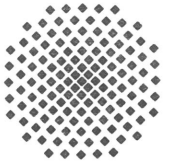


Universität Stuttgart

Institut für Nachrichtenvermittlung und Datenverarbeitung

Prof. Dr.-Ing. habil. P. J. Kühn

55. Bericht über verkehrstheoretische Arbeiten

TRANSPORTSYSTEME FÜR HOCHGESCHWINDIGKEITSNETZE: PROBLEMANALYSE, LÖSUNGSANSÄTZE UND ENTWURF EINER EXEMPLARISCHEN IMPLEMENTIERUNGSSTRUKTUR

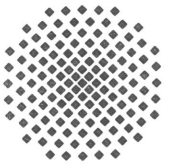
von

Martin Siegel

1994

Druck: E. Kurz & Co., Druckerei + Reprografie GmbH., Stuttgart

ISBN 3-922403-65-4



University of Stuttgart

Institute of Communications Switching and Data Technics

Prof. Dr.-Ing. habil. P. J. Kühn

55th Report on Studies in Congestion Theory

TRANSPORT SYSTEMS FOR HIGH SPEED NETWORKS: PROBLEM ANALYSIS, PRINCIPAL SOLUTIONS AND DESIGN OF AN EXAMPLE IMPLEMENTATION STRUCTURE

by

Martin Siegel

1994

TRANSPORT SYSTEMS FOR HIGH SPEED NETWORKS: PROBLEM ANALYSIS, PRINCIPAL SOLUTIONS AND DESIGN OF AN EXAMPLE IMPLEMENTATION STRUCTURE

Abstract

Communication systems of today are characterised by an increasing difference between possible transmission rates, and actual achievable throughput at the application. More than one billion bits per second (1 Gbit/s) are possible with modern physical transmission technologies, e.g., fibre optic. Users of a complete seven layer ISO/OSI[1] protocol stack have to accept 20 packets per second as a good performance. Packet lengths vary between 64 and a few 1.000 bytes. Typically, mean values for packet lengths are low and, thus, 20 packets per second corresponds to some tenth of Kbit/s. Restrictions concerning type of service and heterogeneity can be used to reduce the number and complexity of protocol functions. Naturally, the application oriented layers 5 to 7 of the ISO/OSI reference model can profit most of these restrictions. The achievable 1.000 packets per second or some Mbit/s above the transport layer 4, however, are neither sufficient for applications as multi media nor do they fit to the planned information highways.

The intention of this work was to find methods to overcome this enormous difference. Main emphasis was put on the layers 1 to 4 of the ISO/OSI reference model, the transport system. The undertaken problem analysis showed:

1. A concentration on single items can lead to unsatisfying and even wrong statements concerning the overall system performance.
2. The logical protocol functionalities have a minor influence on the system performance, only. Implementation aspects, on the other hand, are significant and can not be ignored.

A detailed design of a complete transport system with all hardware and software components was the one single possibility to achieve the mentioned goal. Four aspects were important:

1. Conformity to OSI-standards
2. Universal usability
3. Economy
4. Feasibility, i.e., usage of standard technologies.

The structure of the report is given below.

---

[1] International Organization for Standardization, Open Systems Interconnection

Chapter 1 contains an overview of the problem area. The motivation for the work is given and related goals are derived. Content and structure of the work is described, too.

In **Chapter 2** basic definitions and used nomenclature are introduced. Main trends within the communication arena are shown. The discussion is not restricted to the private sector. Thus, trends in the public, telephony oriented wide area are part of this chapter as well as the computer oriented communication in local areas.

An extensive analysis of the problems which lead to the poor performance of existent communication systems is performed in **Chapter 3**. All functions of a transport system are summarised. Aside from looking at logical protocol functions, evaluations of implementation specific aspects are done. The discussion of typical implementation variants is used to show severe draw backs.

Chapter 4 presents solutions to each problem area mentioned in the previous chapter 3. The ongoing projects all over the world are mentioned in this chapter. Known studies of single aspects are covered, too. As in the problem analysis, implementation aspects get same attention as logical functionalities of the protocol layers.

The implementation structure of the designed transport system is presented in **Chapter 5**. All designed hardware components are described. This are six ASICs for time critical or time consuming operating system functions, e.g., timers, memory administration, flexible dual port control, process coupling or intelligent direct memory access. Some aspects, which do not follow well known ´regular´ circuit design rules, and therefore, can be of interest to the reader, are discussed in more detail. A very fast and rather simple solution for calculating and verifying the original ISO/OSI layer 4 checksum is one of these new things.
A further focus is set on inter working of all single components and software design.

Chapter 6 explains used design and verification methods. All different hardware and software design and verification steps and the related tools are covered. The developed abstract simulation model and a new type of simulation are one main part of this chapter. That simulation is a mixture of hardware oriented behavioural modelling and performance evaluation methods known from congestion theory. Main simulation results and achieved performance figures of the transport system are presented.

The first part of **Chapter 7** summarizes the work. Finally, an outlook on extensions and further topics concludes the report.

