eine gewisse Zeitdauer an die spezifische Nutzung gebunden wird, und der Anwendungdienst wird deshalb einen dem Dienstleistungstyp 'Verbindungsaufbau' vergleichbaren Dienstleistungstyp 'Binde bestehende Darstellungsverbindung an die Nutzung im Rahmen der anwendungsspezifischen Kooperationsbeziehung' unterstützen müssen. Die Darstellungsverbindung bildet dann zusammen mit einer solchen Bindung eine Anwendungsassoziation.

Zur Vereinfachung ergeht die Konvention, daß eine Darstellungsverbindung für die gesamte Dauer ihres Bestehens dieselbe Anwendungsassoziation trägt, und daß umgekehrt eine Assoziation nur für die Dauer einer Darstellungsverbindung wirksam ist. Damit ergibt sich eine 1:1-Beziehung zwischen Darstellungsverbindungen und Assoziationen, und die ACSE-Dienstleistung zum Erzeugen einer Assoziation kann direkt auf die Darstellungsdienstleistung zum Aufbau einer Darstellungsverbindung abgebildet werden. Ebenso ist die ACSE-Dienstleistung zur Aufgabe einer Assoziation auf die Darstellungsdienstleistung zum Abbau einer Verbindung abbildbar.

Neben den beiden bestätigten Dienstleistungstypen zum Erzeugen und zum Beendigen einer Assoziation (Associate, Release) sind weiterhin eine Anzeige über den vorzeitigen Abbruch einer Assoziation aus medieninternen Gründen (PAbort) und eine unbestätigte Dienstleistung zur Anzeige des vorzeitigen Abbruchs durch einen Dienstnehmer (UAbort) vorgesehen.

Interessant ist die Parametrisierung des AssociateReq-Primitivs, weil man daran sehen kann, wie einerseits Parameter des Darstellungsdienst-Primitivs (ConnectReq) direkt enthalten sind - und dort zum Teil schon vom Kommunikationssteuerungs-Dienstleistungsprimitiv ConnectReq übernommen wurden - , und wie andererseits die beim Aufbau der Darstellungsverbindung vorgesehenen Nutzdaten weiter aufgespalten werden in zusätzliche ACSE-spezifische Parameter.

Parameter, die bereits vom Darstellungsdienst her vorgesehen sind, sind: Initiatoradresse, Beantworteradresse, Darstellungskontextanzahl, Darstellungskontextliste, Default-Darstellungskontextname, Verbindungseigenschaften, Darstellungsanforderungen, Kommunikationssteuerunganforderungen, Synchronisationsfolgenummer, Kommunikationssteuerungsrechtevergabe, Kommunikationssteuerungsverbindungsidentifikation).

Auf ACSE-Ebene kommen folgende Parameter hinzu: Anwendungsinstanzname des Initiators, Anwendungsinstanzname des Beantworters, Anwendungskontextname, Nutzinformation.

Anwendungsinstanznamen (application entity titles) sind von den Zugangspunktadressen des Transportsystems, die als Initiator- und Beantworteradressen genannt werden, zu unterscheiden. Sie identifizieren die Dienstnehmer als Individuen unabhängig von der Adresse momentan benutzter Zugangspunkte. Ein Beispiel soll dies verdeutlichen. Wenn z.B. der Handlungsreisende A. Müller mit der Personalnummer 007 von einem Telephonapparat mit der Nummer 4711 aus seine Firma anruft, so ist A. Müller 007 mit dem Anwendungsinstanznamen und 4711 mit der Initiatoradresse vergleichbar. Durch den Anwendungsinstanznamen ist es möglich, daß A. Müller von verschiedenen Apparaten aus seine Firma erreicht und dennoch als dasselbe Individuum erkannt wird.

Der Anwendungskontextname identifiziert das Anwendungsprotokoll (z.B. FTAM), gemäß dem die Darstellungsverbindung betrieben werden soll. Es ist vorgesehen, diese Namen in einer weltweit verbindlichen Liste zu führen, um Eindeutigkeit zu gewährleisten. Im Beispiel des Handlungsreisenden könnten so z.B. zwischen Firma und ihren Agenten bestimmte Gesprächstypen - wie 'Standortmeldung und Frage nach neuen Zielen', 'Abfrage aktueller Angebotseckdaten', 'Abstimmung der Konditionen eines Angebots und Angebotsbestätigung', 'Auftragsvereinbarung' - ausgemacht sein, die zur Beschleunigung jeweils nach bestimmten vorab festgelegten Mustern zu führen sind.

Nicht weiter reglementiert ist der Parameter Nutzinformation. Er entspricht dem Parameter Nutzdaten der niederen Dienste und wird hier, da es sich um eine Dienstleistung der Anwendungsschicht handelt, Nutzinformation genannt, um zu unterstreichen, daß an der Schnittstelle eines Anwendungsdienstes Daten immer im Zusammenhang mit der von der Anwendung bestimmten Bedeutung der Daten ausgetauscht werden.

7.11.2.2 Transaktionskontrolle - CCR

Während die ACSE-Dienstleistungstypen so universell sind, daß sie i.a. von allen Anwendungen aus benötigt werden, haben die Dienstleistungstypen zur Transaktionskontrolle ein wesentliches spezialisierteres Anwendungsgebiet und werden deshalb nur von einigen Anwendungstypen benötigt. Sie sollen die Kommunikation unterstützen, die von einer (auch mehr als 2 Instanzen umfassenden) Menge von Anwendungsinstanzen benötigt wird, wenn eine Transaktion über zwischen den Instanzen verteilten Datenbeständen auszuführen ist. Die Ausführung verteilter Transaktionen - besser: die Durchführung des Commit einer verteilten Transaktion - folgt dabei dem Zwei-Phasen-Commit-Protokoll.

Transaktion

Der Begriff Transaktion ist aus dem Bereich Datenbanken bekannt. Gegeben sei ein Datenbestand auf dem einzelne Aktionen (Lesen, Schreiben, Verändern von Komponenten) ausgeführt werden können. Eine Folge solcher Aktionen, die logisch zusammengehören, wird Transaktion genannt. Man geht davon aus, daß mehrere Auftraggeber nebenläufig Transaktionen auf der gemeinsamen Datenbasis ausführen, und daß die Datenbasis wichtig ist, d.h. dort per Transaktionen durchgeführte Änderungen sollen dauerhaft 'sicher' gespeichert werden.

Es gilt das ACID-Prinzip (Atomicity, Consistency, Isolation, Duration/Persistenz). Eine Transaktion soll obwohl intern in eine Folge von Aktionen gegliedert nach außen (für die übrigen nebenläufigen Zugreifer der Datenbasis) als unteilbare Einheit wirken, d.h. sie soll entweder ganz oder gar nicht ausgeführt werden. Sie soll konsistente Datenbasiszustände in konsistente überführen. Obwohl mehrere Transaktionen nebenläufig ausgeführt werden können, soll der Effekt auf die Datenbasis so sein, als ob jede Transaktion isoliert unter exklusivem Zugriff auf die Datenbasis ausgeführt worden wäre (auch Serialisierbarkeit genannt). Letzendlich soll die Datenbasis zuverlässig gespeichert sein und auch bei Ausfällen soll der von einer erfolgreich durchgeführten Transaktion hergestellte Zustand nicht verloren gehen.

Eine Transaktion läuft folgendermaßen ab. Ein Anwender vereinbart zunächst den Beginn einer neuen Transaktion. Dann ruft er die einzelnen Aktionen auf (Schreiben, Lesen, Verändern). Nun hat er die Wahl, die Transaktion ungeschehen zu machen (dazu ruft er 'Rollback' auf) oder den erfolgreichen Abschluß der Transaktion anzufordern (dazu ruft er 'Commit' auf). Nach Commit-Aufruf kann das Datenbasisverwaltungssystem u.U. erkennen, daß ein Commit nicht möglich ist, da dadurch andere (nebenläufige) Transaktionen gestört werden würden. Dann reagiert es ebenfalls mit Rollback und siganlisiert dies dem Nutzer. Will ein Nutzer sichergehen, daß ein Commit-Aufruf auch erfolgreich bearbeitet wird, kann er vorher 'PrepareCommit' aufrufen. Er erhält entweder eine positive Anzeige (sie signalisiert ihm, daß ein anschließendes Commit erfolgreich sein wird), oder eine negative. Auch nach einer positiven Anzeige kann er aber dennoch Rollback statt Commit fordern.

Einfaches Beispiel zu einer verteilten Transaktion

Folgendes einfache Beispiel soll zur Erläuterung dienen (siehe dazu auch Bild 7.8). Eine Bank A hat den Auftrag erhalten, die Überweisung eines Betrags xx von einem Konto K1, das bei einer Bank B geführt wird, auf ein Konto K2, das bei einer Bank C geführt wird, durchzuführen. Die Bank A hat ein eigenes Konto K bei sich, auf das sie ihre Gebühren überweist (ob das alles im Bankwesen so üblich ist, sei hier nicht von Interesse).

Die Ausführung des Auftrags ist eine verteilte Transaktion. Sie besteht aus drei lokalen Transaktionen (je eine bei einer der drei Parteien A, B und C). Die lokalen Transaktionen sind einfach und enthalten je nur eine Aktion (i.e., K:=K+Gebühr bzw. K1:=K1-xx-Gebühr bzw. K2:=K2+xx).

Die Banken können nur über unzuverlässige Telekommunikationsverbindungen kommunizieren. Außerdem ist es möglich, daß eine Bank ihre lokale Transaktion nicht erfolgreich abschließen kann.

CCR-Konzept für verteilte Transaktionen

Nach dem CCR-Konzept ist die 3-Parteien-Beziehung der drei Banken auf eine Baumstruktur aus 2-Parteien-Verbindungen abzubilden, wobei jeweils ein Knoten zu seinen Söhnen in Master-Slave-Beziehung steht. Im Beispiel könnte ein Baum mit der Wurzel A und den beiden Söhnen B und C gewählt werden. Kommunikation erfolgt zwischen A und B sowie zwischen A und C, aber nicht zwischen B und C erfolgen (im allgemeinen müssen B und C nicht unbedingt Blätter sein sondern könnten ihrerseits Master von Subtransaktionen werden; z.B. das Konto K1 wird in einer Filiale von B geführt und für Abbuchungen müssen auch Daten einer zweiten Filiale hinzugezogen werden).

Der Master wird mit jedem Slave den Beginn einer Slave-Transaktion vereinbaren und dann die entsprechenden Aktionsfolgen bei den Slaves anfordern. Dies ist noch nicht kritisch.

Kritisch ist das Commit. Master A kann z.B. zuerst Commit bei Slave B anfordern, und B kann das Commit erfolgreich durchführen. Dann wird A Commit bei C anfordern. Kann C jetzt Commit nicht ausführen, sondern muß Rollback veranlassen, dann ist die verteilte Transaktion nicht mehr atomar. B kann nicht mehr zurück. A kann nicht nach vorn.

Im verteilten System muß deshalb das Commit mit besonderen Verfahren durchgeführt werden. CCR setzt hierzu auf das (im Bereich verteilter Datenbanken wohlbekannte) Zwei-Phasen-Commit-Protokoll.

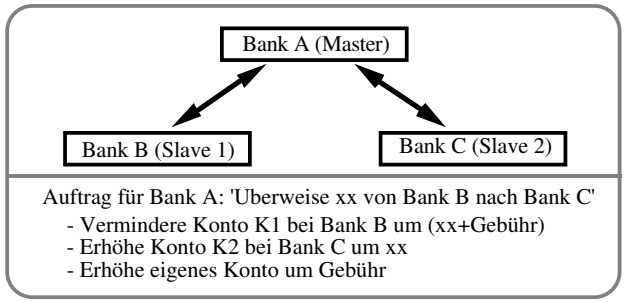

Bild 7.8: Beispiel zu CCR

Zwei-Phasen-Commit-Protokoll

Zentral ist, daß das Commit in 2 Phasen abgewickelt wird. In der ersten Phase fordert der Master PrepareCommit von den Slaves. Jeder Slave muß nun ermitteln, ob er bei späterer Anforderung Commit erfolgreich durchführen kann oder nicht. Jeder Slave gibt eine Rückmeldung an den Master. Bei positiver Rückmeldung hat er sich verpflichtet, einer späteren Commit-Anweisung unbedingt zu folgen. Hat der Master alle Rückmeldungen eingesammelt, beginnt die zweite Phase. Waren alle Rückmeldungen positiv, fordert er von allen Slaves Commit an. War eine Rückmeldung negativ (oder will er selbst die Transaktion inzwischen nicht mehr erfolgreich abschließen), fordert er von allen Slaves Rollback an. Auf diese Weise erhält man Atomizität auch bei verteilten Transaktionen.

Weitere Verteilungsaspekte

Durch die Entfernung der Partner treten jetzt noch weitere Besonderheiten auf. Ein Partner kann z.B. durch Defekte im ihn tragenden Rechner ausfallen, während die anderen lauffähig bleiben. Oder aber, die Darstellungsverbindung zwischen zwei Partnern kann für eine unbestimmte Zeit ausfallen. In solchen Fällen sollte die Transaktion insgesamt ebenfalls entweder überhaupt nicht durchgeführt worden sein, oder aber definiert nach Beseitigung des Ausfalls fortsetzbar. Hierzu müssen die Partner ihre lokalen Aktionen für sich gesehen bereits als Transaktionen behandeln, so daß einerseits eine Rücknahme auf Anforderung möglich ist und andererseits - vom Zeitpunkt der Benachrichtigung über die Durchführbarkeit der Aktion - diese auf Anforderung auch sicher ausgeführt werden kann (u.U. über Systemausfälle hinweg). Hierdurch wird die Problematik von Instanzenausfällen - unter der Annahme, daß solche Ausfälle zeitlich befristet sind - gelöst.

Verbindungsausfälle sollen ebenfalls als zeitlich befristet angenommen werden. Verliert z.B. A die Verbindung zu B, so soll von einer später wieder möglichen Kommunikation ausgegangen werden. Ferner erfolge ein Verbindungsverlust nicht unbemerkt sondern werde beiden betroffenen Partnern angezeigt, so daß die Partner im Falle einer solchen Anzeige den nur dann möglichen Verlust von Nachrichten einkalkulieren können. Zum Wiederaufsetzen nach Verbindungsverlust benötigen sie eine weitere Kommunikationsdienstleistung, mit der beim Partner dessen Stand in der Abwicklung erfragt werden kann.

Einen entsprechenden Ablauf des Masters zeigt Bild 7.9.

Aktionen des Masters:

```
1)   Begin 'Lokale Transaktion'
2)   Sende 'Begin Untertransaktion 1' an B
3)   Sende 'Begin Untertransaktion 2' an C
4)   Sende 'K1:=K1-xx-Gebühr' an B
5)   Sende 'K2:=K2+xx' an C
6)   Eigene Aktion 'K:=K+Gebühr'
7)   Prüfe, ob eigene Transaktion erfolgreich beendbar.
     Wenn nein: 23
8)   Sende 'Prepare Commit' an B
9)   Sende 'Prepare Commit' an C
10)  Erwarte Rückmeldungen von B und C
     Falls eine von beiden nicht positiv: 23
11)  Sende 'Commit' an B
12)  Sende 'Commit' an C
13)  Commit für eigene Transaktion
14)  Erwarte Rückmeldungen von B und C
15)  Falls Rückmeldungen ausbleiben: 17
16)  Fertig: Transaktion erfolgreich beendet

17)  (* Rückmeldung auf Commit blieb aus*)
     Sende 'Restart' an B bzw. C bzw. beide
18)  Erwarte Statusmeldung(en)
     Falls Status=comitted: Fertig: Transaktion erfolgreich beendet
19)  Sende 'Commit' an B bzw. C bzw. beide
20)  Erwarte Rückmeldung(en)
21)  Falls Rückmeldung(en) ausbleiben: 17
22)  Fertig: Transaktion erfolgreich beendet

23)  (* Mindestens eine Untertransaktion nicht erfolgreich beendbar: Rollback *)
     Sende 'Rollback' an B
24)  Sende 'Rollback an C'
25)  Rollback eigene Transaktion
26)  Erwarte Rückmeldungen von B und C
27)  Falls Rückmeldungen ausbleiben: 29
28)  Fertig: Transaktion nicht durchgeführt, Zustand wie vorher

29)  (* Rückmeldung auf Rollback blieb aus  *)
     Sende 'Restart' an B bzw. C bzw. beide
30)  Erwarte Statusmeldung(en)
     Falls Status=rollback: Fertig: Transaktion nicht durchgeführt
31)  Sende 'Rollback' an B bzw. C bzw. beide
32)  Erwarte Rückmeldung(en)
33)  Falls Rückmeldung(en) ausbleiben: 29
34)  Fertig: Transaktion nicht durchgeführt
```

Bild 7.9: Zwei Phasen Commit Protokoll am Beispiel

CCR-Dienstleistungstypen

Nach dieser Erläuterung am Beispiel soll die Erörterung der CCR-Dienstleistungstypen mit einer Aufzählung der Typen abgeschlossen werden:

- unbestätigte Anforderung einer Transaktion eines Slave durch den Master (Begin); die Transaktion wird durch Parameter spezifiziert,
- (-) wie die einzelnen Aktionsanforderungen übertragen werden ist nicht durch CCR festgelegt,
- unbestätigte Aufforderung eines Slave durch den Master, auf die vorangegangene Transaktionsanforderung nun zu reagieren (PrepareCommit);

diese Dienstleistung ist aus Sicht des Zwei-Phasen-Commit-Protokolls unbedingt notwendig, aber nicht aus Sicht der CCR-Norm; nämlich dann nicht, wenn der PrepareCommit-Wunsch implizit in einer Aktionsanforderung übertragen wird,
- unbestätigte Benachrichtigung des Master über die Bereitschaft zur erfolgreichen Transaktionsdurchführung durch einen Slave (Ready),
- unbestätigte Benachrichtigung des Master darüber, daß ein Slave die geforderte Transaktion nicht durchführen kann (Refuse),
- bestätigte Vereinbarung des Transaktionsabschlusses zwischen Master und einem Slave mit dem Master als Initiator (Commit),
- bestätigte Vereinbarung der Rücknahme der geforderten Transaktion zwischen Master und Slave mit Master als Initiator (Rollback),
- bestätigte Dienstleistung zum Erfragen und Rückmelden des momentanen Stands der Transaktion aus der Sicht des gefragten Partners (Restart).

CCR-Protokoll

Das CCR-Protokoll bildet die CCR-Dienstleistungstypen auf Presentation/Session-Dienstleistungen ab. Zu beachten ist, daß das CCR-Protokoll nicht identisch ist mit dem Zwei-Phasen-Commit-Protokoll. Das Zwei-Phasen-Commit-Protokoll muß in den Anwendungsprozessen selbst (in unserer Schicht 8) verfolgt werden. Das CCR-Protokoll realisiert nur den Transfer von gemäß Zwei-Phasen-Commit-Protokoll auszutauschenden Nachrichten.

7.11.2.3 'Zuverlässige' Transferdienstleistungen - RTSE

RTSE soll Datentransferdienstleistungen zur Verfügung stellen, die zuverlässig sind und somit eine im Vergleich zu CCR vereinfachte Gestaltung kooperierender Anwendungsprozesse ermöglichen. Nun kann es allerdings - wie schon häufig erwähnt - im echten Telekommunikationssystem völlig zuverlässige Transfers nicht geben:
- das Datenpaket kann verloren gehen,
- die Transferzeit für das Datenpaket ist nicht vorhersagbar.

Die zentrale RTSE-Dienstleistung 'Transfer' (daneben gibt es mit ACSE vergleichbare zur Einbettung in eine Assoziation) folgt dem Paradigma 'Einschreiben mit Fristsetzung und Rückschein'. Bild 7.10 verdeutlicht den Ablauf. Ein Absender sendet mittels TransferReq(Gültigkeitsdauer, Kennung, Datum) ein Datum ab. Der Dienst wird es nur ausliefern, wenn es innerhalb der per Gültigkeitsdauer gesetzten Frist ausgeliefert werden kann. Dann wird der Dienst selbst eine positive Rückmeldung erzeugen. Kann das Datum nicht rechtzeitig ausgeliefert werden, erzeugt der Dienst eine negative Rückmeldung.

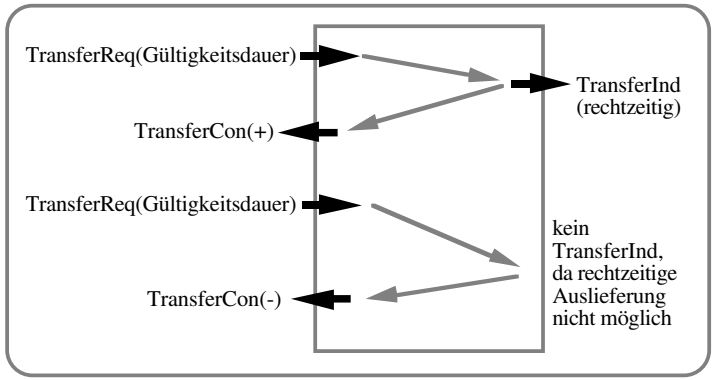

Bild 7.10: RTSE - Transfer

Erhält ein Absender eine positive Rückmeldung, kann er davon ausgehen, daß das Datum rechtzeitig ausgeliefert wurde. Erhält er eine negative, kann er davon ausgehen, daß das Datum dem Empfänger überhaupt nicht ausgeliefert wurde. Wenn der Absender keine Rückmeldung erhält, muß er denselben Auftrag (durch Parameter 'Kennung' erkennbar) nocheinmal absetzen. Der Empfänger kann per 'Kennung' Wiederholungen erkennen und ignorieren.

Das RTSE-Protokoll sieht die Nutzung der Kommunikationssteuerungsdienstleistungen zur Aktivitätssteuerung für die Abwicklung eines Reliable Transfers vor (1 Aktivität je Transfer).

7.11.2.4 Abgesetzte Operationen - ROSE

Mit den Remote Operation Service Elements werden Dienstleistungstypen vereinbart, welche die zur abgesetzten Ausführung von Operationen erforderliche Kommunikation unterstützen. Es wird von einer Assoziation zwischen 2 Anwendungsinstanzen ausgegangen, innerhalb derer eine Instanz die andere zur Ausführung von Operationen veranlassen will und u.U. auch umgekehrt. Die Instanz, die eine zu ihr abgesetzte Operationsausführung veranlaßt, werde im folgenden Client genannt, die ausführende Instanz Server.

Bild 7.11 zeigt ein entsprechendes Szenario. Ein Anwendungsprozeß 'Client' ruft eine Operation XX auf, die entfernt durch einen Anwendungsprozeß 'Server' auszuführen ist.

Im Normalbetrieb ergibt sich folgender Ablauf einer Operationsausführung:
1) Der Client benachrichtigt den Server über seinen Wunsch nach der Ausführung einer bestimmten Operation mit bestimmten Datenargumenten (Eingabeparametern).
2) Der Server hat die gewünschte Operation erfolgreich ausgeführt und teilt dies dem Client unter Übertragung der Ergebnisargumente (Ausgabeparameter) mit.

Zu ROSE gibt es zwei Protokollvarianten:
- Aufsetzen auf Presentation / Session; hier ist der Anwendungsprozeß nicht reglementiert bezüglich der Behandlung von Besonderheiten. Es kann der Abbruch der Verbindung (mit der Möglichkeit des Verlusts von Nachrichten) auftreten. Hierzu, wie auch zum möglichen zwischenzeitlichen Nachrichtenverlust erfolgen Anzeigen gemäß Darstellungsdienst / Kommunikationssteuerungsdienst, und es liegt in der Verantwortung der Anwendungsinstanzen, darauf im Sinne der Anwendung geeignet zu reagieren.

- Transfer der ROSE-PDUs per RTSE-Transfers; hier gelten bei Besonderheiten die RTSE-Konventionen.

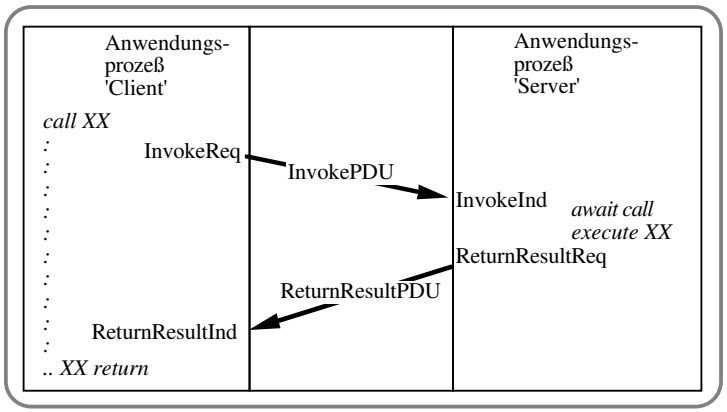

Bild 7.11: ROSE - Remote Operation Service Elements

Im letzten Fall werden Operationsaufrufe mit 'exactly once'-Semantik durchgeführt, d.h., eine gerufene Operation wird je Aufruf - auch bei Auftreten von Besonderheiten - genau einmal ausgeführt. Im ersten Fall hängt es von der Art der Session-Nutzung ab, ob 'exactly once', 'at least once' oder 'at most once' realisiert wird.

Im ROSE-Konzept ausdrücklich behandelte Besonderheiten beziehen sich darauf, daß einerseits der Server die geforderte Operationsausführung sofort ablehnen kann (z.B. Operationstyp unbekannt, oder Operation für diesen Client nicht statthaft), und daß andererseits die Ausführung der Operation zwar in Angriff genommen wird, aber vom Server aufgrund von Besonderheiten in der Operationsausführung nicht erfolgreich beendet werden kann. Im ersten Fall benachrichtigt der Server den Client über die Ablehnung unter Nennung eines Ablehnungsgrunds, im zweiten Fall benachrichtigt er ihn über die aufgetretene Fehlersituation und nennt dabei u.U. sehr detaillierte Fehlerparameter, die in ihrer Auslegung vom Typ der Operation bestimmt sind.

Die derart insgesamt benötigten Dienstleistungstypen sind alle unbestätigt:

- Operationsanforderungsbenachrichtigung (Invoke (Operationscode, Aurufkennung, Operationsargument)),
- Operationsdurchführungsbenachrichtugung (ReturnResult (Aufrufkennung, Ergebnis)),
- Operationsablehnungsbenachrichtigung (Reject (Aufrufkennung, Ablehnungsgrund)),
- Operationsfehlerbenachrichtigung (ReturnError (Aufrufkennung, Fehler, Fehlerparameter)).

Der Parameter Operationscode ist eine Zahl in der der Typ der Operation (d.h., der Operationsname) codiert ist. Der Parameter Aufrufkennung kann vom Aufrufer bei Invole freizügig gewählt werden. Server-Nachrichten enthalten immer die Kennung des Invokes auf die sich die Servernachricht bezieht. Operationsargument ist der Eingabeparameter der Operation (es gibt nur einen, aber man kann ja mehrere - per ASN.1 SET bzw. SEQUENCE - zu einem zusammenfassen). Ergebnis ist der Ausgabeparameter.

Derzeit wird pei der Operationsparameterübergabe nur die Übergabe 'by value' unterstützt. Übergaben 'by name' oder 'by reference / by pointer' bereiten im verteilten System generell Probleme (es gibt aber Lösungsmöglichkeiten) sind derzeit nicht erfaßt.

Es soll noch erwähnt werden, daß es im Zusammenhang mit der Nutzung einer Assoziation durch ROSE-Dienstleistungen vorgesehen ist, die Assoziation gemäß besonderen Nutzungsmodalitäten zu klassifizieren, und die Modalitäten bereits mit der Erzeugung der Assoziation durch die CASE/ACSE-Dienstleistung Assoziationserzeugung zwischen den beiden Anwendungsinstanzen abzustimmen. So ist es von Interesse, ob innerhalb der Assoziation nur eine bestimmte der beiden Anwendungsinstanzen die Rolle des Client spielen soll, oder ob beide dazu berechtigt sind. Ferner interessiert, ob Operationen 'synchron' oder 'asynchron' aufgerufen werden sollen. Hierbei steht synchron dafür, daß nach der Anforderung einer Operationsausführung bis zur Endemeldung (ReturnResult / ReturnError / Reject) keine weitere Operation angefordert werden darf, während der asynchrone Betrieb es gestattet, unbeachtet der Rückmeldungen beliebig neue Ausführungsanforderungen abzusetzen (hier wird dann die Aufrufkennung wichtig).

7.12 Protokolle der Anwendungsschicht

Auch in der Anwendungsschicht zeigen sich die bereits bei der Kommunikationssteuerungsschicht festgestellten Verhältnisse.Der erbrachte Kommunikationsdienst ist zwar für die Nutzung in einer sehr differenzierten Kooperationsbeziehung konzipiert, die Dienstnehmer müssen, um den Dienst sinnentsprechend zu nutzen, in eigener Verantwortung komplexere Ablauffestlegungen einhalten (z.B. Regime-Klammerung bei FTAM), und es wird ein breites Spektrum unterschiedlicher Dienstleistungstypen unterstützt. Aber andererseits werden alle Dienstleistungen im groben direkt auf Dienstleistungen des untergeordneten Dienstes abgebildet. Hierdurch ergeben sich sehr einfache Ablauffestlegungen für die Instanzen und den größten Raum in der Protokollbeschreibung nimmt die Festlegung der den einzelnen Dienstleistungstypen zugeordneten PDU-Typen ein. Dies kann nun durch Definition abstrakter PDU-Formate erfolgen.

Dies zeigt sich auch in den beiden Protokollvorschlägen zur Assoziationskontrolle und zur Nebenläufigkeitssteuerung (CASE - ACSE und CCR, ISO DIS 8650), die die gemeinsamen Dienstleistungstypen für Anwendungen erbringen.

Die Erzeugung und Aufgabe von Assoziationen geht so direkt im Auf- und Abbau von Darstellungsverbindungen auf. Gegenüber dem Darstellungsverbindungsaufbau zusätzliche Kontrollparameter der Dienstleistung Assoziationserzeugung werden in den beiden zugeordneten ACSE-PDU-Typen Assoziationsanforderung und Assoziationsbestätigung hinterlegt, und als Nutzdaten der Darstellungsdienstleistung 'Verbindungsaufbau' übertragen.

Das CCR-Protokoll sieht ebenfalls ein den Dienstleistungstypen entsprechend gefächertes Spektrum von PDU-Typen vor. Auch die Abbildung von CCR-Dienstleistungen auf Darstellungsdienstleistungen bleibt direkt und nutzt die schon vom Kommunikationssteuerungsdienst her gegebenen Möglichkeiten zur Strukturierung des Transfers durch Vereinbarung und Gebrauch von Synchronisationspunkten. So wird z.B. die Dienstleistung zur Anforderung des Beginns einer Transaktion auf die Kommunikationssteuerungsdienstleistung zur Vereinbarung eines Hauptsynchronisationspunkts abgebildet. Die Benachrichtigung über die Bereitschaft zur Aktionsdurchführung wird durch Typendaten-Transfer übertragen. Die Ablehnung entspricht einer Resynchronisation und eine Commit-Vereinbarung geht in einer Vereinbarung des nächsten Hauptsynchronisationspunkts auf. Eine zusätzliche Gliederung des Übertragungsgeschehens durch Nebensynchronisationspunkte ist möglich, wird aber vom CCR-Protokoll nicht ausdrücklich reglementiert.

7.13 Anwendungsdienste / Anwendungen

Den Abschluß des Kapitels soll eine kurze (und nicht vollständige) Übersicht über einige zur Zeit in Entwicklung oder Gebrauch befindlichen Anwendungsdienste / Anwendungstypen bilden.

Internet - Anwendungen:
- Telnet; abgesetztes terminal,
- FTP; File transfer,
- Mail; elektronische Post.

'Value added' Telecommunication Services (sogenannte VANs, value added (communication) networks):
- Teletex (deutsch: Bildschirmtext),
- E-Mail nach CCITT X.400,
- Directory (Verzeichnisdienst) nach CCITT X.500.

'Traditionelle' ISO/OSI-Standardisierungsbestrebungen:
- Virtual terminal,
- Job Transfer and Management,
- File Transfer, Access, and Management.

Büroautomation:
- Directory,
- Office Document Architecture, Office Document Interchange (ODA/ODIF),
- Filing,
- Remote Data Base Access,
- Printing,
- Time,
- Authorisation and Authentication.

Fabrikautomatisierung:
- MAP/MMS (Manufactoring Automation Protocols/Manufactoring Message Specification),

Handel, Verwaltung und Transport:
- EDIFACT (Electronic Data Interchange For Administration, Commerce, and Transport).

Electronic Banking / Electronic Sales:
- Electronic Funds Transfer,
- 'Point of Sale'-Buchung.

Verteilte allgemeine Informationssysteme (z.B. World Wide Web WWW).

8. Implementierung

Nachdem in den vorangegangenen Kapiteln die logische Architektur von Telekommunikationssystemen eingeführt wurde, soll nun auf ihre Realisierung mittels Rechnerknoten und Datenübertragungseinrichtungen eingegangen werden. Datenübertragungseinrichtungen und ihre Ankopplung an Rechner wurden in Kap. 2 schon besprochen, so daß als zentrales Thema die Implementierung der Kommunikationssystem-Schichten ab Schicht 2 verbleibt.

Die logische Architektur gliedert eine Schicht in Dienstschnittstellen und Instanzen, und im folgenden soll die Aufgabe der Implementierung einer Schicht auch unter Gliederung in die beiden entsprechenden Teilaufgaben 'Dienstschnittstellenimplementierung' und 'Instanzenimplementierung' erörtert werden.

Wir konzentrieren uns dabei auf die software-mäßige Implementierung, wie sie - in der Regel für höhere Schichten - unmittelbar in einem Universalrechner, der auch die eigentlichen Anwendungsprozesse trägt, oder - üblicherweise für die Schichten 2 und 3 - in dem Universalrechner vorgelagerten, auf der Anschalteinrichtung zum Netz hin untergebrachten dedizierten Rechnern vorgenommen wird.

Mit der durch den technischen Fortschritt zunnehmenden Möglichkeit, Übertragungseinrichtungen sehr hoher Bitrate (Hochgeschwindigkeitsnetze, > 100 Mbit/s) kostengünstig einzusetzen, wird auch die hardware-mäßige, nicht programmgesteuerte Implementierung von Protokollen durch firmwaregesteuerte oder auch festverdrahtete speziell dafür entwickelte VLSI-Chip-Sätze eine immer größere Rolle spielen, weil die Verarbeitung der hohen Datenraten durch derzeitige frei programmierbare Standardprozessoren nicht gewährleistet ist. Andere Lösungsansätze dazu basieren auf der Verwendung dedizierter Mehrprozessor-Systeme. Hierauf soll jedoch nicht weiter eingegangen werden.

8.1 Implementierung von Dienstschnittstellen

Die logische Architektur eines Telekommunikationssystems sieht Instanzen vor, die an Dienstzugangspunkten durch den Austausch von Schnittstellenereignissen vertikal im Verlauf der Zeit miteinander kommunizieren. Im folgenden Abschnitt sollen die wesentlichen Mechanismen vorgestellt werden, die zur Abbildung der logischen Schnittstelle innerhalb der Implementierung des einen Ort - d.h. einen bestimmten Rechner im Rechnernetz - betreffenden Teils verwendet werden.

Die Schnittstellenmechanismen der Implementierung hängen dabei in ihrer geeigneten Verwendbarkeit sehr stark von den Randbedingungen der Implementierung ab. Dies sind einerseits, der verwendete Rechner und das zugrundgelegte Betriebssystem in seiner Bauart, seinen funktionellen und Leistungseigenschaften. Dem soll im folgenden durch die Erläuterung unterschiedlicher, alternativ einsetzbarer Mechanismen Rechnung getragen werden.

Andererseits ergibt sich eine starke Abhängigkeit zur gewählten Form der Implementierung der Instanzen, und beim Entwurf der Implementierung eines Telekommunikationssystems insgesamt sollte die Konzeption der Instanzen- und der Schnittstellenrealisierung unbedingt aufeinander abgestimmt erfolgen. Hierzu soll im folgenden nur grob unterschieden werden, ob ein Schnittstellenereignis in-

nerhalb eines Prozesses im Sinne des Betriebssystems stattfindet oder ob die vertikale Kommunikation Prozeßgrenzen zu überwinden hat, also in eine Form von Interprozeßkommunikation abgebildet werden muß. Ersteres findet sich häufig bei den Schnittstellen der anwendungsorientierten Dienste, letzteres ist vor allem bei der Transportdienst-Schnittstelle von großem Nutzen, weil sich hier die Grenze zwischen standardmäßig vom System her zu unterstützenden Interrechner-Kommunikationsfähigkeiten und anwendungsprozeß-bezogenen, spezifischen Dienstleistungen befindet.

Abschließend soll angemerkt werden, daß die Implementierung einer Dienstschnittstelle vom ISO / OSI-Modell her, das nur die logische Architektur betrifft, überhaupt nicht reglemeniert ist. Es besteht kein Zwang, diese logische Schnittstelle in die Implementierung eines Systems so abzubilden, daß in der Struktur der Implementierung eine direkt entsprechende und in sich geschlossene Software-Schnittstelle sichtbar ist.

8.1.2 Schnittstelle über Prozeßgrenzen

Bei einer Schnittstelle über Prozeßgrenzen wird davon ausgegangen, daß oberhalb und unterhalb der Schnittstelle eine Menge von Prozessen existiert, wobei jeweils ein Prozeß oder auch eine bestimmte Gruppe von Prozessen eine Instanz darstellt. Im folgenden soll zunächst der Begriff des Prozesses näher erörtert werden, bevor auf Mechanismen zur Kommunikation zwischen Prozessen und ihre Anwendung zur Implementierung von Dienstschnittstellen eingegangen wird. Kommunikation zwischen Prozessen umfaßt dabei die beiden Teilaspekte der gegenseitigen Abstimmung des zeitlichen Ablaufs verschiedener Prozesse (Synchronisation) und der Übergabe von Daten von einem Prozeß an andere (Datenaustausch).

8.1.2.1 Prozeß

Der Begriff Prozeß soll im Sinne von Software-Prozeß verstanden werden, wie er i.a. von den heute üblichen Mehrprozeß-Betriebssystemen wie z.B. UNIX, VAX/VMS unterstützt wird. Mit Prozeß wird dort der Verbund einer Menge von Betriebsmitteln - Code, Arbeitsspeicher, Prozessorzeit, u.U. noch weitere wie E/A-Gerätenutzungszeit, Peripherspeicherbereiche - bezeichnet, der zur einmaligen Ausführung eines Programms, des Codes des Prozesses, benötigt wird. Den Prozeßcode könnte man sich z.B. als PASCAL-Hauptprogramm vorstellen, und er dient i.a. der Bewältigung einer logisch weitgehend in sich geschlossenen Aufgabe. Verschiedene Ausführungen desselben Programms entsprechen verschiedenen Prozessen.

