Zusammenfassung der Vorlesung RvS

Andreas Schröder

03. - 06. März 2003

Inhaltsverzeichnis

1 Übertragungssysteme (3.1) 5
1.1 Art des nachrichtentechnischen Kanals . . . 5
1.2 Serielle oder parallele Übertragung . . . 5
1.3 (Takt-)Synchronisation zwischen Quelle und Senke . . . 5
1.4 Codierung . . . 5
1.5 Qualitätsparameter eines Kanals . . . 5
1.6 Betriebsarten . . . 6
1.7 Sammelkanäle . . . 6

2 Medien (3.1.2) 6
2.1 Medienqualität . . . 6
2.2 Modelle für Störeinflüsse . . . 6
2.3 Verwendete Medien . . . 7
2.4 Geteilte Medien . . . 7
2.5 Übertragungsverfahren . . . 7

3 Telekommunikationsprotokolle und -Dienste (4) 7
3.1 Beschreibung der Dienstfunktionalität (4.1.1) . . . 8
3.1.1 Dienstzugangspunktadressen . . . 8
3.1.2 Schnittstellenereignisse . . . 8
3.1.3 Weg-Zeit-Diagramme . . . 8
3.1.4 Zustandsübergangsdiagramme . . . 8
3.2 Verbindungsorientierte Dienste . . . 9
3.2.1 Verbindungseigenschaften . . . 9
3.2.2 Dienstleistungsqualität . . . 9
3.2.3 Richtungsbetrieb . . . 10
3.2.4 Rechtevergabe . . . 10
3.2.5 Verbindungsklassen . . . 10
3.2.6 Rückstau . . . 10
3.2.7 Vorgänge bei verbindungsorientierten Diensten (4.1.2) . . . 10
3.3 Datagrammdienste (4.1.3) . . . 11
3.4 Telekommunikationsprotokolle (4.2) . . . 11
3.4.1 Ablauffestlegungen . . . 11
3.4.2 PDU-Formatfestlegungen . . . 11
3.4.3 Beispiel Alternating Bit Protocol . . . 12
3.4.4 Protokollkorrektheit . . . 12
3.4.5 Protokoll-Leistungsfähigkeit . . . 12

4 Logische Architektur von Kommunikationssystemen (5) 13
4.0.6 Aufgaben der Transportschichten . . . 14
4.0.7 Aufgaben der anwendungsorientierten Schichten . . . 15

5 Protokollmechanismen (6) 15
5.1 Zuteilung geteilter Medien (6.1) . . . 15
5.1.1 Zentrale Zuteilung . . . 15
5.1.2 Dezentrale Zuteilung . . . 16
5.1.3 Backoff-Algorithmus . . . 17
5.1.4 Leistungsaspekte . . . 17
5.1.5 Maßnahmen zur Steigerung der Leistung . . . 17
5.2 Fehlerbehandlung (6.2) . . . 17
5.2.1 Verfahren zur Fehlerbehebung . . . 18
5.2.2 Fehlererkennender Code (CRC) . . . 18
5.2.3 Sequenzzahlen und Zeitstempel . . . 18
5.2.4 Quittieren . . . 18
5.2.5 Zeitüberwachung (Timeout) . . . 19
5.3 Fehlerbehebung (6.3) . . . 19
5.4 Längenanpassung (6.4) . . . 19
5.5 Flusskontrolle (6.5) . . . 19
5.6 Weiterleiten (6.6) . . . 20
5.7 Überlastkontrolle (6.7) . . . 20
5.8 Übertragungsleistungsanpassung (6.8) . . . 21
5.9 Schutz und Sicherheit (6.9) . . . 21
5.10 Verwaltung (Management) (6.10) . . . 22

6 Dienstleistungen der Schichten des OSI-Modells 22
6.1 Schicht 1: Physical Layer . . . 22
6.2 Schicht 2: Data-Link Layer . . . 23
6.2.1 Schicht 2a: Media Access Control . . . 23
6.2.2 Schicht 2b: Logical Link Control . . . 23
6.3 Schicht 3: Network Layer . . . 23
6.4 Schicht 4: Transport Layer . . . 23
6.5 Schicht 5: Session Layer . . . 23
6.6 Schicht 6: Presentation Layer . . . 24
6.7 Schicht 7: Application Layer . . . 24
6.8 Anmerkungen . . . 24

7 Rechnernetzanwendungen - Paradigmen (9.2.2) 24
7.1 System kommunizierender Prozesse . . . 24
7.2 Clients und Server . . . 25
7.3 Geteilte Objekte und kooperierende Prozesse . . . 25
7.4 Datenbank und kooperierende Prozesse . . . 25
7.5 Gemeinsame Aktionen und kooperierende Prozesse . . . 26
7.6 System aktiver Objekte . . . 26
7.7 Nachrichtenprozesse . . . 26
7.8 System wandernder Objekte . . . 26

8 Systemunterstützung (9.3) 27
8.1 Netzwerkfähige Station . . . 27
8.2 Verteilte Programmiersprachen . . . 27
8.3 Verteilte Betriebssysteme . . . 27
8.4 Netzwerk-Betriebssysteme . . . 27
8.5 Netzwerk-Betriebssystem nach dem Client-Server-Modell . . . 28
8.6 Verteilte Datenbanken . . . 28
8.7 Beispiele der Systemunterstützung . . . 28

9 Verteilte Algorithmen - Schnappschuss (10.2.4) 29
9.1 Konsistenz . . . 29
9.2 Einfacher Algorithmus . . . 30
9.3 Verbesserter Algorithmus . . . 30

10 Vermittlungssysteme (3.2) 32
10.1 Vermittlungseinrichtungen für Durchschaltevermittlung . . . 32
10.2 Vermittungseinrichtungen bei der Speichervermittlung . . . 32
10.3 ATM . . . 33

11 Implementierung (8) 33
11.1 Schnittstellen . . . 33
11.1.1 Prozess . . . 33
11.1.2 Synchronisation . . . 34
11.2 Datenaustausch . . . 34
11.2.1 Geteilter Speicher . . . 34
11.2.2 Fremdzugreifbarer Speicher . . . 34
11.2.3 Auftragsübergabe . . . 34
11.2.4 Datenaustausch mit Rendezvous . . . 35
11.2.5 Nachrichtenaustausch . . . 35
11.2.6 Monitore . . . 35
11.2.7 Geteilte Puffer . . . 35
11.3 Prozessinterne Schnittstelle . . . 35
11.4 Implementierung der Instanzen . . . 36
11.4.1 Zustandsdarstellung . . . 37
11.4.2 Aktuelles Eingabe-Ereignis . . . 37
11.4.3 PDU-Analyse . . . 37
11.4.4 PDU-Aufbau . . . 37
11.4.5 Aktivierung . . . 38
11.4.6 Selektion . . . 38
11.4.7 Zeitüberwachung . . . 38
11.5 Grundlegende Software-Architekturen . . . 38
11.5.1 Zentralverteiler . . . 38
11.5.2 Sammlung von Ereignisbearbeitungsprozeduren . . . 39
11.5.3 Instanzen und Prozesse . . . 39
11.5.4 Leichtgewicht-Prozesse . . . 39
11.5.5 Integration mehrerer Instanzen . . . 39
11.6 Programm- und Programmierumgebung . . . 40
11.6.1 Systemsprachen . . . 40
11.6.2 Höhere Programmiersprachen . . . 40
11.6.3 Einbettungssysteme . . . 40

11.6.4 Schnelle Prototyp-Erzeugung 40
11.6.5 Implemetierungsgeneratoren 40

12 Verteilte Algorithmen -Terminierungserkennung (10.2.5) 41
12.1 Vektormethode 41
12.2 Kreditverfahren 41

1 Übertragungssysteme (3.1)

Ein Übertragungssystem besteht aus einer Quelle, einer Senke und einem Nachrichtentechnischen Kanal, der wiederum aus einem Signal-Umformer, einem Medium und einem Rückformer besteht. Das Signal der Quelle $x(t)$ wird durch den Umformer in ein Signal $x'(t)$ zur Übertragung über das Medium aufbereitet. Innerhalb des Mediums wirken Störeinflüsse, sodass sich das Signal am Eingang des Rückformers als aus dem Signal $x'(t)$ und einem Störsignal $z(t)$ zusammensetzt. Das Signal $y'(t)$ wird durch den Rückformer nun zu einem Signal $y(t)$ aufbereitet, welches der Senke übergeben wird.

1.1 Art des nachrichtentechnischen Kanals

- Zeit- und signalkontinuierlich (analoges Signal)
- Zeitdiskret und kontinuierliches Signal (Sample and Hold)
- Zeitkontinuierlich und diskretes Signal
- Zeit- und Signaldiskret (übliches digitales Signal)

1.2 Serielle oder parallele Übertragung

1.3 (Takt-)Synchronisation zwischen Quelle und Senke

- durch gemeinsame, aber unabhängig von einander bestimmte Konventionen
- durch die Übertragung des Taktrasters auf einem eigenen Kanal
- durch die Übertragung mit dem Signal
- durch die punktuelle Synchronisation (mit Hilfe des Signals)

1.4 Codierung

- Quellencodierung: Zuordnung von Bitmustern zu den zu übertragenden Daten, zum Beispiel ASCII
- Kanalcodierung: Hinzufügen von Sicherungssequenzen, Synchronisationshilfen

1.5 Qualitätsparameter eines Kanals

- Schrittgeschwindigkeit: Baudrate, [1/s], Anzahl der Signalschritte, nach Nyquist kann die Schrittgeschwindigkeit nicht höher als $2 * B$, (B = Bandbreite des Kanals) sein
- Übertragungsgeschwindigkeit: [Bit/s], wird durch die Anzahl pro Signalschritt übertragener Bits und die Schrittgeschwindigkeit festgelegt, nach Shannon können pro Signalschritt nicht mehr als $\frac{1}{2} ld(1 + R)$ Bits übertragen werden (R = Rauschabstand in dB = mittlere Signalleistung / mittlere Rauschleistung)

- Signallaufzeit
- Schrittverzerrung: Abweichung vom isochronen Taktraster
- Bitfehlerwahrscheinlichkeit

1.6 Betriebsarten

- Simplex
- Halbduplex
- Vollduplex

1.7 Sammelkanäle

Kanäle, an denen mehrere Quellen und Senken angeschlossen sind. Hier sind Verfahren der Zugriffssteuerung wichtig.

2 Medien (3.1.2)

2.1 Medienqualität

- Bandbreite
- Dämpfungsverzerrung: Abhängigkeit der Dämpfung von der Frequenz
- Laufzeitverzerrung: Abhängigkeit der Signallaufzeit von der Frequenz

2.2 Modelle für Störeinflüsse

- Weißes Rauschen
- Echo-Bildung
- Nebensprechen
- Brumm
- Störimpulse

2.3 Verwendete Medien

- Elektrischer Leiter
- Funk
- Hohlleiter
- Glasfaser

2.4 Geteilte Medien

- Frequenzmultiplex: Jedem Teilnehmer wird ein Frequenzband zur exklusiven Nutzung zugeordnet (dynamisch oder statisch)
- Zeitmultiplex: Jedem Teilnehmer wird das gesamte Medium für einen Zeitschlitz wiederkehren zugeordnet (ebenfalls dynamische (ATM) oder statische Zuordnung)

2.5 Übertragungsverfahren

- Basisbandübertragung: digitales Rechtecksignal wird direkt in das Medium eingespeist
- Amplitudentastung: Durch Veränderung der Amplitude eines Sinus-Signals werden Informationen übertragen
- Frequenztastung: Veränderung der Frequenz eines Sinus-Signals
- Phasentastung: Die Phase eines Sinus-Signals wird verschoben Auch die Kombination mehrerer Verfahren ist möglich (siehe 56k-Modems)

3 Telekommunikationsprotokolle und -Dienste (4)

Telekommunikationssysteme werden in Schichten unterteilt, die jeweils bestimmte Dienste der darunterliegenden Schicht in Anspruch nehmen und der darüber liegenden Schicht Dienstleistungen anbieten.