Inhaltsverzeichnis

Verzeichnisse

Kapitel 1

Einführung 1

Kapitel 2

Grundlagen und Umfeld der Kommunikationstechnik 5

Kapitel 3

Problemanalyse 17

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen und Begriffe

AFI	Authority and Format Identifier - Autorisierungs- und Formatkennung, Teil der Schicht 3 Adresse
ANSI	American National Standards Institute
AR	Anwendungs-Rechner (Englisch: host)
ASCII	American Standard Committee for Information Interchange - Standardisiertes Format zum Datenaustausch (7 Bit)
ASIC	Application Specific Integrated Circuit - anwendungsspezifische integrierte (VLSI-) Schaltkreise
ASN.1	Abstract Syntax Notation No. 1 - abstrakte Syntaxnotation Nr. 1
ATM	Asynchronous Transfer Mode - asynchrone Übertragungsform
BD-product	Bandwidth-Delay-product - siehe BV-Produkt
B-ISDN	Broadband Integrated Services Digital Network - Breitband-ISDN
Back-end	Verbindung von Großrechnern und deren unmittelbarer Peripherie
Backbone	Wörtlich: Rückgrat; Vernetzung von Netzen
BM	Betriebs-Mittel
Bus	Sammelbegriff für komponenteninterne Verbindungsstruktur
BV-Produkt	Bandbreiten-Verzögerungsprodukt, Speicherfähigkeit des Netzes in Bits pro Sekunde
Caching	Zwischenspeichern von Teilinformation in schnellen Speichern
CCITT	Comité Consultatif International Télégraphique et Téléphonique - Standardisierungsgremium für (überwiegend) öffentliche Netze
CEP	Connection End Point, (logischer) Endpunkt einer Verbindung
CIM	Computer Integrated Manufacturing - Rechnergesteuerte Herstellung, Fabrikautomatisierung
CMIS	Common Management Information Service, Schicht 7 Dienst zum Austausch und zur Behandlung von Management Information
CONS/CLNS	Connection Oriented bzw. Connectionless Network Service - verbindungsorientierter / -loser Dienst der Vermittlungsschicht
CPU	Central Processing Unit - zentrale Verarbeitungseinheit (auch Prozessor oder Mikrorechner)
CRC	Cyclic Redundancy Check - zyklische Redundanzprüfung (Prüfsumme)
CS	Circuit Switched - leitungsvermittelnd
CSDN	(public) CS-Data Networks - (öffentliche) CS-Datennetze
CSMA-CD	Carrier Sense Multiple Access with Collision Detection - Vielfachzugriffsverfahren basierend auf dem Abhören des Mediums und der Feststellung von Kollisionen
(D)ARPA	(Defense) Advanced Research Projects Agency - Forschungsabteilung des US-amerikanischen Verteidigungsministeriums
DMA	Direct Memory Access - direkter Speicherzugriff (ohne die CPU)
DQDB	Distributed Queue Dual Bus - Medienzugriffsprotokoll mit verteilter Warteschlange und zwei gegenläufigen Bussen
DoD	Department of Defense - US-Verteidigungsministerium
Dual-ported	Zweitorspeicher, Speicher mit zwei unabhängigen Zugriffsmöglichkeiten
DS	Directory Service - Dienst für die Verwaltung von Verzeichnissen
EBCDIC	Extended Binary Coded Decimal Interchange Code - Standardisiertes Format zum Datenaustausch (8 Bit)
EOT	End of Text - Textende, Markierung für die letzte TPDU innerhalb einer TSDU
FDDI	Fiber Distributed Data Interface - Medienzugriffsprotokoll für Doppelringsystem mit Tokenverfahren
FFOL	FDDI Follow-On LAN - ANSI-Initiative für die Entwicklung eines Gigabit/sec-LANs

FIFO	first in, first out - reihenfolgeerhaltende Abfertigungsstrategie
Frame Relay	transparenter Schicht 2Dienst zum Übertragen von Daten über CS-Netze
Front-end	direkte Vernetzung geographisch getrennter leistungsfähiger Rechenanlagen
FTAM	File Transfer, Access and Management - Schicht 7 Dienst für die Übertragung, Zugriffsregelung und (sonstige) Verwaltung von Dateien
FTP	File Transfer Protocol - Anwendungsprotokoll der Internetfamilie zur Dateienübertragung
Garbage collection	wörtlich: Müllsammlung, Aufsuchen und Nutzbarmachen von Speicherverschnitt
GOSIP	Government OSI Profile - staatliche Festlegung von OSI-Protokolltürmen
HS	High-Speed - Hochgeschwindigkeit
IDP	Initial Domain Part - global festgelegter Adreßteil der Schicht 3 Adresse
IEEE	Institute of Electrical and Electronics Engineers - Internationale Vereinigung der Elektroingenieure
IF-Bereich	Interface-Bereich - Schnittstellen- und Verwaltungsbereich innerhalb des TpS
IMS	Institut für Mikroelektronik, Stuttgart
IND	Institut für Nachrichtenvermittlung und Datenverarbeitung der Universität Stuttgart
Interrupt	Unterbrechung (bzw. deren Anforderung); Möglichkeit, um eine CPU auf (externe) Ereignisse aufmerksam zu machen
IP	Internet Protocol - Vermittlungsprotokoll der (D)ARPA-Protokollfamilie
ISDN	Integrated Services Digital Network - diensteintegrierendes digitales Nachrichtennetz
ISO	International Organization for Standardization - internationales Standardisierungsgremium
JTM	Job Transfer and Manipulation, Dienst für die Übertragung und Manipulation von Aufträgen
KB / MB	Kilo-/ Megabyte; K für $1024 = 2^{10}$, M für 10^6 bzw. 1.048.576 (2^{20})
LAN	Local Area Network - Lokales Netzwerk
LIFO	last in, first out - reihenfolgevertauschende Abarbeitungsstrategie (Stapelmodus)
LLC	Logical Link Control - logische Sicherung, Teilschicht 2b des OSI-RMs
LWE	Lower Window Edge - untere Fensterkante
MAC	Media Access Control - Medienzugangskontrolle, Teilschicht 2a des OSI-RMs
MAN	Metropolitan Area Network - Netze mit für Städte geeigneten Ausdehnungen
MHS	Message Handling System - Dienst zur Behandlung von Nachrichten
NBS	National Bureau of Standards - US-amerikanische Standardisierungsinstitution
OSI	Open Systems Interconnection
OSI-RM	OSI-Reference Model
OTF	On-The-Fly - wörtlich: im Fluge; schritthaltend, in Echtzeit
(C-)PBX	(Computerized) Private Branch eXchange - (Rechnergesteuerte) Nebenstellenanlage
PC	Personal Computer - Kleinrechenanlage
PCD	Process Coupling Device - Prozeßkoppelbaustein
PCI	Protocol Control Information - Protokollsteuerungsinformation
PDU	Protocol Data Unit - Grundeinheit zum Austausch von Information zwischen den (entfernten) Protokollinstanzen
Polling	(zyklisches) Abfragen
PS	Packet Switched - paketvermittelt bzw -mittelnd
PSDN	(public) PS-Data Networks - (öffentliche) PS-Datennetze
QoS	Qualities of Service - Dienstgüte
Refresh	Speicherzyklen zur Auffrischung der Information in dynamischen Speichern
RPC	Remote Procedure Call - Prozeduraufruf auf einem entfernten Rechner
ROSE	Remote Operation Service Element - Schicht 7 Dienst zur Koordinierung entfernter Operationen
SAP	Service Access Point - Dienstzugangspunkt
SDU	Service Data Unit - Dienstdateneinheit
SICS	Swedish Institute for Computer Science - Schwed. Forschungszentrum für Informatik
TCP	Transmission Control Protocol - Transportprotokoll der (D)ARPA Protokollsäule
Telnet	Protokoll der Internetfamilie zur Realisierung eines entfernten Endgeräts

Timer	Zeitgeber
TP4	Transportprotocol class 4 - ISO/OSI- Protokollklasse der Transportschicht
TPDU	Transport PDU - Basisübertragungseinheit der Transportschicht
TpS	Transport-System
TSDU	Transport SDU - SDU zur Interaktion mit dem Transportdienst
TTPN	Timed Transition Petri-Net - Petrinetz mit Zeitinformation, formale Beschreibungsform
UWE	Upper Window Edge - obere Kante eines (Protokoll-)Fensters
VDF	Verteilungsdichtefunktion
VE	Verarbeitungseinheit
VF	Verteilungsfunktion
VLSI	Very Large Scale Integration - Höchstintegration von elektrischen Bauelementen auf einem Halbleiterkristall (ursprünglich ab 100.000 Transistoren)
VT	Virtual Terminal, Dienst zur Handhabung eines virtuellen Endgeräts
WAN	Wide Area Network - Weitverkehrsnetze