Im Normalfall besitzt ein Prozeß keine direkten Querbeziehungen zu anderen gleichzeitig vom System getragenen Prozessen. Das Betriebssystem kapselt den Prozeß von allen anderen ab, so daß er isoliert so ausgeführt wird, als stünde für die Ausführung ein eigener Rechner zur Verfügung. Die Speicherbereiche des Prozesses (z.B. für die im Hauptprogramm vereinbarten Variablen, für den Laufzeitkeller und die Halde einer PASCAL-Programmausführung) sind nur dem zugeordneten Prozeß zugreifbar (Schutzaspekte). Der oder die Prozessoren des Rechners werden dem Prozeß vom Betriebssystem so zugeordnet, daß er - wenn der Rechner nicht gerade überlastet ist - unabhängig vom Fortschritt anderer Prozesse in seiner Ausführung so vorankommt, als stünde ihm ein eigener Prozessor ganzzeitig zur Verfügung (Nebenläufigkeit).

Weitergehend besitzen normale Anwendungsprozesse die Eigenschaft der Terminierung nach endlicher Zeit. Ein Prozeß zur Ausführung eines Programms, das z.B. eine Liste von Primzahlen berechnen und ausdrucken soll, ist - wenn man von möglichen Programmierfehlern (Endlosschleife) absieht - nur für eine endliche Zeit im System. Nach der Installation (Laden des Codes, Definition des Prozesses unter Zuordnung der wichtigen Betriebsmittel), wird der Prozeß für eine Zeitdauer ausgeführt,

bis er programmgemäß terminiert (z.B. im Hauptprogramm die 'end.'-Anweisung erreicht). Anschließend wird er vom Betriebssystem unter Freigabe der belegten Betriebsmittel vernichtet. Zur Vermeidung von - aus Programmierfehlern resultierenden - Endlosprozessen sind oft mit der Installation maximale Laufzeiten vereinbar, die vom Betriebssystem überwacht werden und bei Überschreitung zu einem außerplanmäßigen Abbruch des Prozesses führen.

Im Vergleich zu einem solchen Normalfall, dem terminierenden und in sich geschlossenen Anwendungsprozeß werden in der Regel noch Abarten durch ein Mehrprozeßbetriebssystem unterstützt, die gerade in unserem Anwendungsgebiet der Implementierung von Telekommunikationssystemen wichtig sind.

So kann es Prozesse geben, die, solange keine schwerwiegenden Laufzeitbesonderheiten auftreten, gerade nicht terminieren sollen und deshalb als endlose Prozesse geplant sind (ihr Programmcode enthält eine Endlosschleife). Diese Form wird oft gewählt, um auf einem Rechner Dienstleistungen für andere Prozesse anzubieten, die in gleichartiger Weise während der gesamten Betriebszeit des Rechners unterstützt werden sollen. Ein Beispiel dazu wäre ein Treiberprozeß, dem allein der physikalische Zugriff auf ein bestimmtes E/A-Gerät gestattet ist, und der von anderen Prozessen Aufträge für Aus- und Eingaben an diesem Gerät annimmt und für sie abwickelt.

Solche Prozesse, wie Treiberprozesse, unterscheiden sich von normalen Anwendungsprozessen auch darin, daß sie inhaltlich oft einen Teil des Betriebssystems bilden. Sie übernehmen wesentliche Aufgaben in der Organisation und Abwicklung des normale Anwendungsprozesse betreffenden Ablaufgeschehens, und ihnen müssen hierzu im Vergleich zu den Anwendungsprozessen ein erweiterter Befehlsumfang und erweiterte Zugriffsrechte auf Betriebsmittel zugestanden werden. Sie sind deshalb gegenüber normalen Anwendungsprozessen privilegiert. Ihnen sind z.B. E/A-Befehle der Maschinensprache zugänglich und sie dürfen - unbeachtet gegebenenfalls bestehender Speicherschutzmaßnahmen - auf beliebige Arbeitsspeicheradressen zugreifen. Während im Idealfall Programmierfehler im Programm eines Anwendungsprozesses unter keinen Umständen zu einer negativen Beeinflussung anderer Prozesse wie überhaupt zu einer Störung des Gesamtablaufs führen sollen, ist dies bei einem privilegierten Prozeß nicht ausgeschlossen und sogar in der Regel der Fall.

Am Beispiel des Treiberprozesses kann ebenfalls auf eine dritte Besonderheit eingegangen werden. Er ist in der Abwicklung seiner Aufgaben gegenüber anderen Prozessen nicht in sich geschlossen. Wohl gibt es bei seinem Ablauf einzelne Phasen, in denen er in sich geschlossene Teilaufgaben bearbeitet (z.B. die Ausgabe einer Textzeile auf einem Bildschirm veranlaßt), aber er wird auch Phasen durchlaufen müssen, in denen er mit anderen Prozessen kommuniziert, um anschließende Aktionen in Abhängigkeit zur stattgefundenen Kommunikation durchzuführen (er muß z.B. bereit sein von einem anderen Prozeß den Auftrag zu einer bestimmten Ausgabe anzunehmen und sich die entsprechenden Auftragsparameter mitteilen zu lassen). Die Lösung der dem Prozeß gestellten Aufgabe insgesamt ist ohne Kommunikation des Prozesses mit anderen Prozessen nicht möglich. Ein solcher Prozeß soll über Prozeßgrenzen kommunizierender Prozeß genannt werden, und er enthält Operationen zur Durchführung von Interprozeßkommunikationsaktionen.

8.1.2.2 Synchronisation

Synchronisation bezieht sich darauf, daß ein Prozeß in seinem Programmcode eine Stelle erreicht, bei der er den weiteren Fortschritt davon abhängig macht, daß der andere Prozeß ebenfalls eine bestimmte Stelle erreicht hat. So kann z.B. ein Prozeß A an einer Stelle a alle Aktivitäten einstellen und darauf warten, daß ein anderer Prozeß B eine gewisse Stelle b erreicht hat. Durch Synchronisation kann die Ausführung der beiden Prozesse in ihrem zeitlichen Fortschritt aufeinander abgestimmt werden.

Zur Lösung von Synchronisationsproblemen werden zur Zeit im wesentlichen zwei Mechanismen eingesetzt:

- Warten und Fortsetzen,
- Rendezvous-Synchronisation.

8.1.2.2.1 Warten und Fortsetzen

Mechanismen des Wartens und Fortsetzens werden den Prozessen durch zwei Operationen zugreifbar:

- Warten ; diese Operation wird an der Wartestelle a durch A aufgerufen und blockiert die weitere Ausführung von A,
- Fortsetzen; diese Operation wird durch B an der Stelle b aufgerufen und löst die Blockierung von A.

Bei der genaueren Auslegung der Warte- und Fortsetzoperation ist ferner zu berücksichtigen, daß u.U. B auch schneller als A ist, und die Fortsetzoperation schon aufgerufen hat, bevor A wartet. Hierzu wäre festzulegen, ob Deblockierungsanstöße, die mangels wartendem Partner nicht sofort wirksam werden können, gepuffert werden sollen oder verloren gehen.

8.1.2.2.2 Rendezvous-Synchronisation

Die Rendezvous-Synchronisation ist symmetrisch und kommt mit nur einem Operationstyp aus. Die Operation 'Treffen' wird in beiden Prozessen A und B an den entsprechenden Stellen a und b aufgerufen. Im Falle, daß A die Stelle a erreicht, bevor B b erreicht hat, wird A mit der Operation 'Treffen' blockiert, bis anschließend B ebenfalls die Operation 'Treffen' aufruft. Es ergeben sich also dieselben Eigenschaften wie beim Warten und Fortsetzen. Im umgekehrten Fall - B erreicht b, bevor A die Stelle a erreicht hat - wird nun allerdings symmetrisch verfahren. B wird nun blockiert, bis A an der Stelle a ebenfalls die Operation 'Treffen' aufruft. Mit dem Rendezvous-Mechanismus ergibt sich also eine engere Synchronisation. Es wird sichergestellt, daß die beiden Zeitintervalle, in denen einerseits A und andererseits B mit der Ausführung der 'Treffen'-Operation befaßt sind, sich überschneiden.

8.1.2.2.3 Varianten

Varianten der Synchronisationsoperationen beziehen sich darauf, daß ein Prozeß u.U. verschiedene Synchronisationspunkte durchlaufen soll, um sich z.B. mit verschiedenen anderen Prozessen zu synchronisieren. Hierzu können verschiedene Wartestellen bzw. Treffpunkte i.a. durch Identifikationen unterschieden werden, mit denen die Operationsaufrufe parametrisiert werden. Möglich ist es, daß Wartestellen bzw. Treffpunkte Objekte eines eigenständigen Laufzeitobjekttyps sind, die z.B. bei der Installation eines Prozeßsystems per Bedienerkommando oder auch von einem Prozeß aus durch Aufruf einer Erzeugungsoperation eingerichtet und mit einem logischen Namen versehen werden können.

8.1.2.2.4 Realisierung

Mindestens eine solche Form der Interprozeßsynchronisation wird in der Regel von einem Mehrprozeßbetriebssystem unterstützt, dessen Komponente zur Prozeß- und Prozessorverwaltung entsprechende Operationen anbietet. Diese Operationen werden betriebssystemintern so realisiert, daß wartende Prozesse das wichtige Betriebsmittel 'Prozessor' freigeben (Prozeßumschaltung), um es im Sinne einer guten Ausnutzung des Rechners einem anderen Prozeß zur Verfügung zu stellen. Zur Lösung von Synchronisationsproblemen sollte deshalb auch generell auf diese Operationen zurückgegriffen werden, und die Programmierung einer Wartestelle durch eine während des Wartens fortwährend ausgeführte Schleife (aktives Warten) sollte vermieden werden.

8.1.2.3 Datenaustausch

Datenaustausch bezieht sich darauf, daß ein Prozeß bestimmte Werte berechnet und im Speicher abgelegt hat, die dem anderen Prozeß zur weitergehenden Bearbeitung übergeben werden sollen. Zum sinnvollen Datenaustausch sind immer geeignete Synchronisationsmaßnahmen notwendig. Wenn z.B. Prozeß A eine Datenstruktur D mit Werten belegt, die von B auszuwerten sind, so muß auch der zeitliche Ablauf von B mit dem von A abgestimmt werden, um sicherzustellen, daß B die von A berechneten Werte und nicht zufällig bereits früher am selben Platz befindliche durch zu frühzeitige Auswertung berücksichtigt.

Für den Datenaustausch zwischen verschiedenen Prozessen wurden eine Vielzahl unterschiedlicher Konzepte eingeführt. Die wesentlichsten sind (aufgezählt in der Reihenfolge zunehmender Integration der benötigten Synchronisation und zunehmender Anwendungsnähe):

- Geteilter Speicher;
- Fremdzugreifbarer Speicher;
- Auftragsübergabe;
- Datenaustausch im Rendezvous;
- Nachrichtenaustausch;
- Monitore.

Abschließend wird unter dem Stichwort 'geteilte Puffer' auf eine Kombination unterschiedlicher Mechanismen zur Aufwandsverringerung eingegangen.

8.1.2.3.1 Geteilter Speicher

In vielen Betriebssystemen - oft allerdings nur in sehr beschränktem Umfang - können Bereiche des Arbeitsspeichers zur geteilten Nutzung durch mehrere verschiedene Prozesse eingerichtet werden. Wenn nun z.B. eine Variable V_A eines Prozesses A am selben Ort lokalisiert ist, wie eine Variable V_B eines Prozesses B, so kann z.B. die Darstellung eines Werts, die A durch eine Zuweisung 'V_A:=...' dort hinterlegt hat, in B durch einen Lesezugriff auf V_B verfügbar werden. Durch die geteilte Nutzung der Speicherdarstellung erfolgt der Datenaustausch mit denkbar geringem Aufwand.

Zur sinnvollen Anwendbarkeit geteilten Speichers müssen aber noch zusätzlich Synchronisationsaufgaben gelöst werden. Ist zwischen A und B nur ein einmaliger Datenaustausch per V vorgesehen, so ist eine Reihenfolgesynchronisation zu verwirklichen. B erhält eine Wartestelle, die von A erst nach erfolgtem Schreiben der Variable gelöst werden darf.

Oft wird die Variable V zum mehrfachen Datenaustausch genutzt werden. Z.B. kann A dort periodisch den aktuellsten Wert einer B interessierenden Größe hinterlegen, und B kann periodisch lesend zugreifen, wobei es nicht notwendig ist, daß B jeden von A hinterlegten Wert zur Kenntnis nimmt, wohl aber, daß B einen tatsächlich so von A hinterlegten Wert liest. Verfügt der Rechner über mehrere Prozessoren und erstreckt sich V über mehrere Speicherzellen, oder wird die Speicherdarstellung des von A berechneten Werts mit einer Folge von mehreren Maschinenbefehlen erzeugt, so kann es ohne besondere Synchronisationsmaßnahmen vorkommen, daß B eine unzulässige Wertdarstellung liest, weil der ältere Wert zum Teil schon aktualisiert aber der neuere in seiner Speicherdarstellung noch nicht ganz hergestellt ist.

Als Lösungsmöglichkeit bietet sich hierzu eine Synchronisation nach dem Prinzip des gegenseitigen Ausschlusses an. Befehlsfolgen zur Aktualisierung und solchen zum Lesen der geteilten Variable muß für die Dauer der gesamten Ausführung einer Folge die Variable exklusiv zur Verfügung stehen. Die Nebenläufigkeit der Prozesse A und B muß so eingeschränkt werden, daß zu einem Zeitpunkt höchstens eine einzige Folge in Ausführung ist, und daß, sobald die Ausführung einer Folge begonnen hat, diese vollständig zuendegeführt wird, bevor die Ausführung einer anderen startet. Bei

der Betrachtung des Ablaufs beider Prozesse bilden solche Befehlsfolgen atomare, sich gegenseitig ausschließende Bestandteile.

Zur Sicherstellung des gegenseitigen Ausschlusses müssen alle entsprechenden Befehlsfolgen in den beiden Prozessen durch Synchronisationsoperationen geklammert werden. Einer Folge wird die Operation 'Betreten' vorangestellt. Sie blockiert z.B. den aufrufenden Prozeß A, falls der andere Prozeß B bereits eine geklammerte Folge betreten hat, bis B die Folge verlassen hat. Den Abschluß einer Folge bildet die Operation 'Verlassen', die umgekehrt den sogenannten kritischen Abschnitt der Befehlsfolge zum Zugriff auf die geteilte Variable freigibt und gegebenenfalls einen beim Betreten blockierten Prozeß fortsetzt. Die Operation 'Betreten' kann durch die Synchronisationsoperation 'Warten', 'Verlassen' durch 'Fortsetzen' realisiert werden, wenn initial ein zusätzlicher Fortsetzaufruf durchgeführt wird, und diese Operationen unter Pufferung von Deblockierungsanstößen arbeiten.

8.1.2.3.2 Fremdzugreifbarer Speicher

Fremdzugreifbarer Speicher ist ein Spezialfall des geteilten Speichers. Während ein geteilter Speicherbereich im engeren Sinn als besonderer Speicherbereich (z.B. als Betriebssystem-Objekt 'Common Data') durch besondere Installationsmaßnahmen erzeugt wird und keinem der zugreifenden Prozesse enger zugeordnet ist, ist ein fremdzugreifbarer Speicherbereich Teil des einem bestimmten Prozeß privat zugeordneten Arbeitsspeichers, und es bedarf keiner besonderen Maßnahmen zu seiner Erzeugung. Wenn ein Betriebssystem Fremdspeicherzugriffe unterstützt, können in der Regel alle Zellen des privaten Speichers eines Prozesses auch von anderen Prozessen aus gelesen und manipuliert werden.

In Wiederaufnahme des voranstehenden Beispiels könnte die Variable V bei Prozeß A als normale Variable angelegt sein. Kennt Prozeß B die Identifikation von Prozeß A und die Adresse von V im Adreßraum von A, kann er durch Aufruf einer besonderen Betriebssystemoperation 'Fremdadreßraumzugriff' dort zugreifen. Solche Operationen umgehen Speicherschutzmaßnahmen und sind deshalb in der Regel nur privilegierten Prozessen zugänglich. So kann dieser Mechanismus zur Interprozeßkommunikation nur verwendet werden, wenn Systemprozesse mit anderen Systemprozessen oder Anwendungsprozessen kommunizieren. Hier wird er aber gerne eingesetzt, weil keine besonderen Maßnahmen zur ausdrücklichen Erzeugung geteilten Speichers erforderlich sind.

Die Synchronisationsproblematik bleibt allerdings dieselbe wie beim geteilten Speicher, d.h. zugreifende Befehlsfolgen bilden kritische Abschnitte und sind durch gegenseitigen Ausschluß zu schützen. Zur Kommunikation zwischen Anwendungs- und Treiberprozessen wird hierzu oft eine besondere Form eingesetzt, die den Vorteil besitzt, daß innerhalb des Anwendungsprozesses keine Klammerungen durch Synchronisationsoperationen notwendig werden, die Auftragsübergabe.

8.1.2.3.3 Auftragsübergabe

Der Auftragsübergabe-Mechanismus ist - wie erwähnt - üblicherweise ein Spezialfall der Kommunikation über Fremdspeicherzugriffe. Seien A ein Anwendungsprozeß und B ein Treiberprozeß, d.h. ein auftragsbearbeitender Systemprozeß. Die Variable V dient zur Beschreibung eines Auftrags, den A durch B durchführen lassen will, und u.U. zur Aufnahme der Ergebnisse.

A kann die Variable V zunächst unreglementiert schreibend und lesend benutzen. Erreicht A nun eine Stelle, an der er V an B übergeben will, so ruft er die Operation 'Auftragsübergabe' auf, sie wird mit der Adresse von V und einer Identifikation für B parametrisiert. Das Betriebssystem blockiert mit der Ausführung der Operation zuerst den aufrufenden Prozeß A, dann hinterlegt es die Identifikation von A und die Adresse von V in einer B zugeordneten Datenstruktur, so daß B anschließend per Fremdadreßraumzugriffen die Werte von V lesen und u.U. auch verändern kann. Da A blockiert ist, ist der

gegenseitige Ausschluß gewährleistet. A darf aber erst dann fortgesetzt werden, wenn B seine Zugriffe auf V abgeschlossen hat. Hierzu ist in B ein Aufruf der Operation 'Fortsetzen(A)' vorzusehen.

Nachteil dieses Mechanismus ist, daß A für die Dauer der V-bezogenen Tätigkeit von B blockiert ist. Er kann gemildert werden, wenn die Blockierungszeit möglichst kurz gehalten wird. Hierzu kann bei Lesezugriffen V möglichst umgehend kopiert werden (bei langen Wertdarstellungen ist hier aber der Aufwand zur Kopienerstellung zu berücksichtigen), und A kann nach Erstellung der Kopie sofort fortgesetzt werden.

Zur Kopienvermeidung und auch um umgekehrt-gerichtete Datenübergabe von B an A zu ermöglichen, werden oft besondere Konventionen vereinbart. Die Datenstruktur V gliedert sich in Anzeigenbits, Schreibdaten (zur Übergabe von A an B) und Lesedaten (zur Übergabe von B an A). Die Anzeigenbits (wichtig ist das sogenannte Tätigbit) werden von A vor Aufruf der Übergabeoperation gelöscht und die Schreibdaten belegt. Mit der Übergabeoperation wird ein besonderes Anzeigenbit, das Tätigbit gesetzt. Unter der Konvention, daß A nicht verändernd auf die Schreibdaten und Anzeigenbits sowie überhaupt nicht auf die Lesedaten zugreifen darf, solange das Tätigbit gesetzt ist, braucht A nicht blockiert zu werden. Nach Abschluß der Bearbeitung löscht B das Tätigbit und V steht A wieder zur freien Verfügung. Das Tätigbit dient als Sperranzeige für V. A muß dieses Bit testen, um zu erfahren, ob V noch oder nicht mehr für Zugriffe durch B reserviert ist. Schreib- und Lesedaten bzw. Teile davon bzw. die entsprechenden Speicherbereiche werden oft Schreib- und Lesepuffer genannt. Das Tätigbit ist eine sehr kleine, zur Sperranzeige einer größeren verwendete Datenstruktur. Der gegenseitige Ausschluß für Zugriffe auf die Gesamtstruktur kann durch Beachtung der Sperranzeige gewährleistet werden.

Diese Form der Auftragsübergabe wird in sehr vielen Betriebssystemen verwendet. Oft werden - für den einzelnen Aufruf optional durch Parameter wählbar - beide Formen unterstützt. Der Anwendungsprozeß A kann mit dem Aufruf der Übergabeoperation wählen, ob er für die Dauer der Bearbeitung durch den Treiberprozeß B blockiert werden will (Übergabe mit Warten) oder nicht (Übergabe ohne Warten). Im ersten Fall hat das Tätigbit keine Bedeutung. A wählt üblicherweise die zweite Option, wenn er auch ohne Abschluß der Auftragsbearbeitung durch B noch andere sinnvolle Tätigkeiten durch führen kann, im Verlauf derer die geteilte Variable V nicht zugegriffen wird. Oft erreicht A dann eine Stelle, an der er ohne Zugriffe auf V nicht mehr weiterarbeiten kann. Um dann ein aktives Warten von A durch zyklisches Testen des Tätigbits zu vermeiden helfen, wird von manchen Betriebssystemen für diesen Fall eine Operation zum nachgeschobenen Warten unterstützt. Mit ihr kann sich A unter Prozessoraufgabe blockieren, bis die Auftragsbearbeitung von B abgeschlossen wurde.

8.1.2.3.4 Datenaustausch im Rendezvous

Die Operation 'Treffen' zur Synchronisation zweier Prozesse nach dem Rendezvous-Konzept kann zusätzlich um Datenaustauschfähigkeiten erweitert werden:

- Treffen mit Datenübergabe; ein Prozeß A, der an einen anderen Prozeß B Daten übermitteln will, berechnet diese Daten und legt sie in einem privaten Speicherbereich V_A ab. Anschließend ruft er diese Operation auf, sie blockiert ihn, bis die Datenübergabe stattgefunden hat.
- Treffen mit Datenübernahme; der Prozeß B ruft diese Operation auf und parametrisiert sie mit der Adresse eines ihm privat zugeordneten Speicherbereichs V_B zur Aufnahme der Daten. Er wird blockiert, bis die Datenübergabe stattgefunden hat, d.h. die Daten vom Betriebssstem von V_A nach V_B kopiert wurden.

Im Vergleich zu den vorstehend erläuterten Mechanismen handelt es sich bei dem Datenaustausch im Rendezvous schon um einen nutzernäheren und aufwendigeren Mechanismus. Zu seiner Realisierung werden oft besondere Systemprozesse verwendet, und es wird auf die obigen einfacheren Mechanismen zurückgegriffen. So kann z.B. der Aufruf einer 'Treffen'-Operation als Auftragsübergabe an einen Interprozeßkommunikationstreiber erfolgen.

Sollen in ihrer Speicherdarstellung umfangreiche Daten ausgetauscht werden, dann ist der Zeit- und Prozessorbedarf zur Kopienerzeugung zu berücksichtigen. Als vorteilhaft ist aber doch zu werten, daß nur einmal und zwar direkt von der Quelle zum Ziel kopiert wird, und daß deshalb auch keine betriebssysteminternen Puffer zur zwischenzeitlichen Aufnahme der auszutauschenden Daten benötigt werden, wie das bei den nachfolgenden Mechanismen notwendig wird.

8.1.2.3.5 Nachrichtenaustausch

Unter Nachrichtenaustausch soll verstanden werden, wenn das Betriebssystem Operationen zum gepufferten Datenaustausch zwischen Prozessen anbietet. Dies wird oft auch Botschaftensystem genannt. Wichtig sind zwei Operationen:

- Senden ; ein Prozeß A, der an einen anderen Prozeß B Daten übermitteln will, berechnet diese Daten und legt sie in einem privaten Speicherbereich V_A ab. Anschließend ruft er diese Operation auf, sie blockiert ihn nur sehr kurz, nämlich solange, bis die Daten in einen betriebssysteminternen Puffer P kopiert wurden.
- Empfangen; der Prozeß B ruft diese Operation auf und parametrisiert sie mit der Adresse eines ihm privat zugeordneten Speicherbereichs V_B zur Aufnahme der Daten. Er wird blockiert, bis die Datenübergabe stattgefunden hat, d.h. die Daten vom Betriebssstem von P nach V_B kopiert wurden.
- Testen; diese Operation wird nicht immer unterstützt. Sie ist für den Empfangsprozeß B wünschenswert, wenn er nur nachschauen will, ob momentan eine Nachricht für ihn vorliegt, aber im negativen Fall nicht blockiert werden will, weil er noch andere Aufgaben bearbeiten kann.

Die Pufferung besitzt den Vorteil, daß die Prozesse nur noch sehr lose synchronisiert werden. Der Sender kann nach Absetzen seiner Nachricht sofort weiterarbeiten. Ein Empfänger wird ebenfalls nicht blockiert, wenn bereits eine Nachricht zur Auslieferung an ihn ansteht. Benötigt werden jedoch zwei Kopiervorgänge sowie ein zusätzlicher systeminterner Speicherbereich zur zwischenzeitlichen Aufnahme der Daten.

Varianten des Nachrichtenaustauschs beziehen sich auf die Pufferungsdisziplin und die Adressierung.

Pufferspeicher ist in der Regel beschränkt. Ist bei einer Sendeoperation kein Puffer mehr verfügbar, so kann entweder durch Rückstau reagiert werden (nun wird auch der Sender blockiert, bis wieder Pufferplatz verfügbar ist), durch Ablehnung (dem Sender wird die Ablehung des Auftrags angezeigt) oder durch Verlust (eine der gepufferten Nachrichten geht ohne besondere Anzeige verloren, es findet ein Überschreiben statt).

Bei mehreren gepufferten Nachrichten und einem anstehenden Empfangsaufruf ist ferner festzulegen, welche Nachricht als nächstes ausgeliefert wird. Hierzu wird oft das FIFO-Prinzip verwendet. Es kann durch Priorisierung von Nachrichten durchbrochen werden.

Bei mehreren Sendern und Empfängern ist es von Interesse, daß Sende- und Empfangsaufträge adressiert werden können.

Eine Möglichkeit dazu ist, Sendeaufrufe mit der Prozeßkennung des Adressaten zu versehen. Hier wird jedem Prozeß vom Betriebssystem ein Empfangspufferspeicherbereich zugeordnet (Briefkasten) und die Empfangsoperation benötigt keinen Adreßparameter. Sie kann allerdings oft optional doch mit einer Absenderadresse parametrisiert werden. Dann wird die älteste im Briefkasten befindliche Nachricht des spezifizierten Senders gesucht.

Eine größere Unabhängigkeit der Adressierung von der Prozeßkennungsvergabe und eine größere Flexibilität zur Definition spezieller Kommunikationsstrukturen besteht, wenn besondere Interprozeßkommunikationsobjekte wie Ports oder Kanäle vom Betriebssystem unterstützt werden.

Ein Port ist eine mit Namen identifizierbare Kommunikationsendstelle, die einem Prozeß zugeordnet wird. Ein Sender A sendet an einen bei ihm definierten Port P_A. Ein Empfänger B empfängt von

einem bei ihm definierten Port P_B. Vorher wurden die Ports mittels spezieller Systemaufrufe eingerichtet und verkabelt, z.B. durch Verbindung des Ports P_A mit P_B.

Ein Kanal ist ein unabhängig von den Prozessen erzeugbares und durch Namen identifizierbares Objekt. Ein Sender A sendet in einen von ihm per Namen gewählten Kanal K. Ein Empfänger empfängt von einem ebenfalls von ihm per Namen gewählten Kanal K. Mit der Erzeugung eines Kanals durch besondere Systemaufrufe wird in der Regel der für diesen Kanal zur Verfügung stehende Pufferungsspeicher reserviert. Das Konzept findet sich bei UNIX mit den dort sogenannten Pipes und bei VAX/VMS mit den dort sogenannten Mailboxes. Besonderheit bei beiden Betriebssystemen ist, daß Standard-Ein/Ausgabe-Aufrufe (read / write) als Sende- und Empfangsaufrufe verwendet werden und logische Kanalnamen sowohl Interprozeßkommunikationsobjekte als auch logische Geräte bezeichnen können.

8.1.2.3.6 Monitore

Ein Kommunikationsobjekt Kanal kann abstrakt aus der Sicht der Anwendungsprozesse als spezielle geteilte Datenstruktur aufgefaßt werden, die nur über besondere Zugriffsoperationen, nämlich Senden, Empfangen und Testen erreicht werden kann. Die Datenstruktur ist der dem Kanal zugeordnete Pufferspeicher mit Verwaltungsdaten (Belegtinformation für die Pufferspeicherstücke, Liste der Kennungen der am Kanal wartenden Prozesse). Senden beschreibt den Pufferspeicher, Lesen liest ihn und manipuliert Verwaltungsdaten, Testen liest Verwaltungsdaten.

Es ist sinnvoll, Datenstruktur und Zugriffsoperationen als logische Einheit zu sehen, als abstrakte Datenstruktur, und ihre Funktion nur in Termen der Operationen und ihrer Parameter zu erklären. So entspricht ein nach FIFO-Disziplin arbeitender Kanal in seinem Verhalten einem Objekt des abstrakten Datentyps 'Schlange'. Dies trifft aber noch nicht ganz die Funktion des Kanals. Und zwar wurde eine wesentliche Eigenschaft noch nicht berücksichtigt, nämlich daß die Operationen des Kanals von verschiedenen Prozessen aus nebenläufig aufgerufen werden können, und daß im Verlauf der Ausführung einer Operation Prozesse blockiert und deblockiert werden.

Die aus möglichen nebenläufigen Aufrufen resultierende Problematik der Überlappung verschiedener Operationsausführungen mit der möglichen Erzeugung von Inkonsistenzen in der Datenstruktur wird dadurch gelöst, daß man das Prinzip des gegenseitigen Ausschlusses für die gesamte Dauer von Operationsausführungen festschreibt. Einen derartigen Verbund aus Speicher und Operationen nennt man Monitor. Durch die Einkapselung der Daten und die Verlagerung der zum gegenseitigen Ausschluß nötigen Synchronisationsmaßnahmen in die Operationsausführung kann die Programmierung von Anwendungsprozessen sehr problennah gehalten werden, und die maschinen- und betriebssystemspezifische Realisierung der Datenspeicherung und Synchronisation bleibt im Monitor verborgen.

Mit der generellen Festlegung, daß Operationsausführungen insgesamt als kritische Abschnitte zu werten sind, wird es allerdings unmöglich, einen Prozeß im Verlauf einer Operationsausführung ausdrücklich zu blockieren, wie das z.B. bei der Empfangsoperation auf einen leeren Kanal nötig ist. Die Ausführung stoppt, der kritische Abschnitt kann deshalb nicht verlassen werden, ergo werden alle Prozesse, die eine neue Monitoroperation aufrufen wollen, sofort mit dem Aufruf blockiert. Damit kann keine innerhalb des Monitors eingeschlossene Deblockieranweisung mehr ausgeführt werden, und zur Auflösung der Blockierung des Monitors wäre ein besonderer Deblockieraufruf von außen notwendig. Im Sinne einer Einkapselung der Synchronisation ist es aber unbedingt wünschenswert, alle das Kommunikationsobjekt betreffenden Blockier- und Deblockieranweisungen innerhalb der Operationen vorzusehen.

Die Problematik wird dadurch gelöst, daß ein Blockieraufruf innerhalb einer Operationsausführung O implizit das Ende eines kritischen Abschnitts signalisiert und den Monitor zur Bearbeitung eines anderen Operationsaufrufs O' freigibt. Dort kann dann ein entsprechender Deblockieraufruf vorgese-

hen sein, der mit dem Ende der Bearbeitung von O' wirksam wird, und die Bearbeitung von O in einem zweiten, der Wartestelle folgenden kritischen Abschnitt fortsetzt.

Monitore werden zur Zeit von keinen weitverbreiteten Betriebssystemen direkt unterstützt, können aber auf einigen doch durch besondere Programmierkonventionen sehr effizient realisiert werden (z.B. wenn geteilt zugreifbarer Speicher verfügbar ist). Aufgrund ihrer Problemnähe und Modularisierung bietet sich die Verwendung des Konzepts an, wenn die standardmäßig vom Betriebssystem unterstützten Interprozeßkommunikationsfähigkeiten zur Lösung besonderer Aufgabenstellungen nicht geeignet sind.

8.1.2.3.7 Geteilte Puffer

Bei der Erläuterung der vorgenannten Mechanismen wurde immer davon ausgegangen, daß die zu übertragenden Daten in ihrer Darstellung direkt übertragen oder übergeben werden. Beim Nachrichtenaustausch hat dies - bei langen Datensätzen aufwendige - Kopiervorgänge zur Folge. Ferner muß viel Speicher je Kanal zur Pufferung reserviert werden.

Diese Nachteile lassen sich vermindern, wenn die Darstellung der Daten in geteilt zugreifbarem Speicher erfolgt, der direkt von einem Sender beschrieben und direkt von einem Empfänger gelesen werden kann. Um gleichzeitig den durch ein Nachrichtenaustausch-System angebotenen Adressier- und Synchronisationskomfort zu nutzen, verbirgt man den direkten Datenaustausch hinter einem Nachrichtenaustausch. Als Inhalte von Nachrichten treten nun nicht mehr direkt die Daten, sondern Referenzen auf geteilte Speicherbereiche auf (Referenzübergabe per Nachrichtenaustausch).

Ein Sender A kann so für zu sendende Daten zunächst einen geteilt zugreifbaren Speicherbereich als Puffer P von einem zentralen Betriebssystemdienst D reservieren lassen. D reseviert einen Speicherbereich P im beantragten Umfang und teilt A die Adresse des Bereichs in Form eines Deskriptors Pd mit. Jetzt kennt A die Adresse und kann dort die Sendedaten hinterlegen. Da nur A die Adresse kennt, müssen keine besonderen Maßnahmen zum gegenseitigen Ausschluß getroffen werden.

Beim Senden enthält die Nachricht den Deskriptor Pd als Inhalt, und A gibt mit dieser Weitergabe des Deskriptors seine Zugriffsrechte auf den Puffer P per Konvention auf. Empfängt ein Prozeß P die Nachricht, übernimmt er mit dem Deskriptor Pd auch das Zugriffsrecht auf den Puffer P und die Verantwortung für die Freigabe des Puffers nach Gebrauch.

8.1.2.4 Dienstschnittstellenimplementierung

Nachdem nun die wichtigsten Mechanismen zur Kommunikation zwischen auf demselben Rechner angesiedelten Prozessen vorgestellt wurden, soll auf die Anwendung dieser Mechansimen zur Implementierung einer Dienstschnittstelle eingegangen werden. Der dem logischen Austausch von Ereignissen an Dienstzugangspunkten am nächsten liegende Mechanismus ist der Austausch von Nachrichten über Kanäle. Seine Verwendung soll zunächst einleitend besprochen werden, und auf die wichtigsten Besonderheiten von Dienstschnittstellenimplementierungen soll auf der Grundlage dieses Mechanismus eingegangen werden. Den Abschluß bilden Anmerkungen zur Verwendung anderer Mechanismen.

8.1.2.4.1 Zugangspunktimplementierung durch ein Kanalpaar

Bild 8.1 zeigt eine sehr naheliegende Form der Implementierung einer Dienstschnittstelle durch die Verwendung zweier Kanäle je Zugangspunkt. Der Austausch eines Stimulus entspricht der Übertragung einer Nachricht von Prozeß A zum Prozeß B über den Kanal AB, der einer Reaktion der Übertragung einer Nachricht in umgekehrter Richtung über den Kanal BA. Prozeß A implementiere hierbei eine Dienstnehmer-Instanz I_{i+1} und Prozeß B eine Erbringerinstanz I_i. Die Kanäle seien puffernd und

liefern Nachrichten nach dem FIFO-Prinzip aus, so daß in einer Übertragungsrichtung keine Reihenfolgenvertauschungen auftreten können. Das Absenden einer Nachricht entspricht so direkt der Erzeugung eines Schnittstellenereignisses und der Empfang einer Nachricht der Zurkenntnisnahme des Ereignisses durch den anderen Partner.

Für jeden Schnittstellenereignistyp wird ein zugeordneter Nachrichtentyp definiert, dessen Inhaltskomponenten die Werte der mit dem Ereignis zu übergebenden Parameter enthalten. Über den Kanal AB werden also Nachrichten ausgetauscht, deren Typen den Request- und Response-Primitiven des Dienstes entsprechen, über den Kanal BA den Inidcation- und Confirmation-Primitiven entsprechende Nachrichtentypen.

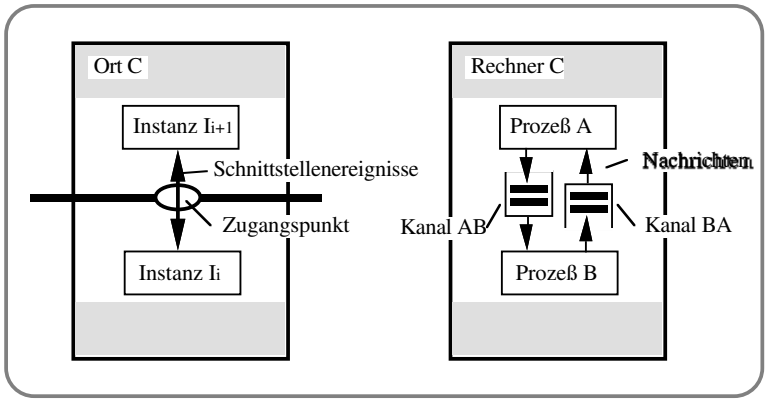

Bild 8.1: Implementierung eines Zugangspunkts als Kanal-Paar

8.1.2.4.2 Lokale Nachrichtenquittungen

Die beschriebene einfache Schnittstellenimplementierung trifft das Konzept der Schnittstellenereignisse im wesentlichen (nebenläufige Erzeuger und Verbraucher, der Erzeuger bestimmt mit dem Ereignis auszutauschende Daten, der Verbraucher nimmt ein Ereignis zusammen mit Datenparametern zur Kenntnis; der Ereignisaustausch findet an einem adressierbaren Ort statt, wobei ein Partner nur die Adresse des Austauschorts, nicht aber die genaue Identität des anderen Partners kennen muß). Aus funktioneller Sicht ist dabei lediglich der Aspekt nicht berücksichtigt, daß z.B. der Partner B ein von A erzeugtes Ereignis aufgrund seines aktuellen Zustands nicht akzeptiert. Im logischen Modell gilt in solchen Fällen das Ereignis als nicht existent.