Einen Dienst kann man anhand seiner Dienstschnittstelle beschreiben, die aus den SAPs (Service Access Points, Dienstzugangspunkte), den an diesen SAPs auftretenden Schnittstellenereignissen, und den dabei vorhandenen Ablauffestlegungen besteht. Damit wird vom internen Aufbau des Dienstes abstrahiert und er kann unabhängig von darunter liegenden Schichten betrachtet werden.

3.1 Beschreibung der Dienstfunktionalität (4.1.1)

3.1.1 Dienstzugangspunktadressen

Diese werden oft einfach durchnummeriert, je nach Anwendung (große Adressräume) werden aber auch hierarchische Strukturen verwendet (zum Beispiel Telefonnummer, bestehend aus Landes- und Ortsvorwahl und Anschlussnummer)

3.1.2 Schnittstellenereignisse

Eine Übliche Konvention für die Benennung von Schnittstellenereignissen ist folgende:

- Bezeichnung der Schicht (L, N, T, S, P, A, ...)
- Dienstelementname (Con, Dta, Dis, Res, Abo, ...)
- Grundtypname (Req, Ind, Res, Con, ...)

Diesen Ereignissen, auch Dienstprimitiven genannt, können nun Parameter zugeordnet werden (zum Beispiel bei TConReq können Quell- und Ziel-Adresse angegeben sein). Die Wertemenge dieser Ereignisse und Parameter wird bei der Spezifizierung der logischen Architektur eines Dienstes noch nicht beachtet.

3.1.3 Weg-Zeit-Diagramme

Dies ist ein üblicher und überschaubarer Weg, die örtlichen (bezogen auf Sender und Empfänger) und zeitlichen Zusammenhänge der einzelnen Schnittstellenereignisse darzustellen. Alternativen im Ablauf werden normalerweise in separaten Diagrammen dargestellt.

3.1.4 Zustandsübergangsdiagramme

Diese Form der Darstellung ist besser geeignet, um komplexe Zusammenhänge zu zeigen. Es werden alle möglichen Abläufe (manchmal auch nur eine Teilmenge) in einem Diagramm dargestellt. Vorlage ist ein erweiterter Mealy-Automat: Es können mehrere alternative Zustandsübergänge für die gleiche Eingabe angegeben werden. Die Auswahl hängt dann von einem Parameter des Gesamtsystems ab, der hier noch nicht angegeben ist. Außerdem können Übergänge mit leerer Eingabe angegeben werden. Diese werden aufgrund von Ereignissen des Gesamtsystems ausgeführt, die ebenfalls hier nicht angegeben sind (zum Beispiel beim Abbruch der Verbindung aufgrund eines Medienfehlers).

3.2 Verbindungsorientierte Dienste

Der Dienst wechselt zwischen zwei Zuständen, "Keine Verbindung "bzw. "Verbindung vorhanden ". Medien, bei denen mehrere Verbindungen unterstützt werden, müssen jedes Schnittstellenereignis genau einer Verbindung zuordnen können. Im Allgemeinen unterstützt ein Medium mehrere Dienstzugangspunkte und in jedem Zugangspunkt mehrere Verbindungen. Um mehrere Verbindungen je Zugangspunkt zu verwalten, wird eine Verbindungsendpunkt-Identifikation eingeführt. Meist wird die Reihenfolgentreue der Nachrichten garantiert.

3.2.1 Verbindungseigenschaften

- Qualität
- Richtungsbetrieb
- Rechte
- Klasse
- Rückstau

3.2.2 Dienstleistungsqualität

Es wird jeweils eine absolute Angabe gemacht und angegeben, wie wahrscheinlich eine Verletzung dieser Angabe ist.

- Verbindungsaufbau

- Durchsatz / Übertragungsleistung
- Nachrichtenlaufzeit / Verzögerung
- Störfehlerrate, je Bit oder je Block
- Verbindungsabbruchwahrscheinlichkeit
- Schutzklasse: Sicherung gegen Abhören, Verfälschen, Verlust, Sicherstellen der Dienstnehmeridentität
- Priorität dieser Verbindung gegenüber anderen Verbindungen im Medium
- Kosten

3.2.3 Richtungsbetrieb

- Simplex: Übertragung nur in eine Richtung möglich
- Halbduplex: Übertragung in beide Richtungen möglich, aber nur abwechselnd
- Vollduplex: Gleichzeitige Übertragung in beide Richtungen möglich

3.2.4 Rechtevergabe

Im einfachsten Fall darf jeder Teilnehmer nach dem Aufbau einer Verbindung mit der Übertragung von Daten beginnen. Allgemein wird mit dem Aufbau der Verbindung festgelegt, welche Rechte (Daten senden, Verbindung beenden, etc.) jeder Dienstnehmer zur Nutzung von Dienstleistungen hat. Diese Rechte können sich im Verlauf der Kommunikation ändern bzw. an einen anderen Teilnehmer übergeben werden).

3.2.5 Verbindungsklassen

Für ein universell einsetzbares Medium wird normalerweise eine große Anzahl von Dienstleistungen definiert, die aber je nach Anwendungsfall nicht genutzt werden. (Zum Beispiel Broadcast, Multicast, ExpeditedDataTransfer). Damit sowohl das Medium als auch der Dienstnehmer sich auf bestimmte Dienstleistungen vorbereiten kann (Reservierung von Ressourcen), kann beim Verbindungsaufbau eine bestimmte Teilmenge der Dienstleistungen gefordert werden.

3.2.6 Rückstau

Bei jeder Datenübertragung muss berücksichtigt werden, dass ein schneller Sender einen langsameren Empfänger überlasten kann. Um dies zu verhindern, können sich die Teilnehmer über einen Rückkanal abstimmen. Eine andere Möglichkeit ist, dass der Übertragungskanal selbst Puffer vorsieht oder den Sender über den Füllstand der Puffer informiert und so unterschiedliche Geschwindigkeiten ausgeglichen werden können.

3.2.7 Vorgänge bei verbindungsorientierten Diensten (4.1.2)

- Verbindungsaufbau (Bestätigt / Unbestätigt)
- Datenübertragung (Bestätigt / Unbestätigt)
- Verbindungsabbau (Bestätigt / unbestätigt)
- Verbindungsabbruch (Benutzerabbruch (User Abort), Medienabbruch (Provider Abort))

3.3 Datagrammdienste (4.1.3)

Es handelt sich immer um den unbestätigten Datentransfer. Die Schnittstellenereignisse (DtaReq und DtaInd) werden nicht in Beziehung zu früheren Ereignissen gesetzt. Es wird nicht für die Reihenfolgentreue der Nachrichten gesorgt.

3.4 Telekommunikationsprotokolle (4.2)

Die Verhaltensfestlegungen mit den Richtlinien zur Nutzung des unterliegenden Mediums bilden ein Telekommunikationsprotokoll. Die Beschreibung wird gegliedert in Instanzverhalten unter Abstraktion vom tatsächlichen Aufbau der Nutzdaten und Formatfestlegungen für die konkrete Syntax der Daten.

3.4.1 Ablauffestlegungen

Sie werden oft durch Mealy-Automatenmodelle mit folgenden Einschränkungen beschrieben.

- Partialität: Nicht für jede mögliche Eingabekombination wird ein Zustandsübergang definiert
- Nichtdeterminismus: Es kann zwischen mehrere "gleichwertigen "Übergängen gewählt werden
- Spontanübergänge: Ein Übergang kann ohne ein Schnittstellenereignis stattfinden
- Kommunikation und Systembildung: Mehrere Automaten können über Fifo-Queues miteinander gekoppelt werden, um ein komplexes System aus mehreren Schichten zu simulieren.
- Variablen: Der Zustandsraum wird durch Variablen strukturiert und somit klein gehalten
- Parameter: Ein- und Ausgaben können ebenfalls durch Variablen strukturiert werden
- Transitionsklausen: Es können Bedingungen für einen Übergang definiert werden und mit jedem ausgeführten Übergang Ergebnisse an Variablen zugewiesen werden

3.4.2 PDU-Formatfestlegungen

Unterhalb der Darstellungsschicht wird jeder PDU genau ein Bitmuster zugeordnet, erst oberhalb der Darstellungsschicht kann zwischen verschiedenen Kontexten gewählt werden. Zur Vereinbarung einer abstrakten Syntax wird oft die Syntaxbeschreibungssprache ASN.1 verwendet (Abstract Syntax Notation). Diese wird dann mithilfe von Standardcodierregeln in entsprechende Bitmuster umgesetzt.

3.4.3 Beispiel Alternating Bit Protocol

- Zieldienst O: Es handelt sich um eine fehlergesicherte, bestätigte Übertragung von Datagrammen in einer Richtung, zwei Dienstzugangspunkte
- Basisdienst U: abwechselnde unbestätigte Übertragung zwischen zwei Dienstzugangspunkten. Verlust oder Verdopplung kann nicht ausgeschlossen werden, Verfälschungen können auftreten, werden dem Dienstnehmer aber angezeigt (dies sind unrealistische Annahmen)
- Instanz E (Empfänger) und S (Sender): Durch Auswertung der Gestört-Anzeige und Verwendung einer Sequenzzahl (1-Bit, alternierend), werden fehlerhafte Daten (gestört oder in falscher Reihenfolge) erkannt und erneut angefordert.

3.4.4 Protokollkorrektheit

Es wird nachgewiesen, dass das Protokoll sich in jedem Fall entsprechend der Dienstbeschreibung verhält. Dazu wird jeder mögliche Ablauf auf seine Korrektheit untersucht. Ein aufwändiger Prozess. Zur Vereinfachung wird ein Graph gebildet, in dem zwischen jeder Ein- und Ausgabe ein Zustand liegt. Dabei können folgende Eigenschaften des Protokolls entdeckt werden

- Verzögerung: Eine Schleife im Graphen ohne Zieldienstereignisse, die endlos durchlaufen werden kann, aus der es aber auch Wege hinaus gibt
- Terminierung: Ein Zustand, von dem kein weiterer Weg möglich ist
- Quasi-Terminierung: Eine Schleife ohne Zieldienstereignisse, aus der es keinen Weg hinaus gibt
- Toter Ereignistyp: Ein Ereignistyp im Graphen, der niemals erreicht werden kann
- Toter Instanzenzustand: Ein Instanzzustand, der nie erreicht werden kann

3.4.5 Protokoll-Leistungsfähigkeit

Es wird untersucht, welche Leistung und Qualität ein Zieldienst erbringen kann, wenn Leistung und Qualität des Basisdienstes bekannt sind. Große PDUs haben den Nachteil, dass es wahrscheinlich ist, dass ein Bitfehler die PDU stört und somit eine erneute Übertragung der PDU nötig wird. Kleine PDUs hingegen verursachen weniger Aufwand bei der Neuübertragung, dafür verursacht jede PDU Aufwand, diese zu übertragen (Protokoll-Overhead).