Formelzeichen

a	Gesamtlänge einer Quittierungs-PDU
a_i	i-tes Oktett eines zu prüfenden Oktettstromes
A	Zahl der Speicherzugriffe für eine PDU
b	Bandbreite des Mediums in Bits pro Sekunde
B	Nutzbandbreite des Mediums in Bits pro Sekunde
c0	Hilfsgröße für die Berechnung des ersten Prüfoktetts
$c0_D$ $c0_S$	wie c0, für Datenteil bzw. Steuerkopf
$c0^n$ $c0^{n-1}$	wie c0, mit zeitlicher Relation
c1	Hilfsgröße für die Berechnung des zweiten Prüfoktetts (original)
$c1_D$ $c1_S$	wie c1, für Datenteil bzw. Steuerkopf
c1'	wie c1, jedoch mit abgewandelter Berechnungsvorschrift
$c1'^n$ $c1'^{n-1}$	wie c1', mit zeitlicher Relation
c[x]	Variationskoeffizient der Größe x
d	Übertragungsverzögerung
e	Geografische Distanz der Kommunikationspartner in Metern
E	Effizienz
E[x]	Mittelwert der Größe x
f(x)	Verteilungsdichtefunktion der Größe x
F(x)	Verteilungsfunktion der Größe x
h	Zahl der Steuerkopfbits einer PDU
i, j	Laufindices
k	Position des ersten Prüfoktetts
l	Gesamtlänge einer Daten-PDU (l = h + n)
L	Zahl verlorener PDUs nach einem Fehler
L_i	Listenlänge des i-ten Elements einer zyklischen Struktur
m	allgemeiner Modul bei Modulorechnung
n	Zahl der Nutzdatenbits einer PDU
N	Zahl der ursprünglich zu übertragenden TPDUs
(N) (N-1)	Schichtennummer innerhalb eines Protokollturmes
NV	Anzahl nicht verworfener PDUs
O	Normierter Wert für den durch Wiederholung erzeugten Mehraufwand
p_A	Wahrscheinlichkeit für Fehlerfreiheit nach einem Fehler
p_L	TPDU-Verlustwahrscheinlichkeit
R	Zahl der pro Stockung durch Wiederübertragung revidierten TPDU-Verluste
R_T	TPDUs pro Sekunde (TPDU-Rate)
S	Zahl der Stockungen
t_a	Zeit für die Erstellung einer Quittung
t_c	Bearbeitungszeit einer PDU
t_{cr}	Zeit für den Verbindungsaufbau in Empfangsrichtung
t_{cs}	Zeit für den Verbindungsaufbau in Senderichtung
t_{dat}	Transferzeit für Nutzdaten und deren Quittungen
t_O	Wert des Wiederholungszeitgebers

t_{next}	Taktzeit einer Fließbandverarbeitung
t_r	mittlere Antwortzeit (roundtrip delay)
t_{rmax}	Maximalwert von t_r
$t_{Schicht}$	Bearbeitungszeit einer Schicht
t_{set}	Gesamtzeit für Verbindungsaufbau ($t_{set} = t_{cs} + t_{cr}$)
t_t	Übertragungszeit
t_T	Übertragungszeit einer Dateneinheit aus dem Speicher
t_z	Speicherzykluszeit
v	Verarbeitungszeit für den Empfang einer PDU
V	Zahl der TPDU-Verluste ($V = N \bullet p_L$)
w	Speicherbreite in Oktetts
W	Fenstergröße in PDUs
W_S	Größe des Sendefensters
W_R	Größe des Empfangsfensters
x	erstes Prüfsummenoktett (an der Position k)
y	zweites Prüfsummenoktett (an der Position k+1)

Kapitel **1**

Einführung

Die zweite Hälfte des 20. Jahrhunderts ist geprägt durch den steigenden Einfluß der Informationsverarbeitung, welche ihrerseits wesentlich von funktionierenden Möglichkeiten zur Kommunikation bestimmt ist. Bis in die 70-er Jahre hinein war die Kommunikationstechnik fast ausschließlich auf die Unterstützung der sprachlichen Kommunikation zwischen Menschen ausgerichtet. Die Information wurde dabei entsprechend ihrem grundsätzlichen Charakter zeit- und frequenzkontinuierlich (*analog*) über physikalisch vorhandene Verbindungen transportiert. Schon bald nachdem die *Steuerungen* von Vermittlungssystemen von der elektro-mechanischen Relaistechnik überwiegend auf Digitaltechnik umgestellt worden waren, ergaben sich auch beim Wechsel von der analogen zur digitalen *Übertragungs*technik wirtschaftliche Vorteile, welche bei weitem den notwendigen Aufwand zur Analog- / Digital- und entsprechenden Rückwandlung überwogen.

Inzwischen sind viele neue Kommunikationsformen hinzugekommen. Neben weiteren Typen von zwischenmenschlicher Kommunikation entstand eine grundsätzlich verschiedene Kategorie: die Kommunikation zwischen informationsverarbeitenden Maschinen, welche in den letzten Jahren überproportionale Steigerungsraten zu verzeichnen hatte. Trotz der durch die Digitalisierung der Sprache gegebenen technischen Möglichkeit, die beiden Grundformen der Kommunikation miteinander zu koppeln, wurde dies

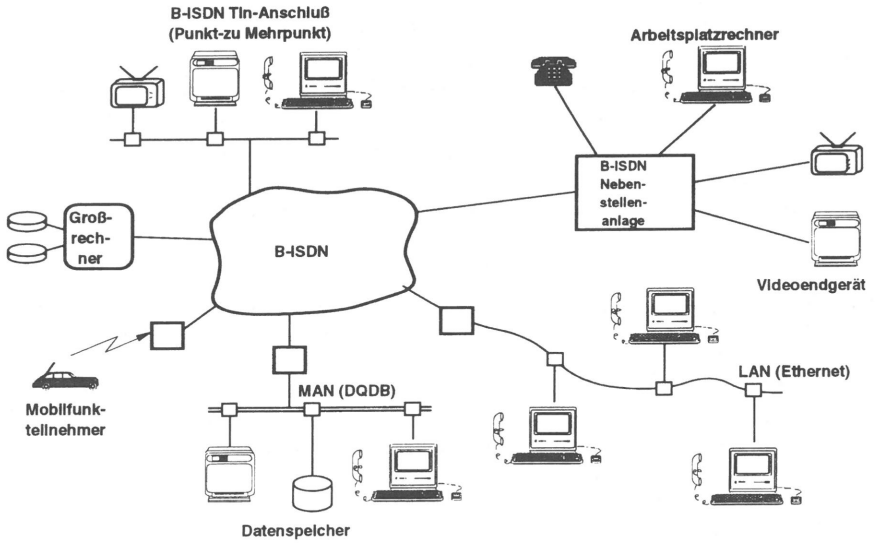


Bild 1.1: *Szenario eines zukünftigen Kommunikationssystems*

zunächst kaum in Betracht gezogen.