In der gewählten Implementierung erzeugt A eine entsprechende Stimulus-Nachricht und B empfängt die Nachricht, wird sie aber ignorieren, weil er sie als momentan verbotenen Stimulus ansieht. Da die Nachricht einfach ignoriert wird, bleibt A ohne Kenntnis darüber, ob im logischen Sinn ein entsprechender Stimulus stattgefunden hat oder nicht. Zum Beheben dieses Defekts sollten - mindestens für Stimuli - zusätzliche Nachrichtentypen zur lokalen Quittierung von Schnittstellenereignissen vorgesehen werden. Der vertikale Nachrichtenaustausch zur Übertragung von Schnittstellenereignissen folgt dann einem einfachen zwischen A und B vereinbarten Protokoll gemäß dem jede ein Primitiv übertragende Nachricht vom Empfänger durch eine zugeordnete Quittungsnachricht in Gegenrichtung zu bestätigen oder abzulehnen ist. Zur gegenseitigen Zuordnung sind Primitiv-Nachrichten vom Absender zusätzlich mit einer Identifikation zu versehen, die in der Quittung dann ebenfalls genannt wird.

Solche Quittungsnachrichten erhöhen den beim Betrieb einer Schnittstelle entstehenden Aufwand. Oft werden sie deshalb durch billigere Fremdadreßraumzugriffe ersetzt. Der Absender hinterlegt in einer Primitiv-Nachricht die Adresse von Anzeigenbits, die der Empfänger entsprechend Annahme oder Ablehnung setzt und bei Ablehnung u.U. durch den Code eines Ablehnungsgrunds ergänzt.

8.1.2.4.3 Verbindungsendpunkte

Bei verbindungsorientierten Diensten, wenn z.B. Prozeß A über denselben Zugangspunkt mehrere verschiedene Verbindungen gleichzeitig unterhält, kann es für die Programmierung von A sehr störend sein, daß verschiedene Verbindungen betreffende Ereignisse in dieselben Kanäle eingereiht werden. Eine mögliche - aber u.U. sehr aufwendige - Lösung ist es, je Verbindungsendpunkt mit dem Aufbau einer Verbindung zusätzlich zwei Kanäle zur Darstellung des Endpunkts einzurichten. Eine andere Lösung ist es, die Empfangsoperation so zu erweitern, daß sie selektierend die nächste einen bestimmten Endpunkt betreffende Nachricht gezielt auswählen kann.

8.1.2.4.4 Ausbleibende Partnerreaktion

Eine Dienstnehmerinstanz A wird nach geglückter Anforderung einer Dienstleistung oft untätig auf eine Reaktion des Dienstes (z.B. Eintreffen des Confirmation-Ereignisses) warten. Hierzu kann sie die bei leerem Kanal zur Blockierung führende Empfangsoperation sehr gut verwenden. Allerdings kann bei bestimmten Diensten - aufgrund von Störungen - die Reaktion ausbleiben oder erst so spät eintreffen, daß A in einer Ausnahmebehandlung vorher andere Aktionen hätte ausführen sollen. Hier ist die unbedingte Blockierung bis zum Eintreffen einer Nachricht nicht verwendbar und ein Ausweg wäre das prozessorbeanspruchende aktive Warten durch zyklisches Testen des Kanals BA. Für solche Fälle ist eine Erweiterung der Empfangsoperation interessant, die es erlaubt beim Aufruf eine maximale Wartezeitdauer anzugeben, so daß der Aufrufer auch bei ausbleibender Reaktion nach dem Ablauf der Wartezeit fortgesetzt wird.

8.1.2.4.5 Lange Nachrichten

Die Nachrichten zur Darstellung eines Schnittstellenereignisses können, je nach zulässiger Maximallänge für mit dem Ereignis auszutauschende Nutzdaten, schon im Normalfall mehrere hundert Bytes lang sein. Es entsteht bei direkter Übergabe ein nicht zu vernachlässigender Kopier- und Pufferaufwand, so daß die Variante der Referenzübergabe unter Verwendung geteilt zugreifbarer Puffer berücksichtigt werden sollte.

8.1.2.4.6 Sehr lange Nachrichten

Insbesondere bei anwendungsorientierten Diensten können Nutzdaten, die mit einem einzigen Schnittstellenereignis übertragen werden, noch wesentlich länger sein (z.B. mehrere MByte), so daß eine unfragmentierte Übergabe einer entsprechenden einzigen Nachricht aus technischen Gründen weder wünschenswert noch machbar ist.

Eine einfache mögliche Lösung ist es zwar, peripherspeicher-basierte Puffer und Referenzübergabe zu verwenden. Allerdings ist dazu der bei Peripherspeichern wesentlich erhöhte Schreib- und Leseaufwand zu berücksichtigen.

Für die schnellere Übergabe im Arbeitsspeicher bleibt nur der Weg über den Austausch von Segmenten. Ein einziges Schnittstellenereignis wird durch eine Folge mehrerer Nachrichten dargestellt, wobei jede Nachricht eine kleinere und noch im Arbeitsspeicher gut handhabbare Portion der Schnittstellendaten per Wert oder auch per Referenz überträgt.

8.1.2.4.7 Rendezvous-Austausch

Die Verwendung von Treffpunkten anstelle von Kanälen zur Implementierung von Dienstschnittstellen ist möglich, aber wird aufgrund der engen Synchronisation zwischen Dienstnehmer- und Erbringer-Prozeß seltener vorgesehen. Ein Dienstzugangspunkt kann dabei durch ein Paar von Treffpunkten dargestellt werden. Im ersten Ansatz entspricht ein Schnittstellenereignis einem einzigen Austausch von Daten, und es sind ebenfalls die obengenannten Besonderheiten von Dienstschnittstellen zur geeigneten Aufweitung des einfachen Konzepts zu berücksichtigen.

Zusätzlich ist die wesentlich engere Synchronisation der Prozesse zu berücksichtigen. Es werden hier auch Sender blockiert, wenn der Partner momentan nicht zur Annahme von Daten bereit ist. Im Programm der Prozesse ist deshalb dafür Sorge zu tragen, daß an allen vom Prozeß betroffenen Treffpunkten in genügend kurzen Zeitabständen Annahme-Bereitschaft besteht, um den zügigen Ablauf des Gesamtsystems zu gewährleisten. Hier verlagert sich dann die bei Kanälen im Rahmen der Interprozeßkommunikation abgewickelte Pufferung von Daten in die einzelnen Empfängerprozesse.

8.1.2.4.8 Spezielle Monitore

Bei der Erläuterung der Schnittstellenimplementierung per Kanälen oder Treffpunkten wurde auf verschiedene Besonderheiten von Kommunikationsdienst-Schnittstellen eingegangen, und es wurde auf geeignete Erweiterungen und Anpassungen der Kanaloperationen 'Senden' und 'Empfangen' hingewiesen. Kanäle in derartiger Ausprägung werden von Standard-Mehrprozeßbetriebssystemen in der Regel nicht angeboten, so daß zu ihrer Implementierung zusätzliche Maßnahmen getroffen werden müssen. Ein vom Standpunkt des strukturierten Entwurfs empfehlenswerter Weg dazu ist es, die gewünschte Funktionalität in einem eigens dafür konzipierten Monitor einzukapseln.

8.1.3 Prozeßinterne Schnittstelle

Oft werden aus Effizienzgründen - oder weil das Betriebssystem des Rechners nur sehr beschränkt mehrprozeßfähig ist - die Implementierungen mehrerer Instanzen in demselben Prozeß zusammengefaßt. Dann können die innenliegenden Dienstschnittstellen ohne Verwendung aufwendiger Interprozeßkommunikationsmechanismen implementiert werden, Synchronisationsmaßnahmen mit den enthaltenen Prozeßumschaltungen entfallen.

Die Auslegung prozeßinterner Schnittstellen hängt natürlich ebenfalls stark von dem zur Implementierung der Instanzen verfolgten Konzept ab. Die folgenden Abschnitte stellen die wichtigsten Schnittstellenkonzepte ohne nähere Berücksichtigung der Umsetzung eines Protokolls in ein Programm vor. Grob können dazu unterschieden werden:

- ereignispuffernde Schnittstelle,
- aktionsorientierte Schnittstelle.

8.1.3.1 Ereignispuffernde Schnittstelle

Zentrales Programmobjekt einer ereignispuffernden Schnittstelle ist eine Datenstruktur, die - ähnlich den Kanälen einer Prozeßgrenzen überschreitenden Schnittstelle - anhängige Schnittstellenereignisse in Form von Substrukturen speichert. Da die Struktur nur innerhalb desselben Prozesses zugegriffen wird, kann sie im privaten Speicher des Prozesses gehalten werden und es sind keine Synchronisationsmaßnahmen erforderlich.

Die Datenstruktur kann dem Konzept abstrakter Datentypen direkt entsprechend entworfen werden und entspricht in ihrer Grundfunktionalität einer FIFO-Schlange für Ereignisbeschreibungen. Zur

Beschreibung eines Ereignisses kann dabei ein Verbund aus Ereignistypcodierung, Verbindungsendpunktkennung, Adreßparametern, sonstigen Kontrollparametern und Nutzdaten bzw. Nutzdatenverweis verwendet werden. Die Zugriffsoperationen sind:

- Initialisieren; Erzeugung der initialen Datenstruktur, in der keine Schnittstellenereignisse gespeichert sind.
- Hinzufügen; Hinterlegen der Beschreibung eines neu erzeugten Ereignisses.
- Lesen; Lesen der Beschreibung des ältesten gespeicherten Ereignisses.
- Löschen; Löschen der Beschreibung des ältesten gespeicherten Ereignisses.
- Testen; Abfrage, ob momentan überhaupt ein Ereignis gespeichert ist.

Für die genauere Auslegung der Struktur soll auf die vorangegangenen Ausführungen zur Kanalverwendung verwiesen werden. Insbesondere stellt sich auch prozeßintern die Problematik, lange und sehr lange Nutzdaten-Parameter von Ereignissen zu handhaben.

Im Gegensatz zur Interprozeßkommunikationslösung, bei der i.a. zur Vereinfachung der Synchronisation für jede der beiden Ereignisrichtungen an einem Zugangspunkt getrennte Kanäle vorgesehen werden, ist prozeßintern auch die Variante üblich, alle Ereignisse - auch unterschiedlicher Richtungen und unterschiedlicher Zugangspunkte - in derselben Schlange zu hinterlegen, weil hier die Synchronisationsproblematik entfällt. Sind innerhalb desselben Prozesses mehrere Dienstschnittstellen zu implementieren, weil die Instanzen von mehr als zwei Schichten im selben Prozeß dargestellt werden, wird sogar oft auch für unterschiedliche Dienstschnittstellen nur eine einzige gemeinsame Datenstruktur verwendet. Zur Verbesserung des Ablaufverhaltens der Instanzenimplementierung und zur Vermeidung von Speicherplatzengpässen kann es dabei notwendig werden, eine Ereignisbeschreibung zusätzlich um eine Prioritätsangabe zu ergänzen, und das Einreihen der Ereignise in die Schlange prioritätsgesteuert stattfinden zu lassen, um bestimmte Ereignisse vorrangig bearbeiten zu können. Auch nach Zugangspunkt, Verbindungsendpunkt, Richtung und Ereignistyp selektierende Lese- und Löschzugriffe werden dazu oft vorgesehen.

Bei einer ereignispuffernden Schnittstelle sollte im Sinne einer klaren und änderungsfreundlichen Programmstruktur - soweit es Effizienzüberlegungen zulassen - die Implementierung der Datenstruktur vor der Zugriffsumgebung verborgen werden, und die Zugriffsoperationen werden demnach als Prozeduren im Programm ausdrücklich vorgesehen.

8.1.3.2 Aktionsorientierte Schnittstelle

Während bei der ereignispuffernden Schnittstelle die Erzeugung eines Schnittstellenereignisses auf das Einspeichern einer Ereignisbeschreibung in einer Datenstruktur zur irgendwann nachfolgenden Bearbeitung abgebildet wird, will die aktionsorientierte Schnittstelle das speicheraufwendige Puffern vermeiden. Grundkonzept ist hier, daß die Erzeugung eines Ereignisses möglichst unmittelbar auf seine Bearbeitung abgebildet wird. Eine Anweisungsfolge, die ein Ereignis an einer Stelle erzeugt, enthält bei ereignispuffernder Schnittstelle einen Aufruf der Prozedur 'Hinzufügen'. Bei aktionsorientierter Schnittstelle enthält die Anweisungsfolge an dieser Stelle aus Strukturierungsgesichtspunkten zwar in der Regel ebenfalls einen Prozeduraufruf, diesmal jedoch den einer Prozedur zur direkten Bearbeitung des erzeugten Ereignisses. Ein Ereignis, das an einer Schnittstelle auftritt, löst also möglichst unmittelbar den Aktivitätsstrom (Activity Thread) aus, der zu seiner vollständigen Bearbeitung erforderlich ist.

Im ersten Ansatz werden für jeden Ereignistyp eine eigene Prozedur vereinbart, deren Eingabeparameter das erzeugte Ereignis darüberhinaus näher kennzeichnen (Adressen, Endpunktidentifikation, Kontrollparameter, Nutzdaten bzw. Nutzdatenverweis). Zur lokalen Quittung über die Annahme oder Ablehnung des Ereignisses kann ein Ausgabeparameter vorgesehen werden. Der Prozedurrumpf

enthält die Anweisungsfolge, die zur protokollgemäßen Bearbeitung vorgesehen ist, also bereits direkt ein Stück Protokollimplementierung.

Zur Handhabung langer und sehr langer Nutzdaten sind ebenfalls besondere Maßnahmen vorzusehen (Verweisübergabe, portionsweise Übergabe anhand zusätzlich eingeführter Ereignistypen).

Die aktionsorientierte Schnittstelle kann sehr effizient implementiert werden und zieht auch oft eine sehr effiziente Implementierung betroffener Protokollinstanzen nach sich. Viele ereignisbeschreibende Daten müssen nicht über eine längere Zeitdauer in der Halde sondern können nur kurzfristig im Laufzeitkeller gehalten werden. Eine ausdrückliche Selektion des als nächstes abzuarbeitenden Ereignisses entfällt, da alle anhängigen Ereignisse zusammen mit der Reihenfolge ihrer Bearbeitung sich direkt in der Folge der Prozedur-Schachteln im Laufzeitkeller niederschlagen.

Trotz dieser Vorteile wird diese Form der Schnittstelle nur sehr selten verwendet. Da ihre Verwendung einen sehr sorgfältigen Entwurf der Instanzenimplementierung voraussetzt. Ein wesentliches Problem ist hier, daß eine ereigniserzeugende Anweisungsfolge in ihrem weiteren Fortschritt bis zur vollständigen Bearbeitung des erzeugten Ereignisses blockiert wird. Wenn nach dem logischen Konzept z.B. derselbe Aufrufer mehrere Stimuli konkurrierend absetzen soll, ist bei der Implementierung deshalb genau zu prüfen, in welcher Reihenfolge die Ereignisse erzeugt werden, und ob bei der Ereignisbearbeitung gegenseitige Abhängigkeiten bestehen.

Weiterhin deckt sich die logische Auffassung über die Aktionsfolge zur vollständigen Bearbeitung eines Ereignisses oft nicht mit dem innerhalb eines Prozedurrumpfes tatsächlich programmierbaren Umfang. Hierdurch werden logisch zusammengehörige Aktionsfolgen im Programm auf verschiedene Prozedurrümpfe verteilt, und zwar oft so, daß Teile verschiedener logischer Aktionsfolgen sich im selben Prozedurrumpf mischen.

Dies soll an folgendem einfachen Beispiel verdeutlicht werden. Eine Instanz A der Schicht i könne einen Stimulus S für eine Instanz B der Schicht i-1 erzeugen. Die Bearbeitung von S durch B würde in logischer Sicht zunächst im Absetzen eines Stimulus S' an eine Instanz C der Schicht i-2 und anschließend im Erwarten einer Reaktion R' von C bestehen. Je nach Ausprägung der Reaktion R' würde der Stimulus S' für C wiederholt, oder aber die Bearbeitung des Stimulus S beendet.

Der Rumpf einer Prozedur PS zur Erzeugung und integrierten Bearbeitung von S kann nur den Beginn dieser Aktionsfolge enthalten. Im Rumpf von PS wird der Stimulus S' durch Aufruf der Prozedur PS' erzeugt und anschließend muß der Prozedurrumpf von PS bereits enden, da das Erwarten eines Ereignisses im Schnittstellenkonzept nicht vorgesehen ist. Die nachfolgenden Aktionen (Testen der Ausprägung von R', gegf. Wiederholung der Erzeugung von S') müssen im Rumpf der bei der Erzeugung von R' durch C aufgerufenen Prozedur PR' vorgesehen werden.

Ein weiterer, die klare Strukturierung einer Protokollimplementierung erschwerender Nachteil ist ebenfalls an diesem Beispiel ersichtlich. Die Prozeduren, die die Schnittstelle zwischen zwei Instanzen darstellen, im Beispiel wären das PS' und PR' für die Schnittstelle zwischen B und C, werden aus zwei Richtungen aufgerufen. PS', als Stimulusprozedur, wird von oben, d.h. von B, aufgerufen; PR', als Reaktionsprozedur, von unten, nämlich von C. Weiterhin wird in diesem Fall sogar PS' nicht nur von oben sondern auch noch aus dem Rumpf von PR' heraus, also mittelbar ebenfalls von unten, aufgerufen.

8.1.3.3 Mischformen

Die Abwägung der Vor- und Nachteile der ereignispuffernden und der aktionsorientierten Schnittstelle ergibt oft die Wahl einer Mischform. Verbreitet ist es dabei, Ereignisse einer Richtung einheitlich nach einem Konzept abzubilden, und es soll gewährleistet werden, daß die schnittstellenbildenden

Prozeduren einer Schnittstelle zwischen einer Instanz A der Schicht i und einer Instanz B der Schicht i-1 immer nur in einer Richtung aufgerufen werden:

- Downcall-Schnittstelle; hier besteht die Schnittstelle nur aus Prozeduren, die von der logisch übergeordneten Instanz A aus aufgerufen werden.
- Upcall-Schnittstelle; hier besteht die Schnittstelle nur aus Prozeduren, die von der logisch untergeordneten Instanz B aus aufgerufen werden.

8.1.3.3.1 Downcall-Schnittstelle

Die Downcall-Schnittstelle sieht vor, daß Stimuli aktionsorientiert und Reaktionen ereignispuffernd behandelt werden. Die der übergeordneten Instanz A zum Aufruf verfügbaren Schnittstellenprozeduren sind Bearbeitungsprozeduren für die absetzbaren Stimuli und Ereignistest- bzw. Abholprozeduren für die vorgesehenen Reaktionen. Diese Form wird oft bei anwendungsorientierten Diensten verwendet. Bei Zusammenfassung aller anwendungsorientierten Instanzen mit der eigentlichen Nutzerinstanz in demselben Prozeß entspricht so das Hauptprogramm, also die Spitze der Aufrufhierarchie, der eigentlichen Nutzerinstanz.

8.1.3.3.2 Upcall-Schnittstelle

Die Upcall-Schnittstelle sieht umgekehrt vor, daß Stimuli ereignispuffernd und Reaktionen aktionsorientiert behandelt werden. Die hier nun der untergeordneten Instanz B verfügbaren Schnittstellenprozeduren bearbeiten abgesetzte Reaktionen direkt und erfragen Stimuli. Diese Form wird oft bei sehr hardwarenaher Implementierung transportorientierter und insbesondere mediennaher Schichten eingesetzt. An einer E/A-Anschlußstelle auftretende Reaktionen hardwaremäßig implementierter Telekommunikationsdienste sowie durch einen Hardware-Zeitgeber ausgelöste Wecksignale bewirken Unterbrechungen, die hardwaremäßig und sehr schnell zur Ausführung einer Unterbrechungsroutine führen. Hier bilden dann Unterbrechungsroutinen die Spitze der Aufrufhierarchie.

8.2 Implementierung von Instanzen

Die Aufgabe der Implementierung einer Schicht eines Telekommunikationssystems wurde bereits in die Umsetzung von Dienstschnittstellen, d.h. Dienstzugangspunkten, und die Realisierung von Instanzen gegliedert, und verschiedene Prinzipien zur softwaretechnischen Abbildung von Dienstschnittstellen wurden vorgestellt. Im folgenden liegt der Schwerpunkt auf der Realisierung von Instanzen.

In der logischen Architektur ist eine Instanz eine aktive Einheit, die im Verlauf der Zeit autonom eine ihr zugeordnete und in sich geschlossene Verhaltensvorschrift verfolgt. Entsprechende Objekte in einem Digitalrechner sind Programmausführungen, also Prozesse im Sinn eines Betriebssystems, und das Programm korrespondiert zur Verhaltensvorschrift.

Im unmittelbar folgenden Abschnitt sollen zunächst die einzelnen zur Realisierung einer Instanz notwendigen Funktionen mit hierfür geeigneten Programmstrukturen vorgestellt werden. Danach wird die Verteilung der Funktionen von Instanzen auf Prozesse besprochen und abschließend wird auf Programmiersprachen und Programmiersysteme für die Protokollimplementierung eingegangen.

8.2.1 Instanzen

Eine Instanz kann - wie schon gezeigt - durch einen erweiterten Mealy-Automaten dargestellt werden. Die Erweiterungen sind dabei die indeterministische Auswahl von Zustandsübergängen, die Integration spontaner Übergänge, die durch Datenvariablen induzierte Feinstruktur des Zustandsraums und die durch Parametrisierung der Ein- und Ausgaben mit Datenparametern vorgenommene verfeinerte Gliederung der Ein- und Ausgabealphabete. Der Zustandsraum wird durch die im Zustandsdiagramm als Knoten notierten Hauptzustände grob und durch die zusätzliche Berücksichtigung der aktuellen Wertbelegung von Datenvariablen fein strukturiert. Ein Ein- oder Ausgabealphabet sieht grob für jeden Schnittstellenereignistyp ein zugeordnetes Zeichen vor und detailliert sich in die einzelnen möglichen Kombinationen aus Grundtyp und Datenparameterwertbelegungen. Wenn eine Instanz sich in einem Zustand befindet, von dem ein Spontanübergang ausgeht, kann sie, ohne daß von außen eine Beeinflussung erfolgt ist, den spontanen Übergang durchführen. Hierdurch wird es einer Instanz erlaubt, einen nächsten Schritt optional auszuführen. Mehrere verschiedene Übergänge können vom selben Ausgangszustand aus und abhängig vom selben Eingabezeichen vorgesehen sein, um darzustellen, daß die Instanz in bestimmten Situationen die Freiheit besitzt, zwischen verschiedenen erlaubten nächsten Schritten auszuwählen.

Diese Modellsicht einer Instanz liegt den meisten als Implementierungsvorgabe verwendeten Protokollspezifikationen zugrunde. Im folgenden wird deshalb von der derartigen Konkretisierung des Problems der Umsetzung einer Instanz in ein Programm ausgegangen, daß ein Programm zur Simulation des zeitlichen Ablaufs des Automatenmodells der Instanz zu entwerfen ist. Hierzu sind folgende Teilfunktionen erforderlich:

- **Aktivierung**; das Eintreffen eines Schnittstellenereignisses entspricht dem Auftreten der nächsten Eingabe. Es muß die Programmausführung aktivieren, damit das Ereignis weiterbearbeitet werden kann. Treffen über einen Zeitraum hinweg keine Schnittstellenereignisse ein, dann sollte die Programmausführung dagegen - um Prozessorzeit zu sparen - blockiert sein. Spontane Übergänge werden durch Ereignisse aktiviert, die im logischen Modell nicht ausdrücklich vorgesehen sind und die z.B. Besonderheiten der Implementierungsumgebung signalisieren.
- **Übernahme**; die Aktivierung hat das Auftreten eines als nächstes zu bearbeitenden aktuellen Eingabeereignisses signalisiert. Die genaue Ausprägungsform des Ereignisses ist in einer Speicherdarstellung hinterlegt. Diese Speicherdarstellung muß dem Instanzenprogramm nun durch die Übernahme zugänglich werden. Weiterhin ist das Ereignis mit dem Zeitpunkt der Übernahme als für zukünftige Instanzenschritte nicht mehr existent zu markieren, um zu verhindern, daß dasselbe Ereignis mehrfach Aktivitäten auslöst.
- **Analyse**; das aktuell eingetroffene Ereignis - die aktuelle Eingabe - muß in seiner genauen Ausprägung zur Kenntnis genommen werden. Dazu muß die Speicherdarstellung des Ereignisses analysiert werden.
- **Selektion**; die genaue Ausprägung des aktuellen Eingabeereignisses bestimmt in Kombination mit dem aktuell vorliegenden Zustand der Instanz die genaue Ausprägung des im nächsten Schritt durchzuführenden Zustandsübergangs. Der folgende Zustandsübergang ist vorgaben-entsprechend auszuwählen. Sind ausgehend von derselben Kombination aus Eingabe und aktuellem Zustand mehrere verschiedene Transitionen unter indeterministischer Auswahl möglich, soll nicht 'gewürfelt' werden, sondern die Auswahl ist unter Implementierungsaspekten zu treffen, d.h. die aktuelle Betriebsmittel- und Auslastungssituation sollen in die Auswahl zur Steigerung der Leistungsfähigkeit des implementierten Gesamtsystems eingehen.
- **Transition**; die Durchführung des gewählten Zustandsübergangs besteht darin, daß Ausgaben erzeugt werden und der Folgezustand eingenommen wird:
  - **Konstruktion**; das Erzeugen einer Ausgabe, eines Schnittstellenereignisses, setzt zunächst die Berechnung und Speicherung einer die Ereignisausprägung codierenden Speicherdarstellung voraus.

- **Auslieferung**; die Speicherdarstellung eines Ereignisses wird an der Schnittstelle zu einer anderen Instanz des Systems hinterlegt, um von ihr übernommen werden zu können. Auch die Aktivierung der Zielinstanz muß sichergestellt sein.
- **Zustandsänderung**; die in der Implementierung gewählte Darstellung des aktuellen Instanzenzustands muß so modifiziert werden, daß sie dem Folgezustand der Transition entspricht.

Um dem logischen Modell eines autonomen Automaten zu entsprechen, muß die vorgenannte Aktionsfolge zyklisch wiederholend durchführbar sein. Längerfristige Blockierungen dieses Ablaufs dürfen nur dadurch bedingt auftreten, daß längerfristig keine Eingabeereignisse zu Bearbeitung anstehen. Die Bearbeitung eines Ereignisses muß in befristeter und als kurz zu bezeichnender Zeit abgewickelt sein. Kurz ist dabei relativ zu werten zu den Zeitkonstanten der gemäß Protokoll vorgesehenen Zeitkonstanten, zu den Erwartungen der Zieldienstnehmer über die Dauer der Abwicklung von Dienstleistungen und zu Anforderungen des Basisdienstes über die fristengerechte Reaktion seiner Dienstnehmer auf Anzeigen. Aufgrund dieser Anforderung wird in der Regel bei der Implementierung von Instanzen auch auf die dynamische Allokierung und Freigabe exklusiv benötigter Betriebsmittel (z.B. Arbeitsspeicher) verzichtet, um unvorhersehbare Verzögerungen zu vermeiden. Favorisiert wird die statische Vorab-Reservierung eines am Maximalbedarf orientierten festen Satzes von Betriebsmitteln, wobei der echte Maximalbedarf anhand der Flußkontrollparameter des Protokolls abgeschätzt werden kann, oder die Überschreitung eines angenommenen Maximalbedarfs durch Rückstau-Maßnahmen verhindert wird.

Folgende Unterprobleme können aus der obigen Aufteilung isoliert werden:

- Schnittstellenimplementierung; sie liegt den Funktionen der Aktivierung, Übernahme und Auslieferung zugrunde und hat in der Regel Querbeziehungen zur den Funktionen der Analyse und Konstruktion.
- Zustandsdarstellung; sie liegt den Funktionen der Selektion und der Zustandsänderung zugrunde. Ferner beeinflußt sie u.U. auch die Funktionen der Analyse, wenn die Darstellung des aktuellen Eingabeereignisses in die Zustandsdarstellung integriert wird.
- PDU-Analyse und Aufbau; wichtige Komponente einer Ereignisdarstellung ist bei Ereignissen der Basisdienstschnittstelle das Nutzdatum. Es entspricht einer PDU und folgt in seiner Syntax den Protokollfestlegungen. Die Analyse von PDUs ist folglich wichtige Basisfunktion der Eingabeereignis-Analyse, die Konstruktion von PDUs ist wichtige Basisfunktion der Konstruktion der Speicherdarstellung von Ausgabeereignissen.
- Aktivierung; die Synchronisation einer Instanz mit ihrer Ablaufumgebung ist ein Kernproblem, wenn in einer mehrschichtigen Architektur, bei der sich im selben Rechnerknoten mehrere Instanzen (u.U. auch mit sonstigen dort laufenden Prozessen) Prozessoren teilen, insgesamt eine gute Leistungsfähigkeit erreicht werden soll.
- Selektion; die Auswertung von Ereignis- und Zustandsinformation zur Auswahl der nächsten Transition kann - bei einer sehr großen Zahl unterschiedlicher Kombinationen - zu Laufzeit- und Speicherproblemen führen (Berechnungszeit und Codespeicher).

Sie sollen - mit Ausnahme der schon abgehandelten Schnittstellen-Realisierung - im folgenden in dieser Reihenfolge in eigenen Abschnitten behandelt werden.

Anschließend soll noch auf die Realisierung der telekommunikationsprotokoll-typischen Zeitüberwachungsmaßnahmen eingegangen werden, bevor abschließend auf häufig zu findende Programm-Architekturen zur Instanzen-Implementierung insgesamt eingegangen wird.

8.2.1.1 Zustandsdarstellung

Die Sicht des erweiterten Mealy-Automaten gliedert den Zustandsraum in eine Hauptzustandskomponente (die Zustandsknoten im Diagramm) und in weitere, die Nebenzustandskomponenten, die im Diagramm als Datenvariablen zu finden sind.

Die Instanz S aus Kap. 4.2 hat z.B. vier Hauptzustände 'SuR', 'Q0E', 'RZ' und 'Q1E' und die Nebenzustandskomponenten 'n' und 'm' vom Datentyp Oktettfolge (die noch vorkommende Variable b kann vernachlässigt werden, weil sie nur als Platzhalter für beliebige ankommende Sequenzzahlen dient). Der Zustandsraum der Instanz ergibt sich somit als Menge aller Kombinationen aus {SuR, Q0E, RZ, Q1E} × 'Wertbereich des Typs Oktettfolge' × 'Wertebereich des Typs Oktettfolge'.

Es ist anzumerken, daß die Aufteilung eines Zustandsraums in eine Haupt- und unterschiedliche Nebenzustandskomponenten willkürlich ist. Es lassen sich z.B. auch ein erweiterter Automat mit nur einem einzigen Hauptzustand oder (bei endlichen Variablenwertebereichen) einer ohne jegliche Nebenzustandskomponenten definieren, die denselben Ablauffestlegungen wie Bild 3.8 entsprechen.

Wir gehen im folgenden davon aus, daß der Zustandsraum in der Vorgaben-Spezifikation problembezogen sinnvoll gegliedert wurde. Die Hauptzustände fassen Feinzustände mit im groben vergleichbarem Folgeverhalten zusammen, so daß auch in der Implementierung eine separate Aufschlüsselung nach Hauptzuständen beibehalten werden soll, um zur Lesbarkeit des Programms beizutragen und nur in Details voneinander abweichende Aktionen in gemeinsamen Anweisungsfolgen abzuwickeln. Die Nebenzustände fächern den Zustandsraum sehr fein auf, sie sind in der Spezifikation bereits durch Datenvariablen dargestellt und diese Darstellung soll auch in das Instanzenprogramm übernommen werden, weil nur so das Auffächerungsschema ersichtlich bleibt.

Dessen ungeachtet kann es sehr oft ratsam sein, die Vorgaben-Spezifikation vor dem Programmentwurf in einen äquivalenten Automaten mit einer näher an den Implementierungsrandbedingungen ausgerichteten Struktur des Zustandsraums zu überführen.

Für die Verwaltung der aktuellen Nebenzustandskomponenten werden aufgrund der hohen Anzahl möglicher Werte grundsätzlich Datenstrukturen eingesetzt. Zur Darstellung des - vergleichsweise wenig voneinander zu unterscheidende Werte vorsehenden und bereits eine Grobselektion der Transitionen unterstützenden - aktuellen Hauptzustands ist auch die Kopplung mit dem aktuellen Befehlszählerstand bei der Programmausführung möglich.

8.2.1.1.1 Hauptzustand

Bei einer Instanz treten üblicherweise nur eine geringe, überschaubare Anzahl in der Größenordnung von zehn bis wenigen hundert unterschiedlicher Hauptzustände auf.

8.2.1.1.1.1 Datenstruktur-Darstellung

Zur Darstellung des aktuellen Hauptzustands bei Programmablauf kann problemlos eine eigens dafür vorgesehene Variable verwendet werden, deren Datentyp einem geeigneten Aufzählungsdatentyp entspricht. Ein Wechsel des aktuellen Hauptzustands erfolgt durch Zuweisung eines neuen Werts an die Variable.

8.2.1.1.1.2 Befehlszähler-Darstellung

Bei der Darstellung des aktuellen Hauptzustands per Befehlszähler verzichtet man auf eine ausdrücklich zur Führung des Hauptzustands vorgesehene Variable. Der aktuelle Hauptzustand der Instanz entspricht vielmehr dem, daß die Programmausführung momentan nur bestimmte, dem vorliegenden Hauptzustand zugeordnete Bereiche des Programms erfaßt. Ein Wechsel des aktuellen Hauptzustands erfolgt durch Verzweigung (Sprung).

Für jeden möglichen Hauptzustand kann so im Hauptprogramm der Instanz eine Sprungmarke vorgesehen werden. Jede an einer solchen Marke beginnende Anweisungsfolge übernimmt die gesamte Funktionaufeinanderfolge 'Aktivierung - Übernahme - Selektion - Transition', wobei jede Transition

mit dem Wechsel des Hauptzustands, nämlich einem Sprung auf die Marke des nächsten Hauptzustands, endet.

Nachteil dieser Technik ist, daß für jeden Hauptzustand die gesamte Funktionsfolge separat zu programmieren ist (dies kann durch Verwendung von Prozeduren gemildert werden). Vorteil ist, daß die Teilfunktionen Aktivierung, Übernahme, Selektion und Transition speziell auf diejenigen Eingabeereignisse und diejenigen Zustandsübergänge zugeschnitten werden können, die vom betroffenen Hauptzustand aus vorgesehen sind. Dies kann zur Verminderung der Programmausführungszeit beitragen. Auch können dadurch Aktivierungsprobleme u.U. leichter lösbar sein.

Ein Beispiel dazu wäre die Instanz S aus Bild 3.8 unter der Annahme, daß sie in einem eigenen Prozeß realisiert ist, und daß die beiden Zugangspunkte A_O und A_U je durch ein eigenes Paar von Interprozeßkanälen implementiert sind. In den Zuständen 'SuR' und 'RZ' kann zur Aktivierung gezielt ein Empfangsaufruf auf den A_O-Eingabe-Kanal, in den Zuständen 'Q0E' und 'Q1E' kann gezielt ein Empfangsaufruf auf den A_U-Eingabekanal zur Instanz abgesetzt werden. Bei Verschmelzung beider Aktivierungsstellen zu einem Programmstück müßten entweder beide Kanäle zyklisch getestet werden, oder das Betriebssystem bzw. die Hilfsfunktionen zur Realisierung von Kanälen müßten das Mehrfachwarten unterstützen.

8.2.1.1.2 Nebenzustände

Die Darstellung der Nebenzustände erfolgt generell durch Datenstrukturen, die vorwiegend in permanenten Variablen auf Hauptprogrammebene gehalten werden. Die für diese Variablen eingeführten Datentypen sollten problemnahe Strukturierungsmöglichkeiten der Programmiersprache (Verbunde, Felder) nutzen, um die Lesbarkeit und die Inbezugsetzung zur Vorgabe-Spezifikation zu fördern. Komponenten des Zustandsraums, deren Wertbelegung nur temporär relevant ist, können im Laufzeitkeller gehalten werden (Prozedurparameter, prozedur-interne Variablen).

Der Entwurf der Datenstrukturen muß speziell im Hinblick auf die zu implementierende Protokollinstanz und die dort benötigten Auslese- und Veränderungsoperationen erfolgen. Er kann allgemein nicht näher eingegrenzt werden. Im folgenden werden deshalb nur noch einzelne Komponenten angesprochen, die bei gängigen Protokollen benötigt werden.

8.2.1.1.2.1 Aktuelles Eingabe-Ereignis

Die Übernahme eines aktuellen Eingabeereignisses führt zur Hinterlegung einer Beschreibung der Ausprägung des Ereignisses. Auf sie greift die anschließende Selektion zu. Sehr oft ergeben sich weitere Zugriffe auch aus den späteren Funktionen bis hin zur Konstruktion und Auslieferung von Ausgabeereignissen, wenn - aus Effizienzgründen, um das Umkopieren von Daten zu vermeiden - die Darstellung des aktuellen Ereignisses implizit diese späteren Funktionen parametrisiert.

Die verwendete Datenstruktur entspricht in ihrer Zusammensetzung oft der folgenden von Problemseite her naheliegenden Form. Sie ist ein Verbund aus:

- Ereignistypkennung; z.B. UDtIndBA oder ODtReq, die anschließenden Komponenten sind in Datentyp und Anzahl abhängig vom Ereignistyp.
- Kontrollparameter des Dienstschnittstellenereignisses; wie sie im Konzept des betreffenden Ziel- oder Basisdienstes vorgesehen sind, unter Vereinbarung geeigneter Datentypen für die verschiedenen Parameterarten (z.B. Basisdienstadresse, Zieldienstadresse, Verbindungsendpunktkennung, Bitfehlerwahrscheinlichkeit).
- Nutzdaten; bei einem an der Zieldienstschnittstelle aufgetretenen Ereignis bleiben die Nutzdaten insgesamt uninterpretiiert. Bei einem an der Basisdienstschnittstelle aufgetretenen Ereignis ist das Nutzdatum eine angekommene PDU und muß weiter (in einer PDU-Substruktur) erschlossen werden:

- PDU-Typkennung; sie kennzeichnet das Grundformat der PDU, die anschließenden Komponenten der Substruktur sind in Datentyp und Anzahl abhängig vom PDU-Typ,
- Kontrollparameter der PDU,
- Nutzdaten der PDU.

Als Datentyp für eine Ereignisbeschreibung kann ein nach Ereignistyp varianter Verbundtyp gewählt werden. Eine andere Lösung ist es, einen festen Verbund aus allen Komponenten aller Typen vorzusehen, der typ-entsprechend lückenhaft belegt wird. Das gleiche gilt für die PDU-Substruktur.