Die Leistungsfähigkeit kann mit verschiedenen Methoden durchgeführt werden:

- theoretische Betrachtungen: Es wird mithilfe mathematischer Modelle versucht, eine Voraussage über die Abhängigkeit der Leistung des Zieldienstes von der Leistung des Basisdienstes zu treffen
- Simulation: Gerade bei mehreren voneinander abhängigen Protokollinstanzen wird es zunehmend schwieriger, eine theoretische Analyse durchzuführen. Hier kann eine Simulation nützlich sein.

4 Logische Architektur von Kommunikationssystemen (5)

Ein Telekommunikationssystem wird in Schichten eingeteilt, die aufeinander aufbauen, um einem Dienstnehmer bessere Leistungen anzubieten, als dies der Basisdienst kann. Dabei wird nach folgenden Regeln vorgegangen:

- Jede Schicht soll eine in sich logisch geschlossene Funktion erbringen
- Die Schnittstellen zwischen den Schichten sollen überschaubar und nur wenigen Konzepten folgend aufgebaut sein
- Die als Zieldienst angebotene Funktion soll unabhängig von der internen Implementierung der Schicht definiert werden können

Die Schichten können in zwei Bereiche gegliedert werden

- Transportsystem: Dies sind die dem physikalischen Medium näheren unteren Schichten, die die grundlegenden, fehlerbehobenen Kommuni-kation ermöglichen
- Anwendungsorientierte Schichten: Diese Schichten stellen Dienstleistungen zur Verfügung, die Dienste zur Nutzung durch eine Anwendung bereitstellen, die aber über die reine Datenübertragung hinausgehen (Sitzungsverwaltung, Datendarstellung, Anwendungsprotokolle) Transportsystem

Das Transportsystem soll unabhängig von der Art der zu unterstützenden An-wendung, sondern universell einsetzbar sein. Es sollen alle technischen Aspekte der Datenübertragung über physikalische Medien verdeckt werden, sodass die anwendungsorientierten Schichten frei von Transportaspekten ausgelegt wer-den können.

Da die Dienste des Transportsystems direkt von Anwendungsprozessen verwendet werden, sollte der Transportdienst SAPs für jeden Anwendungsprozess, im Idealfall sogar mehrere je Prozess zur Verfügung stellen und so einem Prozess die direkte Adressierung eines Anderen gegenüber dem Transportsystem zu ermöglichen. (Ende-zu-Ende-Kommunikation)

Der Transportdienst sollte die Dienstleistungen DatagrammTransfer für verbindungslose Übertragungen und Verbindungsauf- und Abbau und (bestätigte) Datenübertragung für verbindungsorientierten Transfer bereitstellen. Zum

Verbindungsaufbau wird hier eine Abstimmung über die Qualität der Verbindung nötig. Der Sender teilt seine Qualitätsanforderungen dem Transportdienst mit. Dieser stuft die Anforderungen auf so zurück, dass er in der Lage ist, sie zu erfüllen und übermittelt diese veränderte Verbindungsanforderung an den Empfänger. Dieser kann die Anforderungen seinen Fähigkeiten entsprechend weiter verringern und schickt eine Bestätigung an den Sender zurück (Bestätigter Verbindungsaufbau). Dieser kann nun entscheiden, ob er die Verbindung mit den ausgehandelten Parametern nutzen möchte, oder sie direkt wieder ab-bauen.

4.0.6 Aufgaben der Transportschichten

Der Transportdienst muss folgende Eigenschaften erbringen, die das physikalische Medium nicht in der Lage ist, zu liefern.

- Zuverlässigkeit
- Flexible Dienstqualität
- Variable Nachrichtenpaket-Länge bzw. gar keine Paketeinteilung der Daten
- Synchronisation: Ein Dienstnehmer muss nicht auf bestimmte zeitliche Anforderungen des Mediums Rücksicht nehmen.
- Konnektivität: Ein physikalischer Kanal ist im Allgemeinen eine Verbindung zwischen zwei Endpunkten. Das Transportsystem muss das Routing von Daten übernehmen.
- Adressierung: Es wird ein einheitlichen Adressraum geschaffen, der i. a. mehrere physikalische Medien umfasst
- Mehrfachnutzung von Verbindungen

4.0.7 Aufgaben der anwendungsorientierten Schichten

- Beseitigung der Problematik der verschiedenen Quellencodierungen, d.h. der unterschiedlichen binären Codierung von Nachrichten (Darstel-lungsschicht)
- Die Steuerung von mehrere Austauschvorgänge umfassenden Kommunikationsbeziehungen (Kommunikationssicherungsschicht)

5 Protokollmechanismen (6)

Ein Kommunikationsdienst kann niemals völlig zuverlässig arbeiten. Es können immer Störungen auftreten, die von der Diensthierarchie nicht verdeck werden können. Daher kann es keine zentrale Instanz geben, die den Gesamtablauf nach einem festgelegten Muster steuert. Die lose gekoppelten Instanzen müssen nebenläufig arbeiten. An jedem Ort befindet sich eine autonome Instanz, die entsprechend ihrer Sicht des Gesamtsystems mit den anderen Instanzen koope-riert. Um eine Aufgabe zu lösen, müssen sich aber alle Instanzen an ein festge-legtes Verhaltensschema halten, an ein Protokoll.

5.1 Zuteilung geteilter Medien (6.1)

Solange ein Medium nur im Simplex-Betrieb zwischen einem Sender und einem Empfänger betrieben wird, ist keine explizite Zuteilung des Mediums erforderlich. Aber schon beim Halbduplexbetrieb muss das Senderecht jeweils einem Teilnehmer vergeben werden. Nun darf kein Empfänger mehr senden, da die Datenübertragung sonst gestört wird. Im Vollduplex-Betrieb auf einem geteilten Medium mit vielen Telnehmern ist diese Problematik noch größer. Verfahren zur Behandlung dieser Problematik sind

- Zeitmultiplex: Jedem Teilnehmer wird das ganze Medium für eine bestimmte Zeit (Zeitschlitz, Timeslot) zugewiesen.
- Frequenzmultiplex: Jedem Teilnehmer wird ein Frequenzband zugewiesen, auf dem er ständig übertragen darf. Die Zuteilung kann statisch oder dynamisch erfolgen und nach zentraler oder dezentraler Kontrolle

5.1.1 Zentrale Zuteilung

Bei der zentralen Zuteilung werden die Stationen von einem Master zyklisch abgefragt, ob sie Daten übertragen möchten. (Polling)

5.1.2 Dezentrale Zuteilung

Vorteil der dezentralen Zuteilung ist die einfache Ein- und Ausgliederung von Stationen. Nachteil ist, dass eine Datenübertragung innerhalb eines bestimmten Zeitrahmens nicht garantiert werden kann.

- Aloha (slotted / non-slotted):
  Bei Sendebedarf sendet ein Teilnehmer seine Daten. Bei Verfälschung werden die Daten ignoriert und es erfolgt nach zufälliger Wartezeit eine erneute Übertragung. Bei slotted darf nur zu bestimmten Zeitpunkten gesendet werden. Damit vermindert man die Zeit, zu der eine Kollision stattfinden kann.
- CSMA (slotted / non-slotted, 1- / p- / non-persistent):
  Ein Sender beobachtet das Medium auf Übertragungsaktivität. Wenn er senden möchte und das Medium ist aktiv, berechnet er nach einem Backoff-Algorithmus eine zufällige Wartezeit und versucht erneut zu senden. Bei 1-persistent wird solange das Medium beobachtet, bis es frei zu sein scheint und dann gesendet (Schlecht, wenn viele Sender gleichzeitig übertragen wollen). Bei p-persistent wird solange gewartet, bis das Medium frei zu sein scheint, dann eine Zufallszahl z berechnet. Ist z kleiner als p, so wird gesendet, ist z größer, so wird der ganze Vorgang erneut begonnen. So wird vermieden, dass nach einer längeren Belegtphase mehrere potentielle Sender gleichzeitig zu senden beginnen.
- CSMA/CD:
  Zusätzlich zu CSMA wird hier während der eigenen Übertragung das Medium beobachtet und Verfälschungen des eigenen Sendesignals festgestellt. Bei einer Kollision wird die Übertragung sofort unterbrochen

und ein Jam-Signal gesendet. Damit wird das Medium nach einer Kollision schnell wieder für die nächste Übertragung frei. Ein erneuter Sendeversuch wird erst nach einer durch einen Backoff-Algorithmus be-rechneten Wartezeit gestartet.

- Zirkulierendes Recht:
  Über den Stationen ist eine zyklische Ordnung definiert. Ein Token zirkuliert zwischen den Stationen, der das Senderecht symbolisiert. Will eine Station senden, wartet sie auf den Token, entnimmt ihn dem Netz und sendet dafür ihre Daten. Nach erfolgter Übertragung wird der Token wieder eingespeist. Es wird zwischen Multiple Token (Token wird direkt nach erfolgter Datenübertragung eingespeist), Single Token (Token wird eingespeist, bis die Daten durch den Ring gelaufen sind und der Anfang der Daten wieder bei der sendenden Station eintrifft) und Single Packet (Token wird eingespeist, wenn die Daten wieder vollständig bei der sendenden Station eingetroffen sind) unterschieden. In LANs ist zwischen diesen Verfahren fast kein Unterschied festzustellen (Leitungslänge 500m, Paketlänge ca. 16km). Die Verfahren können bei echten Ringen (Token Ring) als auch bei Sammelleitung (Token Bus) unter willkürliche Definition einer zyklischen Ordnung verwendet werden. Problematische ist der Verlust des Tokens (führt zur Wahl eines Stellvertreters, der einen neuen Token generiert) und mehrere Token im Ring.

5.1.3 Backoff-Algorithmus

- Statisch: Es wird eine zufällige Wartezeit mit fester Verteilungsfunktion berechnet.
- Dynamisch: Es wird eine Verteilung gewählt, die von der Anzahl vorangegangener fehlgeschlagener Sendeversuche abhängt.

Üblich sind truncated binary exponential Backoff-Algorithmen (zum Beispiel bei Ethernet (1-persistent)). Mit steigender Anzahl an Versuchen steigt der Erwartungswert der Zeit exponentiell bis zu einem Maximalwert an.

5.1.4 Leistungsaspekte

Von Interesse sind der Durchsatz und die Zeitdauer für die erfolgreiche Übertragung eines Paketes. Diese sind abhängig von physikalischen Größen (Medien-Länge, Ausbreitungsgeschwindigkeit und Sende/ Empfangsbitrate) und Kenngrößen des Medien-Betriebs (Anzahl der Stationen, Paketgröße, angebote-ne last).

5.1.5 Maßnahmen zur Steigerung der Leistung

- Verringerung der Belegung des Mediums durch eine Kollision: Bei CSMA/CD wird die Übertragung eines gestörten Paketes abgebrochen
- Verringerung der Verletzlichkeitsdauer eines Paketes: Dies wird durch slotted-Varianten erreicht

- Drosselung der Sendeversuche bei hoher Last: Wie das Verhalten bei Stau auf Autobahnen: Bei zu hoher angebotener Last sinkt der Durchsatz bis zum Stillstand. Verbesserung durch (exponentielle) Backoff-Algorithmen

5.2 Fehlerbehandlung (6.2)

Es muss zwischen Nachrichtenverfälschungen (Fehler in einer PDU) und Zusammenhangs-Besonderheiten (Reihenfolgevertauschung, Nachrichtenverlust, Phantomnachrichten, Duplizierung von Nachrichten) unterschieden werden. Zusammenhangs-Besonderheiten werden oft durch die unterschiedliche Sicht von Stationen auf den Zustand der Übertragung verursacht (zum Beispiel nach temporärem Ausfall eines Mediums).