Die eigenständige Behandlung jedes Kommunikationsdienstes, also unabhängige Infrastrukturen und Techniken, ist heute jedoch nicht länger zu begründen, weder inhaltlich noch technologisch. Nicht zuletzt überwiegt wiederum auch der wirtschaftliche Vorteil einer einheitlichen Technik bei weitem den dazu notwendigen Aufwand. Diese *Integration* aller bestehender Arten von Kommunikation und damit insbesondere das Zusammenwachsen der Telefon- und Rechnerwelt ist daher *der* bestimmende Faktor innerhalb der heutigen Kommunikationstechnik. Bild 1.1 zeigt das ideale Szenario eines zukünftigen Kommunikationssystems.

Alle Arten von Kommunikation sind darin vereint. Eine beliebige Kommunikation zwischen allen angeschlossenen Teilnehmern bzw. Endgeräten soll möglich sein. Das System soll so flexibel ausgelegt sein, daß sowohl alle heutigen Dienste als auch neue Dienste mit noch unbekannten Charakteristiken einbezogen werden können. Neben den verschiedensten Ausprägungen der stetig an Bedeutung - und Akzeptanz - zunehmenden multimedialen Kommunikation ist speziell die Einbeziehung jeglicher Art von Mobilkommunikation ein vorrangiges Ziel.

1.1 Anforderungen an moderne Kommunikationssysteme

Aus den angesprochenen Entwicklungstendenzen in Bezug auf allgemeine Integration und der damit verbundenen Flexibilität ergeben sich die folgenden Anforderungen an zukünftige Netze:

- Leichte Zusammenarbeit mit anderen öffentlichen und privaten Netzen
  (bestehenden und neu entwickelten), d.h. Standardisierung
- Unterstützung unterschiedlichster Dienste
  (asynchron, synchron, isochron; verbindungsorientiert oder verbindungslos)
- Garantie sowohl für Durchsatz als auch Maximalverzögerung
- Multi-Media-Fähigkeit
- Integriertes Benutzer-Schnittstellen-Protokoll (UIP, User Interface Protocol)
- Unterstützung unterschiedlicher Dienstattribute
  (Punkt-zu-Punkt, Multicast, Broadcast; unidirektional, bidirektional, asymmetrisch)
- Effizienz, d.h. hohe mittlere Auslastung
- Skalierbarkeit in Bezug auf geographische Größe, Bandbreite und Anschlußzahl
- Verfügbarkeit und Wartbarkeit
  (Fehlertoleranz, integrierte Diagnose- und Managementunterstützung)
- Fairness
- Wirtschaftlichkeit.

Die Vielzahl der teilweise widersprüchlichen Anforderungen schließt Lösungen, die auf bestehenden Netzen basieren, weitgehend aus. Sowohl im öffentlichen Bereich (Breitband-ISDN, ATM) als auch im privaten Umfeld (Hochgeschwindigkeits-und Gigabit-LANs) sind daher jeweils seit einiger Zeit Aktivitäten zur Vorbereitung neuer Kommunikationssysteme im Gange. Allerdings stehen dabei jeweils überwiegend mediennahe Aspekte im Vordergrund. Dies entspricht lediglich den untersten 2 Schichten des von der ISO (International Organization for Standardization) entwickelten 7-schichtigen Referenzmodelles für die Kommunikation in heterogenen Systemen (Open Systems Interconnection Reference Model, OSI-RM). Diese ungleiche Verteilung des Entwicklungspotentials ist einer der Gründe dafür, daß zwischen dem Leistungsvermögen der unteren und höheren Schichten ein Unterschied besteht.

1.2 Ziele der Arbeit

Das ursprüngliche Ziel der Arbeit war es, Schwachstellen von OSI-konformen Kommunikationssystemen aufzufinden, entsprechende Verbesserungen zu erarbeiten und zu bewerten. Schwerpunkt der Betrachtungen sollten die *logischen* Funktionalitäten der höheren Schichten des OSI-RMs sein, also im wesentlichen die in den ISO/OSI-Standards beschriebenen reinen Protokollkomponenten.

Eine wesentliche Intention war dabei das Lokalisieren von Protokollmechanismen, bei denen eine Umsetzung in Hardware bis hin zu anwendungsspezifischen integrierten Schaltkreisen (ASICs) besonders hohe Effizienzsteigerungen bewirken sollte.

Abhängig von den Randbedingungen (geographische Ausdehnung, Verkehrscharakteristik, Übertragungskapazität, Betriebssystem etc.), ergaben sich bei der Bewertung von ersten vorgenommenen Optimierungen allerdings erhebliche Abweichungen. So führten vorgenommene Änderungen, z.B. bei Protokollmechanismen zur Quittierung von empfangenen Paketen, je nach angenommener Umgebung entweder zu dem geplanten besseren Leistungsverhalten oder zu einer wesentlich verschlechterten Leistungsfähigkeit.

Die Betrachtung von Implementierungsgesichtspunkten führte schließlich zur zentralen Zielsetzung:

> *Die Arbeit soll nachweisen, daß die ursprünglich für langsame und fehleranfällige Netzwerke entwickelten ISO/OSI-Protokolle mit einer entsprechend gewählten Implementierungsstruktur auch für viele Anwendungen in Hochgeschwindigkeits-Kommunikationssystemen eingesetzt werden können.*

Um diese Zielvorgabe erreichen zu können, war es unabdingbar, ein komplettes Kommunikationssystem zu entwerfen. Die ursprünglich geplante Konzentration auf einzelne wenige, voneinander unabhängige, Aspekte wurde damit durch eine breiter angelegte systemorientierte Vorgehensweise ersetzt. Dem pragmatischen, eher ingenieurgemäßen, Aspekt der (wirtschaftlichen) Realisierbarkeit wurde dabei ein ebenso hoher Stellenwert zugeordnet wie den theoretischen Betrachtungen.

1.3 Gliederung der Arbeit

Im folgenden **Kapitel 2** werden zunächst einige für das Verständnis der weiteren Arbeit notwendigen Grundlagen und Begriffe eingeführt. Die bestimmenden Tendenzen innerhalb der Kommunikationstechnik sowohl im öffentlichen als auch privaten Bereich werden ebenfalls vorgestellt.

Kapitel 3 gibt die Begründung für die Durchführung der Arbeit an. Dazu werden die generellen Aufgaben eines Transportsystems beschrieben, wobei neben den logischen protokollorientierten Funktionen die "sonstigen" implementierungsspezifischen Funktionalitäten der Systemumgebung gesonderte Beachtung finden. Die Diskussion von heute üblichen Implementierungsvarianten einiger exemplarisch ausgewählter Teilaspekte zeigt Schwachpunkte auf.

Die vielfältigen Ansätze zur Behebung dieser Schwachpunkte werden in **Kapitel 4** aufgeführt. Auch hier wird eine getrennte Betrachtung der protokoll- und der systemorientierten Ansätze vorgenommen.