Für Testzwecke und zur Implementierungsunterstützung können noch weitere Komponenten vorgesehen werden (Prozeß- und Kanalkennungen, Kennungen von Rückkanälen etc.).

Der Aufwand der Analyse-Funktion kann dadurch vermindert werden, daß die Darstellung erzeugter Ereignisse identisch ist mit der Beschreibung aktueller Eingabeereignisse. Die Konstruktion in einer anderen Instanz nimmt dann die Analyse in der Zielinstanz vorweg. In der Zielinstanz fällt dann nur noch, und zwar auch nur bei vom Basisdienst erzeugten Ereignissen, die PDU-Analyse an.

8.2.1.1.2.2 Dienstleistungszusammenhang

Üblicherweise nicht aus dem Hauptzustand hervorgehende Angaben über den Zusammenhang des aktuellen Zustands zur aktuellen Dienstleistungserbringung betreffen die Adressen des Initiators und des Responders der aktuell abgewickelten Dienstleistung. Weitere solche Parameter können Zeitangaben und geforderte Qualitäten sein.

8.2.1.1.2.3 Verbindungszusammenhang

In höheren transportorientierten Schichten, die einen verbindungsorientierten Dienst erbringen, steht jede Dienstleistungsabwicklung im Zusammenhang mit einer Zieldienstverbindung und es können gleichzeitig mehrere Verbindungen unterstützt werden. Aus Zieldienstzugangspunktadresse und Verbindungsendpunktkennung (bei Eingabeereignissen der Zieldienstschnittstelle) bzw. aus Basisdienstzugangspunktadresse und Verbindungsendpunktkennung (bei Eingabeereignissen der Basisdienstschnittstelle) muß die betroffene Zieldienstverbindung identifizierbar sein. Hierzu dient eine Tabelle aller momentan unterstützten Verbindungen, die Verbindungstabelle.

Das Automatenmodell ist in solchen Fällen nicht mehr direktes Modell einer Instanz sondern liefert nur das Muster für die Unterstützung einer einzigen Verbindung, wobei von der realisierten Instanz jedoch die nebenläufige Unterstützung mehrerer Zieldienstverbindungen erwartet wird. Die Instanz ist also eine Automatenmenge, wobei jeder Automat einer aktuellen bestehenden Zieldienstverbindung zugeordnet ist.

Die Verbindungstabelle besitzt für jede Verbindung einen Eintrag, der neben der Selektion der von einem aktuellen Ereignis betroffenen Verbindung auch die Speicherung des aktuellen, diese Verbindung betreffenden Instanzenzustands gewährleistet. Neben den lokalen Adressen und Endpunktkennungen werden hier auch die Partneradressen und zusätzlich alle (außer in manchen Fällen vielleicht der Eingabeereignisbeschreibung) unter den vorangegangenen Punkten zur Zustandsdarstellung aufgeführten Angaben als Substrukturen gespeichert.

8.2.1.1.2.4 Nutzdaten-Verwaltung

Zur Behandlung der Nutzdaten von Schnittstellenereignissen oder PDUs soll auf die Ausführungen zu geteilten Puffern im Abschnitt 'Dienstschnittstellenimplementierung' verwiesen werden. Da die Nutzdaten oft eine größere Länge in der Darstellung besitzen ist es sehr effizienzsteigernd, das Kopieren durch Verweisübergabe und Verwendung geteilter Puffer zu vermeiden.

Bei Berücksichtigung des Ein- und Auspackens von Nutzdaten in PDUs durch Protokollinstanzen steht man vor dem Problem, daß ein Datenblock nicht insgesamt unverändert weitergegeben wird, sondern daß - beim Auspacken - z.B. nur ein Mittelstück des Blocks relevant bleibt oder - beim Einpacken - ein gegebener Block das Mittelstück eines neuen bildet. Um hier das Umkopieren zu vermeiden werden anspruchsvollere Speicherabbildungen für Puffer (gestückelte Speicherung und Verkettung) erforderlich. Inwieweit der erhöhte Verwaltungs- und Zugriffsaufwand sich durch Kopieraufwandverminderung lohnt, muß im Einzelfall eines Telekommunikationssystems unter Berücksichtigung des zu erwartenden Nutzungsprofils entschieden werden.

8.2.1.1.2.5 Verwaltung des Basisdienstes

Die ausdrückliche Verwaltung der Zugangmöglichkeiten einer Instanz zum Basisdienst ist oft nicht erforderlich, weil diese mit dem Systementwurf statisch feststehen. Hier genügen oft statisch belegte oder per Administrationseingriffen nur selten veränderte Tabellen zur schnellen Realisierung von Adreßabbildungsfunktionen.

Die dynamische Zuordnung aktuell genutzter Basisdienstzugangspunkte, Endpunktkennungen und Verbindungen zu Zieldienstverbindungen kann in den meisten Fällen innerhalb der Verbindungstabelle erfolgen.

Eigene ausdrücklich zur Verwaltung der Basisdienst-Nutzung eingeführte Datenstrukturen betreffen ihre verbindungsübergreifende Nutzung. Zu nennen sind Wegetabellen bei Instanzen mit Weiterleitungsfunktionen. Ferner empfiehlt sich bei Instanzen mit Multiplexfunktionen eine separate Verwaltung der aktuell bestehenden Basisdienstverbindungen, ihrer Qualitäten und Auslastung.

8.2.1.2 PDU-Analyse

Die PDU-Analyse untersucht ein per Schnittstellenereignis des Basisdienstes eingetroffenes Nutzdatum auf die korrekte Wertdarstellung eines erlaubten PDU-Typs hin und liefert in der PDU-Substruktur in der Beschreibung des aktuellen Ereignisses die zur Auswertung durch die Instanz aufgeschlüsselten Darstellungen der PDU-Bestandteile.

Sieht ein Protokoll Verfälschungssicherung per fehlererkennendem oder korigierendem Code vor, dann wird die Prüfung des Kanalcodes und Ermittlung des zugeordneten Quellencodes in einer anfänglich aufgerufenen Unterfunktion vorgenommen.

Die nachfolgende, eigentliche Analyse wird mit dem Typ des Schnittstellenereignisses parametrisiert, das die PDU angeliefert hat, und kann so berücksichtigen, daß bei einem gegebenen Protokoll bestimmte PDU-Typen nur im Zusammenhang mit bestimmten Ereignistypen geliefert werden dürfen.

8.2.1.2.1 Paketweise Bearbeitung

Im Sinne einer klaren Strukturierung von Programmablauf und Funktionsdurchführung ist es sinnvoll, eine PDU insgesamt durch eine in sich geschlossene Funktion (z.B. Prozedur 'Analysiere PDU') vollständig analysieren zu lassen (sie kann natürlich bei Entdeckung eines Syntaxfehlers sofort abbrechen). Dies wird in der Regel auch dann vorgesehen, wenn die Analyse nicht laufzeitintensiv und die PDU-Substruktur nicht speicherintensiv ist.

8.2.1.2.2 Segmentweise Bearbeitung

Vor allem wenn eine vollständige PDU-Beschreibung sehr speicherintensiv wäre, wie dies bei Protokollen der Anwendungsschicht vorherrschen kann, empfiehlt es sich die PDU-Analyse in mehreren

Phasen durchzuführen. In der ersten Phase wäre dann neben einem - nicht alle Fehler erkennenden Syntaxtest - nur soweit zu analysieren, daß die anschließende Selektion stattfinden kann. Im Verlauf der Durchführung der Transition können dann bedarfsweise weitere Analysephasen zwischengeschaltet werden. Eine solche Phasengliederung der Analyse kann durch geeignete Formatfestlegung der PDUs wesentlich unterstützt werden.

Bei sehr langen (z.B. längenmäßig nicht beschränkten) PDUs, wie sie ebenfalls bei manchen Anwendungsprotokollen vorgesehen sind, sei ferner auf die Ausführungen zu sehr langen Nachrichten im Abschnitt 'Dienstschnittstellenimplementierung' verwiesen. Soll Peripherspeichereinsatz vermieden werden, muß segmentweise Übergabe per längeren Schnittstellenereignisfolgen unter Einführung zusätzlicher Schnittstellenereignistypen zur Auslieferung von Folgesegmenten vorgesehen werden.

Eine vollständige PDU-Beschreibung ist hier aus Speichergründen nicht möglich, so daß jeweils nur ausschnitthafte Beschreibungen und Informationen zur Fortsetzung des Analysevorgangs gehalten werden.

8.2.1.3 PDU-Aufbau

Die Konstruktion einer PDU ist das Pendant zur Analyse und ihr gegenüber dadurch vereinfacht, daß davon ausgegangen werden kann, daß nur zulässige PDUs zur Konstruktion in Auftrag gegeben werden. Mit der Beauftragung steht fest, welchen Typs die aufzubauende PDU ist. Man benötigt also z.B. nicht eine einzige Prozedur, die alle vorgesehenen PDUs aufbaut, sondern kann für jeden PDU-Typ eine eigene Aufbau-Prozedur einführen.

8.2.1.3.1 Paketweise Bearbeitung

Auch bei der PDU-Konstruktion wird man, wenn es die PDU-Länge zuläßt, einen Block insgesamt in einer einzigen Funktion aufbauen. Die Konstruktionsprozedur kann dazu entweder explizit mit allen erforderlichen Parametern versehen aufgerufen werden, oder sie liest die zur Parameterbestimmung erforderlichen Angaben aus der Zustandsdarstellung. Letzteres vermeidet Umkopiervorgänge im Laufzeitkeller.

8.2.1.3.2 Segmentweise Bearbeitung

Die segmentweise Konstruktion wird beim Erzeuger sehr langer PDUs erforderlich. Bei Anstoß der Bearbeitung eines Folgesegments muß auch hier gewährleistet sein, daß der entsprechenden Konstruktionsprozedur Fortsetzinformation zur Verfügung steht.

8.2.1.4 Aktivierung

Der eingesetzte Mechanismus zur Aktivierung einer Instanz mit dem Auftreten eines Eingabeereignisses ist abhängig von der gewählten Form der Schnittstellenimplementierung. Grundsätzlich bestehen zwei Möglichkeiten:

- das Programm der Instanz wird in einem eigenständigen Prozeß ausgeführt,
- das Programm der Instanz wird durch Prozeduraufruf von einem anderen Prozeß her aktiviert.

8.2.1.4.1 Erwarten von Ereignissen

Im Fall der Programmausführung durch einen eigenständigen Prozeß enthält das Programm Wartestellen, an denen (z.B. durch eine Empfangsoperation auf einen Interprozeßkommunikationskanal) das Eintreffen des nächsten Ereignisses erwartet wird. Um Prozessorzeit zu sparen, sollte dem War-

ten unter Prozessoraufgabe (Blockieren) der Vorzug vor dem aktiven Warten per Schleife gegeben werden.

Besondere Maßnahmen müssen getroffen werden, wenn die Quelle des nächsten Ereignisses nicht vorhersehbar ist, sondern z.B. ein beliebiger aus einer Menge von Interprozeßkommunikationskanälen das nächste Ereignis abgeben kann. Hier stellt sich das Problem des Oder-Mehrfachwartens. Idealerweise sollte ein Operationsaufruf verfügbar sein, der mit der Kanalmenge adressiert wird, und den Prozeß solange blockiert, bis in einem beliebigen der Kanäle das nächste Ereignis eintrifft.

Liegen in mehreren Kanälen bereits Ereignisse vor, wählt eine solche Mehrfach-Empfangsoperation oft ein beliebiges Ereignis aus. Das kann unerwünscht sein, wenn z.B. im Sinne einer Implementierungsstrategie zur Betriebsmittel-Schonung bestimmte Ereignisquellen bevorzugt behandelt werden sollen. Zum Beispiel ist es üblich, Ereignisse, die die Bereitschaft eines Partners zur Übernahme von Datenpaketen signalisieren, bevorzugt vor solchen zu behandeln, die der Instanz neue zusätzlich zu verwaltende Datenpakete anliefern. Kann in diesen Fällen nicht auf einen komfortablen Interprozeßkommunikationsdienst zurückgegriffen werden, empfiehlt es sich, vor dem Mehrfachempfangsaufruf zunächst die Kanäle ohne Warten zu testen.

Besteht keine Möglichkeit zum Mehrfachwarten unter Prozessoraufgabe per Mehrfachempfangsoperation, muß auf das aktive Warten zurückgegriffen werden. Die einzelnen Eingabe-Kanäle werden zyklisch ohne Warten getestet, bis ein nichtleerer Kanal gefunden ist. Um dieses unproduktive Schleifen im Prozessorbedarf herabzusetzen, können am Ende eines Testzyklus Zeitdauer-Wartestellen vorgesehen werden, die den Prozeß per Betriebssystemoperation für ein bestimmtes Zeitintervall blockieren. Die Länge des Intervalls sollte groß sein, um den Prozessor zu schonen, sie sollte klein sein, um die Bearbeitung zwischenzeitlich eingetroffener Ereignisse nicht zu lange hinauszuzögern, was Auswirkungen auf die Wahl der Zeitkonstanten von Protokoll-Zeitüberwachungsmaßnahmen haben kann.

8.2.1.4.2 Ausführungsstart durch Ereignisse

Bei der unter 'Aktionsorientierte Schnittstelle' besprochenen Schnittstellenumsetzungsform aktiviert bereits die Erzeugung eines Ereignisses die Bearbeitung durch den Empfänger, weil dessen Aktionen in die Erzeugungsprozedur integriert sind. Übernahme-, Selektions- und Transitionsaktivitäten des Empfängers können in einer solchen Prozedur in einer speziell auf den Typ des durch die Prozedur erzeugten Ereignisses zugeschnittenen Weise programmiert sein.

Das Programm einer Instanz besteht aus einer Menge von Prozeduren und den Datenstrukturen zur Darstellung des Instanzenzustands, die i.a. von allen Prozeduren aus zugegriffen werden.

Problematisch ist, daß wenn die Instanzen-Prozeduren von verschiedenen Prozessen (Ereigniserzeugern) aus aufgerufen werden, daß die Zustandsdarstellung dann den Charakter von geteilten Daten besitzt. Das Programm der Instanz kann gleichzeitig mehrfach aktiviert sein, nebenläufige Zugriffe und daraus resultierende Konflikte müssen i.a. durch gegenseitigen Ausschluß aufgelöst werden, was bei Berücksichtigung der zentralen Bedeutung der Zustandsdarstellung zu einem erheblichen Prozessoraufwand führen kann.

8.2.1.5 Selektion

Die Selektion der als nächstes auszuführenden Transition kann - wie bereits erwähnt - für eine Instanz an einer zentralen Programmstelle vorgesehen sein und dort alle Ereignis-Zustand-Kombinationen berücksichtigen. Bei Hauptzustandsdarstellung durch den Befehlszähler oder bei aktionsorientierter Schnittstelle kann die Selektionsfunktion aufgeteilt werden in einzelne, bei den einzelnen Programmstellen oder Prozeduren aufgerufene Teilfunktionen, wobei jede Teilfunktion nur eine spezielle Untermenge der insgesamt möglichen Ereignis-Zustand-Kombinationen berücksichtigen muß.

Unbeachtet dessen, ob eine Teilselektion oder eine Selektion über alle Kombinationen zu realisieren ist, können die beiden folgenden grundlegenden Mechanismen eingesetzt werden.

8.2.1.5.1 Programmierte Selektion

Bei der programmierten Selektion wird die Transitionsauswahl (z.B. in Form geschachtelter IF-THEN-ELSE-Kaskaden oder geschachtelter CASE-Anweisungen) ausprogrammiert. Aufgrund des für den Auswahl-Code benötigten Speicheraufwands, und weil sich leicht sehr unübersichtliche Programme ergeben, wird die programmierte Selektion in der Regel nur für Teilselektionen, insbesondere für die abschließende Teilselektion bei mehrphasiger Durchführung, eingesetzt.

8.2.1.5.2 Tabellengesteuerte Selektion

Die tabellengesteuerte Selektion basiert auf der Verwendung einer mit dem Programmentwurf statisch festgelegten Transitionstabelle. Sie hat die Struktur einer zweidimensionalen Matrix. Elemente sind die Beschreibungen einzelner Transitionen (z.B. in Form der Adresse der entsprechenden Transitionsprozedur oder in Form von Steuerparametern für eine allgemeine Transitionsprozedur). Indizes der einen Dimension sind die Hauptzustände, die u.U. auch nach wichtigen Nebenzustandsklassen weiter aufgefächert sind. Indizes der anderen Dimension sind die Eingabeereigistypen, die u.U. ebenfalls nach wichtigen Ereigniskontrollparameter-Klassen weiter aufgegliedert sind.

Die Selektion findet sehr laufzeiteffizient durch Einstich in die Matrix statt, und der Speicheraufwand zur Darstellung der Matrix ist in der Regel im Vergleich zur programmierten Selektion ebenfalls wesentlich vermindert. Deshalb wird die tabellengesteuerte Selektion vorzugsweise verwendet.

Varianten können die mehrstufige Auswahl über verschiedene Tabellen vorsehen. Bei Teilselektion kann jeder Funktion eine eigene Tabelle zugeordnet werden.

8.2.1.6 Zeitüberwachung

Typisch für ein Telekommunikationsprotokoll ist es, daß in bestimmten aktuellen Zuständen Transitionen nicht nur durch echte Schnittstellenereignisse sondern auch durch den Ablauf von Zeitfristen aktiviert werden können.

Im Automatenmodell einer Instanz wird dies üblicherweise dadurch beschrieben, daß Ereignisse mit Weckern ausgetauscht werden:

- Weckerstart-Ereignisse werden von der Instanz zu Beginn einer Zeitfrist erzeugt. Ein solches Ereignis beauftragt einen Wecker, nach Ablauf der Frist ein Fristablauf-Ereignis an die Instanz zu senden.
- Stornierungsereignisse werden von der Instanz nach einem Weckerstart-Ereignis erzeugt, wenn sich zwischenzeitlich die Notwendigkeit zur Fristüberwachung erübrigt hat (z.B. Quittung rechtzeitig eingetroffen).
- Ablauf-Ereignisse werden von einem Wecker erzeugt, wenn ein bisher nicht stornierter Weckauftrag vorliegt und dessen Frist abgelaufen ist.

Das Ausgabealphabet des Automaten wird also um die Start- und Stornierungsereignisse für die verwendeten Wecker erweitert, das Eingabealphabet um die Ablauf-Ereignisse. Man geht der Einfachheit halber davon aus, daß bei demselben Wecker immer nur höchstens ein Auftrag aktiv sein kann und verwendet bei Bedarf demgemäß mehrere Wecker.

Diese Beschreibung von Zeitüberwachungsmaßnahmen im Automatenmodell impliziert, daß jeder Wecker eine eigene lokal angesiedelte Instanz des Systems ist, mit der die Protokollinstanz zusätzlich zum Ereignisaustausch mit lokaler Dienstnehmer- und lokaler Basisdienst-Instanz ebenfalls Ereignisse austauscht. Damit können Zeitereignisse von Seiten der Protokollinstanzenimplementierung behandelt werden wie sonstige Schnittstellenereignisse.

Eine besondere Problematik bildet allerdings die Implementierung der Wecker. Sie haben zwar eine relativ einfache Funktionalität, aber bei einer - eine Vielzahl von Verbindungen unterstützenden - Protokollinstanz kann es schon vorkommen, daß mehrere hundert Wecker in Gebrauch sind. Hier z.B. die Wecker als eigenständige Prozesse mit gesonderten Interprozeßkommunikationskanälen zu implementieren führt zur Überlastung der meisten Mehrprozeßbetriebssysteme.

8.2.1.6.1 Zeitaufträge über Prozeßgrenzen

Sollen die Zeitereignisse über eine Prozeßgrenze ausgetauscht werden, ist es sinnvoll, alle Wecker in einen einzigen Zeitauftragsbearbeiter-Prozeß zu integrieren. Dem Prozeß wird ein Eingabe-Kanal zugeordnet, in den alle lokalen Protokollinstanzen Start- und Stornierungsereignisse hinterlegen können. Der Prozeß bearbeitet die Ereignisse durch Aufnahme in seine privat geführte Auftragsverwaltung. Ferner überwacht er zyklisch den Auftragsbestand unter Berücksichtigung der aktuellen Uhrzeit und erzeugt beim Überschreiten einer Frist ein entsprechendes Ablaufereignis, das in einen Eingabe-Kanal der Auftraggeberinstanz abgelegt wird.

Die Nachricht zur Beschreibung eines Startereignisses enthält den Kanalnamen des gewünschten Rückmeldekanals, die Fristdauer und eine von der Protokollinstanz gewählte Auftragskennung. Die Stornierungsnachricht enthält Rückmeldekanal-Namen und Auftragskennung zur Identifikation des zu stornierenden Auftrags. Die Ablauf-Nachricht enthält die Auftragskennung, damit die Protokollinstanz den zugeordneten Start-Auftrag identifizieren kann.

Manche Betriebssysteme bieten bereits standardmäßig einen vergleichbaren Zeitauftragsdienst an, andere unterstützen nur bestimmte zur Realisierung des Weckerprozesses benötigte Grundfunktionen:

- lesender Zugriff auf eine Echtzeituhr,
- Blockieren des aufrufenden Prozesses um eine angegebene Zeitspanne,
- Zyklisches Aktivieren eines Prozesses entsprechend einem Taktraster.

8.2.1.6.2 Prozeßinterne Zeitaufträge

Soll der Aufwand zur Überwindung von Prozeßgrenzen beim Austausch von Zeitereignissen vermieden werden, so ist ein entsprechender Zeitauftragsdienst innerhalb des Instanzenprozesses zu realisieren. Die Schnittstelle dazu können drei Prozeduren Start, Stornierung und Abfrage bilden. Start und Stornierung führen zur Aktualisierung einer Auftragsdatenstruktur. Von besonderer Problematik ist die Abfrage, sie hat Aktivierungs- und Übernahmefunktion für die eigentliche Protokollinstanz, und integrierte Blockieraufrufe dürfen nicht verhindern, daß die Instanz durch andere eintreffende Ereignisse aktiviert wird, während umgekehrt auch das Warten auf echte Schnittstellenereignisse nicht die Aktivierung durch Ablauf-Ereignisse verzögern darf.

Ist die Instanz ein eigenständiger Prozeß, so werden die übrigen Eingabeereignisse über Interprozeßkommunikationskanäle angeliefert. Die Aktivierung durch echte Schnittstellenereignisse erfolgt durch den Empfang von Ereignissen an einer Wartestelle. Vorher muß getestet werden, ob ein Zeitauftrag fällig ist, wenn ja wird das Auftreten des Ablaufereignisses angenommen. Falls nein, wartet der Prozeß an den Eingabekanälen und muß aber von dieser Wartestelle auch gelöst werden, wenn ein Zeitauftrag zwischenzeitlich fällig wird. Eine Lösung dazu ist es, vor dem Warteaufruf die Zeitdifferenz zum nächsten Fristablauf zu ermitteln und das Warten mit dieser Zeitspanne als maximale Wartezeit zu verbinden.

Ist die Instanz eine Prozedursammlung, so wird sie nur aktiv, wenn ein echtes Schnittstellenereignis auftritt. Für diese Fälle können zu Beginn jedes Prozedurrumpfs Zeitabfragen und das bedingte Einschieben der Bearbeitung von Fristablaufereignissen vorgesehen werden. Problematisch ist, wenn Zeitaufträge fällig sind, die Instanz aber deshalb nicht aktiv werden kann, weil über eine längere Zeitdauer keine echten Schnittstellenereignisse auftreten. Hierzu kann eine zusätzliche Prozedur vorgesehen werden, deren Rumpf nur das Testen und die bedingte Bearbeitung von Ablaufereignissen vorsieht. Über zusätzliche Maßnahmen ist sicherzustellen, daß diese Prozedur auch in Ruhephasen aufgerufen wird. Dies kann z.B. per Konvention von der Dienstnehmer-Instanz gefordert werden, oder diese Prozedur kann bei hardware-naher Implementierung von der Interruptroutine der Hardware-Uhr des Rechners aus zyklisch aktiviert werden.

8.2.1.7 Grundlegende Software-Architekturen

Die bisher beschriebenen Einzelstrukturen zur Umsetzung von Instanzenfunktionen in Programme öffnen in unterschiedlicher Kombination und zusammen mit den verschiedenen Mechanismen zur Schnittstellenimplementierung ein sehr breites Spektrum an Varianten für die Gesamtarchitektur des Programms einer Instanzenimplementierung. Im folgenden sollen zwei häufiger zu findende Varianten etwas näher umrissen werden, in denen wenige Mechanismen konsequent verfolgt werden.

8.2.1.7.1 Zentralverteiler

Typisch für die Zentralverteiler-Architektur ist, daß die Funktionen der Aktivierung, Übernahme, Analyse und Selektion in einem zentralen Programmstück erbracht werden, dem Verteiler. Er wird zyklisch durchlaufen und verzweigt am Ende eines Durchlaufs in ein der ausgewählten Transition zugeordnetes Programmstück. Ein besonders typisches Bild ergibt sich, wenn eine einzige Protokollinstanz durch einen exklusiv hierfür vorgesehenen Prozeß realisiert wird, und wenn es nur eine einzige Datenstruktur (Interprozeßkommunikationskanal) mit der Disziplin einer FIFO-Schlange zur Speicherung ankommender Schnittstellenereignisse gibt. Die Fristablaufmeldungen von Zeitüberwachungsmaßnahmen und Anzeigen über lokale Betriebsmittelbesonderheiten zur Auslösung spontaner Instanzenübergänge sollten im Sinne einer klaren Programmstruktur ebenfalls als Eingabeereignisse in der FIFO-Eingabeschlange hinterlegt werden.

Das folgende Modula-Programm implementiert die Instanz S aus Bild 3.8. Es umfaßt eine - im Normalfall endlos auszuführende - Schleife im Rumpf des Hauptprogramms, die prozeßprivaten Datenstrukturen zur Zustandsspeicherung, eine Tabelle zur Selektion, einen Satz von Transitionsdurchführungsprozeduren und Prozeduren für unterschiedliche Hilfsfunktionen (Konstruktion, Übergabe). Mit jedem Durchlauf der Hauptschleife wird genau ein Eingabeereignis abgearbeitet und genau eine Transition durchgeführt. Die Problematik der Datenübergabe wird in diesem Programm der Einfachheit halber durch Umkopieren gelöst. Auch sonst wurden einige mögliche Optimierungen aus Lesbarkeitsgründen weggelassen, und die ankommenden IPC-Nachrichten werden nicht vollständig auf zulässigen Aufbau hin geprüft.

```
module InstanzS ;
     from Operatingsystem import
          IPCmessagetype,     (* Datentyp einer IPC-Nachricht *)
          (* = record
               Sender : integer ; Sendername
               Data : bytestr ; eigentliche Nachricht
               end *)
          Attach ,            (* Anmelden des Prozesses bei IPC *)
          Send ,              (* Senden in IPC-Kanal:
                                   Send ( Channelno : integer ;
```

```
                                   Msg : bytestr ) *)
     WaitForReceipt ;    (* Warten auf und Lesen der nächsten
                            IPC-Nachricht aus Eingabekanal des
                            Prozesses:
                            WaitForReceipt
                                 (var Msg : IPCmessagetype) *)
const
     ByteStrMaxLng = 777 ;
     UserIn = 111 ;      (* andere Instanzen: Kanalnr. *)
     MediumIn = 222 ;
     UserName = 333 ;    (* Instanzenidentifikationen *)
     MediumName = 444 ;
     SName = 555 ;
type
     bytestr   =    record        (* Nutzdaten-Datentyp *)
                    l : integer ;
                    s : array [0..(ByteStrMaxLng-1)] of char ;
                    end ;
     mainevttypes
               = (  Ignore,            (* Haupteingabeereignistypen *)
                    ODtReq, UDtIndBA) ;
     subevttypes                       (* Untertypen *)
               = (Sequnum0, Sequnum1, Corrupted) ;
     evtdescr =     record        (* Eingabeereignisbeschreibung *)
                    MainType : mainevttypes ;
                    SubType : subevttypes ;
                    Data : bytestr ;
                    end ;
     mainstates = ( SuR, Q0E, RZ, Q1E) ; (* Hauptzustände *)
     transproc = procedure ;
     refproc = pointer to transproc ;
     tabsubevtline = array subevttypes of refproc ;
     tabevtline = array mainevttypes of tabsubevtline ;
     selecttab = array mainstates of tabevtline ;
var
     SelTab : selecttab ;        (* Selektionstabelle *)
     MainState : mainstates ;    (* Hauptzustand *)
     LastPDU : bytestr ;         (* Nebenzustand *)
     ActEvt : evtdescr ;         (* Eingabeereignis *)
     ActMsg : IPCmessagetype ;   (* Ereignis-IPC-Nachricht *)
     Trans : refproc ;           (* Zeiger auf Transitionsprozedur *)

procedure  Pass ;       (* Übernahme von ActMsg nach ActEvt *)
     var I : integer ;
     begin
     if ActMsg.Sender=UserName then
          ActEvt.MainType := ODtReq ;
          ActEvt.SubType := Seqnum0 ;
          ActEvt.Data:=ActMsg.Data ;
     elsif ActMsg.Sender=MediumName then
          ActEvt.MainType := UDtIndBA ;
          if ActMsg.Data.s[0]<>0C then
               ActEvt.SubType:=Corrupted ;
          else                              (* PDU-Analyse *)
               if ActMsg.Data.s[1]=0C then
                    ActEvt.SubType:=Sequnum0 ;
               else
                    ActEvt.SubType:=Sequnum1 ;
```



```
               end ;
          ActEvt.Data.l:=ActMsg.Data.l-2 ;
          for I:=2 to ActMsg.Data.l-2 do
               ActEvt.Data.s[I-2]:=ActMsg.Data.s[I] ;
               end ;
          end ;
     else
          ActEvt.MainType := Ignore ;
          end ;
     end Pass ;

procedure ConstructPDU ( Seq : integer ) ;
     var I : integer ;  (* PDU (Seq,ActEvt.Data) -> LastPDU *)
     begin
     LastPDU.l:=ActEvt.Data.l+1 ;
     LastPDU.s[0]:=chr(Seq) ;
     for I:=0 to ActEvt.Data.l-1 do
          LastPDU.s[I+1]:=ActEvt.Data.s[I] ;
          end ;
     end ConstructPDU ;

procedure PassUserData ( var Usd : bytestr ) ;
     var I : integer ;  (* Nutzdaten aus ActEvt --> Usd *)
     begin
     Usd.l:=ActEvt.Data.l-2 ;
     for I:=2 to ActEvt.Data.l-1 do
          Usd.s[I-2]:=ActEvt.Data.s[I] ; end ;
     end PassUserData ;

procedure Req0 ;   (* Transition: SuR -- ODtReq --> Q0E *)
     begin
     ConstructPDU(0) ;             (* PDU-Konstruktion
                                      und Nebenzustandsmodif. *)
     Send(MediumIn,LastPDU) ;      (* PDU-Übergabe *)
     MainState:=Q0E ;              (* Zustandswechsel *)
     end Req0 ;

procedure Req1 ;   (* Transition: RZ -- ODtReq --> Q1E *)
     begin
     ConstructPDU(1) ;             (* PDU-Konstruktion
                                      und Nebenzustandsmodif. *)
     Send(MediumIn,LastPDU) ;      (* PDU-Übergabe *)
     MainState:=Q1E ;              (* Zustandswechsel *)
     end Req1 ;

procedure Q0n ;    (* Transition: Q0E -- UDtIndBA(1-/01) --> Q0E *)
     begin
     Send(MediumIn,LastPDU) ;      (* PDU-Übergabe *)
     end Q0n ;

procedure Q1n ;    (* Transition: Q1E -- UDtIndBA(1-/01) --> Q1E *)
     begin
     Send(MediumIn,LastPDU) ;      (* PDU-Übergabe *)
     end Q1n ;

procedure Q0p ;    (* Transition: Q0E -- UDtIndBA(00) --> RZ *)
     var Usd : bytestr
     begin
     PassUserData(Usd) ;
     Send(UserIn,Usd) ;            (* ODtCon-Übergabe *)
```

```
     MainState:=RZ ;              (* Zustandswechsel *)
     end Q0p ;


procedure Q1p ;    (* Transition: Q1E -- UDtIndBA(00) --> SuR *)
     var Usd : bytestr ;
     begin
     PassUserData(Usd) ;
     Send(UserIn,Usd) ;           (* ODtCon-Übergabe *)
     MainState:=SuR ;             (* Zustandswechsel *)
     end Q1p ;


begin
Attach ;           (* Initialisierung *)
SelTab :=( ( (nil,nil,nil),(Req0,nil ,nil ),(nil ,nil ,nil ) ),
           ( (nil,nil,nil),(nil ,nil ,nil ),(Q0p ,Q0n ,Q0n ) ),
           ( (nil,nil,nil),(Req1,nil ,nil ),(nil ,nil ,nil ) ),
           ( (nil,nil,nil),(nil ,nil ,nil ),(Q1n ,Q1p ,Q1n ) ) ) ;
MainState := SuR ;
while true do         (* Arbeitsschleife *)
     WaitForReceipt (ActMsg) ;       (* Aktivierung *)
     Pass ;                          (* Übernahme *)
     Trans := SelTab  [MainState,MainType,SubType] ;
     if Trans <> nil then Trans^ end;     (* Transition *)
     end ;
end InstanzS .
```

Großer Vorteil dieser Architektur ist ihre klare Struktur mit deren enger Korrespondenz zum Automatenmodell. Kleinere Protokolländerungen schlagen sich im wesentlichen nur in der Selektionstabelle und einzelnen Transitionsdurchführungsprozeduren nieder. Jeder Transitionstyp kann in sich geschlossen in einer eigenen Prozedur abgehandelt werden, und Querbeziehungen zwischen diesen Transitionsprozeduren bestehen nur über die Datenstruktur zur Zustandsdarstellung.

8.2.1.7.2 Sammlung von Ereignisbearbeitungsprozeduren

Die Implementierung einer Instanz als Prozedursammlung ergibt sich bei aktionsorientierter Schnittstelle in der Ausprägung in Reinform oder in den Mischformen der Down- oder Upcall-Schnittstelle. Die Grundstruktur des Instanzenprogramms wird direkt durch die Schnittstellenform geprägt.

Im folgenden soll als Beispiel ein Programm für die Instanz S aus Bild 3.8 dienen, das auf der Verwendung von Downcall-Schnittstellen aufbaut.

Die A_O-Schnittstelle wird durch eine Menge von Prozeduren zur Erzeugung von Stimuli (hier nur eine Prozedur ODtReq, da nur ein Stimulus-Typ unterstützt wird) und eine einzige Prozedur zum Erwarten und Abholen der nächsten Reaktion WaitForOReact gebildet. Jede Rückkehr von WaitForOReact signalisiert dem Aufrufer eine aufgetretene Reaktion (da nur ein Reaktionentyp vorgesehen ist, ist im Beispiel kein Parameter zur Typ-Anzeige vorgesehen). Das Instanzenprogramm besteht aus diesen beiden Prozeduren (und einigen von dort aus aufgerufenen Hilfsprozeduren), und die Instanz kann nur aktiv sein, wenn der Dienstnehmer-Prozeß eine der beiden Prozeduren aufruft.

Die A_U-Schnittstelle entspricht in Konzept der A_O-Schnittstelle. Auch hier gibt es Prozeduren zur Weitergabe von Stimuli und eine zum Erwarten von Reaktionen.

```
implementation module InstanzS ;
     (* to User export
```



```
     ODtReq ( Usd : bytestr ) ; User erzeugt ODtReq
     WaitForOReact ( var Usd : bytestr ) ; User wartet auf ODtCon *)
from Medium import
     UDtReqAB,
          (* UDtReqAB ( Usd : bytestr ) ; UDtReqAB erzeugen *)
     WaitForUReact ;
          (* WaitForUReact ( var Corrupt : boolean ;
                              var PDU : bytestr ) ;
                        Warten auf Anzeige von Medium *)

const
     ByteStrMaxLng = 777 ;
type
     bytestr   =    record        (* Nutzdaten-Datentyp *)
                    l : integer ;
                    s : array [0..(ByteStrMaxLng-1)] of char ;
                    end ;
     mainstates = ( SuR, Q0E, RZ, Q1E) ; (* Hauptzustände *)
var
     MainState : mainstates ;    (* Hauptzustand *)
     LastPDU : bytestr ;         (* Nebenzustand *)

procedure ConstructPDU ( Seq : integer ; Usd : bytestr ) ;
     var I : integer ;  (* PDU (Seq,Usd) -> LastPDU *)
     begin
     LastPDU.l:=Usd.l+1 ;
     LastPDU.s[0]:=chr(Seq) ;
     for I:=0 to Usd.l-1 do
          LastPDU.s[I+1]:=Usd.s[I] ;
          end ;
     end ConstructPDU ;


procedure PassUserData ( PDU : bytestr ; var Usd : bytestr ) ;
     var I : integer ;  (* Nutzdaten aus PDU --> Usd *)
     begin
     Usd.l:=PDU.l-1 ;
     for I:=1 to PDU.l-1 do
          Usd.s[I-1]:=PDU.s[I] ; end ;
     end PassUserData ;


procedure ODtReq ( Usd : bytestr ) ; (* ODtReq-Erzeugung *)
     begin
     if MainState=SuR then        (* Selektion *)
          ConstructPDU(0,Usd) ;   (* PDU-Konstruktion *)
          UDtReqAB(LastPDU) ;     (* Übergabe *)
          MainState:=Q0E ;        (* Hauptzustandswechsel *)
     elsif MainState=RZ then
          ConstructPDU(1,Usd) ;   (* PDU-Konstruktion *)
          UDtReqAB(LastPDU) ;     (* Übergabe *)
          MainState:=Q0E ;        (* Hauptzustandswechsel *)
     else (* Ereignis ignorieren *)
          end ;
     end ODtReq ;


procedure WaitForOReact ( var Usd : bytestr ) ;
     (* O-Reakt erwarten, es gibt nur den Typ ODtCon *)
     var PDU : bytestr ; Corr : boolean ;
     begin
     if MainState=Q0E then        (* Selektion, vorweg *)
          loop
```

```
            WaitForUReact (Corr,PDU) ;(* Aktivierung, Übernahme *)
            if not Corr
               and (PDU.l>0) then       (* Analyse, Selektion *)
                 if PDU[0]=0C then exit end ;     (* --> Q0p *)
                 end ;
            UDtReqAB(LastPDU) ;    (* Transition: Q0n *)
            end ;
         PassUserData(PDU,Usd) ;      (* Übergabe *)
         MainState:=RZ ;              (* Zustandswechsel *)
    elsif MainState=Q1E then
         loop
            WaitForUReact (Corr,PDU) ;(* Aktivierung, Übernahme *)
            if not Corr
               and (PDU.l>0) then       (* Analyse, Selektion *)
                 if PDU[0]=1C then exit end ;     (* --> Q0p *)
                 end ;
            UDtReqAB(LastPDU) ;    (* Transition: Q1n *)
            end ;
         PassUserData(PDU,Usd) ;      (* Übergabe *)
         MainState:=SuR ;             (* Zustandswechsel *)
    else (* Aufruf ablehnen *)
         Usd.l := -1 ; (* Fehleranzeige *)
         end ;
    end WaitForOReact ;

begin
MainState:=SuR ;        (* Initialisierung *)
end InstanzS .
```

Diese Prozedursammlung-Lösung besticht im Vergleich zur Zentralverteiler-Lösung dadurch, daß weniger permanente Datenstrukturen benötigt werden. Die Eingabeereignis-Warteschlange und der Aufwand für Zugriffe und Verwaltung entfallen, weil erzeugte Ereignisse gleich abgearbeitet werden. Es ergibt sich aber insgesamt eine schlechtere Lesbarkeit. Die Funktionen der Aktivierung und Selektion sind jetzt an mehreren Stellen in Teilfunktionen vorhanden.