5.2.1 Verfahren zur Fehlerbehebung

- Einführung von Kontrollparametern zur Reihenfolgesicherung
- Einführung von Redundanz in den PDUs zur Kontrolle des Inhalts

5.2.2 Fehlererkennender Code (CRC)

Bei der Verwendung fehlererkennender Codes wird jedem Quell-Code eine Prüfsumme angehängt (Kanalcode). Beim Empfänger wird versucht, diesen Kanalcode wieder auf ein Quell-Codewort abzubilden. Ist dies nicht möglich, so ist die Nachricht verfälscht worden. Es gibt aber immer Verfälschungen, die ein mögliches Kanalcodewort in ein anderes mögliches überführen und so unentdeckt bleiben. Übliches Verfahren ist die CRC-Prüfsumme. Das Quellcodewort wird mittels Modulo-2-Artihmetik (Überträge werden weggelassen) durch einen standardisierten Divisor geteilt und der sich ergebende Rest an das Quellcodewort angehängt. Beim Empfänger wird dieses Kanalcodewort erneut durch den gleichen Divisor geteilt. Sind die Daten fehlerfrei, so darf sich kein Rest ergeben. In modernen Protokollen ist die Fehlererkennung eine dem Protokoll untergelagerte Hilfsfunktion.

5.2.3 Sequenzzahlen und Zeitstempel

Zur Erkennung von Reihenfolge-Besonderheiten werden Sequenzzahlen verwendet, die auf einen vorgegebenen Wertebereich beschränkt sind. Beim Alternating Bit Protocol zum Beispiel auf 1 Bit. Ein den Sequenzzahlen ähnliches Verfahren ist die Benutzung logischer Uhren. Es wird sich nicht auf die Ganggenauigkeit rechnergesteuerter Uhren verlassen, sondern logische Uhren werden durch bestimmte Protokollmaßnahmen weitergeschaltet und definieren so eine zeitliche Ordnung zwischen Kommunikationsereignissen.

5.2.4 Quittieren

Ein Empfänger quittiert den Empfang entweder jeder Nachricht (Stop-and-Go-Verfahren, wie beim Alternating Bit Protocol, wenn man die 1-Bit-Sequenzzahl als Quittung auffasst) oder er quittiert den Empfang mehreren Nachrichten (Sliding-Window-Verfahren). Möglich ist bei Duplex-Betrieb das Piggy-Backing. Quittungen werden dann innerhalb von Nutzdatenpaketen in die Gegenrichtung

übertragen. Es können Vereinbarungen getroffen werden, nur fehlerhafte oder nur fehlerfreie Nachrichten zu quittieren.

5.2.5 Zeitüberwachung (Timeout)

Werden zum Beispiel nur positive Quittungen verschickt, so kann der Sender nach dem Ausbleiben einer Quittung über eine längere Zeit vermuten, dass ein Fehler aufgetreten ist und die entsprechende Nachricht erneut versenden. Die Festlegung der Zeitdauer ist wichtig. Ist sie zu kurz kann auch bei korrekter Funktionsweise eine Störung vermutet werden und so zusätzliche Kommunikation verursacht werden. Ist sie zu lang, vergeht eine zu lange Wartezeit, bis eine Störung entdeckt wird.

5.3 Fehlerbehebung (6.3)

Unterschieden wird zwischen zwei Verfahren, der Forward Error Correction (FEC) und dem Automated Repeat Request (ARQ). Bei FEC werden Quellcodewörter auf Kanalcodewörter mit Redundanz abgebildet, sodass der Empfänger die Möglichkeit hat, fehlerhafte Kanalcodewörter trotzdem auf entsprechende Quellcodewörter abzubilden. Da die Berechnung aufwendig ist, wird dieses Verfahren nur bei extrem teuren und langsamen Verbindungen benutzt, wenn eine erneute Anforderung der Daten nicht vertretbar ist. Bei ARQ wird eine verfälschte Nachricht einfach ignoriert und somit auf den Verlust der Nachricht zurückgeführt. Dies vereinfacht den Protokollentwurf. Durch den Verlust handelt es sich um eine Reihenfolgebesonderheit, die durch automatische Wiederholung dieser Nachricht, oder das Zurücksetzen in einen früheren Zustand behoben werden kann.

5.4 Längenanpassung (6.4)

Während die unteren Schichten fast immer mit Paketen fester oder nur in grenzen variabler Länge arbeiten, wollen vor allem die anwendungsorientierten Schichten beliebig lange Datenblöcke oder -Ströme übertragen. Dazu werden die Daten entweder fragmentiert (auf mehrere Pakete aufgeteilt) oder geblockt, also mehrere kleine Dateneinheiten in ein Paket verpackt.

5.5 Flusskontrolle (6.5)

Es kann passieren dass Sender und Empfänger unterschiedlich schnell sind. Ist der Empfänger langsamer als der Sender, so muss dies berücksichtigt werden. Es wird zwischen folgenden Verfahren unterschieden:

- Verlust beim Empfänger: Es wird auf eine Reihenfolgebesonderheit zurückgeführt und damit durch erneute Anforderung der Nachricht behoben. Sehr ineffizient.
- Flussbeschränkung im Sender: Der Sender beschränkt seine Senderate auf einen für den Empfänger passenden Wert. Zum Beispiel durch das Lecker-Eimer-Verfahren.

- Flussabstimmung zwischen Sender und Empfänger: Üblicherweise vergibt der Empfänger dem Sender einen Kredit, den dieser mit dem Senden von Daten vermindert. Ist der Kredit aufgebraucht, darf der Sender erst wieder senden, wenn er einen erneuten Kredit vom Empfänger erhalten hat. Oft wird ein Kredit von 1 (also Stop and Go) verwendet, ähnlich dem Quittieren jeder Nachricht. Eine andere Möglichkeit ist, beim Aufbau einer Verbindung bereits entsprechende Ressourcen beim Empfänger zu reservieren oder den Aufbau der Verbindung abzubrechen bzw. die Anforderungen der Verbindung herabzustufen.

5.6 Weiterleiten (6.6)

Ein Transportdienst muss die Weiterleitung zwischen mehreren Stationen unterstützen, während die Stationen über mehrere Medien hinweg miteinander verbunden sind. Dazu dient die Weiterleitung. Es gibt zwei Verfahren, den Weg der Daten über dazwischenliegende Stationen zu bestimmen.

- Quellengesteuertes Routing: Der Absender bestimmt im Voraus den Weg der Daten
- Transitknotengesteuertes Routing: Jede Station, die die Daten empfängt, aber selbst nicht End-Empfänger ist, entscheidet selbst (meist auf Grundlage statischer oder dynamischer Tabellen), an welche Station er die Daten weiterschickt. Dabei muss darauf geachtet werden, dass die Daten irgendwann ihren Empfänger erreichen und nicht unter Umständen endlos im Netz zirkulieren (Bei IP gibt es in jedem Paket ein Feld TimeToLive, das mit einem Wert initialisiert wird (oft 64) und an jeder Station um eins verringert wird. Erreicht das Feld 0, bevor es am Ziel ist, wird es verworfen (Zurückführung auf Verlust der Nachricht))

Es wird unterschieden, ob für jedes Paket ein neuer Weg gesucht wird, oder ob alle Pakete einer Verbindung über denselben, einmal gesuchten Weg geleitet werden.

Eine einfache Möglichkeit des Routings ist das Fluten. Ein empfangenes Paket wird an alle anderen Stationen gesendet.

5.7 Überlastkontrolle (6.7)

Die Überlastkontrolle soll vermeiden, dass die Übertragungsleistung dann, wenn mehr Last angeboten als abgenommen wird, absinkt. Die Überlastkontrolle ist für ein Gesamtnetz verantwortlich, während die Flusskontrolle für einen einzelnen Übertragungskanal zuständig ist.

Es können entweder vorab Ressourcen reserviert werden, um eine Überlast zu verhindern, oder es werden Maßnahmen eingeleitet, die eine bestehende Überlast bekämpfen. Zur Beseitigung von Überlast können zum Beispiel an einer Station, die mit der Verarbeitung von Nachrichten überlastet ist, diese einfach ignoriert werden oder es können Verfahren zur Strom-Drosselung eingesetzt werden:

- Flusskontroll-Mechanismen, um die angebotene Leistung zu beschränken
- Zirkulierende Berechtigungsmarken

- Adaptive Routingverfahren, um Engpässe zu umgehen
- Veränderung von Protokoll-Parametern (Paketlänge, Timeout-Konstanten, etc.)

5.8 Übertragungsleistungsanpassung (6.8)

Wenn sich die geforderte Bandbreite eines Ziel- und eines Basisdienstes unterscheiden, so können Verfahren zur Übertragungsleistungsanpassung verwendet werden. Entweder werden mehrere Verbindungen des Zieldienstes über eine Leitung des Basisdienstes übertragen (Multiplexen) oder es wird eine Zieldienst-Verbindung gleichzeitig über mehrere Basisdienst-Verbindungen betrieben (Bündelung von Verbindungen). Beide Verfahren sollen eine möglichst gute Ausnutzung der vorhandenen Ressourcen sicherstellen und so Kosten minimieren.

5.9 Schutz und Sicherheit (6.9)

Im ISO/OSI-Modell werden 5 Sicherheitsdienste unterschieden

- Authentifikation: Sicherstellen der Identität der Kommunikationspartner
- Zugriffskontrolle: Betriebsmittel dürfen nur von Berechtigten verwendet werden
- Vertraulichkeit: Das Mithören einer Datenübertragung muss verhindert werden. Dies kann bis zur Erzeugung von Dummy-Traffic führen, um Verkehrsflussanalysen zu verhindern.
- Integrität: Die Verfälschung von Daten muss verhindert werden.
- (Rechts-)Verbindlichkeit: Das Senden bzw. der Empfang von Nachrichten kann bewiesen werden.

Um diese Ziele zu erreichen, können unterschiedliche Mittel eingesetzt werden

- Verschlüsselung (von Nachrichten oder auch Passwörtern, etc.)
- Sicherheitsprotokolle (Schlüsselverteilung, Authentifizierungsprotokolle)
- Zugriffskontrollmechanismen (Access Control Lists (ACL), befristete Tickets)
- Physikalischer Schutz der Subsysteme

5.10 Verwaltung (Management) (6.10)

Vor allem in größeren Netzen entsteht ein erheblicher Verwaltungsaufwand, der folgendermaßen gegliedert werden kann:

- Fehlerverwaltung
- Abrechnung
- Konfigurationsverwaltung

- Leistungsverwaltung
- Sicherheitsverwaltung

Zur Unterstützung dieser Prozesse wird ein System aus Agenten und Managern vorgeschlagen (OSI-Modell). Die Agenten sind in Form von Prozessen implementiert und stellen Schnittstellen zu den verwaltbaren Parametern und zu Ereignissen einzelner Betriebsmittel zur Verfügung. Die Management-Prozesse erlauben den Zugriff auf die Manager und bilden die Benutzerschnittstelle. Sie werden unterteilt nach Managern für einzelne Betriebsmittel, für eine Schicht oder für ein gesamtes Netz.