Im **Kapitel 5** wird die Implementierungsstruktur des gewählten Systems hergeleitet und getrennt nach Hard- und Softwarefunktionen detailliert vorgestellt. Die Beschreibung einzelner Komponenten und einiger ihrer besonders hervorzuhebenden konzeptuellen internen Aspekten, sowie eine Darstellung des Zusammenwirkens der Einzelkomponenten bilden den Schwerpunkt dieses Kapitels.

Die Verifizierung des Systems und eine Beschreibung des eingesetzten Entwurfsablaufes - inklusive einigen Anmerkungen zur benutzten Entwurfsumgebung - ist im **Kapitel 6** dokumentiert. Neben einer rein funktionellen Simulation, die überwiegend auf der Ebene von Gattern aufsetzt, wurde auch eine statistische Simulation des Systems durchgeführt.

Eine Zusammenfassung sowie einige Anregungen für weiterführende Arbeiten werden im abschließenden **Kapitel 7** gegeben.

Kapitel **2**

Grundlagen und Umfeld der Kommunikationstechnik

In diesem Kapitel werden für die Arbeit relevante Begriffe und Modellvorstellungen aus dem Bereich der Kommunikationstechnik eingeführt. Die heutige Situation wird zusammengefaßt. Sich abzeichnende Tendenzen werden betrachtet, da diese die maßgeblichen Randbedingungen der Arbeit beinhalten.

2.1 Heutige Kommunikationsnetze

Der Bereich der elektrisch und elektronisch unterstützten Kommunikation besteht aus zwei Teilbereichen:

1. (Sprach-)Kommunikation zwischen Menschen
2. (Daten-)Kommunikation zwischen Maschinen

Die Anfänge der zwischenmenschlichen elektrisch unterstützten Kommunikationsform liegen schon über 100 Jahre zurück [185] und führten u.a. zu den weltumspannenden öffentlichen Telefonienetzen. Im Vordergrund stand und steht die dialogorientierte Fernübertragung von Sprache.

Die resultierenden Anforderungen an die zulässige Verzögerung erforderte ursprünglich die Einrichtung einer durchgehenden elektrischen Leitung zwischen den Kommunikationspartnern, woraus sich die Bezeichnung leitungsvermittelnd oder kurz CS (circuit switched) ableitet. Obwohl den Gesprächsteilnehmern von Telefonsystemen heutzutage kein physikalischer Draht mehr exklusiv für die gesamte Gesprächsdauer zugeteilt wird, liegt auch modernen CS-basierten Telefonsystemen aus logischer Sicht diese anschauliche Funktionsweise zugrunde.

Sieht man von der etwa 60 Jahre alten Frühform der Intermaschinenkommunikation des Fernschreibens ab, ergab sich die Notwendigkeit für die zweite - für die Arbeit wesentliche - Kommunikationsform in größerem Umfang erst in den 60-er und 70-er Jahren dieses Jahrhunderts durch die Verbreitung der Rechnertechnik [12]. Neben der Kommunikation auf Bauteil-, Platinen- und Geräteebene entstand schon bald ein Bedarf für den Austausch von Daten zwischen räumlich voneinander entfernten Maschinen.

Durch den nicht mehr unmittelbar beteiligten fehlertolerierenden Faktor Mensch überwog die Forderung nach Fehlerfreiheit die nach geringer Verzögerung. Die bestehenden Telefonienetze waren folglich nur bedingt für diese Kommunikationsform geeignet. Es entwickelten sich deshalb - zunächst innerhalb nichtöffentlicher Netze - Netztechnologien, deren wesentliches Merkmal die Loslösung von der CS-Vorstellung des permanenten elektrischen Kontakts zwischen den Kommunikationspartnern ist [40, 44]. Stattdessen werden einzelne Informationsteile (Datenpakete) als Basiseinheit angesehen, wodurch der Begriff paketvermittelnd oder PS (packet switched) entstand. Merkmale der für den privaten Bereich entwickelten Techniken werden nachfolgend eingeführt.

Bedingt durch rechtliche und wirtschaftliche Aspekte ist man zur Datenkommunikation über größere Entfernungen nach wie vor auf öffentliche Kommunikationsnetze angewiesen. Neben der Nutzung der bestehenden Telefonnetze entstanden auch eigene öffentliche Datennetze. Aus diesem Grunde wird auch der öffentliche Bereich angesprochen.

2.1.1 Öffentlicher Bereich

Die offensichtlichste und einfachste Möglichkeit, Maschinen über das öffentliche Netz kommunizieren zu lassen, ist das Einrichten von festen Leitungen. Nutzbandbreiten von lediglich einigen tausend Bits pro Sekunde, unverhältnismäßig lange Zeiten für einen Verbindungsaufbau sowie hohe und nicht dem tatsächlichen Verkehrsaufkommen angepaßte Gebühren sind gravierende Nachteile [12, 35, 42]. Da viele Telefonnetze noch analoge Signale übertragen, sind auch entsprechende Wandler von bzw. zu der digitalen Darstellungsform der Rechner erforderlich (Modems).

Zur Verringerung der Leitungskosten wurden Konzentratoren verschiedenster Komplexität entwickelt, an die mehrere Endgeräte angeschlossen werden können. Mit dem Mieten mehrerer Leitungen des öffentlichen Netzes und einer Verhinderung jeglichen fremden Zugriffs können quasi-private WANs (Wide Area Networks, Weitverkehrsnetze) entstehen. Deren Kosten und Effizienz stehen jedoch in einem ungünstigen Verhältnis. Außerdem boten die öffentlichen Netzbetreiber keine Möglichkeit an, mehrere solcher privater Datennetze untereinander zu verbinden.

Die Forderung nach der Entwicklung eigenständiger Datennetze für den Fernverkehr wurde daher immer dringender. Entscheidende Bedeutung fiel dabei der Vereinheitlichung des Zugangs zu einem solchen Netz zu. Resultierende Standards sind z.B. die CCITT X- und I-Serien. Neben Standards für den Zugang zu paketvermittelnden öffentlichen Datennetzen (Packet Switched Data Networks, PSDNs) wie z.B. X.25 [8, 82, 128], wurden auch solche für den expliziten Zugang zu leitungsvermittelnden öffentlichen Datennetzen (Circuit Switched Data Networks, CSDNs) festgelegt, z.B. X.21. CSDNs leiten den Verkehr transparent durch das komplette Netz. Es werden - im Gegensatz zu PSDNs -keinerlei Mechanismen zur Fehlerbehandlung oder Steuerung des Informationsflusses vom Netz durchgeführt. Stattdessen ist bei CSDNs der Anwender selbst für die Wahrung der Dienstgüte und die dazu notwendigen Protokollmechanismen zuständig.

Bekanntester Vertreter eines PSDNs ist das ursprünglich US-amerikanische und inzwischen weltweit operierende DARPA Internet [5]. Das deutsche, mit dem Internet verbundene Netz, welches im Weitverkehrsbereich auf X.25 basiert, wird DATEX-P genannt.