8.2.2 Instanzen und Prozesse

In diesem Abschnitt soll der Aspekt der Aufteilung einer Instanz auf Prozesse auch unter Berücksichtigung dessen behandelt werden, daß in einem Rechnerknoten i.a. mehrere übereinanderliegende Instanzen zu implementieren sind. Varianten sind:

- mehrere Prozesse je Instanz,
- ein Prozeß je Instanz,
- mehrere Instanzen im selben Prozeß.

Für eine Aufgliederung in mehrere Prozesse spricht grundsätzlich, daß dadurch eine klare Gliederung der Implementierung in in sich geschlossene autonome Einheiten unterstützt wird. Kann man standardmäßig vom Betriebssystem unterstützte Interprozeßkommunikationsmechanismen für die Schnittstellen verwenden, wird dies zusätzlich durch ein einheitliches Schnittstellenkonzept gefördert und die Installation und Wartung des Systems ist in getrennt handhabbaren Teilen sehr gut möglich.

Dafür, die Prozeßanzahl gering zu halten, spricht, daß viele Betriebssysteme nur eine sehr beschränkte Anzahl von Prozessen unterstützen können. Die Zuteilung des Prozessors an viele Prozesse, die Verwaltung einer großen Anzahl von Prozessen und die Interprozeßkommunikationsmechanismen sind oft prozessorzeit- und speicherintensiv, belasten also den Rechnerknoten in hohem Maße, vor

allem wenn, wie in einer mehrschichtigen Hierachie oft der Fall, viele Instanzenprozesse immer nur sehr kurze Tätigkeitsphasen haben.

8.2.2.1 Prozeßmengen

Die Aufspaltung einer Instanz auf mehrere Prozesse wird oft dann vorgesehen, wenn die Instanz einen verbindungsorientierten Dienst unterstützt, und dabei mehrere Zieldienstverbindungen gleichzeitig bestehen können. Dazu können ein einziger Prozeß zur Verwaltung der Zieldienstzugangspunkte und mehrere Prozesse, nämlich je einer zur Unterstützung genau eines Verbindungsendpunkts, eingeplant werden. Ein Verbindungsendpunktprozeß kann jetzt direkt dem aus der Vorgabe-Spezifikation entnommenen Automaten entsprechen, weil er nur noch eine einzige Verbindung zu unterstützen hat. Die Verbindungsendpunktprozesse können so ausgelegt werden, daß sie sich einen gemeinsamen Code teilen.

Während der Zugangspunkt-Prozeß statisch mit der Einrichtung des Zieldienstzugangspunkts installiert wird, kann bei den Verbindungsendpunktprozessen - außer der Möglichkeit, statisch für die Maximalanzahl von Verbindungen schon vorab alle Prozesse zu installieren - auch die Möglichkeit der dynamischen Erzeugung eines Prozesses mit einer Endpunkt-Einrichtung und dem Vernichten des Prozesses mit dem Abbau der Verbindung gewählt werden.

8.2.2.2 Leichtgewicht-Prozesse

Um die Vorteile der problemnahen Programmstrukturierung in mehrere Prozesse beizubehalten, während die Nachteile des hohen, für die Überwindung von Prozeßgrenzen nötigen Aufwands gemildert werden sollen, wird das Konzept der sogenannten Leichtgewicht-Prozesse verwendet.

Aus der Sicht des Betriebssystems des Rechners werden dabei mehrere Instanzen in einen einzigen Prozeß zusammengefaßt. Dieser Prozeß trägt ein speziell auf die Instanzenimplementierung zugeschnittenes eigenes Mehrprozeßbetriebssystem (L-System), das die sogenannten Leichtgewichtprozesse (L-Prozesse) unterstützt.

Ein L-Prozeß wird als separater Prozeß konzipiert, um z.B. eine Instanz oder Teilfunktionen einer Instanz zu realisieren. Zur Installation wird der Prozeßcode zum Code des L-Systems dazugebunden, und dem L-System wird bei der Initialisierung der Auftrag zur Erzeugung und Unterstützung des L-Prozesses übergeben.

Das L-System unterstützt i.a. folgende Funktionen:

- Geteilte Datenstrukturen zwischen L-Prozessen,
- Verwaltung von geteilten Puffern,
- Prozessorzuteilung an L-Prozesse,
- Interprozeßkommunikation unter L-Prozessen, oft durch spezielle L-Prozeß-Auftragsschlangen,
- Schnittstelle des L-Prozeßsystems nach außen, d.h. zum Standardbetriebssystem hin,
- Zeitauftragsdienstleistungen.

Auf diese Weise wird durch das L-System unter Verdeckung des Standardbetriebssystems eine speziell auf die Erfordernisse der problemnahen Implementierung von Telekommunikationsprotokollen zugeschnittene Umgebung erzeugt, und es ergibt sich neben der Möglichkeit zur aufwandsgeminderten Realisierung von Mehrprozeß-Systemen damit auch die Chance, bei Portierung von Kommunikationssoftware die Programme der L-Prozesse unverändert oder nur wenig verändert zu übernehmen.

8.2.2.3 Integration mehrerer Instanzen

Steht ein Leichtgewichtsbetriebssystem nicht zur Verfügung, weil seine Entwicklungskosten gescheut werden oder die von ihm verursachte Grundbelastung des Rechners als zu hoch angesehen wird, so sollen oft Mehrprozeßlösungen überhaupt vermieden werden. Mehrere Instanzen sollen dann in einen gemeinsamen Prozeß integriert werden.

8.2.2.3.1 Zentralverteiler-Architektur

Als Lösung bietet sich die Zentralverteiler-Architektur an (Server-Architektur). Als Beispiel soll angenommen werden, daß 4 übereinanderliegende Instanzen (ein 'Turm') jeweils separat in Zentralverteiler-Architektur implementiert wurden. Man kann nun die 4 Programme mischen, so daß der Verbund das Programm nur noch eines Prozesses bildet. Dazu werden die 4 Eingabeereignis-Schlangen zu einer Eingabe-Ereignisschlange zusammengefaßt, in der die Eingabeereignisse aller 4 Instanzen (u.U. prioritätsgesteuert) gespeichert werden. Zur Ereignisbeschreibung wird ein gemeinsames Schema verwendet und die 4 Selektionstabellen werden in eine einzige Tabelle integriert. Damit kann die Selektionsfunktion die Selektion aller 4 Instanzen übernehmen. Es wird nur noch eine Arbeitsschleife benötigt. Die übrigen Teile der Instanzenprogramme, d.h. die umfangreichen Transitionsprozeduren und die Datenstrukturen zur Zustandsspeicherung können unverändert übernommen werden.

Hier ergibt sich ein enger Querbezug zum Ansatz der Leichtgewichtprozesse. Ereigniswarteschlange, Arbeitsschleife, Selektionstabelle und Selektionsfunktion sind mit einem direkt auf das spezielle Problem zugeschnittenen Prozeßumschalter vergleichbar, die Ereigniswarteschlange übernimmt ferner Interprozeßkommunikationsaufgaben. Eine Ausführung einer Transitionsprozedur entspricht einem dynamisch bei Bedarf erzeugten Leichtgewichtprozeß.

8.2.2.3.2 Prozedursammlung

Ein spezieller Fall ergibt sich bei einer Software-Architektur der Instanzenprogramme als Prozedursammlungen. Es handelt sich nun um eine Null- bzw. N-Prozeß-Lösung. Die Instanzen sind nicht durch eigenständige Prozesse zur Laufzeit vertreten, sondern werden mit dem 'Leben' der aufrufenden Umgebung, also der Anwendungsprozesse und/oder der Gerätetreiber-Prozesse ausgeführt. Betrachtet man einen 'Turm' aus mehreren Instanzen, so wird z.B. ein Aufruf eines Anwendungsprozesses zu einer sequentiellen, den Turm durchlaufenden Ausführungskette von Prozedurausführungen führen. Die Instanzen entsprechen zur Laufzeit also vielen solcher in sich sequentieller und zueinander nebenläufiger 'Threads', man spricht von einer 'Activity Thread'-Architektur.
Diese Architektur kann sehr effizient sein, da Prozeßwechsel an Instanzengrenzen vermieden werden. Problematisch ist, daß Instanzen-Zustandsdaten i.allg. von allen Threads zugegriffen werden und ihre konsistente Aktualisierung durch geeignete Synchronisation der Threads an den kritischen Abschnitten gewährleistet werden muß. Weiterhin können Schutzaspekte gegen diese Lösung sprechen. Bei Down-Calls werden die Instanzen ohne zusätzliche Maßnahmen unter der Identität von nicht-privilegierten Anwendungsprozessen ausgeführt.

8.2.2.3.3 Kombination Server/Activity-Thread

Zentralverteiler-Architektur und Prozedursammlung lassen sich mit Vorteil kombinieren. Aus dem Zentralverteiler-Ansatz stammt die eine zentrale Warteschlange. Sie wird aber nur für Ereignisse verwendet, die von außen (also von Anwendungsprozessen oder von Gerätetreibern) an den Turm gegeben werden. Das Hauptprogramm arbeitet in der zentralen Schleife diese Warteschlange seriell ab. Mit jeder Ereignisentnahme wird ein Thread durch den Turm zur Ereignisbearbeitung per Aufruf der Prozedur der zunächst betroffenen Instanz gestartet. Diese Prozedur wird, wenn sie ein Turm-internes Ausgabeereignis erzeugt, direkt die Verbraucherprozedur der betroffenen Instanz aufrufen etc.

Dadurch wird die Turm-interne Ereignispufferung vermieden. Auch müssen keine nebenläufigen Threads synchronisiert werden, da Threads durch die Hauptschleife nur seriell entstehen. Es ist allerdings darauf zu achten, daß nur nach-kurzer-Zeit-terminierende Threads auftreten dürfen, wenn die Lebendigkeit der Implementierung nicht gestört werden soll.

8.2.3 Programm- und Programmierumgebung

Abschließend soll kurz angesprochen werden, in welcher Umgebung Instanzen implementiert werden.

8.2.3.1 Systemimplementierungssprachen

Die grundlegende Umgebung eines Instanzenprogramms wird natürlich durch die Hardware der Rechnerknoten-Konfiguration und (wenn vorhanden) durch das Betriebssystem gebildet. Letzeres ist in der Regel bei Knoten, die auch Anwendungen tragen, verfügbar. Es kann bei kleineren reinen Vermittlungsknoten fehlen und fehlt sehr oft, wenn Protokollfunktionen durch Rechner in Minimalkonfiguration abgewickelt werden, die dem eigentlichen Knotenrechner vorlagert und speziell dafür vorgesehen sind (z.B. CPU und Speicher, untergebracht auf Anschaltungsplatine). Die unmittelbare Programmierumgebung dafür wird durch die für den Rechner vorgesehene Systemimplementierungssprache (z.B. Assemblersprache, C) definiert. Die Systemimplementierungssprache enthält Elemente zum direkten Ansprechen der speziellen E/A- und Ablauf- und Speicherorganisations-Befehle des Prozessors (der Prozessoren) und zum Aufruf der Betriebssystemfunktionen.

Bei vielen Implementierungen von Instanzen transportorientierter Schichten wird diese Systemimplementierungsumgebung unmittelbar benutzt, weil der Telekommunikations-Transportdienst als eine allen Anwendungen zur Verfügung zu stellende Betriebssystemerweiterung gewertet wird, und somit direkt in das Betriebssystem integriert werden soll. Dies wird durch Verwendung desselben Programmiersystems erleichert.

8.2.3.2 Höhere Programmiersprachen

Instanzen anwendungsorientierter Schichten sind dagegen in der Regel Teil von Anwendungsprogrammen. Die Programmierumgebung dafür sind die bekannten für Anwendungen verwendeten höheren Programmiersprachen. Das von einem Hersteller für eine solche Sprache gelieferte Programmiersystem beschränkt sich dabei selten auf den Standardsprachumfang, sondern unterstützt zusätzliche Elemente zum direkten Zugriff auf Betriebssystemfunktionen, so daß auch in einer solchen Sprache Interprozeßkommunikationsmechanismen realisiert und verwendet werden können, und somit Mehrprozeßlösungen möglich sind.

Im Bereich der von Postverwaltungen betriebenen Systeme gilt die CCITT-Empfehlung, Programme einheitlich in der speziellen Programmiersprache CHILL (CCITT High Level Implementation Language) zu halten. Sie ist im Sprachumfang mit Modula-2 vergleichbar, bietet darüberhinaus aber Elemente zur Definition von Prozessen, zur Prozeßablaufsteuerung und Interprozeßkommunikation an.

8.2.3.3 Einbettungssysteme

Wie sichtbar geworden ist, gibt es unabhängig vom implementierten Protokoll generell auftretende Teilprobleme (Zeitüberwachung, Ereignisschnittstelle und Aktivierung, Pufferverwaltung), und es

erhebt sich damit die Frage nach einer über die Standardprogrammierumgebung hinausgehenden speziellen Unterstützung.

Im einfachsten Fall wird sie als Laufzeitunterstützung durch einen Satz von Bibliotheksprozeduren realisiert. Weitergehende Laufzeitunterstützung kann ein vorgegebenes Leichtgewichtbetriebssystem bieten. Beide Möglichkeiten bewirken für die Programmierung eine Erweiterung der verwendeten Programmiersprache um vordefinierte Prozeduren.

Soll darüberhinaus eine Sprachanpassung an den Problembereich erfolgen, werden oft Vorübersetzer verwendet. Sie erweitern die Programmiersprache nun echt und unterstützen neue syntaktische Konstrukte, etwa zur Definition von Selektionstabellen, zur Vereinbarung der Datentypen von Datenstrukturen zur Zustands-, Ereignis- und PDU-Darstellung, sowie zur besonderen Kennzeichnung von Transitionsprozeduren.

Bei Anwendungsprotokollen ist hier insbesondere auch der Problembereich der Konstruktion und Analyse unterstützbar, wenn von formalen, in ASN.1 gehaltenen Formatfestlegungen ausgegangen werden kann. Sie können maschinell in PDU-Konstruktions- und Analyseprozeduren sowie Datentypen zur programminternen PDU-Darstellung abgebildet werden.

8.2.3.4 Schnelle Prototyp-Erzeugung

Ideale problemnahe Programmierumgebung ist ein Programmiersystem das Instanzenspezifikationen direkt ausführt, also ein Übersetzer und ein Laufzeitsystem für die Protokollspezifikationssprache. Wird eine formale Spezifikationssprache (wie z.B. ESTELLE, LOTOS, SDL) verwendet, dann ist das prinzipiell möglich und es werden für einige Sprachen auch bereits entsprechende Systeme angeboten.

Der Vorteil ist, daß der Programmentwurf mit dem dazu benötigten Personalaufwand und seinen vielen Fehlerquellen entfällt. Der Nachteil ist die durch die mechanische Umsetzung bewirkte Ineffizienz im Ablaufverhalten. Die Umsetzung folgt einigen Standardregeln und berücksichtigt weder besondere Eigenschaften des speziell zu implementierenden Protokolls noch solche der Zielrechnerkonfiguration. Deshalb werden solche Programmiersysteme zur Zeit nur sehr selten und wenn, dann meistens nur zur Erstellung von Prototypen für Vorstudien eingesetzt. Die Prototypen besitzen oft einen um einen Faktor von mehreren hundert erhöhten Laufzeitbedarf.

8.2.3.5 Implementierungsgeneratoren

Als Vorstadium eines Spezifikationssprachen-Programmiersystems können Programmrahmengeneratoren angesehen werden. Wie solche enthalten sie einen Übersetzer, dessen Quellsprache eine Spezifikationssprache ist. Allerdings wird die Zielsprache nicht verdeckt. Zielsprache ist eine höhere Programmiersprache oder Systemimplementierungssprache, und es wird vorausgesetzt, daß hierfür ein Programmiersystem zur Verfügung steht, mit dem sie weiterbearbeitet werden, d.h. daß der Entwickler die erzeugten Programme besichtigt und modifiziert.

Die erzeugten Programme sind dabei sehr oft lückenhaft. Kritische Aufgabenbereiche, wie Aktivierung und Kommunikation mit anderen Instanzen bleiben offen und müssen vom Programmentwickler selbst entworfen werden. Für in industrieller Umgebung eingesetzte Systeme findet man oft die Angabe, daß durch den Generator zwischen 20% und 45% des Programmentwicklungsaufwands eingespart werden.

Die Generatoren treten oft in Kombination mit bestimmten Einbettungssystemen auf. Das erzeugte Programm greift auf vorgegebene Laufzeitunterstützungsfunktionen zurück.

9. Rechnernetz-Anwendungen

Nachdem in den vorangegangenen Kapiteln auf die Kommunikation in verteilten Systemen eingehend eingegangen wurde, soll das vorliegende Kapitel sich endlich mit dem eigentlichen Zweck eines Rechnernetzes befassen, nämlich produktive Anwendungen zu tragen.

Daß der Kommunikation viele Kapitel gewidmet wurden und daß die Anwendungen nun mit nur einem Kapitel vergleichweise kurz behandelt werden, soll nicht zu dem Trugschluß verleiten, daß bei der Entwicklung einer Rechnernetz-Anwendung der überragende Teil des Arbeitsaufwands in den Kommunikationsbereich fällt. Es ist eher das Gegenteil der Fall. Die Implementierung der transportorientierten Kommunikationsdienste zwischen Stationen (ISO/OSI-Schichten 1..4) wird in der Regel standardmäßig vorhanden sein, also kaum Arbeitsaufwand verursachen. Die Implementierung der anwendungsorientierten Kommunikationsdienste (ISO/OSI-Schichten 5..7) ist in engem Zusammenhang mit dem Entwurf der eigentlichen Anwendung und den dort benötigten produktiven Funktionen zu leisten, gehört also eigentlich zur Anwendungsentwicklung dazu.
Die Gründe für die Beschränkung des Umfangs der Behandlung von Rechnernetz-Anwendungen liegen vielmehr in der für diese Vorlesung gewählten Schwerpunktbildung. Es sollte in erster Linie das für Rechnernetze im Vergleich zu einzelnen Rechnern wesentliche, nämlich die Telekommunikation zwischen Rechnern, zur Sprache kommen.

Die Behandlung von Rechnernetz-Anwendungen soll so auch auf Aspekte konzentriert werden, die sich auf die Besonderheiten verteilter Anwendungen gegenüber lokalen Anwendungen beziehen. Insbesondere soll hier auch nicht auf die spezifische produktive Funktionalität verschiedener Anwendungstypen und davon abhängiges eingegangen werden.

Es sollen kurz wesentliche Besonderheiten von Rechnernetz-Anwendungen im allgemeinen gegenüber lokalen Anwendungen aufgezählt werden.
Anschließend soll auf für Rechnernetz-Anwendungen geeignete Architekturkonzepte und zur Unterstützung ihrer Entwicklung bzw. ihres Ablaufs vorhandene Unterstützungssysteme eingegangen werden. Unter dem Stichwort Architektur befassen wir uns dabei - eher von der Problemseite kommend - mit der logischen Strukturierung einer Anwendung. Unter dem Stichwort Systemunterstützung befassen wir uns - von der Maschinenseite kommend - mit Möglichkeiten, die Basismaschine Rechnernetz der problemnäheren Programmierung zu erschließen.

9.1 Besonderheiten von Rechnernetz-Anwendungen

Die wichtigsten Besonderheiten von Rechnernetz-Anwendungen gegenüber rein lokalen Anwendungen folgen daraus, daß Rechnernetz-Anwendungen in Teile gegliedert sind, die sich an unterschiedlichen Orten (Stationen) befinden:

- Lose Kopplung: Zwischen den Teilen ist nur unzuverlässige Kommunikation möglich.
  Die Anwendung zerfällt also in weitgehend autonom arbeitende Komponenten. Eine zentrale Ablaufsteuerung kann direkt nicht realisiert werden. Keine Komponente kann eine aktuelle und umfassende Sicht des globalen Systemzustands besitzen.
  Komponenten (und auch Kommunikationsverbindungen) können unabhängig voneinander aus-

fallen. Die Anwendung kann in verschiedene voneinander isolierte Teilsysteme zerfallen bzw. sollte in der Lage sein, abgespaltene Teilsysteme zu reintegrieren.

- Heterogenität: Die Stationen (bzw. deren Betriebssysteme) können von unterschiedlicher Bauart sein. Es gibt viele unterschiedliche Typen von Betriebsmitteln (u.a. auch neue: Verbindungseinrichtungen, Datennetz, Netz-Hilfsdienste) mit unterschiedlicher Funktionalität und Leistungsfähigkeit sowie einer oft sehr differenzierten Kostenstruktur.
- Verteilte Administration: Installation und Pflege der Anwendung müssen je Station separiert für die dortigen Komponenten durchgeführt werden, da in der Regel kein zentrales Verwaltungssystem zur Verfügung steht.
  Erschwerend kann hinzukommen, daß verschiedene Stationen organisatorisch in unterschiedliche Verantwortungsbereiche fallen.
- Flexible Konfiguration: Die Größe der Basismaschine Rechnernetz bedingt, daß die Basismaschine mit hoher Wahrscheinlichkeit modifiziert wird. Einzelne Teile - z.B. Stationen, Betriebssysteme, Peripheriegeräte - können durch andere ersetzt werden oder es können Teile hinzukommen bzw. entfernt werden. Die Anwendung sollte so entworfen sein, daß Modifikationen der Basismaschine nicht umfassende Modifikationen der Anwendung nach sich ziehen.
- Offenheit (im Sinne von ISO/OSI): Manche Anwendungen gliedern sich in Teile, die von separaten Entwicklergruppen implementiert wurden. Wichtig ist hier, daß Interaktionsaspekte implementierungsunabhängig in einer Weise spezifiziert wurden, die es erlaubt, einzelne Teile ohne Kenntnis der konkreten Implementierung anderer Teile zu entwerfen.

Weitere Besonderheiten von Rechnernetz-Anwendungen stammen daher, daß die Integration von Nachrichtenverarbeitung- und Transport der rechnergestützten Behandlung sehr umfassende Problembereiche erschließt. Es sind deshalb eigentlich nicht Besonderheiten von Rechnernetz-Anwendungen sondern von 'großen' Anwendungen im allgemeinen:

- Umfang: Viele Funktionen, große Datenmengen, viele Komponenten.
- Hohe Lebensdauer: Anwendungen sollen nach einmaliger Installation über einen sehr langen Zeitraum (Jahre) ohne Betriebsunterbrechung ablaufen. Es sind deshalb verstärkt Fehler- und Ausfalltoleranzaspekte zu berücksichtigen.
- Überschneidung mit anderen Systemen: Es treten während des Ablaufs der Anwendung Interaktionen mit anderen Anwendungen (und auch mit Menschen) auf.
  Es ergeben sich integrierte Anwendungen, die zwar einen enger abgegrenzten Aufgabenbereich wahrnehmen, die hierzu aber mit anderen Anwendungen interagieren, so daß eine Menge solcher Anwendungen selbst wieder als eine einzige Anwendung angesehen werden kann.
- Anpaßbarkeit, Modifizierbarkeit: Größere und sehr langlebige Anwendungen werden mit hoher Wahrscheinlichkeit in Zukunft auch zur Lösung von Problemen herangezogen werden, die zum Zeitpunkt des Entwurfs noch nicht absehbar waren.
- Wichtigkeit: Unkorrekte Funktionalität oder mangelnde Leistungsfähigkeit können zu größeren Schäden (z.B. wirtschaftliche Verluste) führen.
- Gesellschaftliche Aus- und Einwirkungen: Ein großer Problembereich erfaßt einen größeren Ausschnitt der realen Welt. Damit sind oft juristische Aspekte (z.B. Schutz/Vertrauenswürdigkeit; Arbeitsrecht; Vertragsrecht, falls z.B. Rechner verbindliche Abmachungen treffen) direkt zu berücksichtigen. Weitergehend ist zu beachten, daß innovative größere Anwendungen zur Veränderung gesellschaftlicher Gegebenheiten beitragen können.

9.2 Architektur

Wie erwähnt, sollen unter dem Stichwort Architektur einige Konzepte / Modellvorstellungen / Paradigmen zu Struktur und Aufbau von Anwendungen behandelt werden, die beim Entwurf von Rechnernetz-Anwendungen dienlich sein können. Sie sollen die problemnahe Gestaltung einer Anwendung ermöglichen und sind deshalb eher als logische Architekturkonzepte zu sehen. Gleichwohl soll angemerkt werden, daß - ein entworfenes System soll ja später auch mit vertretbarem Arbeitsaufwand

in eine effiziente und leistungsfähige Implementierung umgesetzt werden können - sie sich am Machbaren orientieren müssen, also auch als Programmarchitekturkonzepte zu sehen sind.

9.2.1 Spektrum

Eine echte Rechnernetz-Anwendung zerfällt in der Implementierung in einzelne Komponenten, die sich an verschiedenen Orten befinden. Von einzelnen Komponenten werden verschiedene Funktionen wahrgenommen. Zwischen den Komponenten finden Interaktionen statt.

Beim Entwurf einer Anwendung wird man deshalb grob folgende Teilaufgaben bewältigen müssen:

- Identifikation von einzelnen Funktionen,
- Gliederung der Anwendung in Komponenten, Zuordnung von Funktionen zu Komponenten,
- Festlegung der zwischen den Komponenten benötigten Interaktionen.

Im folgenden soll das Spektrum von Komponenten, Funktionen und Interaktionen kurz umrissen werden.

9.2.1.1 Komponenten

Es folgt eine Aufzählung von häufig zu findenden Komponenteneigenschaften:

- aktiv ↔ passiv,
- verletzlich, ausfallbehaftet ↔ sicher, persistent,
- permanent vorhanden ↔ temporär vorhanden,
- statisch definiert ↔ dynamisch definiert,
- lokal bekannt ↔ global bekannt,
- lokal zugreifbar ↔ abgesetzt zugreifbar ↔ global zugreifbar,
- örtlich festgelegt ↔ wandernd / mobil ↔ zentral vorhanden ↔ global vorhanden,
- billig ↔ teuer.

Paradigmen führen bestimmte Komponententypen ein. Dabei erscheint oft als zentral die Unterscheidung zwischen aktiven Komponenten (Prozessen) und passiven Komponenten (Betriebsmittel). Die passiven lassen sich weitergliedern in Zustandsspeicher (Datenstrukturen) und Kopplungskomponenten.

Es folgt die Aufzählung einiger Beispiele dazu unter Nennung interessanter Varianten:

- Prozeß / Prozeßfamilie / Prozeßteam
  - replizierte Prozesse, Stand-by-Prozesse,
  - überwachte Prozesse, Überwacher,
  - Master, Slaves,
  - Client-Prozesse, Server-Prozesse.
- Datenstrukturen, Betriebsmittel
  - mit / ohne Zugriffskontrolle (Synchronisation, Organisation),
  - geteilt ↔ privat für einzelne Prozesse,
  - Original, Kopie / Replikat,
  - Speicherbereiche, Datenstrukturen, Dateien, Monitore, Datenbanken, Objekte (im Sinne objektorientierter Systeme: Einkapselung, Vererbung, Klassenbildung).
- Prozeß / Datenstruktur - Verbund
  - aktive Objekte,
  - Zugriffsserver, Verwalter.
- Kopplungskomponenten

- Interprozeßkommunikationsobjekte (Pipes, Sockets),
- Telekommunikationsverbindungen, Zugangspunkte,
- Assoziationen.

9.2.1.2 Funktionen

Funktionen sollen hier nicht inhaltlich sondern nach allgemeineren 'äußerlichen' Kriterien unterschieden werden.

Grob unterscheidet man zwischen aktiver, stimulierender Funktionalität und passiver, reagierender Funktionalität.
Aktive Funktionen besitzen ein selbständiges, zeitlich oft unendliches Verhalten, entsprechende Komponenten sind Prozesse oder Agenten.
Passive Funktionen führen nur auf Anforderung Aktionen aus. Sie können durch in endlicher Zeit terminierende Verhaltensphasen beschrieben werden. Entsprechende Komponenten sind Prozeduren oder Operationen (u.U. auch Objekt-Methoden).

Eine weitere Unterscheidung kann nach der Wichtigkeit der Funktionen erfolgen. So kann man zwischen eher im Vordergrund eines Systems und eher im Hintergrund eines Systems liegenden Funktionen unterscheiden.

Geht man schon etwas näher auf die speziellen Inhalte eines Systems ein, dann können Funktionen unterschiedlichen Rollencharakter besitzen, z.B. auftragserteilende - auftragsbearbeitende Funktion; Sekretär-, Verteiler-, Verwalter-, Überwacherfunktion; optimierende, sichernde Funktion.

Abschließend soll erwähnt werden, daß beim Entwurf oft verschiedene logische Betrachtungsebenen eingeführt werden und daß dieselbe Funktion auf verschiedenen Betrachtungsebenen unterschiedlichen Charakter besitzen kann. So kann z.B. auch eine auf höherer Ebene als passiv betrachtete Funktion auf tieferer Ebene aktive Elemente enthalten.

9.2.1.3 Interaktionen

Interaktionen zwischen Komponenten lassen sich aus technischer Sicht gliedern in:
- Kommunikationsinteraktionen; hier führen die zusammenarbeitenden Komponenten Nachrichtenaustausch aus,
- Kooperationsinteraktionen (im engeren Sinne); hier manipulieren die Komponenten gemeinsame Zustandsspeicher.

Sie können nach unterschiedlichen Aspekten näher aufgeschlüsselt werden:
- zuverlässig - unzuverlässig,
- Effekte flüchtig - Effekte persistent,
- Interaktionen überwacht, rücksetzbar, wiederholbar,
- Konsistente, inkonsistente Teilsichten erzeugend,
- teuer - billig, schnell - langsam.

Es ist anzumerken, daß Kooperationsinteraktionen auch im verteilten System (in dem - wie bekannt - in der Implementierung alle Interaktionsformen auf Nachrichtenaustausch abgebildet werden müssen) auf höherer logischer Betrachtungsebene sinnvoll sein können.

Auf höherer Betrachtungsebene sind dann häufig auch Kombinationen der beiden Formen zu finden. Sie enthalten bereits inhaltliche Denkmuster. Beispiele dafür sind 'Prozedur- / Operationsaufruf und Rückkehr (Fernaufruf, Remote Operations)', 'Prozeßerzeugung, Start, Terminierung', 'Moderierter Nachrichtenaustausch (Schwarzes Brett, Konversation)', 'Gemeinsame Aktionen in Telekonferenz'.

Häufig benutzte Interaktionstypen sind:
- Nachrichtenaustausch zwischen zwei Partnern grundlegend: Benachrichtigung, bestätigte Benachrichtigung.
- Nachrichtenaustausch zwischen zwei Partnern umfassender: Verbindung, Datenstrom-Übertragung, Aufruf- / Rückkehr-Übertragung, Dialog.
- Nachrichtenaustausch zwischen mehreren Partnern: Sammelbenachrichtigung, Umlauf, Konversation.
- Datenabstimmung: Informationstreffen / Rendezvous.
- Kooperation: Verständigungsbereich / Schwarzes Brett, Monitor-Nutzung, Datenbankzugriffe, Arbeitstreffen / Joint Actions / Gemeinsame Unterprozesse.

9.2.2 Paradigmen

Im folgenden sollen einige beim Entwurf verteilter Anwendungen geeignete Modelle zur Gliederung und Konstruktion der Anwendungen vorgestellt werden. Je Ansatz finden sich aufeinander abgestimmte Ausprägungen an Komponenten-, Funktionen- und Interaktionsmustern. Die Ansätze schließen sich im allgemeinen nicht gegenseitig aus. In einem größeren Projekt wird man oft mit Mischformen arbeiten. Auch können auf verschiedenen Betrachtungsebenen unterschiedliche Modelle vorherrschen.

9.2.2.1 System kommunizierender Prozesse

Eine Anwendung wird in eine Menge von Prozessen / Agenten strukturiert. Die Prozesse können miteinander per Nachrichtenaustausch kommunizieren.

Datenstrukturen / Zustandsspeicher / Betriebsmittel werden durch Prozesse verdeckt. Es gibt nur private, von je einem einzigen Prozeß exklusiv zugegriffene Zustandsspeicher, so daß sie in den Entwurf des Gesamtsystems nur indirekt eingehen.

Systeme kommunizierender Prozesse sind ein sehr grundlegendes Konzept. Es ist implementierungsnah und führt, da die in der Implementierung Aufwand verursachenden Elemente im Modell schon sichtbar sind, bei sinnvoller Anwendung zu effizienten Systemen.
Nachteilig ist, daß eine einfache abstrakte Sicht des Systems nicht unterstützt wird. Bei der Betrachtung des Gesamtsystems ist die Nebenläufigkeit der Prozesse zu beachten.

9.2.2.2 Clients und Server

Dies ist eine in der derzeitigen Praxis weit verbreitete Variante zu 'Systeme kommunizierender Prozesse'. Sie unterstützt eine vereinfachte abstrakte Sicht eines Systems. Prozesse treten in zwei Grundmustern auf:
- Client; ein Client-Prozeß vergibt Aufträge durch Aufruf von Server-Operationen.
- Server; ein Server-Prozeß verwaltet systemweit relevante Zustandsspeicher und kapselt sie ein. Er wird nicht von sich aus tätig sondern führt nur Aufträge von Clients durch.

Die vereinfachte abstrakte Sicht auf ein Client-Server-System konzentriert sich auf Clients und global relevante passive Komponenten.
Die Serverprozesse können als Implementierungsdetails in den Hintergrund treten. Die passiven Komponenten und ihre Rollen im System können nach 'Information-Hiding'-Prinzipien (Einkapselung) abstrakt modelliert werden.

Der stattfindende Nachrichtenaustausch wird ebenfalls verdeckt. In abstrakter Sicht ist nur zu beobachten, wann welcher Client welche Server-Operation ausführen läßt.

Komplexe Client-Server-Systeme werden oft hierarchisch strukturiert. Eine in eine höhere Schicht als Server eingehende Komponente kann in tieferer Schicht selbst wieder in Clients und Server, unter Umständen auch in symmetrisch kooperierende Prozesse, zerfallen. Hier treten dann letzendlich Prozesse auf, die sowohl Client- als auch Server-Charakter haben.
Dies ist ebenfalls oft bei integrierten Anwendungssystemen (sie bestehen aus mehreren zusammenarbeitenden Anwendungen) der Fall. Aus dem Blickwinkel einer einzelnen Anwendung tritt die betrachtete Anwendung als Client, die übrigen Anwendungen treten als Server auf, die Anforderungen der betrachteten Anwendung bedienen.

9.2.2.3 Geteilte Objekte und kooperierende Prozesse

Bereits im Ansatz 'Clients und Server' kann, wenn man Server in den Hintergrund treten läßt und stattdessen die von ihnen verwalteten Zustandsspeicher / Betriebsmittel als global relevant in den Vordergrund rückt, sich ein Modell ergeben, in dem Client-Prozesse und eingekapselte geteilte Zustandsspeicher dominieren. In abstrakter Sicht greifen die Client-Prozesse auf geteilte Zustandsspeicher zu und können somit per Kooperation (im engeren Sinne) interagieren.

Dem objektorientierten Ansatz folgend können die geteilten Betriebsmittel als Objekte aufgefaßt werden, als Objekte im Sinne der objektorientierten Programmierung, die sich durch drei Konzepte auszeichnen:

- Einkapselung: Der Objektzustand ist nur per im Zusammenhang mit dem Objekt definierten Operationen zugreifbar.
- Klassenbildung: Die Beschreibung von Systemen, in denen mehrere gleichartige Objekte vorkommen, wird durch die Möglichkeit zur Definition von Objektklassen und zur Erzeugung von Objektinstanzen erleichtert.
- Vererbung: Wiederverwendung, Variantenbildung und Erweiterungen werden durch Vererbungsmechanismen unterstützt.

An dieser Stelle soll keine umfassende Beschreibung objektorientierter Ansätze - die ja nicht speziell im Zusammenhang mit verteilten Systemen von Interesse sind - gegeben werden. Es soll abschließend nur auf spezielle Aspekte objektorientierter Ansätze für verteilte Systeme hingewiesen werden:

- Nebenläufige Prozesse rufen Objektoperationen: Beim 'traditionellen' objektorientierten Entwurf treten Prozesse in den Hintergrund, man konzentriert sich auf die Objektwelt und geht gern von einer sequentiellen Aufrufumgebung aus. Im verteilten System würde eine strenge Sequentialisierung zu geringer Leistung führen.
- Orte: Objekte und Prozesse werden (jedenfalls in der Implementierung) von bestimmten Stationen des Rechnernetzes getragen. Je nach Distanz können deshalb Operationsaufrufe unterschiedlich schnell und teuer sein. So gibt es Ansätze, die den Objektort berücksichtigen. Stichworte dazu sind: 'Wandernde Objekte (sie werden im Netz nach leistungsoptimierenden Kriterien bewegt)', 'Replizierte Objekte (Kopien vermindern den Kommunikationsaufwand)', 'Globale Objekte (sie sind in der abstrakten Sicht systemweit einheitlich erreichbar)', 'lokale Objekte (sie sind ortsgebunden und nur für den Zugriff von diesem Ort aus vorgesehen)', 'abgesetzt zugreifbare Objekte (lokale Objekte, die auch von anderen Orten aus - allerdings teurer - zugreifbar sind)'.
- Fehlertoleranz: Langlebigkeit, geforderte Leistungsfähigkeit und Verteilung erfordern die Unterstützung von Fehlertoleranzmechanismen schon beim Entwurf. So können in manchen Ansätzen flüchtige und persistente Objekte unterschieden werden. Wiederherstellungsoperationen können vorgesehen sein.