6 Dienstleistungen der Schichten des OSI-Modells

6.1 Schicht 1: Physical Layer

In dieser Schicht ist definiert, wie das physikalische Medium zur Übertragung der Daten beschaffen ist, also zum Beispiel, um welche Übertragungstechnik (Kupfer, Glasfaser, Funk, etc.) es sich handelt, wie die einzelnen Stationen an das Medium angeschlossen werden (Steckverbindungen, etc.) und wie einzelne Bits über das Medium signalisiert werden können (Spannungspegel, Stromstärken, o. ä.).

6.2 Schicht 2: Data-Link Layer

Hier werden auf der sendenden Seite Datenpakete in einzelne Bits zur Übertragung über das physikalische Medium gewandelt und auf der empfangenden Seite wieder zurück in Pakete zur Weitergabe an das Network Layer. Diese Schicht wird unterteilt und zwei Subschichten.

6.2.1 Schicht 2a: Media Access Control

An dieser Stelle wird festgelegt, wie mehrere Stationen auf ein geteiltes Medium zureifen dürfen. Verfahren hierzu sind in LANs Ethernet mit CSMA/CD (Carrier Sense Multiple Access / Carrier Detect), Token Ring, zum Anschluss an ein WAN aber auch PPP(Point to Point Protocol).

6.2.2 Schicht 2b: Logical Link Control

Hier wird eine grundlegende Fehlerkorrektur (zum Beispiel durch CRC) durchgeführt und die Flusskontrolle eingeführt (damit ein schnellerer Sender keinen langsameren Empfänger überlastet). Hier werden außerdem die SAPs (Service Access Points) definiert.

6.3 Schicht 3: Network Layer

In dieser Schicht findet die Adressierung und Routenbestimmung der Datenpakete statt. Es werden logische (Computer)-Namen in physikalische Adressen gewandelt (mit Arp und Rarp bei Ethernet). Zu große Datenpakete können hier zur Übergabe an das Data Link Layer in kleinere Einheiten aufgespalten werden.

Verwendete Protokolle: IP, Arp, Rarp, ICMP, RIP, IPX

6.4 Schicht 4: Transport Layer

Hier wird das erste Mal im Protokoll-Stapel eine verbindungsorientierte Datenübertragung angeboten. Außerdem stellt das Transport Layer weitergehende Fehlerkorrektur und die korrekte Reihenfolge der Datenpakete (durch Sequenznummern) sicher. Verlorengegangene Datenpakete oder doppelt eingetroffene werden durch diese Schicht durch erneute Anforderung bzw. Ignorieren der Pakete ebenfalls korrigiert. Diese Schicht sendet Quittungen zur Bestätigung eingegangener Pakete bzw. zum Veranlassen einer erneuten Übertragung. In dieser Schicht werden zu übertragende Datenströme in einzelne Pakete verpackt (Fragmentierung bzw. Bündelung).

6.5 Schicht 5: Session Layer

In dieser Schicht wird ein längerer Dialog bzw. eine Interaktion zwischen zwei kommunizierenden Stationen verwaltet. Dazu können zum Beispiel Wiederaufsetzpunkte (Checkpoints) vereinbart werden, die es ermöglichen, nach dem Ausfall einer Verbindung an einem solchen Checkpoint die Kommunikation wieder fortzusetzen. Hier werden zusätzlich Assoziationen zwischen Stationen angelegt (diese werden normalerweise einfach auf eine Verbindung der Netzwerk-Schicht abgebildet) und es können Sende-Rechte ausgetauscht werden (zum Beispiel durch Übertragung eines Sende-Token).

6.6 Schicht 6: Presentation Layer

Die Darstellungsschicht wird dazu verwendet, die Darstellung von Daten zu verändern. Werden zum Beispiel Daten zwischen verschiedenen Rechnerarchitekturen ausgetauscht, könnte diese Schicht dafür sorgen, dass die Daten der einen Architektur zunächst in eine einheitliche Netzwerk-Syntax konvertiert werden und beim Empfänger wieder in die entsprechende Rechner-Syntax der empfangenden Architektur. Es ist weniger aufwendig, eine gemeinsame Netzwerk-Syntax der Daten zu vereinbaren, als in dieser Schicht Konverter für alle möglichen Kombinationen an Datenformaten vorzusehen.

6.7 Schicht 7: Application Layer

In dieser Schicht werden Protokolle implementiert, die direkt mit einer Anwendung interagieren. Diese Schicht enthält meist eine Sammlung oft verwendeter Protokolle wie zum Beispiel (http, smtp, ftp, smb (Windows Dateifreigabe, etc.)

6.8 Anmerkungen

Vergleich zwischen ISO/OSI und TCP/IP Application Application Presentation Session Transport Transport Network Internet Data Link Subnet Physical

Es ist auch im ISO/OSI-Modell üblich, die Schichten 2 bis 4 auf Betriebssystem-Ebene zu implementieren, sodass allen Anwendungen eine gemeinsame Schnittstelle zum Netzwerk angeboten werden kann; die Schichten 5, 6 und 7 werden aber auch hier fasst immer zusammengefasst in einem Anwendungsprozess implementiert.

7 Rechnernetzanwendungen - Paradigmen (9.2.2)

Im Folgenden sind einige übliche Vorgehensweisen für die Implementierung von verteilten Anwendungen aufgeführt.

7.1 System kommunizierender Prozesse

Die Anwendung wird in Form mehrerer miteinander durch Nachrichtenaustausch kommunizierender Prozesse implementiert. Jeder Prozess verwaltet seine Betriebsmittel privat für sich, und sie werden von ihm exklusiv genutzt. Dies ist ein sehr grundlegendes Konzept und implementierungsnah.

7.2 Clients und Server

Hier handelt es sich um eine zurzeit weit verbreitete Variante zum System kommunizierender Prozesse. Es treten zwei Gruppen von Prozessen auf:

- Server, die sich passiv verhalten, dafür aber gemeinsam genutzte Betriebsmittel zur Verfügung stellen. Meist gibt es mehr Clients als Server.
- Clients: Clients nutzen die von den Servern bereitgestellten Ressourcen bzw. vergibt Aufträge an die Server.

Bei komplexen Client-Server-Systemen können Server selbst wieder die Rolle von Clients übernehmen, die Aufträge an andere Server vergeben. Oft wird dann eine hierarchische Struktur verwendet. (Multi-Tier-Architektur)

7.3 Geteilte Objekte und kooperierende Prozesse

Falls im Modell die Server keine andere wesentliche Aufgabe erfüllen, als Betriebsmittel (hauptsächlich Zustandsspeicher) zur Verfügung zu stellen, so kann man in einer abstrakten Sicht diese Ressourcen als Objekte auffassen, die von den Clients geteilt, also gemeinsam genutzt werden. Hierbei können objektorientierte Vorgehensweisen, wie Kapselung, Vererbung und Zusammenfassende Beschreibung einer Objektmenge als Klasse verwendet werden.

Es gibt einige Unterschiede zur traditionellen objektorientierten Strukturierung einer lokal ablaufenden Anwendung:

- Bei verteilten Anwendungen muss immer die Nebenläufigkeit der beteiligten Prozesse berücksichtigt werden. Eine strenge Serialisierung würde zu schlechter Leistungsfähigkeit führen.
- Objekte und Prozesse sind an Orte gebunden. Aufrufe an entfernte Objekte benötigen eine längere Zeit als bei lokalen Objekten
- Es muss eine gewisse Fehlertoleranz vorgesehen werden. Es wird dazu meist zwischen flüchtigen und gespeicherten (persistenten) Objekten unterschieden, deren Zustand wiederhergestellt werden kann

7.4 Datenbank und kooperierende Prozesse

Nebenläufige Prozesse, die auf einen gemeinsamen Datenspeicher zugreifen, gibt es schon lange in Form von Datenbanken (die nicht speziell für verteilte Anwendungen entworfen wurden). Dieses Modell geht also von Prozessen aus, denen zur Kooperation eine zentral gesteuerte Datenverwaltungsanwendung (DBMS, Database Management System) zur Verfügung steht.

Man muss die Nebenläufigkeit der Prozesse nicht explizit berücksichtigen, sondern kann jeden Prozess für sich getrennt entwerfen, da Zugriffssynchronisation und Konsistenzerhaltung im DBMS implementiert sind. Nachteil dieses Modells ist, dass die zentrale Datenbasis oft einen Leistungsengpass darstellt und durch sie die Erweiterbarkeit des Systems eingeschränkt werden kann.

Es gibt Varianten dieses Schemas, bei dem von mehreren Datenbanken ausgegangen wird, die jeweils nur einen Teil ihrer Datenbasis global zur Verfügung stellen. In Multidatenbanksystemen wird die logische Zentralität der Daten fast völlig aufgehoben. Hier stehen dann aber meistens keine Transaktionen mehr zur Verfügung.

7.5 Gemeinsame Aktionen und kooperierende Prozesse

Bei diesem Modell steht nicht die zentrale Datenbasis im Fordergrund, sondern der einzelne Prozess. Es werden Ausführungsphasen der Prozesse unterschieden, in denen nur lokal gearbeitet wird (diese Phasen sollten der Leistung wegen bevorzugt werden) und Phasen der Kooperation, in denen die Prozesse auch Betriebsmittel (z.B. Zustandsspeicher) teilen können. Es wird für jeden Prozess eine oder mehrere Rollen definiert, die er entsprechend der Ausführungsphase annimmt.

7.6 System aktiver Objekte

Der Ansatz der Objektorientierung kann auch für verteilte Anwendungen angepasst werden. Hier enthalten die Objekte aber zusätzlich aktive Elemente, die von sich aus zustandsverändernd wirken können.

7.7 Nachrichtenprozesse

Anstatt Prozesse als fest an einen Ort gebunden zu betrachten, werden hier die Nachrichten als Mittel zum Transport von Prozessen (nicht im Sinne von Betriebssystemprozessen, sondern als Folge von Aktivitäten) angesehen. Es wird bei der Implementierung aber sicher vermieden, echten Programmcode zwischen Stationen auszutauschen. Stattdessen wird dieses Modell zu einer späteren Entwurfsphase eher in ein Knotenprozesssystem überführt.

7.8 System wandernder Objekte

Dies ist eine Kombination des Modells aktiver Objekte mit Nachrichtenprozessen. Es handelt sich also um Objekte die zwischen Stationen wandern können.

8 Systemunterstützung (9.3)

Nachdem eine verteilte Anwendung entworfen wurde, muss sie implementiert werden. Dazu finden sich verschiedene Komponenten, die die Implementierung unterstützen und erleichtern sollen.

8.1 Netzwerkfähige Station

Eine netzwerkfähige Station ist die einfachste Form der Unterstützung. Es handelt sich hierbei einfach um Es handelt sich dabei um eine Plattform, auf der in irgendeiner Form ein Transportdienst zur Verfügung gestellt wird.

8.2 Verteilte Programmiersprachen

Hierbei handelt es sich meist um eine Erweiterung einer bekannten Programmiersprache um Elemente zur Definition nebenläufiger Prozesse und Vorrichtungen zur Interprozess-Kommunikation. Hierbei wird zunächst aber meist davon ausgegangen, dass die Prozesse trotzdem lokal auf einer Maschine laufen. Es gibt aber erste Ansätze, die eine Anwendung direkt auf ein ganzes Rechnernetz abbilden. Dabei muss aber immer berücksichtigt werden, dass es sich dann um eine lose, unzuverlässige Koppelung handelt und dass meist sehr unterschiedliche Maschinen zur Ausführung verwendet werden (Heterogenität der Ausführungsumgebung).