Mit der allgemeinen Digitalisierung sowohl der Übertragung als auch der Vermittlung in den weiterhin leitungsvermittelnd arbeitenden Telefonnetzen steht dem Anwender ein digitaler Netzzugang zur Verfügung, der direkt für die Datenübertragung ausgenutzt werden kann. Für diesen Anwender erscheint das Netz dann wie ein eigenständiges CSDN. Tatsächlich werden Sprache und Daten durch die Vereinheitlichung der Übertragungs- und Vermittlungstechnologie gemeinsam in einem Netz integriert. Daher auch die Bezeichnug ISDN (integrated services digital network , diensteintegrierendes digitales Nachrichtennetz, [3, 190, 324]).

2.1.2 Privater Bereich

Noch in den frühen 80-er Jahren dominierte das Prinzip zentraler Großrechner mit sternförmig angeschlossenen Endgeräten (Terminals) ohne lokale Intelligenz und Speicher. Klein-Rechenanlagen (Personal Computers, PCs) und Arbeitsplatzrechner mittlerer Leistungsfähigkeit (workstations) wurden erst als neue Produkte am Markt eingeführt [23, 217].

Der heute bedeutende Bereich der lokalen Netze (Local Area Networks, LANs) begann durch die Verbreitung des bekannten Ethernet und dessen einfachen Mehrfachzugriffprinzips des Mithörens und Erkennens von Kollisionen (CSMA-CD, Carrier Sense Multiple Access with Collision Detection) gerade erst an Bedeutung zu gewinnen. Bild 2.1 zeigt exemplarisch einige Aspekte heutiger LANs auf [12].

Standen anfangs die physikalische Anordnung des Netzwerkes und die Leistungsfähigkeit der Medienzugriffsverfahren im Brennpunkt des wissenschaftlichen Interesses (siehe z.B. [18, 86]), so gewannen

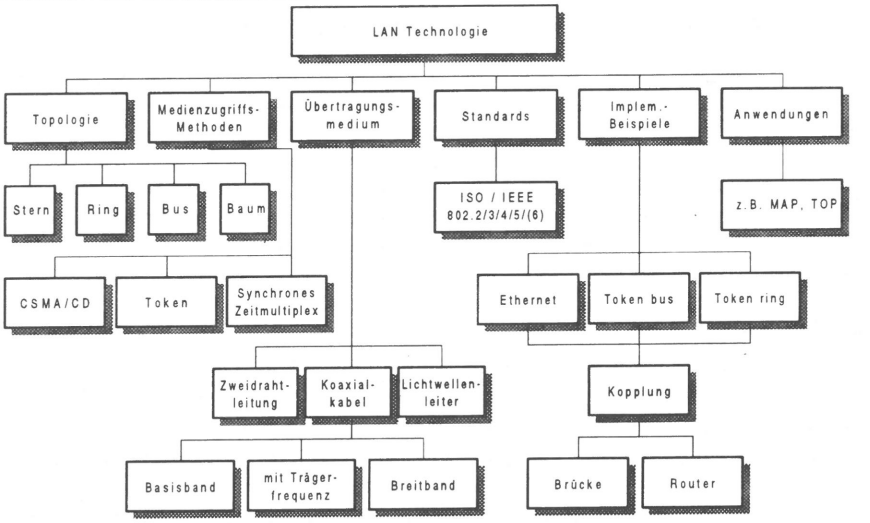


Bild 2.1: *Teilaspekte lokaler Netzwerke*

später eher pragmatische Aspekte etwa der Standardisierung und Ausrichtung auf spezifische Anwendungsfelder sowie verstärkt Fragen der Kopplung von Netzen im Vordergrund [244].

Obwohl der Bedarf für lokale Vernetzung zu Beginn der 80-er Jahre noch gering war und die von Ethernet bereitgestellte Bandbreite von 10 Mbit/s selbst für die meisten heutigen Anwendungen mehr als ausreichend ist sowie von vielen Rechenanlagen nicht verarbeitet werden kann, wurde schon früh damit begonnen, Konzepte für Netze mit höheren Bandbreiten (bis etwa 150 Mbit/s) und größeren maximalen geographischen Ausdehnungen (über 100 km) zu erarbeiten [263, 6].

Ein Teil der Hochgeschwindigkeits-LANs (HS-LANS, High-Speed LAN) ist für die Einbeziehung von Sprache ausgelegt. HS-LANs mit dieser hybriden Fähigkeit werden zur Unterscheidung auch als MAN (Metropolitan Area Network) bezeichnet [46, 88, 325].

Durch die technische Möglichkeit, Sprache zu integrieren, soll die klassische Trennung von Telefonie in Form der Nebenstellentechnik (private branch exchanges, PBXs) und reiner Datenkommunikation überwunden werden [310]. Gleiches gilt auch in umgekehrter Richtung: die klassische Nebenstellentechnik ist zunehmend bestrebt, Datenverkehr einzubinden. Teilweise kommen in modernen PBXs sogar LAN-Mechanismen und ein gemeinsames Medium zum Einsatz [272, 279].

In den Kapiteln 2.2.2 und 4.1.3.1 werden die wichtigsten heute diskutierten HS-Netze vorgestellt. Die Verwendung des Begriffs 'Netz' bezieht sich dabei primär auf die Charakteristik des Medienzugriffs. Fast alle Vorschläge wurden auch in Prototypimplementierungen umgesetzt. Für die Arbeit ergibt sich daraus der Schluß, daß die mediennahen Netzaspekte neuer Netze in der Regel nicht nur theoretisch festgelegt werden, sondern daß auch die Leistungsfähigkeit der verfügbaren Technologie mitberücksichtigt wird. Von der frühzeitigen Verfügbarkeit zugehöriger Bausteine kann damit ausgegangen werden.

Bedingt durch die infrastukturelle Bedeutung von Kommunikationsnetzen stellte sich heraus, daß nur die Konzepte eine Chance auf breite Akzeptanz haben, welche international standardisiert sind. Aus der obigen Liste sind das FDDI [375 - 380, 261] und DQDB [374, 126, 266].

2.2 Tendenzen der Kommunikationstechnik

Die Inhalte und Randbedingungen der Kommunikationstechnik besitzen einen sehr dynamischen Charakter. Zum einen erlauben die ständig verbesserten Technologien sowohl bei Bauelementen und Systementwurf als auch im Bereich der Übertragungstechnik die wirtschaftliche Erstellung immer leistungsfähigerer und intelligenterer Kommunikationssysteme. Zum anderen entwickeln sich ständig neue Anwendungen, die teilweise neue Anforderungen an ein Kommunikationssystem stellen. Im folgenden werden einige grundsätzliche Tendenzen der Kommunikationstechnik aufgezeigt, die daraus resultierenden Anforderungen abgeleitet sowie aktuelle Entwicklungen - wiederum getrennt nach öffentlichem und privatem Bereich - eingeführt.