9.2.2.4 Datenbank und kooperierende Prozesse

Die Nebenläufigkeit der einen geteilten Zustandsspeicher zugreifenden Prozesse, die Langlebigkeit eines Systems und die Erweiterbarkeit eines großen Systems traten als Anforderungen nicht erst mit Rechnernetzen auf. Im Bereich der Datenbanken / Informationssysteme werden diese Ziele schon länger mit dem Ansatz 'Datenbank' verfolgt.
Es gibt einen zentralen, sicheren und konsistenten Speicher für den Zustand einer Anwendung, die Datenbasis. Um eine logische abstrakte Sicht und die Erweiterungsfähigkeit (Stichwort Datenunabhängigkeit) zu unterstützen hat sie einem konzeptuellen Schema zu entsprechen. Die Nutzung wird problemnah in Termen des konzeptuellen Schemas definiert. Die tatsächliche Speicherung wird separiert und findet im Datenbasisverwaltungssystem statt.
Das Datenbasisverwaltungssystem führt die fehlertolerante physikalische Speicherung der Daten und die Kontrolle der (nebenläufigen) Zugriffe in einer Weise durch, daß man aus logischer Sicht von einem zentralen persistenten Zustandsspeicher ausgehen kann, auf dem in logischen Termen definierte komplexe Operationen - als würden sie unter gegenseitigem Ausschluß stattfinden - ausgeführt werden können (die komplexen Operationen heißen Transaktionen und bestehen aus Folgen einfacher Operationen).

Dieses Grundmodell geht also von einer Menge nebenläufiger Prozesse aus, denen zur Kooperation (im engeren Sinne) eine einzige zugriffssynchronisierte zentrale geteilte Datenstruktur zur Verfügung steht. Die Nebenläufigkeit der Prozesse muß nicht vorrangig berücksichtigt werden, man kann jeden Prozeß weitgehend separiert von den anderen entwerfen, da Zugriffssynchronisation vorhanden ist und Prozeßzugriffe auf die Datenstruktur in konsistenzerhaltenden Transaktionen erfolgen. Die leistungsfähige Implementierung der Datenbasis wird aus dem problemnahen Entwurf ausgeklammert (wie die Datenbasis und das Datenbasisverwaltungssystem dann im Rechnernetz tatsächlich implementiert werden - z.B. zentrale Datenbank mit abgesetztem Zugriff, verteilte Datenbank - bleibt späteren Entwurfsphasen vorbehalten).
Damit hat dieser Ansatz viele Vorteile für den problemnahen Entwurf und findet deshalb bei größeren Anwendungen häufig Verwendung.

9.2.2.5 Gemeinsame Aktionen und kooperierende Prozesse

Beim Ansatz 'Datenbank und kooperierende Prozesse' steht die Datenbasis, d.h. das konzeptuelle Model der Datenbasis, im Vordergrund, Prozesse werden nachrangig entworfen. Daß Aktionen damit in den Hintergrund treten, daß für eine große Anwendung ein umfassendes Datenmodell mit hohem Aufwand entwickelt werden muß und daß in der späteren Implementierung die zentrale Datenbasis oft als Leistungsengpaß auftritt, wird als Nachteil angesehen.

Beim Ansatz 'Gemeinsame Aktionen kooperierender Prozesse' rückt man deshalb die Prozesse wieder in den Vordergrund. In den Prozessen finden die interessierenden nützlichen Aktionen statt. Sie sind an Stationen lokalisiert. Sie besitzen ein Verhalten im Lauf der Zeit und werden folglich verschiedene Arbeitsphasen durchlaufen. Dabei können Phasen unterschieden werden, in denen sehr effizient rein lokal gearbeitet werden kann, und solche, in denen Kooperation stattfindet. Primäres Ziel (aus Leistungsgründen), das schon beim Entwurf verfolgt werden soll, ist es, lange lokale Arbeitsphasen zu ermöglichen. Hierzu sind die Betriebsmittel bei den Prozessen lokal angesiedelt und sollen nicht zentral gehalten werden.
Die Kooperation wird nun deshalb nicht in die Fassung einer zentralen, allgültigen Datenstruktur gepreßt, sondern man geht davon aus, daß eine Menge von Prozessen, die sich in einer Kooperationsphase befinden, einige ihrer Betriebsmittel für diese Kooperationsphase teilbar machen. Man unterscheidet zwischen verschiedenen Kooperationstypen und definiert für jeden Prozeß die Rolle (das Verhaltensmuster), die er im entsprechenden Kooperationstyp zu spielen hat.

9.2.2.6 System aktiver Objekte

Der Ansatz objektorientierter Modelle kann auch dadurch an die Verwendung bei verteilten Systemen angepaßt werden, daß man den Objektbegriff erweitert. Wesentlich ist dabei der Übergang vom passiven zum aktiven Objekt. Solche Objekte 'mit Eigenleben' kapseln nun neben Zustand und Zugriffsoperationen auch ein oder mehrere Prozesse ein, die von sich aus zustandsverändernd wirken können und u.U. auch Operationen fremder Objekte aufrufen können.

9.2.2.7 Nachrichtenprozesse

Der Ansatz 'System kommunizierender Prozesse' ging davon aus, daß die Aktionen eines Prozesses an einem Ort durchgeführt werden. Das Programm eines Prozesses schreibt vor, was an einer Station, einem Knoten des Rechnernetzes, an Verarbeitung durchzuführen ist, ist also sozusagen ein Knotenprogramm. Die Entfernungen zwischen Knoten werden durch Nachrichten - 'nackte' Daten - überbrückt.
Der Ansatz der Nachrichtenprozesse kehrt diese Verhältnisse um. Hier sind die wesentlichen Zustandsspeicher des Systems an Knoten lokalisiert. Prozesse sind potentiell in Bewegung. Sie wandern jeweils zu der Station, an der sie als nächstes Zustandsspeicher zugreifen müssen, und nehmen prozeßprivate Daten mit. Das Programm eines Prozesses ist damit sozusagen ein Nachrichtenprogramm.
Während im Knotenprogramm Anweisungen der Arten 'Sende Nachricht an' und 'Empfange nächste Nachricht (von)' auf die Verteilung des Systems eingehen, finden sich im Nachrichtenprogramm stattdessen Anweisungen der Art 'Ziehe in Station x um / Besuche Zustandsspeicher x'.

Es soll erwähnt werden, daß auch im Ansatz 'System kommunizierender Prozesse' Prozesse als mobil definiert werden können, es sich aber dennoch um Knotenprozesse handelt. Man wird dort mobile Prozesse primär nur vorsehen, um einem unterstützenden System Lastausgleich und Leistungsoptimierung durch Prozeßverlagerung zu ermöglichen. Daß Wandern der Prozesse ist dort also im Hintergrund und trägt nicht direkt zur Problemlösung bei.
Im Ansatz 'Nachrichtenprogrammierung' wandern die Prozesse dagegen programmgesteuert und jede Ortsveränderung ist problembezogen.

Weiterhin soll erwähnt werden, daß man bei der Implementierung es natürlich vermeiden wird, tatsächlich Prozeßcode und andere unveränderliche Daten zu übertragen. Man wird deshalb in einer späteren Entwurfsphase das entworfene Nachrichtenprozeßsystem in ein Knotenprozeßsystem überführen.

9.3 Systemunterstützung

Während der vorangegangene Abschnitt Modellvorstellungen umriß, die beim Topdown-Entwurf von Problemseite kommend angewendet werden können, um eine geeignete logische Architektur einer Rechnernetz-Anwendung zu entwerfen, soll nun darauf eingegangen werden, daß man nach endlicher Anzahl von Entwurfsphasen ja letztendlich eine existierende Basismaschine erreichen muß. Es tritt hier also der Blickwinkel des Bottomup-Entwurfs in den Vordergrund.

Die klare Trennung zwischen logischem, problemorientierten Modell der frühen Entwurfsphasen und dessen späterer Abbildung auf gegebene Basismaschinen, wie sie dem reinen Topdown-Entwurf entspricht, findet sich in der Praxis allerdings nur sehr selten, da man durch die allzu späte Berücksichtigung der Basismaschine sehr leicht ineffiziente Lösungen entwickelt. Man bemüht sich deshalb schon

früh um eine Abstimmung zwischen logischem abstraktem Modell und der späteren Implementierung (Mischformen aus Topdown und Bottomup).
Dementsprechend finden sich auch Querbeziehungen zwischen Basismaschinen / Systemunterstützungstypen und Paradigmen. Bestimmte Paare passen in dem Sinne gut zueinander, daß die zugrundeliegenden Modellvorstellungen nicht allzu weit voneinander entfernt sind, bzw. eine aus einer Verfeinerung der anderen hervorgeht oder beide sogar weitgehend identisch sind..

Es sollen zunächst einige allgemeine Typen von Basismaschinen besprochen werden, bevor dann auf einige wenige konkrete Beispiele kurz eingegangen wird.

9.3.1 Arten den Systemunterstützung

Bereits bei der Entwicklung traditioneller lokaler Anwendungen geht man schon lange nicht mehr von der nackten Hardware als Basismaschine aus. Man geht so gut wie immer von der Verfügbarkeit eines Betriebssystems aus. Dies wird bei systemferner Programmierung verdeckt bzw. bei systemnaher Programmierung ergänzt durch die abstrakte Maschine, die vom Programmiersystem der verwendeten (höheren) Programmiersprache zur Verfügung gestellt wird. Unter Umständen kommen dann noch spezielle unterstützende Systeme, wie z.B. ein Datenbankverwaltungssystem, dazu.

Bei Rechnernetz-Anwendungen gilt dies ebenso. Von zusätzlichem Interesse ist jetzt weiterhin, wie der Netzcharakter der Basismaschine 'Rechnernetz' behandelt wird: Muß man z.B. für jede Anwendung den ganzen ISO/OSI-Turm mitentwickeln? Muß man bei heterogenem Netz alle individuellen Eigenheiten einzelner Stationen im Detail verücksichtigen?
Hier ist mindestens eine Trennung der Implementierung des Kommunikationssystems an einer Schnittstelle gängig, an der Dienstleistungen angeboten werden, die für alle möglichen Anwendungen notwendig sind. Für die Implementierung dieser Schnittstelle und der darunter liegenden Schichten sollen für die verschiedenen Rechnertypen, Betriebssysteme und physikalischen Verbindungen geeignete 'fertige' Produkte herangezogen werden können. Die Implementierung der oberhalb liegenden Schichten zählt mit zum Aufgabenbereich der Anwendungsentwicklung. Diese Schnittstelle ist nach dem Konzept des ISO/OSI-Modells die Transportschnittstelle.

Diese Sicht der einer Anwendungsentwicklung zugrundegelegten Basismaschine 'Rechnernetz' wird 'Netzwerkfähige Stationen' genannt. Um das Niveau der Basismaschine weiter anzuheben, darf nicht nur die Kommunikation zwischen Anwendungsprozessen unterstützt werden, sondern es muß eine Homogenisierung des Erscheinungsbilds der unterschiedlichen Stationen gegenüber den Anwendungsprozessen stattfinden, die sich auch auf die übrigen Dienste einer modernen Basismaschine (Rechnerhardware+Betriebssystem) bezieht, wie z.B. Nutzer-, Programm- und Prozeßverwaltung, Ein/Ausgabe, Bedienung, Dateisysteme, Verzeichnisdienste. Hierzu sind gegenwärtig im wesentlichen folgende Stichwörter zu nennen:

- Netz-Betriebssystem,
- Netz-Betriebssystem nach Client-Server-Modell,
- Verteilte Datenbanken,
- Verteiltes Betriebssysteme,
- Verteilte Programmiersprachen.

9.3.1.1 Netzwerkfähige Stationen

Die geringste, nur auf Kommunikationsaspekte beschränkte, Systemunterstützung für verteilte Anwendungen ergibt sich mit dem Konzept der netzwerkfähigen Stationen. Es soll an einem Beispiel verdeutlicht werden.

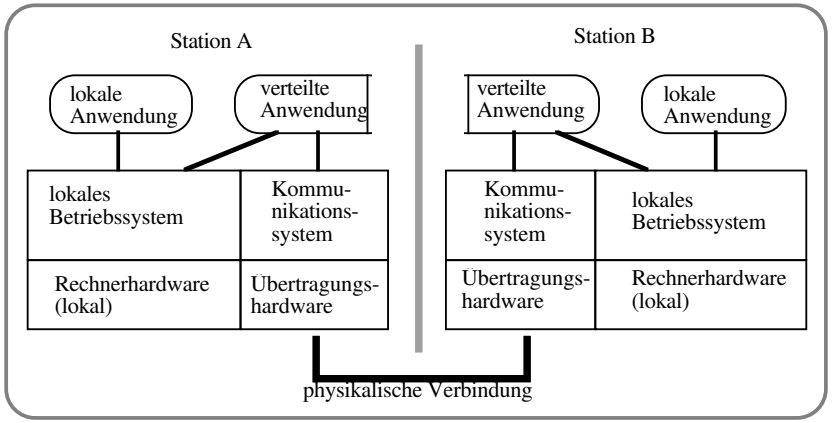

Bild 9.1: Netzwerkfähige Stationen

Bild 9.1 zeigt dazu ein (aus zwei Stationen A und B) bestehendes Rechnernetz. Die Stationen werden separat durch die (u.U. unterschiedlichen) lokalen Betriebssysteme organisiert. Sie sind aber netzwerkfähig durch Übertragungshardware und softwaremäßige Implementierung von Kommunikationsdiensten. Eine verteilte Anwendung besteht aus Prozessen, die je Station zum einen die dortigen lokalen Verhältnisse berücksichtigen müssen, und zum anderen (zur Interaktion mit dem entfernten Partner) auf dem Kommunikationsdienst aufsetzen, wie er an den jeweiligen örtlichen Zugangspunkten angeboten wird.

Als netzweit angebotener Kommunikationsdienst wird dabei häufig der Transportdienst gewählt. Daneben gibt es auch Produkte und Produktfamilien, die sich auf die Funktionalität eines Netzwerkdienstes beschränken, oder solche, die darauf abzielen, lokale und netzweite Interprozeßkommunikation mit einem einheitlichen Spektrum an Dienstleistungstypen abhandeln zu können, das dann auch bereits Elemente eines Kommunikationssteuerungsdienstes enthält.

Bild 9.2 zeigt eine nach diesem Konzept gewählte Grobstruktur einer aus zwei Anwendungsprozessen bestehenden Anwendung in einem aus zwei Stationen bestehenden Rechnernetz. Das Bild zeigt in den beiden Rechtecken die beiden Stationen, die als Basismaschine der Anwendungsprozesse dienen. Sie setzen sich aus der Rechnerhardware, den Verbindungseinrichtungen, den Softwareprodukten zur Bildung des Transportsystems und den beiden lokalen Betriebssystemen zusammen.

Anwendungsprozesse sind Prozesse im Sinn der jeweils lokalen Betriebssysteme. Die Implementierung der anwendungsorientierten Dienste erfolgt im Zusammenhang mit diesen Prozessen. Zwischen Anwendungsprozessen und lokaler Systemumgebung bestehen einerseits die auch für nicht-verteilte Anwendungen übliche Schnittstelle zwischen Programm und Betriebssystem und andererseits eine Schnittstelle zum Transportsystem. Die Transportsystemschnittstelle wird ebenfalls mit den Mitteln des lokalen Betriebssystems realisiert. Die vertikale Kommunikation an Transportdienstzugangspunkten erfolgt durch lokale Interprozeßkommunikation zwischen Anwendungsprozeß und Systemprozessen. Nicht nur die logische Funktionalität des Transportdienstes sondern insbesondere auch die spezielle Implementierung der Dienstschnittstelle sind wichtige Elemente der Implementierungsumgebung der zu entwickelnden Anwendung.

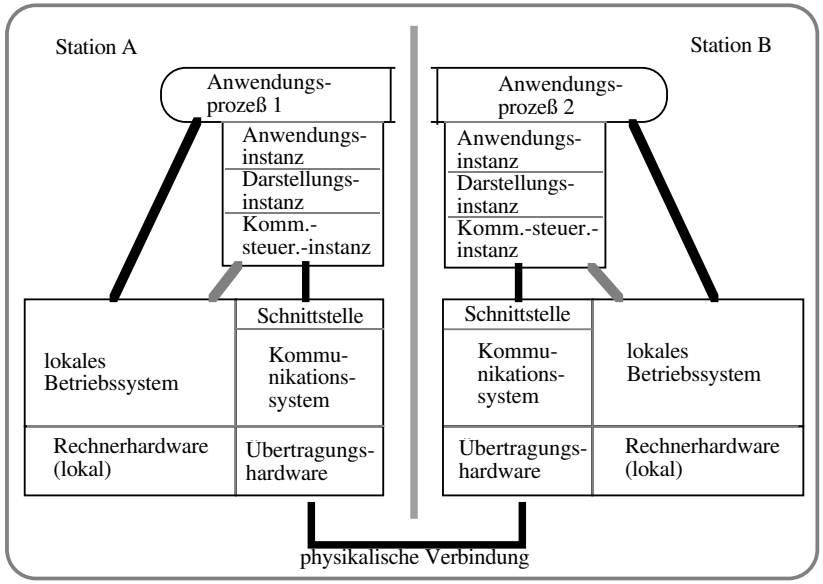

Bild 9.2: Rechnernetz und Anwendung

9.3.1.2 Verteilte Programmiersprachen

Das Stichwort 'Verteilte Programmiersprachen' steht für den Ansatz, der dem Anwendungsentwickler am meisten entgegenkommt. Die Basismaschine, für die er sein Programm zu entwerfen hat, wird durch die Semantik der Programmiersprache in einem einheitlichen Rahmen definiert. Es ist Aufgabe des Compilers und des Laufzeitsystems für die verträgliche Ausführung im konkreten Rechnernetz zu sorgen. Dies ist in Bild 9.3 skizziert.

Gehen wir von einer üblichen höheren Programmiersprache (z.B. Pascal) aus, dann kann im Rahmen eines Programms die Ablaufvorschrift für einen entsprechenden Prozeß problemnah definiert werden. Die Schnittstelle zum Betriebssystem wird durch die Sprachelemente zur E/A verdeckt.

Um nun z.B. Pascal verteilungsfähig zu machen, müßte es im ersten Schritt so erweitert werden, daß Systeme aus nebenläufigen Prozessen programmiert werden können. Sprachkonstrukte zur Definition von Prozeßtypen und zur Instanziierung von Prozessen sowie zur Interprozeßkommunikation sind einzuführen.

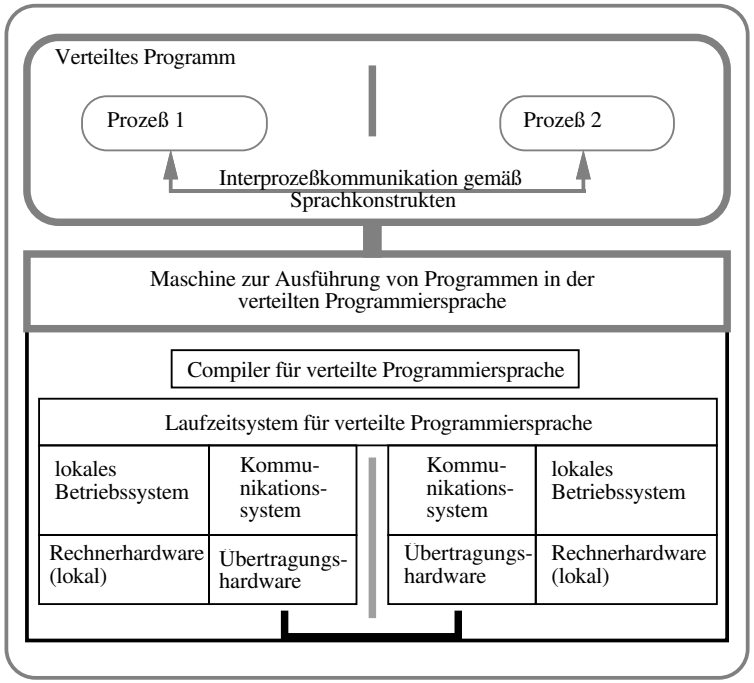

Bild 9.3: Verteilte Programmiersprache

Bei solchen nebenläufigen Programmiersprachen wird dabei in der Regel noch davon ausgegangen, daß keine Verteilungsprobleme bestehen. Alle Prozesse laufen auf derselben konkreten Basismaschine und damit in homogener Umgebung ab. Daten die von einem Prozeß an einen anderen übergeben werden, differieren nicht in ihrer Bitmusterdarstellung. Alle von der Sprache angebotenen Operationen sind von allen Prozessen mit jeweils gleichen Kosten aufrufbar. Die Interprozeßkommunikation wird als zuverlässig gewertet. Kommunikationskosten sind gering, insbesondere kann von kurzen Übertragungslaufzeiten ausgegangen werden. Entsprechende nebenläufige Programmiersprachen, wie z.B. Ada, Concurrent Pascal, Pearl, sehen deshalb auch zur Interprozeßkommunikation Mechanismen vor, die zwar konzeptuell einfach sind (Datenaustausch in Rendezvous-Synchronisation, Monitore, geteilte Variable), die aber nur auf einer örtlich eng zusammenhängenden Basismaschine mit geringen Kosten realisiert werden können.

Es gibt Ansätze, solche nebenläufigen Programme - ohne Änderungen und ohne zusätzliche Festlegungen zu treffen - durch Übersetzer und Laufzeitsysteme automatisch auf ein Rechnernetz abbilden zu lassen. Es ergeben sich aber folgende Probleme:

- Die tatsächlich nur lose Kopplung zwischen weit entfernten Prozessen kann nicht verdeckt werden. Wenn das Anwendungsprogramm die Verbindungs- und Partnerausfälle tolerieren soll, muß der Programmentwurf revidiert werden. Zum Erkennen von Verbindungs- und Partnerausfällen müssen die Interprozeßkommunikationskonstrukte der Programmiersprache geeignet erweitert werden.

- Die mögliche Heterogenität der Stationen des Netzes und die entfernungsabhängigen Qualitäten der Kommunikationsmechanismen verlangen eine fein abgestimmte Lokalisierung der Prozesse und Kommunikationsobjekte im Netz. Zur selbsttätigen (u.U. adaptiven) Lokalisierung durch das Laufzeitsystem geeignete Verfahren verursachen eine hohe zusätzliche Rechnerbelastung.

Derartige Ansätze befinden sich gegenwärtig noch weitgehend im Laborstadium. Dabei werden vorwiegend lokale Netze mit sehr guten Übertragungsqualitäten eingesetzt. Heterogenitätsprobleme (Datendarstellung, unterschiedliche Maschinensprachen und lokale Betriebssysteme) werden überwiegend durch die Beschränkung auf Stationen verträglicher Bauart gemildert. Zur Vereinfachung des Laufzeitsystems beschränkt man sich häufig auf verteilungsgeeignete Interprozeßkommunikationsmechanismen (puffernde Kanäle mit zeitüberwachbaren Zugriffsoperationen) und macht die Lokalisierung der Prozesse auf Sprachebene bestimmbar.

Es gibt ferner Vorschläge und Labormuster für verteilte Programmiersprachen, die von der besprochenen Grundlinie, nämlich daß eine 'traditionelle' auf den Paradigmen 'Prozeß', 'Prozedur' und 'Datenstruktur' beruhende nebenläufige imperative Programmiersprache um verteilungsbezogene Fähigkeiten erweitert wird, abweichen.
Wichtige Stichworte sind hier 'Verteilte objektorientierte Programmiersprache', 'Verteilte funktionale Programmiersprache', 'Verteilte logische Programmiersprache'. Hierauf soll an dieser Stelle nicht näher eingegangen werden.
Angesprochen werden soll jedoch, daß man bei der Zielrichtung verteilter Programmiersprachen zwei sehr verschiedene Grundrichtungen erkennen kann:

- Unterstützung *echter* verteilter Anwendungen; dies ist die Zielrichtung, die auch im Rahmen dieser Vorlesung von Interesse ist. Es geht darum langlebige Anwendungen zu programmieren, die sich schon von Problemseite her dadurch auszeichnen, daß bestimmte Funktionen an einzelnen Orten stattfinden müssen (Beispiel: Flugbuchungssystem mit problembedingten Orten 'Reisebüros').
  Sprachen mit dieser Zielrichtung unterstützen Ortsfestlegungen, die Berücksichtigung von Entfernungskosten und die Integration von Fehlertoleranzmaßnahmen.
- Verteiltes Supercomputing; hier soll das Rechnernetz (das in der Regel ein LAN ist) dazu verwendet werden, rechenintensive nicht-verteilte Probleme schnell dadurch zu lösen, daß die Parallelverarbeitungsfähigkeit der im Netz zur Verfügung stehenden Prozessoren genutzt wird. Da LANs oft sehr zuverlässige Kommunikation erlauben und vom Problem her zwar u.U. Parallelität aber keine Verteilung vorhanden ist, können solche Sprachen Verteilungsaspekte verdecken (z.B. Ortstransparenz).

9.3.1.3 Verteilte Betriebssysteme

Sehr verwandt zum Ansatz der verteilten Programmiersprache ist der des verteilten Betriebssystems. Auch hier soll den Anwendungsprozessen eine möglichst homogene Umgebung, diesmal durch die Programmschnittstelle eines Betriebssystems geschaffen werden. Dieses Betriebssystem ist logisch für das gesamte Rechnernetz ein einziges System, das die unterliegenden Betriebsmittel der Stationen und Verbindungseinrichtungen insgesamt verwaltet, so daß alle funktionellen lokalen Unterschiede verdeckt werden.

Bild 9.4 zeigt die grundsätzliche Struktur. Die Hardware und lokalen Betriebssysteme der verschiedenen Stationen werden vom verteilten Betriebssystem verdeckt. Auch lokale Anwendungen können nur indirekt über Operationen des verteilten Betriebssystems auf Betriebsmittel zugreifen.

Auch diese Lösung wird aus Aufwandsgründen nur sehr selten, und wenn, dann nur in Teilaspekten und unter Einsatz von Stationen gleicher Bauart sowie beschränkt auf LANs verwendet. Das verteilte Betriebssystem muß idealerweise netzweit bekannte 'globale' Betriebsmittel definieren, deren Realisierung hohen Aufwand verursacht.

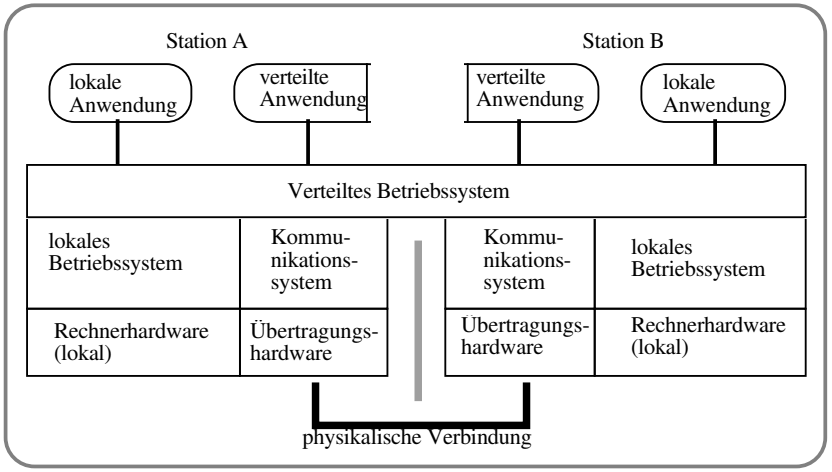

Bild 9.4: Verteiltes Betriebssystem

Weiterhin ist dieser Ansatz auch für sehr große Netze deshalb nicht geeignet, weil dort in der Regel Subnetze und auch einzelne Stationen in ihrer Konfiguration und Nutzung separaten Organisationen unterstehen (z.B. verschiedene Eigentümer), die nicht bereit sind, wesentliche Teile ihrer Verfügungsgewalt an das zentrale verteilte Betriebssystem abzutreten.

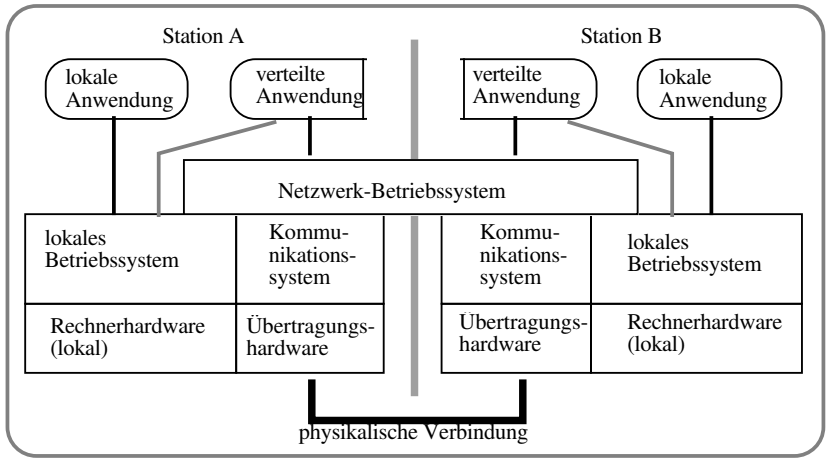

Bild 9.5: Netzwerk-Betriebssystem

9.3.1.4 Netzwerk-Betriebssysteme

Mit dem Ansatz des Netz-Betriebssystems wird im Vergleich zum verteilten Betriebssystem vom umfassenden Zuständigkeits- und Homogenisierungsanspruch abgerückt. Bild 9.5 skizziert die hier gegebene Struktur.

Lokale Anwendungen finden ihre gewohnte Umgebung. Verteilte Anwendungen finden, zumindest was die Interaktionsmöglichkeiten mit entfernten Partnerkomponenten betrifft, eine homogenisierte Umgebung. Bei Systemen, die sich auf Rechner des gleichen Herstellers beziehen (z.B. DECnet für DEC-Rechner, SNA für IBM-Rechner), ist diese Unterstützung sehr umfangreich (Interprozeßkommunikation, netzweit bekannte (aber in der Regel ortsfeste) Dateien, netzweite Administrationsfunktionen). Bei Systemen, die speziell auf Offenheit und damit unterschiedlichste Rechner und lokale Betriebssysteme zugeschnitten sind, beschränkt sich diese Unterstützung oft auf wenige Funktionen (abgesetzter Zugriff zu Dateien und Geräten, elektronische Post).

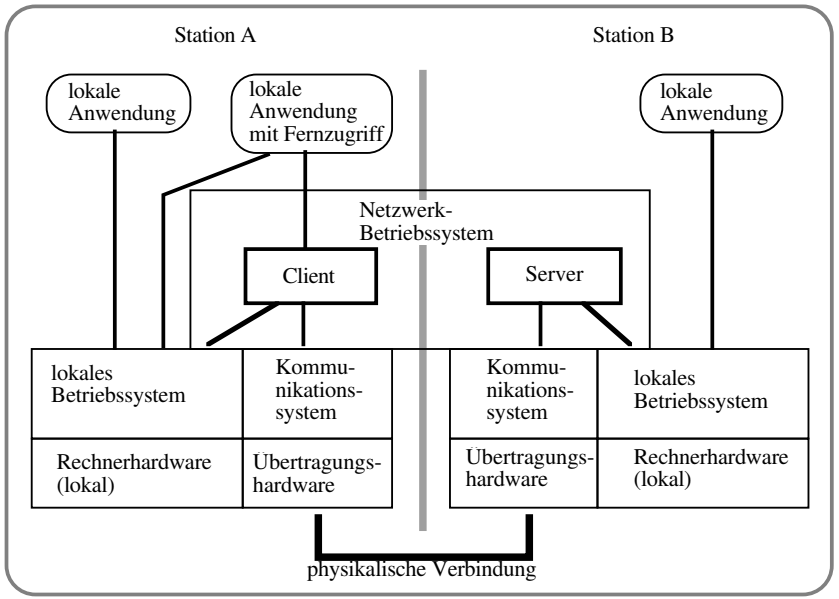

Bild 9.6: Sammlung produktiver Dienste, Client-Server-Modell

9.3.1.5 Netzwerk-Betriebssysteme nach Client-Server-Modell

Als derzeit beim Einsatz offener Rechnernetze favorisierte Struktur hat sich eine gegenüber unterschiedlichen Stationsbauarten und gegenüber dem im Netz angebotenen Dienstleistungsspektrum sehr flexible Variante des Netzwerkbetriebssystems herauskristallisiert, das Netzwerkbetriebssystem als Sammlung voneinander weitgehend unabhängiger produktiver Dienste. Sie werden im Netz nach dem sogenannten Client-Server-Modell realisiert.

Hier können die einzelnen lokalen Komponenten implementierungsunabhängig im Sinne der Offenheit genormt werden (durch sogenannte Service Access Protokolle in Verbindung mit der Spezifikation erbrachter Dienstleistungen). Ferner kann der Funktionsumfang des Netzwerkbetriebssystems von Station zu Station freizügig durch Aufnahme neuer Dienste erweitert werden.

Bild 9.6 zeigt diese Struktur. Je produktivem, netzweit anzubietendem Dienst (z.B. Bedienschnittstelle / abgesetztes Bediengerät, Dateispeicher, Datenbank, Archiv, elektronischer Postmeister, Drucker, Verzeichnis, Nutzerverwaltung / Autorisierung / Authentifikation) werden eine Server-Komponente und für alle anderen Stationen eine Client-Komponente benötigt. Clienten kommunizieren mit einem Server gemäß einem festgelegten Service Access Protokoll, das als ISO/OSI-(7)-Protokoll anzusehen ist. Der Server hat die Aufgabe, den produktiven Dienst zu erbringen. Ein Client vertritt den Server auf einer Station gegenüber den Anwenderprozessen, indem er dort die produktiven Operationen des Servers stellvertretend anbietet.

Dieses Prinzip findet inzwischen bei vielen im Bürobereich installierten Rechnernetzen Anwendung. Auch die gegenwärtig auf dem Markt angebotenen Unix-Workstations entsprechen ihm. Es wird auch von der Normung im ISO/OSI-Bereich unterstützt (vgl. CASE-ROSE).

Abschließend soll zum Begriff 'Client' noch angemerkt werden, daß er oft unterschiedlich ausgelegt wird - auch in diesem Skript. In Kap. 9.2.2.2 'Clients und Server' steht Client für einen Anwendungsprozeß, der in Kundenrolle auftritt. Im obigen Bild 9.6 steht Client für einen Systemprozeß, der die Serveroperationen lokal verfügbar macht.
Wenn man unter Client also einen dienstnehmenden Anwendungsprozeß verstehen will, ersetze man in Bild 9.6 die Beschriftung 'Client' durch 'Clientanschluß' und die 'lokale Anwendung mit Fernzugriff' wird zum 'Client'.
Auch soll angemerkt werden, daß der Clientanschluß nicht unbedingt durch einen Prozeß erfolgen muß. Oft reicht eine Prozedursammlung, die dem Anwendungsprozeß dazugebunden wird.

9.3.1.6 Verteilte Datenbanken

Dem Paradigma 'Datenbank und kooperierende Prozesse' kann man sehr nahe kommen, wenn im Rechnernetz ein Datenbankverwaltungssystem installiert wird, das Anwendungsprozessen den Zugriff auf die verwaltete Datenbasis unabhängig vom Ort der Anwendungsprozesse gestattet.
Hierzu sind grob zwei Varianten möglich:

- Zentrale Datenbank mit Fernzugriff,
- Verteilte Datenbank.

Es soll darauf hingewiesen werden, daß bei beiden Varianten das Paradigma 'Datenbank und kooperierende Prozesse' dasselbe ist. Beim logischen Entwurf einer Anwendung kann - abgesehen von Leistungsaspekten - verdeckt bleiben, auf welche Weise die Datenbasis zur Verfügung gestellt wird.

Bei der zentralen Datenbank mit Fernzugriff gibt es eine ausgezeichnete Station, die die Datenbasis und das Datenbankverwaltungssystem trägt. Um Prozessen anderer Stationen den Zugriff zu ermöglichen, wird zusätzlich ein Zugriffserver installiert (vgl. Netzbetriebssysteme nach Client-Server-Modell). Entfernte Anwendungsprozesse sind Clients.
Die Normung unterstützt diese Variante durch den RDBA-Standard (Remote Database Access). Er setzt die Verwendung der standardiserten Anfragesprache SQL voraus und legt ein 'Remote Operations'-basiertes Service Access Protokoll fest.

Bei einer verteilten Datenbank ist die Datenbasis und das Datenbankverwaltungssystem selbst verteilt. Die Verteilung der Daten kann durch Partitionierung (Aufteilung eines Datenbestandes in disjunkten Teilen auf verschiedene Rechner), Replikation (dieselben Daten werden in Kopie mehrfach an verschiedenen Stationen gespeichert) sowie Kombinationen aus beiden Möglichkeiten erfolgen.

Die Verwaltung nebenläufiger Transaktionen über verteilten und u.U. in Kopien mehrfach gespeicherten Daten kann sehr kommunikationsaufwendig werden. Wird neben den Daten auch die Transaktionskontrolle verteilt, kommt weiterer Kommunikationsaufwand hinzu. Deshalb gibt es bei derzeit marktverfügbaren verteilten Datenbanksystemen oft Restriktionen. So werden manchmal z.B. nur Transaktionen unterstützt, die die an einer einzigen Station verwalteten Daten zugreifen.

Während die zentrale Datenbank mit Fernzugriff - sie geht sehr einfach aus einer Zentralrechner-Datenbank hervor - in Reinform auch in der Praxis häufig vorkommt, gibt es die 'über's ganze Netz verteilte Datenbank' kaum. Man wird weniger leistungsfähige und weniger zuverlässige Stationen nicht als Träger global relevanter Daten und Funktionen einsetzen. Zur vereinfachten Administration wird man auch Stationen davon ausschließen, die organisatorisch abgesetzt sind. Solchen Rechnern erlaubt man dann den abgesetzten Zugriff auf eine ihnen zentral erscheinende - aber in einem kleineren Kreis von Stationen verteilt realisierte - Datenbank. Auf diese Art entsteht eine Mischform.

9.3.2 Beispiele

Zur Verdeutlichung soll im folgenden kurz auf Beispiele zur Systemunterstützung eingegangen werden.