Es gibt im Grunde zwei verschiedene Ziele, die mit verteilten Programmiersprachen verfolgt werden. Zum einen das Supercomputing, wobei ein Problem parallelisiert wird und dann auf mehreren Rechnern gleichzeitig bearbeitet wird und "echte "verteilte Anwendungen, wo mehrere Prozesse zur Erbringung einer Aufgabe miteinander kooperieren.

8.3 Verteilte Betriebssysteme

Ziel ist es, einem Prozess eine möglichst homogene Schnittstelle zu schaffen. Dazu wird ein Betriebssystem verwendet, welches den Charakter des Rechnernetzes mehr oder weniger stark verdeckt. Das Betriebssystem ist logisch für das gesamte Rechnernetz ein Betriebssystem, so dass Unterschiede der einzelnen Stationen verdeckt werden. Diese Lösung ist aufwendig, weil normalerweise immer lokale Betriebsmittel mit hohem Aufwand netzweit zur Verfügung gestellt und verwaltet werden müssen. Solche Betriebssysteme beschränken sich daher meist auf ein LAN und weitgehend gleiche Stationen.

8.4 Netzwerk-Betriebssysteme

Ein Netzwerk-Betriebsystem versucht nicht mehr, das gesamte Rechnernetz zu verdecken. Es stellt lediglich Schnittstellen zur Verfügung, sodass Prozesse eine homogene Umgebung zur Verwendung von Kommunikationsdiensten vorfinden. Bei Systemen desselben Herstellers reichen solche Dienste bis zur netzweiten gemeinsamen Administration, bei auf Offenheit ausgelegten Systemen beschränkt sich die Funktionalität oft auf Mail und den netzweiten Zugriff auf Dateien.

8.5 Netzwerk-Betriebssystem nach dem Client-Server-Modell

Gerade bei offenen Systemen wird dieses Modell oft verwendet. Das Netzwerkbetriebssystem besteht dabei aus einer Menge voneinander unabhängiger produktiver Dienste und Dienstnehmer. Jede Station kann durch geeignete Server-Programme Dienste zur Verfügung stellen, die andere Stationen mithilfe von Clientprogrammen nutzen können. Jede Station kann dadurch in ihrem Dienstangebot dynamisch angepasst werden. Der Client muss nicht unbedingt ein eigenständiger Prozess sein, es kann sich auch um eine Prozedursammlung (Library) handeln, die beim übersetzen eines Programms dazugebunden wird. Unix ist ein solches Netzwerk-Betriebssystem.

8.6 Verteilte Datenbanken

Dem Entwurfsmodell Datenbank und kooperierende Prozesse kann man sehr nahe kommen, wenn man im Netz eine DBMS zur Verfügung stellt. Ob es sich dabei um eine zentrale oder verteilte Datenbank handelt, ist unerheblich, da der Zugriff in jedem Fall zentral verwaltet wird. Bei verteilten Datenbanken können die Daten tatsächlich über mehrere Stationen verteilt sein, oder auf mehrere Stationen repliziert werden, in jedem Fall wird (vor allem bei Transaktionen) der Kommunikationsaufwand wesentlich erhöht.

8.7 Beispiele der Systemunterstützung

- Unix-Sockets: Sie stellen eine übliche Schnittstelle zwischen Anwendung und Transportdienst dar. Sockets sind lokale Interprozesskommunikationsobjekte, die wie puffernde Ports funktionieren.
- Sun-RPC (Remote Procedure Call): Ein Hilfsmittel, bestehend aus einer Bibliothek, die grundlegende Funktionen bereitstellt, mit deren Hilfe Prozeduren in entfernten Prozessen aufgerufen werden können und einem Hilfsprogramm (rpcgen), welches speziell gekennzeichnete Prozeduraufrufe in einem Programm durch Funktionen aus dieser Bibliothek ersetzt. Außerdem gibt es ein Server-Programm, welches einen Verzeichnisdienst bereitstellt (portmapper). Damit kann ein Programm durch einen symbolischen Dienstnamen in einen Identifikator für einen Prozess übersetzen, um dessen RPC-Dienste in Anspruch zu nehmen. Dadurch muss zur Übersetzung des Programms nicht der genaue Prozess mir den aufzurufenden Prozeduren bekannt sein.
- OSF-DCE (Open Software Foundation Distributed Computing Environment): Framework, welches mehrere Services zur Verfügung stellt (RPC, Directory / Naming, Security, Threads, Time (synchronisierte Zeit) und Datasharing (verteiltes Dateisystem und Personal Computer Integration). Wird inzwischen nicht mehr verwendet, wurde von CORBA verdrängt.
- ANSA (Advanced System Network Architecture):
- CORBA (Common Object Request Broker Architecture): System, ähnlich zu RPC, welches aber Objektorientierung versteht. Der ORB (Object Request Broker) ist ein Prozess zur Namensauflösung (Eigentlich Trading,

nicht ein Prozess wird identifiziert, sondern anhand von Attributen ein beliebiger Server) ähnlich dem portmapper bei RPC.

- Java: System bestehend aus der Programmiersprache Java, der Definition einer virtuellen Maschine und ihrer Maschinensprache Bytecode und einer umfangreichen API. Dieses System unterstützt dabei eine RMI (Remote Method Invokation), eine Corba-API, Ojektserialisierung (Zur Übertragung von Objekten über ein Netz verwendbar). Durch die virtuelle Maschine sollen Java-Programme von der Hardware, auf der sie ausgeführt werden, unabhängig sein, solange eine geeignete VM zur Verfügung steht. (Für die meisten üblichen Plattformen heute verfügbar). Es ist auch ein Java-Prozessor möglich, der Bytecode direkt als Maschinensprache verarbeitet.

9 Verteilte Algorithmen - Schnappschuss (10.2.4)

Eine Station eines Rechnernetzes verwendet den Schnappschuss-Algorithmus, um eine konsistente Sicht des gesamten Rechnernetzes, also über alle beteiligten Stationen zu gewinnen. Dies wird zum Beispiel verwendet, wenn von einer nebenläufigen Phase zu einer zentral verwalteten Arbeitsphase gewechselt werden soll. Hierzu ideal wäre ein ätomarer"Broadcast, der alle Stationen zur selben Zeit erreicht, damit sie ihren momentanen Zustand an die Zentralstation übermitteln. Diese Möglichkeit ist in realen Netzen aber leider nicht vorhanden. Um einen Algorithmus zu entwickeln, muss zunächst die Frage geklärt werden, wann ein Schnappschuss als konsistent anzusehen ist.

9.1 Konsistenz

Konsistenz ist sicher gegeben, wenn die Zustände aller Stationen genau zum gleichen Zeitpunkt ermittelt wurden.

Zur folgenden Beschreibung verwendet man die Vereinfachung, dass ein lokal ablaufender Prozess nur aus Phasen der Kommunikation besteht (die eine gewisse Zeit benötigen) und lokal ablaufenden Berechnungen, die hier als atomar, also ohne Bearbeitungszeit, ablaufen. Jeder Versand oder Empfang einer Nachricht wird weiterhin als Ereignis betrachtet.

Ein Zustand kann nun als Konsistent angesehen werden, wenn es möglich ist, dass die die Ereignisse an den einzelnen Stationen gleichzeitig aufgetreten sind. (Dies muss nicht unbedingt tatsächlich der Fall gewesen sein, es reicht die Möglichkeit aus). Dann ist der Schnappschuss konsistent.

Andere (mathematische) Formulierung: Zwischen zwei Ereignisse x_i (an Station S_i) und y_i besteht eine kausale Abhängigkeit, wenn sie (zeitlich geordnet) direkt nacheinander auftreten. Man schreibt dann $x_i \rightarrow_l y_j$. Zwei Ereignisse x_i und y_j sind kausal abhängig, wenn x_i ein Send-Ereignis und y_j das Receive-Ereignis derselben Nachricht ist. ($x_i \rightarrow_g y_j$). Nun bezeichnet $<$ die transitive Hülle von $\rightarrow$.

Nun wird definiert: x ist genau dann kausal unabhängig von y, wenn gilt: $\not(x < y)$ und $\not(y < x)$.

Ein Zustandsvektor $Z = (z_1, z_2, \ldots, z_n)$ heißt nun konsistent, wenn für den dazugehörigen Ereignisvektor $E = (e_1, e_2, \ldots, e_n)$ gilt: Jede zwei Ereignisse e_i und e_j sind kausal unabhängig.

9.2 Einfacher Algorithmus

- Initiator:
  Es gibt eine ausgewählte Station, die den Schnappschuss initiiert und selbst nicht an der Lösung des verteilten Problems teilnimmt. Diese Station sendet nun an alle Stationen die Statusanfrage, wartet in einer Schleife so lange, bis sie von allen Stationen eine Antwort erhalten hat (Verlustfreie Kommunikation wird vorausgesetzt) und sendet dann eine Schnappschussende-Nachricht.

- Stationen:
  Die Stationen wechseln nach Erhalt einer Statusanfrage-Nachricht in einen Modus, in dem sie zunächst ihre Statusantwort abschicken, dann aber ihre eigene Ausführung blockieren. Hierbei werden alle eingehenden Nachrichten empfangen, aber nicht verarbeitet (Mittels einer Methode unreceive in der Bearbeitung zurückgestellt), bis eine Statusende-Nachricht eingeht.

9.3 Verbesserter Algorithmus

Der Nachteil des einfachen Algorithmus ist, dass die Stationen eingefroren werden und somit lange Phasen der Untätigkeit auftreten können. Sinn dieses Verfahrens ist es ja, dass, nachdem eine Station die Statusanfrage bearbeitet hat, keine Kommunikation mit einer anderen Station durchgeführt wird, die von dem Schnappschuss noch nichts weis.

Beim verbesserten Algorithmus kann nun eine Station nach dem Empfang der Statusanfrage weiterarbeiten und auch mit anderen Stationen (die eventuell noch nichts vom Schnappschuss wissen) kommunizieren. Mit jeder Nachricht wird aber eine implizite Schnappschuss-Anforderung gesendet, die vor der eigentlichen Bearbeitung der Nachricht ausgeführt werden muss. Um hierbei mehrfache Statusantworten zu vermeiden, wird jedem Schnappschuss eine eindeutige Schnappschuss-ID zugeordnet.

Nachteil dieses Algorithmus ist, dass es sich um einen veralteten Schnappschuss handeln kann. Die Stationen können schon weitere Nachrichten verarbeitet haben, wenn der Initiator alle Statusmeldungen empfangen hat.

Beim veralteten Algorithmus ist der Schnappschuss solange aktuell, wie die Stationen eingefroren bleiben. Der Initiator kann also solange den Globalzustand sowohl lesend verarbeiten, als auch verändern. Beim verbesserten Algorithmus kann der Initiation hingegen nur lesend mit dem Schnappschuss arbeiten, da er nicht mehr den aktuellen Systemzustand darstellt. Daher eignet sich der verbesserte Algorithmus eher für die Informationsgewinnung (stabile Prädikate, also Eigenschaften, die sich nach einmaligem Auftreten nicht mehr ändern), also zum Beispiel, wie weit die Problemlösung mindestens schon vorangeschritten ist, o. ä.

Ab hier kommen weniger wichtige Themen.