2.2.1 Öffentlicher Bereich

Obwohl die Geschwindigkeit bei der Einführung des schmalbandigen diensteintegrierenden digitalen Nachrichtennetzes ISDN derzeit weit hinter den gesteckten Erwartungen zurückbleibt, sind grundsätzliche Entscheidungen für ein zukünftiges Netz schon gefällt worden [144, 171]:

Basierend auf öffentlich verfügbaren digitalen Übertragunssystemen mit fest vorgegebenen Bandbreiten, der sogenannten synchronen digitalen Hierarchie [275], soll ein universelles und flexibles Netz aufgebaut werden [15]. Da die Charakteristika zukünftiger Dienste sowie deren resultierende Anforderungen an ein Übertragungssystem größtenteils weder bekannt noch abschätzbar sind, hat sich das internationale Gremium der öffentlichen Netzbetreiber CCITT (Comité Consultatif International Télégraphique et Téléphonique) für ein neues asynchrones Übertragungsprinzip als Kern des zukünftigen Breitband-ISDNs (B-ISDN) entschieden [392]. Die Funktionsweise und wesentlichen Merkmale dieses asynchronen Transfer Modes ATM sind in Bild 2.2 grafisch verdeutlicht [13, 32, 76, 206]:

Jeglicher Verkehr wird in kleine Pakete konstanter Länge (sogenannte Zellen) aufgespalten. Die Erzeugung aufeinanderfolgender Zellen einer Kommunikationsquelle als auch deren anschließende Übertragung geschieht ohne feste zeitliche Relation, eben asynchron. Neben der dadurch möglichen Flexibilität bezüglich aktuell benötigter Bandbreite kann ein ATM-basiertes Netz im Mittel auch besser ausgelastet werden als Netze mit anderen Übertragungsverfahren, weil Schwankungen im Verkehrsaufkommen einzelner Teilnehmer bei genügend großer Teilnehmerzahl ein statistisch beschreibbares Verhalten aufwei-

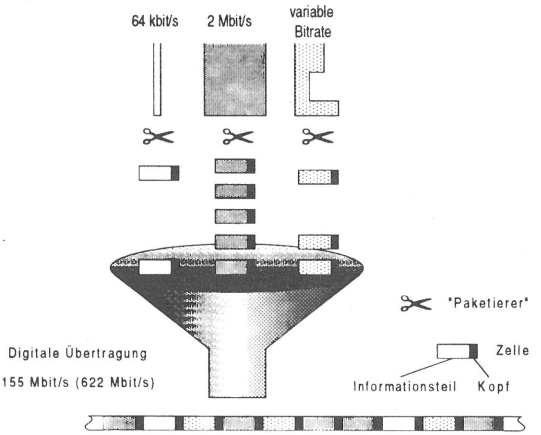


Bild 2.2: *Prinzip von ATM*

sen, welches von ATM ausgenutzt werden kann.

In verschiedenen Ländern sind inzwischen schon Feldversuche gestartet worden oder stehen unmittelbar vor deren Start. Die Entwicklung von ATM-Komponenten wird von allen Großfirmen der Kommunikationstechnik mit Nachdruck betrieben. Es bleibt anzumerken, daß bis heute fast ausschließlich die mediennahen Aspekte von ATM betrachtet werden [125] und nur wenige Aktivitäten sich mit der Problematik höherer Abstraktionsebenen inklusive der Ankopplung an den Anwendungsrechner befassen.

2.2.2 Privater Bereich

Die Kommunikation im privaten Bereich läßt sich drei Kategorien zuordnen (Bild 2.3):

- Zentrale Rechner und zugehörige - meist unmittelbar benachbarte - Peripheriegeräte wie z.B. Speicherwerke oder Drucker sollen miteinander verbunden werden (*Back-end*)
- Geographisch verteilte leistungsfähige Rechenanlagen wollen kommunizieren (*Front-end*)
- Mehrere lokale Netze sollen gekoppelt werden (*Backbone*)

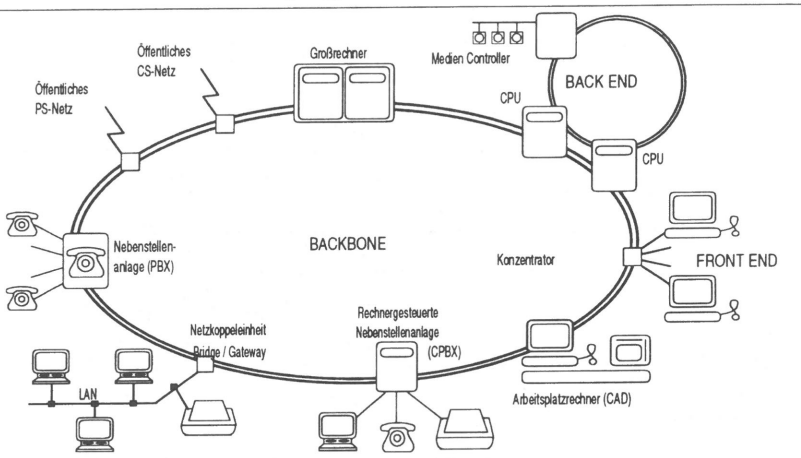


Bild 2.3: *Einsatzmöglichkeiten für zukünftige HSLANs/MANs*

Alleine aus diesen Anwendungsfeldern lassen sich Anforderungen sowohl an erhöhte Bandbreite als auch an größere geographische Ausdehnungen ablesen. Zusätzlich ändern sich auch die Inhalte und Formen der Kommunikation stetig [243]:

- Lokale Netze verbanden anfangs jeweils weitgehend autonome Rechenanlagen; der Austausch von vorverarbeiteten Daten stand im Vordergrund. Heute werden mit LANs verteilte Gesamtsysteme realisiert [25], bei denen Ressourcen und Programme nicht mehr lokal in jeder Rechenanlage verfügbar sein müssen, sondern über das ganze Netzwerk verteilt sein können.
- Gestiegene Qualitätsansprüche - z.B. Grafik statt Text - und die Einführung neuer Dienste wie Bewegtbildkommunikation sowie die Kombination von Diensten - Stichwort: Multimedia [84, 175, 180] - implizieren härtere Anforderungen an Leistungsfähigkeit und Güte von LANs.
- Die Einbeziehung von sprachähnlicher Kommunikation und damit die Überwindung der Zweiteilung der Kommunikationswelt bewirkt weitere bisher unbekannte Anforderungen [243].

In Kapitel 2.1.2 wurden Vertreter von HS-LAN/MAN-Arbeiten genannt, die das Konzeptstadium längst hinter sich haben. Die dort einheitlich anzutreffende Eigenschaft der Skalierung bestehender LAN-Verfahren mit von allen Teilnehmern gemeinsam genutztem Medium ist allerdings beim Übergang zu Band-

breitenbereichen von mehr als einer Milliarde Bits pro Sekunde nur eine von vielen möglichen Varianten: Parallelisierung und Vermaschung werden dabei ebenso diskutiert wie veraltet erscheinende Konzepte mit zentralem Vermittlungsknoten. In Kapitel 4.1.1.1 wird näher darauf eingegangen.