9.3.2.1 Unix-Sockets

Unix-Workstations werden inzwischen wohl generell als netzwerkfähige Stationen ausgeliefert. Hardware-seitig ist ein Rechner mit LAN-Anschluß ausgestattet. Software-seitig sind die ISO/OSI-Schichten des Transportsystems innerhalb des Betriebssystems implementiert (dabei werden jedoch zur Zeit in der Schicht 4 noch nicht ISO/OSI-Protokolle sondern die Internet-Protokolle TCP und UDP eingesetzt; damit steht ein verbindungsorientierter TCP-basierter und ein UDP-basierter Datagramm-Transportdienst zur Verfügung).
Zur Implementierung der Schnittstelle zwischen Anwendungsprozessen und dem Transportdienst werden die sogenannten BSD 4.3 Sockets eingesetzt (verschiedene Hersteller streben hier allerdings schon an - wie z.B. Sun mit dem TLI Transport-Level-Interface -, daß unterschiedliche lokale Schnittstellenmechanismen - je nach Art des vorhandenen Transportsystems - eingesetzt werden können). Sie verdecken das Transportsystem.
Sockets sind lokale Interprozeßkommunikationsobjekte, die als puffernde Ports charakterisiert werden können. Sie werden lokal durch eine Socketnummer identifiziert. Wollen zwei Prozesse kommunizieren, muß jeder der beiden über einen eigenen Socket verfügen. Bei Datagramm-Kommunikation kann ein Prozeß über seinen Port unter Angabe der Adresse des Zielports ein Datenpaket versenden. Bei verbindungsorientierter Kommunikation werden zwei Sockets eigens als Endpunkte der Verbindung eingerichtet. Der Verbindungsbetrieb erfolgt nicht im Paket- sondern im Bytestrom. Das Senden und der Empfang von Daten kann über die Standard-E/A-Operationen des Betriebssystems abgewickelt werden, Sonderoperationen stehen zum Einrichten von Sockets, zum Binden einer Adresse an einen Socket, zum Verbindungsauf- und abbau, zum Mehrfachwarten / Selektieren von aktuell ereignisbehafteten Sockets und auch zum Datentransfer zur Verfügung.
Beim Einrichten eines Sockets kann man per Parameter entscheiden, ob er zur Kommunikation zwischen lokalen Prozessen oder zur Telekommunikation eingesetzt werden soll. Entsprechend muß auch die Wahl des Adreßraums vorgenommen werden. Bei Telekommunikationssockets werden zur Zeit üblicherweise Internet-Adressen verwendet (Internet-Stationsadresse & lokale Socketnummer).

9.3.2.2 Sun-RPC

Sun-RPC ist ein auf Sun-Stations zur Verfügung stehender Verbund aus Bibliothekselementen, Werkzeugen und Konzepten, der die komfortable Programmierung von Client-Server-Systemen und deren Kommunikation per Prozedurfernaufruf unterstützt.

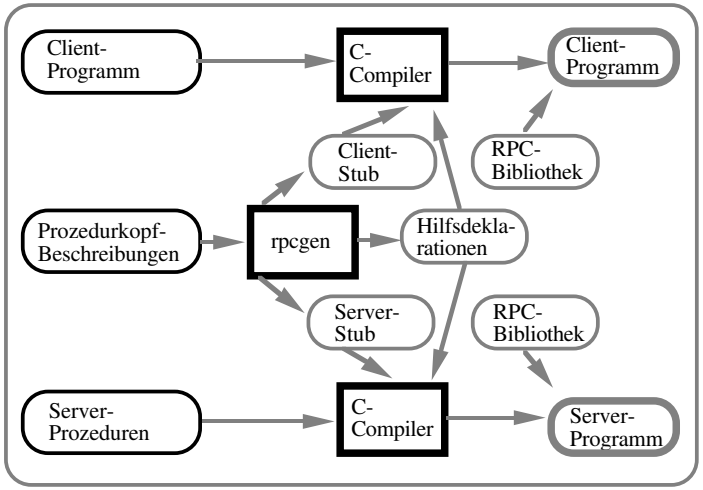

Bild 9.7: Sun-RPC

Ein Server-Programm ist eine Sammlung von Prozeduren. Ein Client-Programm ein Hauptprogramm, aus dem Serverprozeduren aufgerufen werden. Beide werden im wesentlichen so programmiert, als ob man lokal ein Programm durch Zusammenbinden von Hauptprogramm und Prozeduren erzeugen wollte.

Zur Verteilung von Client und Server muß dem Client ein sogenannter Client-Stub (Client-Anschluß) zugebunden werden. Die Server-Prozeduren werden um ein Hauptprogramm, den Server-Stub ergänzt. In den Stubs werden die Kommunikationsaspekte der ISO/OSI-Schichten 5..7 abgewickelt. Sie setzen auf dem Transportsystem auf.

Zur komfortablen Programmerstellung stehen eine Library mit Stub-Grundfunktionen und ein Stub-Generator (rpcgen) zur Verfügung. Er wird mit einer Beschreibung der Syntax der Prozeduren versorgt (sie ist vergleichbar mit einer in ASN.1 gehaltenen und das Makro OPERATION verwendenden Spezifikation von ROSE-Operations). Bild 9.7 skizziert den Ablauf.

Eine weiterer Aspekt betrifft die Problematik des 'Naming and Binding / Verzeichnisdienst'. Wenn ein Server in einer Station installiert wird, wird ihm dort eine Transport-Adresse zugeordnet, die u.U. bei der Installation vom Betriebssystem frei gewählt wird. Im Client-Programm kann also keine Konstante als Zieladresse für RPC-Aufrufe stehen.
Lösung ist Binden zur Laufzeit. Dem Server wird beim Programmentwurf statisch ein logischer Name zugeordnet. Ist der Server installiert und kennt er seine Transportadresse / Socketnummer, trägt er das Paar 'logischer Name, Adresse' in ein Verzeichnis ein (das von der Anwendung 'Portmapper' verwaltet wird). Ein Client, der die Adresse eines Servers noch nicht kennt, kann sie per Anfrage 'Liefere Adresse zu logischem Namen' beim Verzeichnisdienst erfahren.

9.3.2.3 OSF: DCE

DCE (Distributed Computing Environment) ist eine Initiative der OSF (Open Software Foundation, sie wurde 1988 von Hard- und Software-Herstellern, Nutzern, Behörden und Forschungseinrichtungen als Non-Profit-Organisation mit dem Ziel gegründet, eine herstellerneutrale Rechenumgebung zu definieren und aus Produkten zusammenzustellen bzw. entwickeln zu lassen), die darauf ausgerichtet ist, in den 90er Jahren eine Umgebung für Rechnernetz-Anwendungen zur Verfügung zu stellen, die Netz-Anwendungen an sich und insbesondere 'offene' Netz-Anwendungen in dem Sinne der Integration heterogener Stationen zu unterstützt.

Die Strategie ist dabei auf eine schnelle Verfügbarkeit der Umgebung für den Nutzer ausgerichtet. So sollten die wesentlichen Komponenten schon 1992 für die wichtigsten Stationstypen marktverfügbar sein.
Deshalb werden auf konzeptioneller Seite erprobte und gängige Ansätze zugrundegelegt: Client-Server-Architektur für Anwendungen, Client-Server-Netzwerkbetriebssystem-Ansatz zur Architektur der Umgebung, Desktop-Computing im Bereich von Arbeitsplatzrechnern.
Das 'Produkt DCE' soll aus einer Menge einzelner, bestimmte Unterstützungsaufgaben abdeckenden Produkten bestehen, die von unterschiedlichen Herstellern am Markt angeboten werden. Bevor OSF-angeregte Neuentwicklungen in's Auge gefaßt werden, können sich im Rahmen einer Ausschreibung existierende Produkte um die Aufnahme bewerben.
Aufgrund dieser Strategie spiegelt DCE den marktverfügbaren Stand der Technik zur Unterstützung verteilter Anwendungen wider und als wesentlicher DCE-Beitrag kann die gegenseitige Abstimmung und Integration zu einer herstellerunabhängigen Umgebung gelten. Die frühe Marktverfügbarkeit und die Herstellerunabhängigkeit lassen vermuten, daß DCE große Akzeptanz erfahren und von überragender praktischer Bedeutung sein wird.

DCE gliedert sich konzeptionell in eine Menge von Diensten zweier Klassen:

- Fundamental Services
  Dienste zur grundlegenden technischen Unterstützung verteilter Systeme, sowohl Anwendungen als auch Implementierungen von End-Nutzer-Diensten können darauf aufsetzen.
  - Remote Procedure Call (ähnlich Sun-RPC, aber erweitert)
  - Directory / Naming Service (X.500 Directory Implementierung für WAN, Zellen-Verzeichnis für Subnetze, Programme zur Administration und Pflege der Verzeichnisse)
  - Security Services (Produkte zur Authentifizierung und Autorisierung, ähnlich MIT-Athena 'Kerberos')
  - Threads (Lokale Unterstützung von multi-threaded Prozessen bei Multiprozessor-Stationen)
  - Time (Netzweite koordinierte aktuelle Zeit)
- Data Sharing Services
  Dienste zur Unterstützung des (dateien-basierten) Datenaustauschs zwischen Stationen und Anwendungen
  - Verteiltes Dateisystem (Netzwerk-Dateisystem und Unterstützung 'plattenloser' Stationen)
  - Personal Computer Integration (Schnittstellen zwischen PC-Anwendungen und workstation-basierten DCE-Anwendungen)

Bild 9.8 zeigt die wesentlichen DCE-Komponenten und ihre Anordnung zur Unterstützung von Anwendungen. Dunkel umrandet sind die fundamentalen Dienste, hell umrandet direkt die Endnutzer unterstützende Datenaustauschdienste. In beiden Gruppen sind Platzhalter für zukünftig aufzunehmende Dienste vorgesehen (z.B. Abrechnung, Mitteilungssystem).

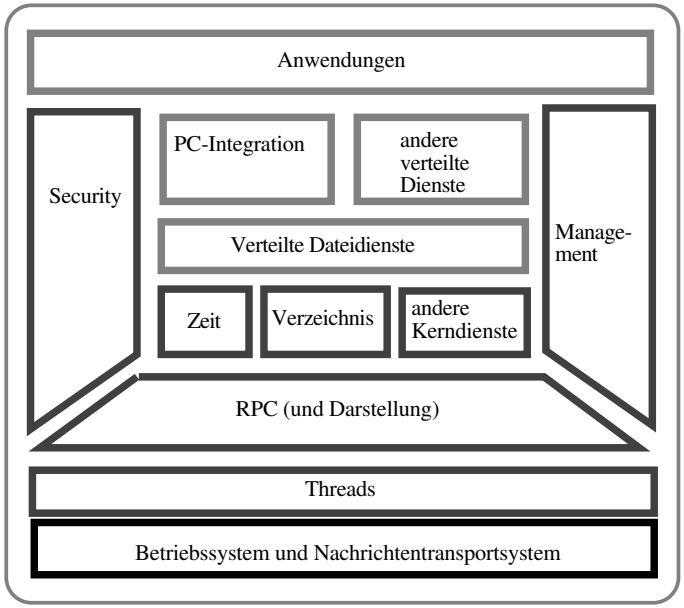

Bild 9.8: DCE

Für die etwas weitere Zukunft sind angestrebt: Definition eines OSF/1-Betriebssystems, Produkte zur objektorientierten Programmierung, zur Transaktionsverarbeitung, zur Systemverwaltung und zum Drucken.

9.3.2.4 ANSA

Ergebnisse des europäischen Forschungsprojekts ANSA (Advanced System Network Architecture) und des Nachfolgeprojekts ISA (Integrated System Architecture) sind inzwischen auf dem Markt erhältlich. Ähnlich zum DCE handelt es sich um eine vereinheitlichte Plattform für verteilte Anwendungen, die auf Betriebs- und Transportsystem aufbauend weiterführende Dienste anbietet. Ähnlich zum DCE bilden Threads und RPCs die erste Stufe grundlegender Dienste. Darüber finden sich bei ANSA aber Dienste, die auf die objekt-orientierte Anwendungsprogrammierung zugeschnitten sind:

- Notification Service,
- Factory,
- Trader,
- Node Manager.

Objekte sind Einheiten, die im Sinne eines Client-Server-Systems produktive Dienste realisieren. Eine Anwendung entspricht einer u.U. dynamisch veränderlichen Menge solcher Objekte in der dynamisch Client-Server-Beziehungen hergestellt, benutzt und freigegeben werden.
Der Notification Service erlaubt es - hauptsächlich Verwaltungsinformationen - in Form unbestätigter Datagramme zwischen Objekten auszutauschen.
Die Factory erzeugt Objekt-tragende Prozesse im Sinne der lokalen Betriebssysteme.

Der Trader übernimmt im Vergleich zum DCE die Aufgabe des Verzeichnisdienstes, das dynamische Binden von Clients und Servern. Dabei wird ein über das Binden hinausgehendes Konzept, das sog. Trading angewandt.
Der Node Manager verwaltet die Objekte eines Knotens. So ist z.B. das Einrichten eines einen neuen Servers eine Dienstleistung des Node Managers. Er greift dazu auf die Factory zurück, um einen tragenden Betriebssystemprozeß zu erzeugen, und auf den Trader, um den neuen Server im System vermittelbar zu machen.
Das Konzept des Tradings (siehe z.B. O. Spaniol, C. Popien, B. Meyer: Dienste und Dienstvermittlung in Client/Server-Systemen, International Thomson Publishing, 1994) stellt gegenüber dem dynamischen Binden des DCE, bei dem symbolische Adressen (z.B. Server-Namen) zur Laufzeit anhand der Einträge eines Verzeichnisdienstes aufgelöst (z.B. auf aktuelle Server-Transportdienst-Adressen abgebildet) werden, eine weitergehende Flexibilisierung dar. Ein Client muß nun nicht mehr identifizierende Attribute (z.B. den Namen) eines Servers kennen, zu dem er in Beziehung treten will. Er wird vielmehr auch gar nicht darauf festgelegt sein, mit einem bestimmten Server in Beziehung zu treten. Der Client ist in der Regel lediglich daraufhin festgelegt, daß er nun einen bestimmten Typ von Dienstleistung in Anspruch nehmen möchte, der von verschiedenen Servern vielleicht sogar in unterschiedlichen Varianten angeboten wird. Er wendet sich dazu an den Trader, der in der Funktion eines Maklers auftritt und aus dem aktuellen Angebot geeignete Server vermittelt. Die Suche nach passenden Servern wird dadurch unterstützt, daß im Trader neben Server-Typ-Angaben und anderen Server-beschreibenden Attributen insbesondere auch die Namen der vom Server unterstützten Operationen (und in Zukunft auch Operationssignaturen) verwaltet werden. Die Server-Einträge werden darüberhinaus in einer hierarchischen Struktur, dem sog. Kontext-Baum, verwaltet, so daß navigierende Suche unterstützt wird.
In der Regel befinden sich mehrere Trader im System, die durch Export und Import von Server-Angeboten kooperieren können.

9.3.2.5 OMG OMA

OMA (Object Management Architecture) wird von einem umfangreichen Hard- und Software-Hersteller-Verbund der OMG (Object Management Group) vorgeschlagen. Es ist - wie ANSA - eine Plattform zur Unterstützung verteilter objektorientierter Anwendungen, deren zentrale Systembildungsfunktionen ebenfalls dem Trading-Konzept folgen. Das Trading wird hier vom ORB (Object Request Broker) durchgeführt und ist in der CORBA (Common Object Request Broker Architecture) festgelegt. Basisfunktionen für ORB und Anwendung werden Object Services genannt. Sie erlauben es, in Betriebssystem-unabhängiger Weise Objekte einzurichten und zu verwalten. Oberhalb des ORB sind Common Facilities vorgesehen, eine Sammlung allgemein verwendbarer Objekte.

9.4 Normung: ODP und DAF

Auch von Seite der Standardisierungsorganisationen, in der internationalen Standardisierungsorganisation ISO sowie beim internationalen Komitee der Postverwaltungen CCITT, sind Bestrebungen im Gange, nicht nur die Kommunikation verteilter Systeme sondern nun auch die verteilten Anwendungen selbst und ihre Unterstützung - jedenfalls in den Offenheitsaspekten - zu normen. Stichwörter sind hierzu 'ISO: ODP (Open Distributed Processing) mit SE-ODP (Support Environment for ODP) und CCITT: DAF (Distributed Applications Framework)'.

Auf ODP soll etwas näher eingegangen werden. Die Arbeiten dazu sind seit 1987 mit der Bildung einer 'ODP Ad hoc Gruppe' im Rahmen von ISO/OSI im Gange.

Ähnlich ISO/OSI ist vorgesehen, zunächst ein Referenzmodell (ODP-RM) zu definieren, in dessen Rahmen dann einzelne ODP-Standards eingeordnet werden können. Da es nun nicht mehr um 'simple' Kommunikation geht, ist das ODP-RM im Vergleich zum OSI-RM komplizierter. So ist man sich bewußt, daß dasselbe ODP-System je nach Standunkt von verschiedenen Blickwinkeln modelliert werden muß. Man unterscheidet:

- Standpunkt eines Unternehmens (Enterprise Viewpoint); hier sind die gloablen Ziele der verteilten Anwendung in Termen des Problembereichs zu erfassen.
- Standpunkt des Informationsflusses (Information Viewpoint); hier sieht man die Anwendung als Informationssystem auf hoher logischer Ebene. Es müssen die vorkommenden Informationen und ihr Fluß in Termen des Problemfelds erfaßt werden (Daten in logischer Bedeutung; Orte und Strukturen des Problemfelds, z.B. Abteilungen, Gruppen, Filialen).
- Standpunkt der problemnahen Anwendungsprogrammierung (Computation Viewpoint); hier soll die Struktur und Organisation der Anwendungsprogramme erfaßbar werden.
- Standpunkt der abstrakten Implementierung (Engineering Viewpoint); hier soll die interne Struktur des Systems (eingesetzte Betriebssysteme, Protokolle, Stationen, Zuordnung von Komponenten zu Stationen etc.) erfaßbar werden.
- Standpunkt der konkreten Implementierung (Technology Viewpoint); hier müssen die konkreten Bausteine eines Systems in ihrer detaillierten Auslegung beschrieben werden.

Das ODP-RM wird also aus 5 Modellwelten bestehen, in denen die Struktur und die Funktionen von ODP-Systemen unter unterschiedlichen Blickwinkeln dargestellt werden können. Die Arbeiten zur Definition des ODP-RM konzentrieren sich dabei auf die (für die Modellierung von Software-Produkten wesentlichen) Standpunkte Information, Computation und Engineering. Es wird aller Wahrscheinlichkeit nach objektorientierte Modellierung zum Einsatz kommen. Die flexible dynamische Konfiguration verteilter objektorientierter Anwendungen soll sich auf das unter ANSA beschriebene Konzept des Trading abstützen.

10. Verteilte Algorithmen

Das vorliegende letzte Kapitel des Skriptums befaßt sich mit dem Teil III, den verteilten Algorithmen, wie sie in abstrakter Sicht von den Anwendungsprozessen einer Rechnernetz-Anwendung ausgeführt werden. Dieses für den Entwurf von Rechnernetz-Anwendungen wichtige Thema soll hier allerdings nur sehr grob umrissen werden.

Es wird zunächst eine abstrakte Sicht der Basismaschine 'Rechnernetz' eingeführt, die als Maschine für verteilte Algorithmen geeignet ist. Anschließend sollen einige wenige Beispiele verteilter Algorithmen vorgestellt werden.

10.1 Basismaschine

In den Teilen I und II wurde die konkrete Infrastruktur von Rechnernetz-Anwendungen erörtert. Sie besitzt zahlreiche Implikationen für die Konstruktion einer Anwendung. Zur Behandlung verteilter Algorithmen soll nun eine abstrahierende Sichtweise eingenommen werden, die zwar noch die wesentlichen Besonderheiten der Rechnernetz-Maschine in einer Form erkennen läßt, die den Entwurf verteilungsfähiger Algorithmen ermöglicht, die aber Details des Netzes und der Telekommunikation im Netz - wie z.B. die sieben ISO/OSI-Schichten, Heterogenität von Stationen und Betriebssystemen, Administrations- und Installationsproblematik, konkrete Leistungs- und Kostenaspekte - verdeckt.

Es gibt mehrere Ansätze, solche abstrakten Basismaschinen für verteilte Algorithmen zu definieren (Man beachte auch den Querbezug zu den im letzten Kapitel unter 'Paradigmen zur Architektur verteilter Anwendungen' vorgestellten Modellen). Im folgenden soll - wie in vielen vom Rechnernetz nicht allzusehr sich entfernenden Vorschlägen - davon ausgegangen werden, daß ein Rechnernetz aus Knoten besteht, zwischen denen - allerdings unzuverlässige - Kommunikation, jedoch keine direkte Kooperation (im engeren Sinne, d.h. kein geteilter Speicher, kein gemeinsamer Takt) möglich ist.

Stationen und Prozesse

An jedem Knoten soll ein Verarbeitungsprozeß, ein Knotenprozeß, ausgeführt werden können. Jeder Knotenprozeß führt einen diesem Prozeß speziell zugeordneten sequentiellen Algorithmus, den Knotenalgorithmus, aus.
Ein Knotenalgorithmus ist über einer Menge lokaler prozeßprivater Variablen definiert und beschreibt die am zugeordneten Knoten stattfindende Folge der Ausführung von Basisoperationen, welche entweder lokale Aktionen auf diesen Variablen oder Nachrichtenaustauschoperationen durchführen. Ein Knotenalgorithmus ist also ein sequentieller Algorithmus im traditionellen Sinn, in den zusätzlich Operationen zur Kommunikation mit anderen Prozessen eingestreut sind.
Ein verteilter Algorithmus ist nun eine endliche Menge von Knotenalgorithmen, die zueinander nebenläufig ausgeführt werden. Die Knotenalgorithmen sollen nur Kommunikation mit Knotenprozessen desselben verteilten Algorithmus vorsehen.
Damit beschreibt ein verteilter Algorithmus ein statisch definiertes geschlossenes System von endlich vielen miteinander kommunizierenden Prozessen. Jeder Prozeß ist ein Knotenprozeß und exklusiv an

genau einer - seiner - Station fixiert. 'Prozeß' ist deshalb gleichbedeutend mit 'Ablauf an einer Station'.

Kommunikation

Die Kommunikationsmöglichkeiten zwischen den Knotenprozessen sollen Telekommunikation widerspiegeln. Hier ist der Nachrichtenaustausch typisch, Rendezvous-Mechanismen werden wegen der engeren zeitlichen Kopplung (und des hinter ihrer Realisierung per Nachrichtenaustausch stehenden Aufwands) seltener zugrundegelegt.
Interessanter wäre es dagegen, verschiedene Formen des Nachrichtenaustauschs (verbindungsorientiert, Datagramm-Austausch; bestätigt, unbestätigt, mit Fristsetzung und Quittung a la RTSE) zu berücksichtigen, da sie starke Leistungs- und Kostenauswirkungen haben (und im Fall von RTSE auch besondere Eigenschaften funktioneller Natur). Hier möchten wir uns aber auf den einfachen unbestätigten Datagramm-Austausch zurückziehen, da wir auf Algorithmenebene von Leistung und realen Kosten (und auch von der absoluten Länge von Zeitdauern) abstrahieren wollen.
Weiterer Spielraum ist durch die an einem Nachrichtenaustausch beteiligte Partneranzahl gegeben. Rundruf- und Umlaufvorgänge sind zur Zeit teilweise in LANs schon mit sehr interessanten Leistungs- und Kosteneigenschaften realisierbar. Es ist zu erwarten, daß solche Dienstleistungstypen an Bedeutung stark zunehmen werden (Standardisierung, Unterstützung auch über LANs hinaus). Um das Modell einfach zu halten, werden wir aber auch hier vom einfachsten, nämlich der Zwei-Partner-Kommunikation, ausgehen.
Die Unzuverlässigkeit der Telekommunikation darf auch auf Algorithmenebene nicht ganz verdeckt werden (sonst würde man statt verteilten Algorithmen nur parallele Algorithmen entwerfen). Aber von Fehlern, die mit untergeordneten Protokollmechanismen abgefangen werden können, soll abstrahiert werden (d.h. Duplikate, Phantomnachrichten und Reihenfolgevertauschungen von Nachrichten mit gleichem Sender und gleichem Empfänger werden nicht berücksichtigt). Verfälschung, Verlust und Verbindungsabbruch gehen indirekt dadurch ein, daß die Dauer einer Übertragung nicht vorhergesagt werden kann. Verluste kann man, wenn man untergeordnete Wiederholungen und eine gewisse Fairness des Kommunikationssystems voraussetzt, ganz ausschließen: Jede Nachrichtenübertragung ist dann in endlicher (aber nicht vorhersagbarer und nicht nach oben beschränkbarer) Zeit erfolgreich. In einer zweiten Variante werden Verluste zugelassen und gehen durch die Laufzeit 'unendlich' in die Betrachtung ein.
Im Fazit sollen zwei Anweisungen zur Verwendung in den Programmen der Knotenprozesse zur Verfügung stehen: SEND (Empfänger, Nachricht) und RECEIVE (VAR Sender, VAR Nachricht). SEND sendet eine Nachricht an einen angegebenen Empfängerknoten. RECEIVE wartet auf die nächste anstehende Nachricht und nimmt sie entgegen. Die Laufzeit von Nachrichten ist unbekannt, in der Variante 'ohne Verluste' aber endlich, in der Variante 'mit Verlusten' u.U. auch unendlich. Reihenfolgenvertauschungen können sich durch differierende Laufzeiten einstellen. Sendet ein Sender mehrere Nachrichten an denselben Empfänger, wird (wenn nichts anderes gesagt wird) davon ausgegangen, daß keine Reihenfolgevertauschung erfolgt.

Stationsausfälle

Stationen im Rechnernetz können unabhängig voneinander ausfallen oder fehlerhaft arbeiten. Dies ist - insbesondere beim Entwurf langlebiger Systeme - zu berücksichtigen. Allerdings öffnet sich damit ein umfassender Problembereich, der Entwurf fehlertoleranter verteilter Algorithmen, der hier nicht weiter ausgeführt werden soll. Wir nehmen (vereinfachend) an, daß Stationsfehler und Stationsausfälle mit untergeordneten Mitteln so behandelt werden, daß sie auf der Ebene des verteilten Algorithmus nicht mehr berücksichtigt werden müssen.
Eine Station soll exakt die Anweisungen des zugeordneten Knotenprogramms ausführen. Für die Ausführungszeit einzelner Operationen nehmen wir nur an, daß sie endlich ist, und lassen so untergeordneten Mechanismen die Möglichkeit aufgetretene Fehler zu beheben (im Falle einer RECEIVE-Anweisung können sich durch unendliches Warten auf nicht-gesendete bzw. verlorengegangene Nachrichten auch unendliche Ausführungszeiten ergeben).

Funktionale Betrachtung - Korrektheit

Bei Beschränkung der Betrachtung eines Algorithmus auf seine Funktionalität wird man von Leistungs- und Aufwandsaspekten abstrahieren. Wie bei der Korrektheit sequentieller Algorithmen interessiert man sich nur dafür, ob der Algorithmus - ausgehend von einem aktuellen Zustand - nach endlicher Zeit das System in einen per Spezifikation gewünschten Zustand überführen wird.
Bei sequentiellen Algorithmen trennt man diese 'totale Korrektheit' in die beiden Eigenschaften 'partielle Korrektheit (der Algorithmus wird, wenn er terminiert, einen spezifikationskonformen Zustand herbeigeführt haben)' und die 'Terminierung (der Algorithmus wird nach endlicher Zeit auch zum Erfolg kommen)'.
Bei nebenläufigen und verteilten Algorithmen ist ebenfalls eine analoge Separierung zweier entsprechender Aspekte üblich. Allerdings müssen sie, da das Gesamtsystem u.U. nie terminieren wird und die Korrektheit sich nicht nur auf Endzustände beziehen kann, etwas verallgemeinert werden:

- Safety - Sicherheit; hier ist es von Interesse, daß das System, wenn es voranschreitet, nur erlaubte Zustände einnimmt (Beispiel: ein Aufzug darf nie in Stock 5 halten, aber die Etagentür von Stock 4 öffnen). Safety entspricht bei sequentiellen Algorithmen die partielle Korrektheit.
- Liveness - Lebendigkeit; hier ist es von Interesse, daß das System auch tatsächlich voranschreitet (Beispiel: wenn bei einem Aufzug ein Zustand vorliegt, in dem ein Ruf-Knopf gedrückt ist, wird in endlicher Zeit ein Zustand eingenommen werden, in dem die Kabine im entsprechenden Stockwerk hält). Liveness entspricht bei sequentiellen Algorithmen die Terminierung.
  Die interessierenden Lebendigkeitseigenschaften nebenläufiger und verteilter Algorithmen können in der Regel aus der sogenannten Fairness elementarer Operationen abgeleitet werden.
  - Fairness einer Operation; man unterscheidet die schwache und die starke Fairness.
    Schwache Fairness einer Operation Op liegt vor, wenn zu jedem Zeitpunkt x gilt, daß Op in einem zu x zukünftigen Zeitpunkt ausgeführt werden wird, es sei denn, das System erreicht ab x unendlich oft Zustände, in denen die Ausführung von Op verboten ist.
    Starke Fairness einer Operation Op liegt vor, wenn zu jedem Zeitpunkt x gilt, daß Op in einem zu x zukünftigen Zeitpunkt ausgeführt werden wird, es sei denn, ab einem Zeitpunkt y>=x erreicht das System nur noch Zustände, in denen die Ausführung von Op verboten ist.

Ausführungszeiten bei funktionaler Betrachtung

Bei einer auf Sicherheitsaspekte beschränkten funktionalen Betrachtung kann man von einer Bewertung der Ausführungszeiten ganz absehen. Lokale Aktionen und ebenso Nachrichtentransfers benötigen eine unbestimmte Zeitdauer >0.
Bei einer Lebendigkeitsaspekte berücksichtigenden funktionalen Betrachtung kann man ebenfalls von einer exakten Bewertung der Ausführungszeiten absehen. Allerdings muß man einfließen lassen, ob eine Aktion überhaupt erfolgreich abgeschlossen wird oder nicht. Wir nehmen an, daß Aktionen eine unbestimmte Zeitdauer >0 aber <∞ benötigen, bei Nachrichtentransfers lassen wir in der Variante 'mit möglichen Nachrichtenverlusten' auch die Zeitdauer ∞ zu.

Nun kann man bei lokalen Aktionen vereinfachend annehmen, daß sie in der Zeitdauer 0 ausgeführt werden, ohne daß dadurch funktionale Betrachtungen verfälscht werden. In dieser 'Atommodell' genannten Vereinfachung benötigt ein Prozeß für lokale Operationen keine Ausführungszeit. So wird er eine Sequenz lokaler Aktionen atomar ausführen und -quasi sofort - die nächste Wartestelle (die nächste RECEIVE-Anweisung) erreichen. Hier verbraucht er natürlich Zeit beim Warten. Wird der RECEIVE-Aufruf durch einen Nachrichtentransfer befriedigt, beginnt unmittelbar die nächste Sequenz lokaler Aktionen. Die in der Realität für lokale Aktionen auftretende Verzögerung wird im Atommodell somit den Nachrichtentransfers mit dazugeschlagen.
Die Abstraktion vom Zeitverbrauch lokaler Aktionen führt in unserem Modell aus folgenden Gründen nicht zu Verfälschungen funktionaler Betrachtungen. Es kommt nur auf relative Zeitbeziehungen an,

die absolute Zeitdauer von Aktionen ist nicht relevant. Lokale Aktionen werden nach Voraussetzung in endlicher Zeit abgeschlossen. Lokale Aktionen greifen nur auf stationsprivate Betriebsmittel zu.

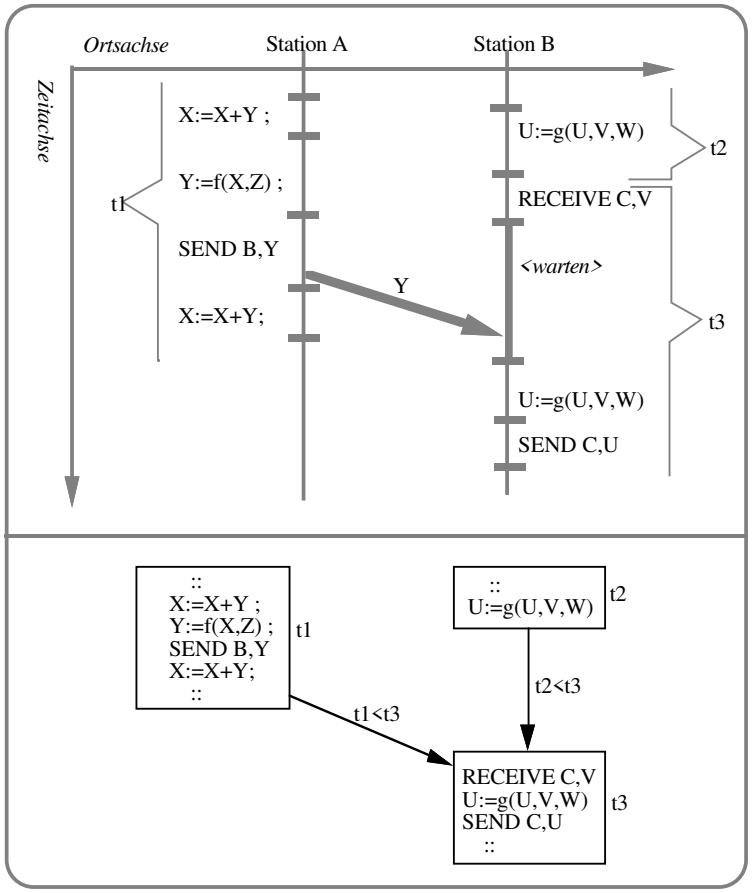

Bild 10.1: Beispiel zum Atommodell

Bild 10.1 stellt zur Verdeutlichung des Atommodells beispielhaft einen möglichen realen Ablauf seinem Bild im Atommodell gegenüber.

Aufwandsbetrachtungen

Aufwandsbetrachtungen (Komplexitätsbetrachtungen) von Algorithmen sind von sequentiellen Algorithmen her aus dem Grundstudium bekannt. Motivation und Nutzen der Aufwandsbetrachtungen verteilter Algorithmen entsprechen denen von sequentiellen und sollen hier nicht ausgeführt werden. Es soll nur kurz auf Unterschiede eingegangen werden.

Bei sequentiellen Algorithmen interessierten zwei Aufwandsgrößen: Zeitaufwand als Abstraktion des Rechenzeitbedarfs und Platzaufwand als Abstraktion des Arbeitsspeicherbedarfs.

Ein verteilter Algorithmus besteht (in unserer Sicht) aus einer Menge sequentieller Algorithmen (den Knotenalgorithmen) und für die lokalen Aktionsfolgen eines Knotens können deshalb mit den bekannten Mitteln Zeit- und Platzaufwandsbetrachtungen durchgeführt werden. Unter Zugrundelegung dieser lokalen Aufwandsgrößen und Berücksichtigung der globalen Abläufe können die Aufwandsgrößen 'Zeit' und 'Platz' für den verteilten Algorithmus bestimmt werden. Bei der Zeitbetrachtung kann die Parallelität berücksichtigt werden. Zeit- und Platzaufwand vernachlässigen aber den für verteilte Algorithmen ebenfalls neu hinzukommenden Aspekt der Kommunikation.

Werden SEND/RECEIVE-Transfers per Telekommunikation abgewickelt und geht man von nicht allzu aufwendigen lokalen Verarbeitungsvorgängen aus, dann wird bei einer Implementierung der Kommunikationsaufwand dominieren: Nachrichtenlaufzeiten betragen ein Vielfaches lokaler Bearbeitungszeiten, Kommunikationsdienstleistungen sind mit in's Gewicht fallenden Kosten (Betriebsmittelbedarf an Stationen, Gebühren) verbunden.

Es werden deshalb Aufwandsgrößen zur Abschätzung des Kommunikationsaufwands eingeführt und der Kommunikationsaufwand geht dominierend in die Beurteilung eines verteilten Algorithmus ein. Üblich sind dazu Abschätzungen der Anzahl von Nachrichtentransfers und Abschätzungen der Länge ausgetauschter Nachrichten.
Bei realitätsnäheren Betrachtungen werden Transfers bzw. übertragene Dateneinheiten auch nach Entfernung (bzw. Kosten der betroffenen Übertragungsstrecke) gewichtet, um deren Unterschiede nicht ganz zu verlieren.
Es soll darauf hingewiesen werden, daß Aufwandsbetrachtungen zwar generelle Aussagen über die Qualität eines Algorithmus zulassen, daß sie aber detaillierte Leistungsuntersuchungen praktischer Rechnernetz-Anwendungen nicht erübrigen. Die Implementierung eines abstrakt sehr effizienten Algorithmus kann bei den praktisch auftretenden Problemfällen von geringerer Leistungsfähigkeit sein als die eines ineffizienten.

10.2 Beispiele

Wir haben im Verlauf der Vorlesung schon einige Beispiele verteilter Algorithmen kennengelernt. Kommunikationsprotokolle sind verteilte Algorithmen, nämlich solche, deren Zweck die Erbringung eines Kommunikationsdienstes ist. Im vorliegenden Kapitel möchten wir uns aber auf verteilte Algorithmen konzentrieren, die von den Anwendungsprozessen einer Rechnernetz-Anwendung ausgeführt werden, die also über die Kommunikation hinausgehende produktive Ziele erfüllen sollen. Auch hierzu wurde schon ein Beispiel besprochen, nämlich das Zwei-Phasen-Commit-Protokoll (vgl. CCR-Erläuterung). Es handelte sich dabei um einen grundlegenden Algorithmus, der recht allgemein verteilte Transaktionen unterstützt. In einer speziellen Anwendung wird man ihn nicht in Reinform isoliert finden. Dort wird er in Spezialisierungen und in Kombination mit anderen Mechanismen vorkommen. Der Wunsch, nun Algorithmen zu sehen, die direkt produktiven Zwecken dienen und die einer Rechnernetz-Anwendung insgesamt zugrundeliegen (z.B. ein Algorithmus zur verteilten Flugbuchung), liegt nahe, kann aber aus Umfangsgründen nicht erfüllt werden. Die folgend skizzierten Algorithmen sind vielmehr ebenfalls grundlegender Natur und können zur Kontrolle der produktiven Aktivitäten einer Anwendung eingesetzt werden.

10.2.1 Echo-Algorithmus

Gegeben sei eine Menge von Stationen in der Konfiguration eines teilvermaschten Netzes, d.h. eine Station hat eine Menge von Nachbarn - nur mit ihnen könne sie per SEND/RECEIVE Nachrichten austauschen -, die nicht alle übrigen Stationen umfaßt.

Das Problem sei, daß eine Station eine Nachricht an alle anderen Stationen versenden will und von allen eine Quittung zurückkommen soll (bestätigter Broadcast).

Die grobe Funktionsweise des sogenannten Echo-Algorithmus nach Chang ergibt sich nach:

- Jede Station gibt eine empfangene Nachricht an alle ihre übrigen Nachbarn weiter (Flooding), wenn sie die Nachricht erstmalig empfängt.
- Stationen mit nur einem Nachbar (Blätter) senden sofort eine Quittung an den Absender.
- Sonstige Stationen senden zu einer erstmalig empfangenen Nachricht dann eine Quittung zurück, wenn von allen Nachbarn Quittungen oder Nachrichten eingetroffen sind.
- Ein Initiator sieht den Broadcast als bestätigt an, wenn von allen seinen Nachbarn Quittungen oder Nachrichten eingetroffen sind.