10 Vermittlungssysteme (3.2)

Auf einem nachrichtentechnischen Kanal kann ein Sender meist nur eingeschränkt bestimmen, welche Stationen seine Nachricht empfangen. Bei Sammelkanälen können meist alle Stationen mithören, bei Punkt-zu-Punkt-Kanälen, natürlich nur der eine Empfänger. In einem größeren Netz gibt es deshalb Vermittlungseinrichtungen, die Nachrichten zu einer vom Sender vorgegebenen Teilmenge (meist nur ein Empfänger, manchmal eine kleine Gruppe) übermitteln. Hierbei kann zwischen zwei Verfahren unterschieden werden.

- Speichervermittlung: (Wie die Briefpost) Jede Nachricht wird nach der Übergabe vom Sender an das Netz einzeln über verschiedene Kanäle zum Empfänger weitergeleitet, wobei sie auch gepuffert werden kann. Einer Nachricht ist kein fester Kanal zugeordnet.
- Durchschaltevermittlung: Zum Beispiel im Telefonnetz vorhanden. Es wird mit dem Aufbau einer Verbindung ein durchgehender Kanal zwischen Sender und Empfänger geschaltet, der exklusiv für diese Verbindung genutzt wird. Er wird erst mit der Trennung der Verbindung wieder aufgehoben. Es wird bei der Vermittlung häufig einfach eine Verbindung zwischen physikalischen Medien hergestellt. Die Vermittlung findet also auf den Schichten 1 und 2 statt.

10.1 Vermittlungseinrichtungen für Durchschaltevermittlung

- Raummultiplex
  Eine Raummultiplex-Vermittlungseinheit besteht aus einem quadratischen Schaltbrett, an das n Eingangsleitungen und n Ausgangsleitungen führen. An den Kreuzungspunkten der Leitungen können Verbindungen hergestellt werden.
- Zeitmultiplex
  Es gibt auch hier Eingangs- und Ausgangsleitungen, jedoch wird jeweils für eine kurze Zeit (Timeslot) nur eine Eingangsleitung mit einer Ausgangsleitung verbunden.

10.2 Vermittungseinrichtungen bei der Speichervermittlung

Das Netz sieht Puffereinrichtungen vor, in denen Nachrichten bis zur Medienverfügbarkeit gespeichert werden können. In diesen Netzen werden meist nur Datagrammdienste erbracht. Es werden hauptsächlich Zeitmultiplex-Systeme verwendet, die nacheinander Datenblöcke aus den Puffern jeder Eingangsleitung übernehmen und zum Beispiel auf eine breitbandigere Leitung wieder ausgeben. Es werden sowohl statische Verfahren verwendet, bei der jeder Eingangsleitung eine konstante Zeit zugeordnet wird, unabhängig von der Menge der zur Übertragung anstehenden Daten, als auch dynamische Verfahren (vor allem ATM,

Asynchronous Transfer Mode), wo den Eingangsleitungen je nach anstehender Datenmenge als auch reservierter Übertragungsleistung mehrere Zeitschlitze zugeordnet werden können. Tritt ein Pufferüberlauf auf, so kommt es zu Nachrichtenverlust.

Während B-ISDN dieses dynamische Verfahren, ATM, verwendet, wird bei normalem ISDN das STM-Verfahren (Synchronous Transfer Mode) mit konstanten Zeitschlitzen verwendet wird.

10.3 ATM

Bei ATM werden die zu übertragenden Daten in 53 Byte große Pakete eingeteilt, die aus 5 Byte Adressierungs- und Kontrollinformationen und 48 Byte Nutzdaten bestehen. Ein ATM-Netz lässt die Überbuchung der Übertragungsleistung zu, geht aber davon aus, dass die Datenquellen voneinander unabhängig senden und daher im Durchschnitt keine Überlastsituation auftritt. Es wird aber beim Verbindungsaufbau eine Übertragungsqualität ausgehandelt, die vom Netz sehr nah am Sender auf Überschreitungen überwacht wird. Maßnahmen zur Überlastvermeidung reichen vom Markieren einzelnen Zellen zur bevorzugten Aussortierung bei Überlast, über das direkte Verwerfen von Zellen bis zum sofortigen Abbruch der Verbindung.

11 Implementierung (8)

11.1 Schnittstellen

Die logische Architektur eines Telekommunikationssystems sieht Schichten vor, die über den Austausch von Schnittstellenereignissen an ihren Dienstzugangspunkten miteinander interagieren. Dieses Modell muss nicht zwingend für die Implementierung übernommen werden (im OSI-Modell werden überhaupt keine Angaben zur Art der Implementierung gemacht).

Die Implementierung der Schnittstellen hängt im Normalfall stark von Randbedingungen des Systems ab (Betriebssystem, Rechnerarchitektur, Leistungseigenschaften, etc.).

11.1.1 Prozess

Es gibt die Möglichkeit, die einzelnen Schichten eines Modells direkt in Prozesse umzusetzen, die die einzelnen Instanzen der Schichten darstellen. Zunächst sind Prozesse aber gekapselte Einheiten, die ohne Unterstützung des Betriebssystems nicht miteinander kommunizieren können. Oft ist für einen Prozess die Terminierung nach endlicher Zeit vorgesehen, was für Protokollinstanzen nicht unbedingt erwünscht ist.

11.1.2 Synchronisation

Um den Arbeitsfortschritt zwischen Prozessen zu synchronisieren, gibt es zwei Varianten

- Blockieren und Fortsetzen
  Ein Prozess ruft eine Methode Warten auf, die ihn blockiert, bis ein anderer Prozess die Methode Fortsetzen aufruft. (unsymmetrisch)

- Rendezvous-Synchronisation
  Jeder Prozess ruft für sich die Methode Treffen auf. Diese Methode blockiert den Prozess solange, bis alle beteiligten Prozesse diese Methode aufgerufen haben

Diese Mechanismen werden immer vom Betriebssystem zur Verfügung gestellt, damit ein wartender Prozess den Prozessor für andere Prozesse freigeben kann.

11.2 Datenaustausch

Hier geht es darum, wie Prozesse auf einer Maschine miteinander kommunizieren können (IPC, Interprocess Communication).

11.2.1 Geteilter Speicher

In einigen Betriebssystemen können bestimmte Speicherbereiche zur Nutzung durch mehrere Prozesse freigegeben werden. Der Zugriff auf diese Speicherstellen muss aber durch geeignete Synchronisationsmaßnahmen innerhalb der Prozesse geregelt werden. Hier bietet sich das Verfahren der Critical Section an, die immer nur von einem Prozess durchlaufen werden kann.

11.2.2 Fremdzugreifbarer Speicher

Wie geteilter Speicher, aber mit dem Unterschied, dass der Speicher im privaten Adressraum eines Prozesses liegt und der Prozess diesen Speicher für andere Prozesse freigeben kann.

11.2.3 Auftragsübergabe

Dieses Verfahren wird vor allem zwischen Prozessen und Systemtreibern eingesetzt. Ein Prozess beschreibt einen privaten Speicherbereich mit den zu übergebenden Daten und ruft eine Funktion "Übergabe "auf (wie Systemcall). Damit wird der Prozess blockiert und es wird einem Systemtreiber erlaubt, diesen Bereich sowohl zu lesen als auch zu beschreiben. Um die Zeit, in der der Prozess blockiert ist, kurz zu halten, kann der Speicherbereich direkt in den Speicher des Treibers umkopiert werden.

11.2.4 Datenaustausch mit Rendezvous

Wie das Rendezvous-Synchonisationsverfahren, nur werden hier beim Treffen der beiden Prozesse Daten ausgetauscht.

11.2.5 Nachrichtenaustausch

Hiermit ist der durch das Betriebssystem unterstützte gepufferte Datenaustausch zwischen Prozessen gemeint (IPC). Es gibt die Methode Senden (kopiert einen Speicherbereich jedes aufrufenden Prozesses in den Systemspeicher), Empfangen (kopiert, falls vorhanden, eine anstehende Datenübertragung in den Speicherbereich des empfangenden Prozesses) und Testen (überprüft, ob Nachrichten abgerufen werden können). Vorteil ist hier, dass die Prozesse im Normalfall nicht blockiert werden (es sei denn, es steht zum Beispiel kein weiterer

Pufferspeicher im System zur Verfügung). Auslieferung kann entweder im Fifo-Modus erfolgen oder durch die Priorisierung von Nachrichten gesteuert werden. Eine Variante sind vom Betriebssystem unterstützte Ports oder Kanäle, die oft mit den üblichen Ein-/Ausgabe-Funktionen gelesen oder beschrieben werden können und somit einen Datenstrom übertragen.

11.2.6 Monitore

Zur Implementierung von Nachrichtenaustausch-Verfahren werden oft Monitore verwendet, also eine einem Objekt ähnliche Zusammenstellung von Datenstrukturen und darauf arbeitenden Methoden. Dabei wird die nötige Synchronisation beim Zugriff mehrerer Prozesse auf einen Monitor in diesem verdeckt.

11.2.7 Geteilte Puffer

Um die beim Nachrichtenaustausch nötigen Kopiervorgänge zwischen Prozess-Speicher und System-Puffer zu umgehen, können geteilte Puffer verwendet werden. Es werden die Nachrichten dann in einem Puffer, der in einem geteilten Speicherbereich eines Prozesses liegt, abgelegt und es werden nur noch Referenzen auf die Daten in Form von Nachrichten ausgetauscht.

11.3 Prozessinterne Schnittstelle

Oft werden aus Effizenzgründen mehrere Schichten innerhalb eines Prozesses implementiert. Hier kann nun zwischen zwei Verfahren des Nachrichtenaustauschs unterschieden werden:

- Aktionsorientierte Schnittstelle
  Jedes auftretende Ereignis wird in Form eines Methodenaufrufs durch die ereigniserzeugende Schnittstelle sofort weiterverarbeitet.

- Ereignispuffernde Schnittstelle
  Es wird im Prozess eine Queue vorgesehen in der die Ereignisse von der erzeugenden Instanz abgelegt werden. Eine verarbeitende Instanz entnimmt die Ereignisse dann wieder zur Weiterverarbeitung. Hier bieten sich Threads zur Implementierung an. Diese Art der Schnittstelle ist sehr ähnlich zu puffernden Kanälen des Betriebssystems.

Bei beiden Methoden sollten Vorkehrungen zur Übergabe besonders langer Nachrichten vorgesehen werden, um unnötigen Kopieraufwand zu vermeiden. Hierzu bietet sich die Übergabe durch Referenz an.

Es gibt eine Mischform, die mit den Begriffen Downcall-Schnittstelle (Prozeduren, die von der übergeordneten Instanz aufgerufen werden) und Upcall-Schnittstelle (Prozeduren, die von der untergeordneten Instanz aufgerufen werden). Die Downcall-Schnittstelle sieht nun vor, dass Stimuli aktionsorientiert und Reaktionen ereignispuffernd übertragen werden. Bei der Upcall-Schnittstelle ist dies genau umgekehrt.

Die Downcall-Schnittstelle wird oft bei anwendungsorientierten Schichten eingesetzt, die Upcall-Schnittstelle bei hardwareorientierten (Dadurch bildet zum Beispiel ein Timer-Interrupt die Spitze einer Aufrufhierarchie, sodass dieses Ereignis schnell verarbeitet werden kann).