Vom Initiator als Wurzel ausgehend wird so ein alle Stationen erfassender Baum aufgespannt. Da der wiederholte Empfang derselben Nachricht Quittungscharakter hat, werden nur 2*n Nachrichtentransfers benötigt um ein n Kanten umfassendes Netz zu erkunden.

Jeder Stationsprozeß verfährt nach folgender Vorschrift:

```
α:   if Initiator-Prozeß (* Wurzel *)
         then
              Initiator:=true ; Engaged:= true ; N:=0 ;
              for each Ncb of Nachbarn do  SEND(Ncb,Nachricht) ;
         else (* Nicht-Wurzel *)
              Initiator:=false ; Engaged:=false ; N:=0 ;
     loop
         RECEIVE (Sender,Data) ;
         if type(Data)=Nachricht
              then
                   if not Engaged
                        then  Engaged:=true ; Master:=Sender ;
                              for each Ncb of Nachbarn\{Master} do
                                   SEND(Ncb,Data) ;
         (* Aktionen für Nachrichten und Quittungen: *)
         N:=N+1 ;
         if N=|Nachbarn|
              then
                   if Initiator  then Broadcast successful!!
                                 else SEND(Master,Quittung) ;
                   goto ω ;
ω:   fertig ;
```

Es ist zu erwähnen, daß sich bei verschiedenen Ausführungen des Algorithmus, auch wenn Initiator und Netz unverändert sind, durch die individuellen und unterschiedlichen Nachrichtenlaufzeiten unterschiedliche Bäume ergeben können. In verteilten Systemen unterliegen die Nachrichtenlaufzeiten Zufallseinflüssen und hierdurch können sich bei unterschiedlichen Ausführungen unterschiedliche Systemabläufe ergeben: Verteilten Systemen innewohnender Indeterminismus.

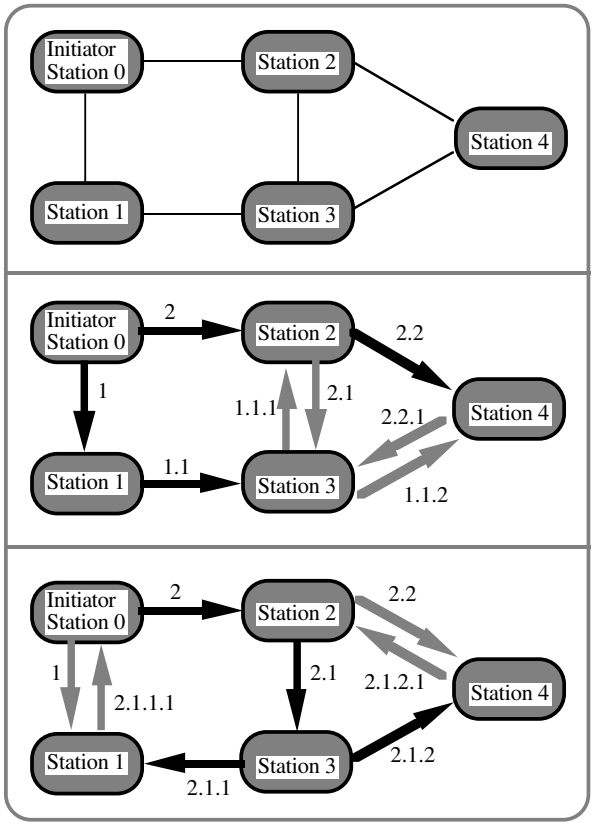

Bild 10.2: Beispiel zum Echo-Algorithmus

Bild 10.2 zeigt ein Beispiel dazu. Im oberen Teil ist das teilvermaschte Netz zu sehen. Der mittlere und der untere Teil sollen zwei verschiedene Abläufe verdeutlichen. Die schwarzen Pfeile sind dabei die Kanten des Baums, Nachrichten die auf nicht-engagierte Zielstationen treffen.
Im mittleren Ablauf kommt die Nachricht 2 vor der Nachricht 1.1.1 bei Station 2 an, Station 3 erhält 1.1 vor 2.1 und vor 2.2.1, Station 4 erhält 2.2 vor 1.1.2.
Im unteren Ablauf benötigt nun die Nachricht 1 relativ lange, um bei Station 1 anzukommen. Sie kommt dort nach 2.1.1 an. Weiterhin ist die Folge 2.1 und 2.1.2 schneller als die Nachricht 2.2.

10.2.2 Vertreter-Auswahl

Gegeben sei eine Menge von Stationen. Problem sei, daß eine ausgezeichnete Station (Repräsentant) eindeutig bestimmt werden soll. Die Initiative zur Vertreter-Bestimmung gehe von einer beliebigen Station aus, u.U. können auch mehrere Stationen nebenläufig eine Vertreter-Bestimmung initiieren.

Eine mögliche Lösung basiert darauf, daß eine Totalordnung über den Stationsadressen definiert werden kann und als Vertreter die Station mit der maximalen Adresse gewählt werden soll.
Damit ist das Problem in das Problem der Maximumsuche überführt.

10.2.3 Maximumsuche

Gegeben sei eine Menge von Stationen, deren Anzahl in keiner Station bekannt ist. Jede Station besitze ein eindeutiges Datenattribut 'Id' aus einer total geordneten Menge.
Problem sei, daß die maximale Id gesucht werde und daß allen Stationen bekannt gemacht werden müsse, ob sie die maximale Id besitzen oder nicht. Die Maximumsuche könne auch nebenläufig mehrfach gestartet werden.

Eine erste Lösung ist konzeptuell sehr einfach. Jede Station verfährt nach folgender Vorschrift:

```
α:  M:= -∞ ;
    if Startwunsch
        then
            M:=Id ;
            for each Ncb of Nachbarn  SEND(Ncb,M) ;
    loop
        RECEIVE (x,OtherId) ;
        if M<OtherId
            then
                M:=max(OtherId,Id) ;
                for each Ncb of Nachbarn  SEND(Ncb,M) ;
        if Terminierung der Maximumsuche
            then
                goto ω ;
ω:  if M=Id
        then
            'ich besitze Maximum'
        else
            'ein anderer besitzt Maximum'
    fertig ;
```

Sie enthält allerdings ein nicht-triviales Teilproblem, nämlich die Terminierungserkennung.
Außerdem entsteht bei einem stark vermaschten Netz hoher Kommunikationsaufwand.
Der Kommunikationsaufwand läßt sich reduzieren, wenn spezielle Topologien zugrundegelegt werden können.

Im unidirektionalen Ring verfahre eine Station wie folgt:

```
α:  M:=-∞ ;
    if Startwunsch
        then
            M:=Id ;
            SEND(rechter Nachbar,M) ;
    loop
        RECEIVE (x,Data) ; (* x ist linker Nachbar *)
        if Type(Data)=Id-Nachricht
            then
                if M<Data.Id
                    then
                        M:=max(Data.Id,Id) ;
                        SEND(rechter Nachb,M) ;
                elseif M=Data.Id
                    then
                        'ich besitze Maximum' ;
                        SEND(rechter Nachb, Terminierungsnachr) ;
        elseif Type(Data)=Terminierungsnachricht
            then
                'ein anderer besitzt Maximum'
                SEND(rechter Nachbar, Terminierungssignal) ;
                goto ω ;
ω:  fertig ;
```

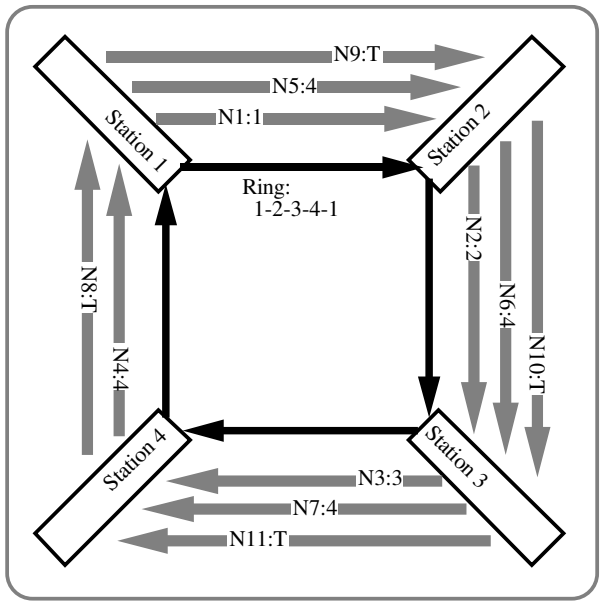

Bild 10.3: Beispiel zur Maximumsuche im Ring

Im unidirektionalen Ring geht der Kommunikationsaufwand zurück. Es gibt je Station nur noch eine Kante, auf die die Station senden kann. Durch die erzwungene Reihenfolge können nicht zum Erfolg führende nebenläufige Anstöße relativ schnell unterdrückt werden. So werden im ungünstigsten Fall bei einem Initiator im Ring mit n Stationen 3n-1 Nachrichten (incl. Terminierungsnachrichten) transferiert. Sind alle Stationen Initiatoren können allerdings im ungünstigsten Fall $(n^2+3n)/2$ entstehen.

Weiterer Vorteil ist, daß nun auch das Teilproblem der Terminierungserkennung gelöst ist. Der Gewinner erkennt die Terminierung und kann sie den übrigen Stationen mitteilen.

Bild 10.3 zeigt beispielhaft einen Ablauf in einem 4-Stationen-Ring. Station 1 initiiert die Maximumsuche. Nach einem Umlauf ist der Station 1 das Maximum bekannt. Mit einem (knappen) weiteren Umlauf erfährt es die gewählte Station 4. Die von der Station 4 versendete Terminierungsnachricht läuft dann abschließend einmal um den Ring.

Eine wichtige Anwendung dieses Algorithmus finden wir im Token-Ring Protokoll nach dem LAN-Standard IEEE 802.5. Dort muß bei der Ring-Initialisierung eine der angeschlossenen Stationen als aktiver Überwacher bestimmt werden.

Im obigen Beispiel wurde die Stationsnummer als Auswahl-Attribut gewählt. Das muß nicht sein. Man könnte z.B. auch die Station mit dem derzeit größten Vorrat an freiem Arbeitsspeicher suchen. Ein Problem, das nun entsteht, ist, daß mehrere Stationen zufällig identische Freispeicher-Werte haben können. Die Attributwerte sind nicht mehr eindeutig. Eine in diesen Fällen häufig angewendete Lösung ist es, nach einem Maximum der (eindeutigen) Verbundwerte [logisches Attribut, Stationsnummer] zu suchen. Bei identischen logischen Attributen wird nun die Station mit der größeren Stationsnummer gewählt.

Liegt kein uni-direktionaler Ring als Netz-Topologie vor (z.B. erscheint ja auch ein ISO/OSI-Transportsystem den Anwendungsprozessen als vollvermaschtes Netz), dann kann der Ring-Algorithmus dennoch angewendet werden. In einer vorbereitenden Phase kann ein logischer Ring bestimmt werden (er sollte so bestimmt werden, daß sich günstige Umlaufzeiten bzw. Umlaufkosten ergeben).

Rückblickend zum Echo-Algorithmus kann man im Vergleich feststellen, daß der soeben vorgestellte Algorithmus zur Maximum-Suche im unidirektionalen Ring von einem anderen Typ ist.
Der Echo-Algorithmus nutzt die im verteilten System gegebene Möglichkeit zur Parallelität, mehrere Stationen können zur selben Zeit wirklich aktiv sein, mehrere Nachrichten können gleichzeitig unterwegs sein. Dadurch gibt es natürlich u.U. unnötige Nachrichten. Auch führt die Konkurrenz der Nachrichten zum Indeterminismus und der Algorithmus ist schwieriger zu durchschauen. Aber die Parallelität kann den Durchsatz fördern.
Die Maximumsuche im unidirektionalen Ring zeichnet sich dadurch aus, daß sich nur ein einziger sequentieller Aktions- und Nachrichtenfluß ergibt. Wenn eine Station aktiv ist, warten die anderen; es ist immer nur eine Nachricht im Transfer (wenn man voraussetzt, daß nicht mehrere Initiatoren nebenläufig die Suche anstoßen). Dies wird durch die starke Einschränkung der Kommunikationsmöglichkeiten (unidirektionaler Ring) erreicht. Hier ist nun der Kommunikationsaufwand gering, Indeterminismen treten nur auf, wenn mehrere Initiatoren die Suche anstoßen. Aber die Sequentialisierung vermindert im Vergleich zu einer parallelen Lösung den Durchsatz.

10.2.4 Schnappschuß

Das Schnappschußproblem ergibt sich, wenn in einem verteilten System, in dem verschiedene Stationen nebenläufig arbeiten, eine Station einen Zustand erreicht, in dem sie eine konsistente Sicht des globalen Systemzustands (Vektor der einzelnen Stationszustände) benötigt, also wenn ein Wechsel von einer wirklich verteilten Arbeitsphase in eine zentrale Arbeitsphase durchgeführt werden muß. Da durch einen Schnappschuß ein solcher Wechsel von 'verteilt' nach 'zentral' ermöglicht wird, ist das

Schnappschußproblem sehr geeignet, um auf abstrakter Ebene den Unterschied zwischen 'verteilt' und 'zentral/lokal/nicht-verteilt' zu erfassen. Man wird insbesondere auch erkennen, daß dieser Wechsel in effizienter und idealer Weise nicht möglich ist.

Schnappschuß-Problem

Gegeben sei eine Menge von Stationen, die einen verteilten Problemalgorithmus ausführen. Hierzu arbeiten sie lokal auf privaten Daten und kommunizieren per Nachrichtenaustausch (SEND/RECEIVE 'Problemnachricht').
Eine Station benötige zu einem beliebigen Zeitpunkt einen Schnappschuß des Gesamtsystems, d.h., sie möchte in einer Datenstruktur über den Globalzustand (Vektor der Zustände der relevanten Problemdaten aller Stationen) verfügen. Hierzu müssen ihr von den anderen Stationen auf Anforderung Zustandsnachrichten geschickt werden. Idealerweise würde ein Schnappschußinitiator von den anderen Stationen Statusmeldungen so anfordern, daß alle Stationen ihre Zustände zum selben Zeitpunkt in Statusmeldungen kopieren.
Im verteilten System kann allerdings ein atomarer Broadcast, der bei allen Stationen zum gleichzeitigen Auftreten der Status-Anfrage führen würde, nicht vorausgesetzt werden. Der Initiator muß den Status-Erfrage-Wunsch an jede der übrigen Stationen per separatem Nachrichtentransfer übermitteln (SEND 'Statusanfrage'). Wir können noch vereinfachend annehmen, daß er diese (n-1) Statusanfrage-Nachrichten gleichzeitig absendet (Atommodell). Aber durch Nachrichtenlaufzeit-Unterschiede (die erheblich sein können), kommen sie zu unterschiedlichen Zeitpunkten bei den Empfängern an. Die von den befragten Stationen daraufhin zum Initiator zurückgesendeten Statusmeldungen werden deshalb Teilzustände wiedergeben, die zu unterschiedlichen Zeitpunkten im System herrschten. Der Initiator kombiniert die Teilzustände nun zum Schnappschuß und erhält einen Globalzustand, der in seinen einzelnen Teilen zwar tatsächlich eingenommen wurde, dessen Kombination möglicherweise so aber in Wirklichkeit zu keinem Absolutzeitpunkt vorhanden war. Die Teile können unterschiedlich alt, der Schnappschuß kann inkonsistent sein.

Konsistenz

Es sei t_i der Zeitpunkt, an dem eine Station S_i die an sie gerichtete Statusanfrage-Nachricht empfängt. Sie bereite daraufhin ihre Statusmeldung mit dem Inhalt z_i, ihrem gegenwärtigen lokalen Zustand, auf und sende ihre Statusmeldung an den Initiator. Dies soll sie nach Eingang der Statusanfrage umgehend durchführen, d.h. zwischen dem Eingang der Statusanfrage und dem Absenden der Statusmeldung empfange S_i keine andere Nachricht (im Programm von Si liege zwischen RECEIVE 'Statusanfrage' und SEND 'Statusmeldung' einfach keine RECEIVE-Anweisung). Nach dem Konzept 'Atommodell' kann dann für die globale Betrachtung vom lokalen Zeitverbrauch abstrahiert werden: der Empfang der Statusanfrage, die Aufbereitung der Statusmeldung und das Absenden der Statusmeldung finden atomar zum selben Zeitpunkt, nämlich t_i, statt.

Der Initiator empfange die so entstandenen Statusmeldungen der Stationen und setze sie zum Schnappschuß $Z=\langle z_1,z_2,..,z_n\rangle$ zusammen. Ein z_i lag zum Zeitpunkt t_i vor, $T=\langle t_1,t_2,..,t_n\rangle$ sei der so zu Z gehörige Zeitvektor.

Gilt $t_1=t_2=..=t_n$, so ist der Schnappschuß Z sicherlich als konsistent zu bezeichnen, Z ist dann ja ein idealer Schnappschuß, dessen Komponenten wirklich gleichzeitig bestimmt wurden.
Die Frage ist nun, wann ein Schnappschuß sonst noch als konsistent bezeichnet werden kann.

Es sei e_i das zum Zeitpunkt t_i an Station S_i auftretende atomare Ereignis, d.h. die lokale Aktionenfolge 'RECEIVE Statusanfrage, Aufbereitung der Statusmeldung, SEND Statusmeldung, etwaige Folgeaktionen, die vor dem nächsten RECEIVE liegen'.
$E=\langle e_1,e_2,.., e_n\rangle$ sei der so zum Schnappschuß $Z=\langle z_1,z_2,..,z_n\rangle$ gehörige Ereignisvektor.
Die informelle Anwort lautet, daß Z konsistent ist, wenn auch eine Ausführung des Algorithmus möglich ist, bei der die e_i zueinander gleichzeitig auftreten können (dies wird durch die Nachrichten-

laufzeiten im System beeinflußt). Zur Präzisisierung muß untersucht werden, unter welchen Umständen dies gegeben sein kann.

Die lokal an einer Station S_i stattfindenden Ereignisse sind zeitlich total geordnet, da wir ja von sequentiellen Knotenprogrammen ausgehen. Sei x_i ein Ereignis an S_i und y_i sein direkter zeitlicher Nachfolger, so schreiben wir '$x_i \rightarrow_l y_i$', die beiden Ereignisse stehen in direkter kausaler Abhängigkeit.

Zwei Ereignisse x_i und y_j an unterschiedlichen Stationen können ebenfalls in direkter kausaler Abhängigkeit stehen, nämlich dann wenn x_i ein SEND für eine Nachricht enthält, die mit dem RECEIVE empfangen wird, mit dem y_j beginnt. Wir schreiben hierfür '$x_i \rightarrow_g y_j$'.

'$x \rightarrow y$' bezeichne generell die direkte kausale Abhängigkeit:

$$x \rightarrow y \;\Leftrightarrow\; (x \rightarrow_l y) \vee (x \rightarrow_g y).$$

'$<$' bezeichne die transitive Hülle von '$\rightarrow$'.
Wenn wir zwei beliebige Ereignisse x und y einer Algorithmusausführung nehmen, und es gilt '$x < y$', dann muß x vor y stattfinden (nach der obigen Definition von '$<$' gibt es dann nämlich eine Kette kausaler Abhängigkeiten von x nach y).

Zwei Ereignisse x und y einer Algorithmusausführung können nun - nur davon abhängig, daß sich geeignete Nachrichtenlaufzeiten einstellen - gleichzeitig auftreten, wenn '$x \parallel y$' gilt mit:

$$x \parallel y \;\Leftrightarrow\; \neg (x < y) \wedge \neg (y < x).$$

Ein Schnappschuß $Z=<z_1,z_2,..,z_n>$ mit zugehörigem Ereignisvektor $E=<e_1,e_2,.., e_n>$ heiße konsistent, wenn es zwischen keinen zwei Ereignissen e_i, e_j eine kausale Abhängigkeit gibt, d.h.

$$\forall_{i,j \in [1..n]}: e_i \parallel e_j$$

(Es soll angemerkt werden, daß es auch andere Definitionen zur Konsistenz von Schnappschüssen gibt).

Einfacher Algorithmus

Im ersten Anlauf betrachten wir einen einfachen Algorithmus zur Ermittlung eines konsistenten Schnappschusses, der sogar einen aktuellen Schnappschuß liefert. Allerdings besitzt er den Nachteil, daß die angefragten Stationen bis zum Abschluß der Ermittlung eingefroren werden.

Neben den Basisoperationen SEND und RECEIVE benötigen wir noch eine neue Operation UNRECEIVE. UNRECEIVE soll die mit dem letzten RECEIVE empfangene Nachricht 'in den Stationsbriefkasten zurücklegen', so daß sie von einem späteren RECEIVE wieder erfaßt werden kann (sie werde aber unten in den Briefkasten gelegt, so daß etwaige andere Nachrichten von nachfolgenden RECEIVEs vorher erfaßt werden).

Zur Vereinfachung nehmen wir an, daß es nur eine Station gibt, von der aus Schnappschüsse initiiert werden sollen, und daß diese Initiatorstation selbst nicht am Problemalgorithmus teilnimmt. Sie verfahre wie folgt:

```
α:  for all Adr from OtherStations do SEND Adr, Statusanfrage ;
    Status:= < > ; Replies:=OtherStations ;
    while Replies<>{} do
        RECEIVE Abs, Statusmeldung ; Status:=Status & Statusmeldung ;
        Replies:=Replies\{Abs} ;
    for all Adr from OtherStations do SEND Adr, Schnappschußende ;
ω:  fertig ;
```


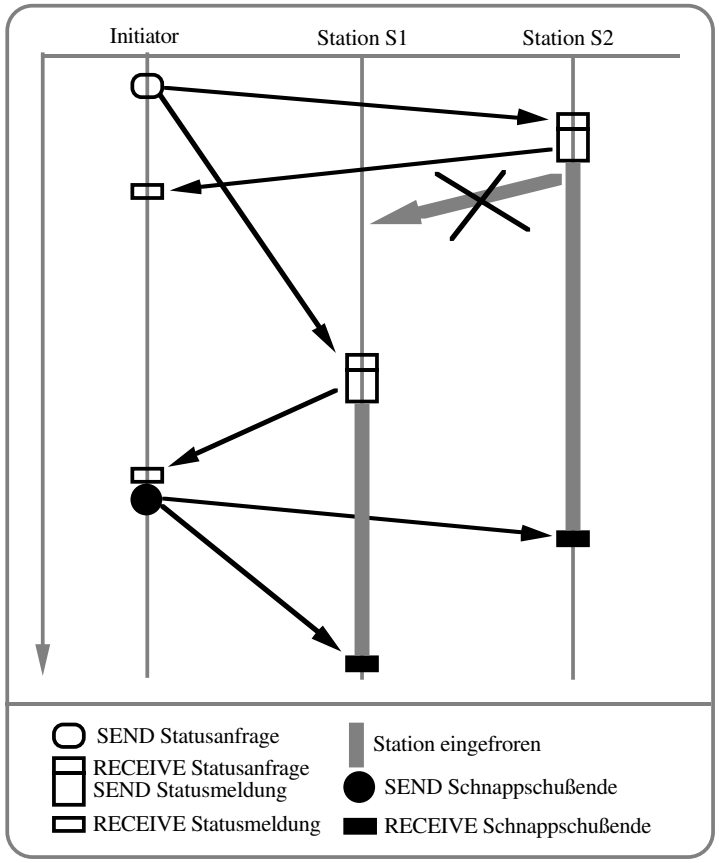

Bild 10.4: Szenario zum einfachen Schnappschußalgorithmus

Bei den angefragten Stationen werde jede Anweisung 'RECEIVE Src, Msg' durch folgende Befehlsfolge ersetzt:

Ersetze **'RECEIVE Src, Msg'** *im Problemalgorithmus durch:*

```
loop
    RECEIVE Src, Msg ;
    if Type(Msg)=Statusanfrage
        then Bereite Statusmeldung auf ; SEND Src, Statusmeldung ;
             loop RECEIVE Src, Msg;
                  if Type(Msg)=Schnappschußende
                      then exit from inner loop ;
                      else UNRECEIVE ;
        else exit from loop ;
```

Bild 10.4 zeigt einen möglichen Ablauf dieses einfachen Schnappschußalgorithmus in einem System aus 2+1 Stationen. S2 erhält die Statusanfrage wesentlich früher als S1 und liefert deshalb ihre Statusmeldung zu einem Zeitpunkt, an dem S1 noch nichts vom Schnappschußwunsch weiß und lokale Aktionen durchführt. Eine Station wird unmittelbar nach dem Absenden ihrer Statusmeldung eingefroren. Hierdurch können lange und stark leistungsmindernde Phasen der Untätigkeit entstehen.

Die zeitliche Unvergleichbarkeit der Statusanfrage/Statusmelde-Ereignisse und damit die Konsistenz wird durch das Einfrieren sichergestellt. Da die 'frühe' Station S2 eingefroren ist, kann sie nicht nach Absenden der Statusmeldung weiterarbeiten. So kann sie auch keine Problemnachricht an S1 senden, die dort vor dem Auftreten des S1-Statusanfrage/Statusmelde-Ereignisses empfangen werden könnte (durchgestrichener Pfeil in Bild 10.4). In einer solchen Situation wären die Ereignisse zeitlich vergleichbar, da eine Kette vom S2-Ereignis - über den Austausch der Problemnachricht - zum S1-Ereignis existieren würde.

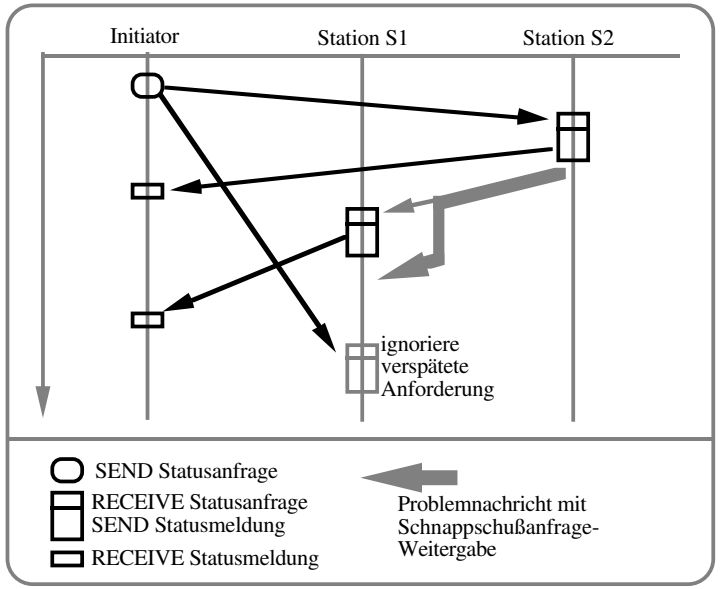

Bild 10.5: Szenario zum verbesserten Schnappschußalgorithmus

Verbesserter Algorithmus

Ein verbesserter Algorithmus vermeidet das Einfrieren der Stationen und damit unnötige Stillstandszeiten. Da die Stationen nach dem Absenden ihrer Statusmeldung nun weiterarbeiten, ist der vom Initiator zusammengesetzte Schnappschuß nun nicht mehr unbedingt aktuell.
Zur Konsistenzsicherung muß die schon in Bild 10.4 angedeutete Situation vermieden werden, in der die Station S1 ihre Statusmeldung schon gesendet hat und später eine Problemnachricht an S2 sendet, die noch keine Statusmeldung erzeugte. Dann liegt das Absenden dieser Problemnachricht in der S1-Zukunft des Schnappschusses, der Empfang aber in der S2-Vergangenheit. Es gibt eine kausale Abhängigkeitskette vom S1-Ereignis zum S2-Ereignis.

Der Algorithmus vermeidet dies dadurch, daß nach dem Absenden einer Statusmeldung jede nachfolgend gesendete Problemnachricht mit einer impliziten Weitergabe der Statusanforderung versehen wird. Empfängt eine Station nun eine solche Problemnachricht, simuliert sie - noch bevor sie die Problemdaten der Nachricht verarbeitet - den Empfang einer Statusanforderung. Wir nehmen dazu an, daß vor dem Auftreten des Problemnachricht-Empfangsereignisses ein Statusanfrage-Ereignis eingefügt wird.

Bild 10.5 zeigt ein entsprechendes Szenario. Hier empfängt S1 eine solche Problemnachricht. Die verspätet eingehende eigentliche Statusanforderung muß natürlich ignoriert werden. Zum Erkennen von Doppelanforderungen können Schnappschußnummern (*ShId*) eingeführt werden.

Die Initiatorstation verfahre wie folgt:

```
Initialisierung:
      ShId:=0 ;
Schnappschußdurchführung:
α:    for all Adr from OtherStations do SEND Adr, <ShId, Statusanfrage> ;
      Status:= < > ; Replies:=OtherStations ;
      while Replies<>{} do
            RECEIVE Abs, <X, Statusmeldung> ;
            Status:=Status & Statusmeldung ;
            Replies:=Replies\{Abs} ;
      ShId:=ShId+1 ;
ω:    fertig ;
```

Bei den angefragten Stationen müssen SEND- und RECEIVE-Anweisungen des Problemalgorithmus durch Anweisungsfolgen ersetzt werden (auch eine Initialisierung der lokalen ShId-Variablen wird benötigt):

```
Initialisierung:
      ShId:=-1 ;
Ersetze SEND Dst,Msg' durch:
      SEND Dst, <ShId, Msg> ;
Ersetze 'RECEIVE Src, Msg' durch:
      loop
            RECEIVE Src, <X, Msg> ;
            if X>ShId then
                  ShId:=X ;
                  Bereite Statusmeldung auf ;
                  SEND Initiator, <ShId, Statusmeldung>;
            if Type(Msg)=Problemnachricht then exit from loop ;
```

Anwendung

Nachdem nun Möglichkeiten zur Schnappschußermittlung gezeigt wurden, stellt sich die Frage, was ein Initiator mit Schnappschüssen anfangen kann. Zu unterscheiden sind die beiden Fälle:

- aktueller Schnappschuß, geliefert vom einfachen Algorithmus,
- (u.U.) veralteter Schnappschuß, geliefert vom verbesserten Algorithmus.

Im Falle des aktuellen Schnappschusses ist die Anwort einfacher. Alle Stationen bleiben solange eingefroren, bis der Initiator ihnen das Schnappschußende signalisiert. Nach dem Einsammeln der letzten Statusmeldung und vor dem Absenden der Schnappschußende-Nachrichten kann der Initiator als einzig aktive Instanz des Systems zentrale Aktionen durchführen, denen der aktuelle Globalzustand

lokal verfügbar ist. Er kann den Globalzustand lesend und schreibend zugreifen (Veränderungen kann er mit den Schnappschußende-Nachrichten an die Stationen zurückspielen).

Im Falle des veralteten Schnappschusses ist die Antwort schwieriger. Da die Stationen nach dem Absenden ihrer Statusmeldungen weiterarbeiten, ist der Initiator nicht-exklusiver Zugreifer des Globalzustands. Deshalb sollte er nur lesend zugreifen und die Frage ist nun, was man aus der Beobachtung eines veralteten Zustands an Information gewinnen kann. Da der Zustand zwar veraltet ist aber doch immerhin vorhanden war, kann der Initiator sich Fragen beantworten der Art 'Wie weit ist die Problemalgorithmus-Ausführung mindestens schon gekommen?'. Hierzu können isotone Beobachtungsfunktionen und - als Spezialfall davon - stabile Prädikate verwendet werden.

Eine isotone Beobachtungsfunktion f bildet den Urbildbereich der möglichen Globalzustände auf einen geordneten Bildbereich so ab, daß für z, z', mit z wurde vor z' ermittelt, gelten muß $f(z)<f(z')$.
Ein Beispiel dafür ist eine Funktion, die angibt, wieviele Eingaben nach Stand des Globalzustands schon vollständig verarbeitet wurden.

Ein stabiles Prädikat p bildet den Urbildbereich der möglichen Globalzustände auf den Bildbereich 'false, true' so ab, daß für z, z', mit z wurde vor z' ermittelt, gelten muß $p(z) \Rightarrow p(z')$, d.h. wenn das Prädikat einmal wahr geworden ist, dann wird es wahr bleiben.
Beispiele für stabile Prädikate sind 'Liegt Terminierung vor' und 'Hat sich ein Deadlock eingestellt'.

10.2.5 Terminierungserkennung

Wenn wir im Normalfall auch davon ausgehen, daß eine 'echte' verteilte Anwendung nicht terminiert, so wird sie doch in einzelnen Betriebsphasen einzelne Teilaufgaben in endlicher Zeit zu lösen haben (z.B. ein Flugbuchungssystem soll einen Buchungsauftrag abwickeln). Wird eine solche Teilaufgabe verteilt bearbeitet, d.h. es tritt nebenläufige, lose gekoppelte Bearbeitung an verschiedenen Orten auf, dann stellt sich oft das Problem, daß eine Instanz, z.B. diejenige, die das Teilproblem in das System schleuste, erkennen will, ob und wann die verteilte Bearbeitung terminiert.

Wie bei der Schnappschußermittlung nehmen wir an, daß eine Menge von Stationen einen verteilten Problemalgorithmus ausführt. Die Stationen arbeiten nicht ganz unabhängig voneinander, sie senden sich gegenseitig Problemnachrichten zu. Ein solches System soll in der Abstraktionsebene des Atommodells betrachtet werden. Unter diesen Annahmen liegt Terminierung dann vor, wenn keine Problemnachricht mehr unterwegs ist (dann wurden alle gesendeten Nachrichten empfangen und verarbeitet, wenn die Verarbeitung zum Absenden zusätzlicher Nachrichten geführt hätte, wären sie - gemäß Atommodell - auch schon unterwegs).

Eine konzeptuell einfache Lösung ist es, Schnappschüsse zu verwenden. Die Station, welche die Terminierung erkennen will, möge periodisch Schnappschüsse erzeugen lassen. Die für die Terminierung relevante Zustandskomponente einer Station gebe über die Anzahl bisher gesendeter und die Anzahl bisher empfangener Nachrichten Auskunft. Stellt der Initiator fest, daß die Bilanz global ausgeglichen ist, liegt Terminierung vor.
Nachteil dieser Vorgehensweise ist der hohe Kommunikationsaufwand, entstehend durch die u.U. hohe Zahl an Schnappschüssen.

Das Doppelzählverfahren baut auf der Idee der Nachrichtenbilanz auf, die Stationsangaben 'Anzahl gesendeter Nachrichten, Anzahl empfangener Nachrichten' werden aber nun nicht im konsistenten Schnappschuß angefordert.
Würde man sich diese - möglicherweise inkonsistente - Bilanz nur einmal liefern lassen, könnte u.U. die Terminierung fälschlich erkannt werden. Bild 10.6 zeigt hierzu ein Szenario mit zwei Problemstationen. Die Sicht des Initiators ist hier inkonsistent, weil der Austausch der Nachrichten a und b zwar von S1, nicht aber von S2 berücksichtigt wurden.

Das Doppelzählverfahren führt nun - anstelle eines Schnappschusses - zwei aufeinanderfolgende und sich zeitlich nicht überlappende Erkundungen durch. Es sei R die Summe der bei der ersten Erkundung gelieferten Empfangsanzahlen, S' die Summe der bei der zweiten Erkundung gelieferten Sendeanzahlen. Gilt R=S', so existierte ein Absolutzeitpunkt, an dem die Bilanz ausgeglichen war, es liegt also Terminierung vor.

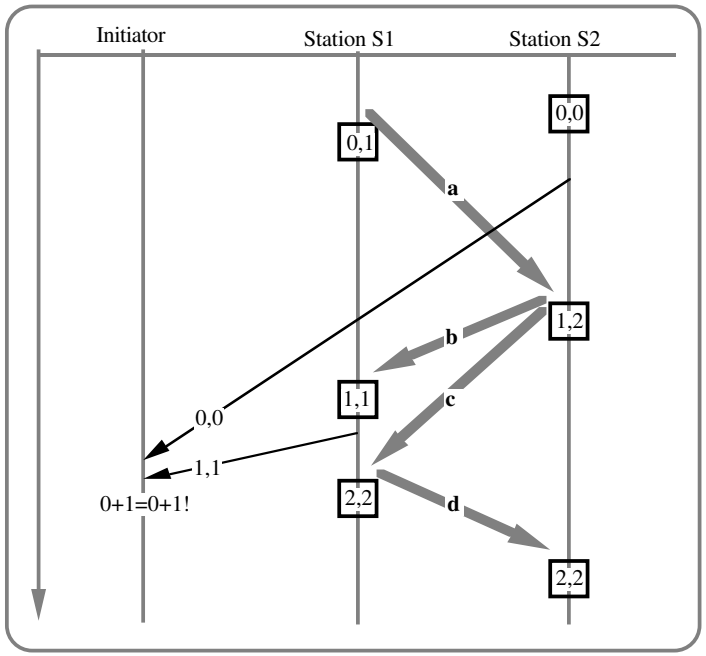

Bild 10.6: Szenario zum Mitzählen der Nachrichten

Ein anderer Lösungansatz ist die Vektormethode. Hier wird ebenfalls in jeder Station mitgezählt, aber etwas differenzierter. Jede Station S_i verwaltet einen Vektor V_i.
Empfängt S_i eine Nachricht, aktualisiert sie V_i gemäß $V_i[i]:=V_i[i]-1$.
Sendet S_i eine Nachricht an S_j, aktualisiert sie V_i gemäß $V_i[j]:=V_i[j]+1$.
Ferner zirkuliere im Netz (und zwar so daß alle Stationen in fairer Weise erreicht werden) eine Kontrollnachricht KV, die ebenfalls als Vektor ausgebildet ist.
Empfängt S_i den Kontrollvektor KV, dann aktualisiere sie KV und ihren eigenen Vektor V_i gemäß '$KV:=KV+V_i$; $V_i:=<0,0,..,0>$'.
Sobald der Kontrollvektor <0,0,..,0> auftritt, liegt Terminierung vor.

Als letztes Beispiel soll das Kreditverfahren kurz skizziert werden. Hier starte eine Station die verteilte Berechnung und sie soll auch die Terminierung feststellen.

- Die Startstation hat initial die Kreditzahl 1.

- Wenn eine Station derzeit einen Kredit q besitzt und m Nachrichten an andere Stationen versendet, gibt sie jeder Nachricht den Kredit 'q/m' mit.
- Senden darf nur, wer einen Kredit >0 hat.
- Wenn die Startstation wieder den Kredit 1 erreicht, liegt Terminierung vor.

=== Ende des Skriptums H. Krumm: 'Rechnernetze und verteilte Systeme' ===