11.4 Implementierung der Instanzen

Im logischen Modell wird eine Instanz durch einen Mealy-Automat dargestellt. Daran orientiert sich auch die Implementierung der Instanz. Die Bearbeitung eines Schnittstellen-Ereignisses kann daher in mehrere Schritte gegliedert werden:

- Aktivierung
  Durch das Eintreffen eines Ereignisses wird die Instanz aktiviert und damit die Bearbeitung eingeleitet
- Übernahme
  Die genaue Form des Ereignisses ist in einer Speicherdarstellung hinterlegt, die von der verarbeitenden Instanz übernommen werden muss.
- Analyse
  Die genaue Form des Ereignisses muss analysiert werden
- Selektion
  Es muss entsprechend der Vorgaben des Protokolls ein Zustandsübergang ausgeführt werden. Sind mehrere Übergänge möglich, sollte nicht zufällig einer gewählt werden, sondern anhand des Gesamtzustandes des Systems der optimale (geringster Aufwand) gewählt werden
- Transition
  Der Zustandsübergang wird ausgeführt, indem das Ereignis als erledigt markiert wird, der neue Zustand eingenommen wird und eine entsprechende Ausgabe erzeugt wird. (Konstruktion, Auslieferung und Zustandsänderung)

11.4.1 Zustandsdarstellung

Ein erweiterter Mealy-Automat wird u. a. spezifiziert durch seine Hauptzustände und die Nebenzustandskomponenten. Üblicherweise wird in der Implementierung der Hauptzustand entweder durch einen Aufzählungstyp dargestellt oder durch den momentanen Wert des Programmzählers. Die Nebenzustandskomponenten werden meist in Variablen auf Programmebene gespeichert (globale Variablen).

11.4.2 Aktuelles Eingabe-Ereignis

Bei der Übernahme des aktuellen Ereignisses wird oft eine Datenstruktur mit folgendem Inhalt gefüllt:

- Ereignistypkennung (z.B. UDtIndBA), ein Aufzählungstyp
- Kontrollparameter (Quell- und Zieladresse, etc.)
- Nutzdaten

11.4.3 PDU-Analyse

Beim Empfang einer PDU von einem Basisdienst muss die PDU zunächst analysiert und in ihre Bestandteile aufgespalten werden. Solange es die PDU-Größe zulässt ist es sinnvoll, die gesamte PDU auf einmal zu verarbeiten (durch eine Prozedur AnalysierePDU). Besonders bei sehr langen PDUs wird oft nur eine Segmentweise Analyse durchgeführt. Dabei wird oft zunächst überprüft, ob die PDU der erwarteten Syntax entspricht und fehlerfrei übertragen wurde; die einzelnen Bestandteile werden erst bei Bedarf verarbeitet.

11.4.4 PDU-Aufbau

Dies ist das Gegenstück der PDU-Analyse. Auch hier wird, sofern es die Größe der PDU zulässt, versucht, die PDU in einem Block aufzubauen. Bei sehr langen PDUs kann es aber auch hier erforderlich sein, segmentweise vorzugehen.

11.4.5 Aktivierung

Bei der Verwendung von IPC wird oft das Warten mit Blockierung des Prozesses verwendet. Der Prozessablauf wird solange blockiert, bis auf dem Empfangskanal eine Nachricht eingeht. Probleme bereitet es, wenn auf mehreren Kanälen gleichzeitig Nachrichten erwartet werden. Das BusyWaiting sollte aber in jedem Fall vermieden werden.

Bei der aktionsorientierten Schnittstelle wird bereits durch die Erzeugung eines Ereignisses die Bearbeitung beim Empfänger angestoßen. Es sind hier keine weiteren Vorkehrungen zur Aktivierung nötig. Probleme bereitet es auch hier, wenn Ereignisse von mehreren anderen Prozessen verarbeitet werden sollen. Die Instanz ist dann an mehreren Stellen aktiv und arbeitet somit auf geteilten Daten.

11.4.6 Selektion

Ist ein Ereignis aufgetreten, muss sich die Instanz für einen Zustandsübergang entscheiden. Dazu kann sie entweder eine (u. U. verschachtelte) if-then-else- bzw. switch-case-Anweisung verwenden, oder (meist effizienter) eine Tabellengesteuerte Selektion vornehmen. Diese Tabelle ist zweidimensional. In der ersten Dimension sind die Hauptzustände des Automaten aufgeführt, in der zweiten die möglichen Ereignisse. An jeder Tabellenposition steht nun zum Beispiel die Adresse einer Prozedur, die zur Transition verwendet wird.

11.4.7 Zeitüberwachung

Viele Instanzen verarbeiten nicht nur Ereignisse, die durch eine andere Instanz erzeugt wurden, sondern auch selbst generierte. Hier handelt es sich vor allem um

- Wecker-Start-Ereignisse
- Stornierungsereignisse
- Ablauf-Ereignisse

Implementiert werden diese Ereignisse oft so, dass sie wie normale Schnittstellenereignisse behandelt werden können. Ein Problem stellt oft die große Anzahl an benötigten Weckern dar (eher ein Problem des Betriebssystems).

11.5 Grundlegende Software-Architekturen

11.5.1 Zentralverteiler

Es handelt sich um eine Architektur, bei der das Programm aus einer Haupt-Schleife besteht, die zyklisch (endlos) durchlaufen wird und jeweils nacheinander die zum Bearbeiten eines Ereignisses nötigen Prozeduren aufruft. Sie besteht meist aus den Anweisungen zum Empfang des Ereignisses (Empfang mit Blockieren), Übernahme der Daten und Berechnung und Ausführung der Transition.

11.5.2 Sammlung von Ereignisbearbeitungsprozeduren

Hier gibt es keine zentrale Programmschleife. Stattdessen handelt es sich um eine Sammlung von Prozeduren, die entsprechend dem Modell der aktionsorientierten Bearbeitung von Ereignissen direkt von der ereigniserzeugenden Instanz aufgerufen werden.

11.5.3 Instanzen und Prozesse

Grundsätzlich gibt es drei Möglichkeiten, Instanzen auf Prozesse abzubilden:

- Eine Instanz je Prozess
- Mehrere Instanzen je Prozess
- Mehrere Prozesse je Instanz

Vorteil bei der Verwendung mehrerer Prozesse ist die übersichtlichere Strukturierung, Nachteil ist der oft nicht zu vernachlässigende Verwaltungsaufwand der Prozesse durch das Betriebssystem.

11.5.4 Leichtgewicht-Prozesse

Um den Verwaltungsaufwand gering, die Strukturierung aber übersichtlich zu halten, bietet sich diese Form von Prozessen an (Threads). Diese teilen sich einen gemeinsamen Speicherbereich, was unter Umständen die Kommunikation erleichtert.

11.5.5 Integration mehrerer Instanzen

Sollen mehrere Instanzen in einem Prozess in Hauptverteiler-Architektur implementiert werden, so bietet es sich an, die Automatenmodelle der Instanzen zu mischen. Es muss hierbei nur eine gemeinsame Transitionstabelle generiert und eine gemeinsame Hauptschleife erstellt werden. Die Bearbeitungsprozeduren können oft unverändert übernommen werden.

Auch bei der Architektur der Prozedursammlung lassen sich mehrere Instanzen in einen Prozess integrieren. Es ergibt sich hierbei ein (oft sehr effizienter) "Prozedurturm ".

Die Kombination beider Arten ist ebenfalls möglich. Hier wird oft eine Hauptschleife verwendet, die Ereignisse von außen entgegennimmt und jeweils die Ausführung eines "Prozedurturms "anstößt. Da die Hauptschleife sequentiell arbeitet, kann es nicht zur geteilten Nutzung von Datenstrukturen kommen (Es ist immer nur ein Prozedurturm in der Ausführung). Es muss dabei aber darauf geachtet werden, dass ein solcher Prozedurturm keine lange Ausführungszeit hat.

11.6 Programm- und Programmierumgebung

11.6.1 Systemsprachen

Diese Sprachen (C, Assembler, ...) werden benutzt, um ein Betriebssystem zu implementieren. Damit bieten sie sich für die transportorienterten Schichten an, da diese oft eher auf Schnittstellen des Betriebssystems und Eigenarten der Architektur zurückgreifen.

11.6.2 Höhere Programmiersprachen

Anwendungsorientierte Schichten werden eher in höheren Sprachen (C++, Java, ...) entwickelt, um die Entwicklungszeit zu verkürzen, da oft schon Möglichkeiten zur IPC, etc. eingebaut sind.

11.6.3 Einbettungssysteme

Gerade bei der Implementierung der Protokollschichten von Rechnernetzen gibt es immer wiederkehrende Probleme. Deshalb gibt es oft vorgefertigte Teilproblemlösungen, die in Form von Programm-Bibliotheken verwendet werden können oder auch durch eine Erweiterung der Programmiersprache, die durch eine Vorverarbeitung (Präprozessor) in die zugrunde gelegte Programmiersprache umgewandelt wird.

11.6.4 Schnelle Prototyp-Erzeugung

Da Protokolle zunächst oft in formellen Spezifikationssprachen beschrieben werden bietet es sich an, hierfür automatische Übersetzer zu schaffen, die aus der Spezifikation eine mehr oder weniger vollständige Implementierung erzeugen. Ein Nachteil ist die oft vorhandenen Ineffizienzen, da nur nach Standardregeln vorgegangen werden kann und keine Spezialitäten des Protokolls oder der zugrunde liegenden Rechnerarchitektur berücksichtigt werden können.

11.6.5 Implemetierungsgeneratoren

Hier wird nicht versucht, ein lauffähiges Programm aus der Spezifikation zu erzeugen, sondern nur ein Quellcode-Grundgerüst, welches oft noch große Lücken (gerade in kritischen Bereichen wie Aktivierung und Kommunikation mit anderen Instanzen) aufweist, die von Hand implementiert werden müssen. Oft werden Implementierungsgeneratoren in Zusammenhang mit Einbettungssystemen verwendet.

12 Verteilte Algorithmen -Terminierungserkennung (10.2.5)

Bei der Bearbeitung eines Problems in einer verteilten Anwendung tritt oft die Frage auf, wann die Bearbeitung abgeschlossen ist. Auch wenn die Anwendung selbst nicht terminiert, kann der Abschluss einzelner Bearbeitungsphasen von Interesse sein.

Betrachtet man ein solches System unter Verwendung des Atom-Modells (Lokale Bearbeitung benötigt keine Zeit, nur die Nachrichtenübermittlung), so kann man sagen, dass eine Problemlösung terminiert ist, wenn keine Nachrichten zur Problemlösung mehr unterwegs sind.

Die einfachste Möglichkeit ist, periodisch Schnappschüsse anzufertigen, die Auskunft über die gesendeten und empfangenen Nachrichten geben. Liegt eine ausgeglichene Bilanz vor, so ist die Problemlösung terminiert. Nachteil ist der übermäßig hohe Kommunikationsaufwand.

12.1 Vektormethode

Jede Station verwaltet einen Vektor V_i. $V_i[j]$ wird inkrementiert, wenn eine Nachricht an Station j gesendet wurde. $V_i[i]$ wird dekrementiert, wenn eine Nachricht empfangen wurde.

Weiterhin zirkuliert ein Kontrollvektor im Netz, der von jeder Station aktualisiert wird, indem ihr eigener Vektor hinzuaddiert wird. Erreicht der Kontrollvektor den Betrag 0, so liegt Terminierung vor.

12.2 Kreditverfahren

Falls eine Station die Berechnung startet und auch die Terminierung feststellen soll, so kann diese Methode verwendet werden.

- Die Startstation hat die Kreditzahl 1
- Hat eine Station den Kredit c und sendet an m andere Stationen, so überträgt sie jeder Station den Kredit c/m.
- Eine Station darf nur senden, wenn ein Kredit ¿ 0 vorhanden ist.
- Erreicht die Start-Station wieder den Kredit 1, so liegt Terminierung vor.