Auch im privaten Bereich sind fast alle Arbeiten sehr nah am Medium, d.h. Übertragungssystem und Medienzugang stehen im Vordergrund der weltweiten Forschungen auf diesem Gebiet.

2.3 Architekturmodelle für offene Kommunikation

Kommunikationssysteme sind - bedingt durch die große Anzahl unterschiedlicher zu erfüllender Aufgaben - durch einen hohen Grad an Komplexität gekennzeichnet. Eine grundlegende Methode der Modellierung ist das Einführen von aufeinander geschichteten Abstraktionsebenen [7, 22]. Jede Schicht stellt der folgenden - höheren - Abstraktionsebene Grundfunktionen zur Verfügung, die benutzt werden können, ohne Details der Funktionsrealisierung kennen zu müssen (Bild 2.4).

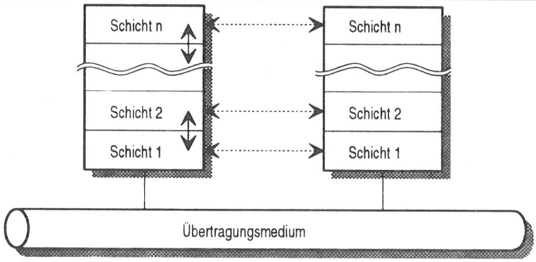


Bild 2.4: *Allgemeines Prinzip der Schichtung*

In einem Kommunikationssystem liegt es nahe, für die beiden logisch beteiligten Seiten Sender und Empfänger eine symmetrische Anordnung der Schichten zu wählen [35, 42]. Eine Funktion für die Schicht (N) wird von der darunterliegenden Schicht (N-1) in Zusammenarbeit mit der Schicht-(N-1) des Kommunikationspartners erbracht. Die Regeln für die Zusammenarbeit zwischen Schichten derselben Abstraktionsebene werden allgemein als *Protokoll* bezeichnet [293].

Die Granularität und damit die Anzahl der Schichten kann frei gewählt werden. Die Modellierung kann sich dabei am realen Systemaufbau orientieren; aber auch logisch abstrakte Aufteilungen - beispielsweise zur Erhöhung des Systemverständnisses - sind möglich.

2.3.1 Ein dreischichtiges Modell

Betrachtet man heutige Kommunikationssysteme, erkennt man drei voneinander unabhängige Komponenten [329], welche sich für eine Schichtung anbieten (Bild 2.5):

- Als Benutzer eines Kommunikationssystems sieht man im wesentlichen die anwendungsbezogene Komponente, welche u.a. für die korrekte Syntax und Semantik von ausgetauschter Information zuständig ist.
- Für die Übertragung der Information ist eine technologiespezifische Komponente erforderlich, die vom physikalischen Netz abhängt und z.B. auch den Netzzugang mit umfaßt.
- Spezielle Kommunikationssoftware führt die Anpassung zwischen den zwei genannten Komponenten durch. Diese Komponente ist insbesondere für den korrekten Informations*transport* verantwortlich (transportorientierte Komponente).

Da jede der drei Schichten des obigen Modells immer noch eine hohe Komplexität aufweist, wurden verschiedene Vorschläge zur weitergehenden Verfeinerung gemacht. Es existieren dabei sowohl hersteller-

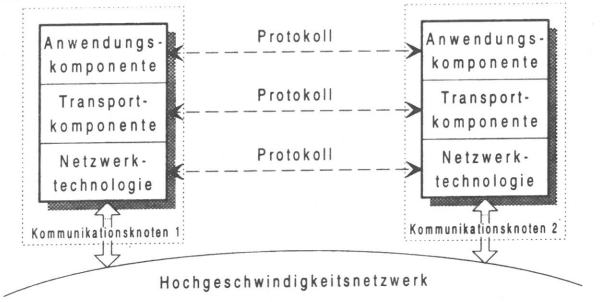


Bild 2.5: *Das Drei-Schichten Modell*

spezifische als auch herstellerunabhängige Architekturmodelle. Die mit SNA und DNA bezeichneten Netzwerkarchitekturen [12] der Firmen IBM bzw. DEC sind prominente Vertreter der herstellerspezifischen Modelle, das OSI-Referenzmodell, welches nachfolgend beschrieben wird, hat sich mittlerweile weltweit als *das* grundsätzliche Modell etabliert.

2.3.2 Das ISO/OSI-Referenzmodell

Um eine offene Kommunikation zwischen Endgeräten verschiedenster Hersteller zu ermöglichen, entwickelte die internationale Standardisierungsorganisation ISO Ende der 70-er Jahre das in [140, 212, 328] beschriebene Referenzmodell für die Verbindung von offenen Systemen (OSI-RM) ein. Die daraus abgeleitete Norm [333] beschreibt die Architekturkonzepte und wesentlichen Strukturen des OSI-RMs. Kernpunkte des Modells sind die Aufteilung der Aufgaben eines allgemeinen Kommunikationsendgerätes in sieben Schichten sowie die Einführung von Diensten und zugehörigen Dienstzugangspunkten als Bindeglieder zwischen den Schichten. Beide Aspekte werden nachfolgend beschrieben.

2.3.2.1 Die 7 Schichten des OSI-Referenzmodells

Bild 2.6 zeigt die 7 Schichten eines Kommunikations-Endknotens gemäß dem OSI-RM, eine an die Bedürfnisse von LANs angepaßte Verfeinerung, sowie eine Zuordnung zu dem im vorigen Kapitel eingeführten dreischichtigen Modell.

Die technologieabhängige Komponente umfaßt neben der Bitübertragungsschicht 1 den Anteil der Datensicherungsschicht 2, der den logischen Medienzugriff übernimmt. Dies ist typischerweise im Bereich der

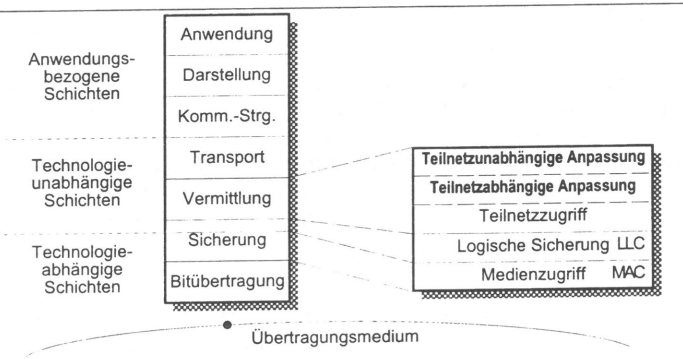


Bild 2.6: *Die 7 Schichten des ISO/OSI-Referenzmodells*

lokalen Netzwerke anzutreffen, da dort überwiegend lediglich ein Medium zur gemeinsamen Benutzung zur Verfügung steht und der Zugriff zu diesem entsprechend verwaltet werden muß. Um diesem Sachverhalt Rechnung zu tragen, wurde die ursprüngliche Schicht 2 des OSI-RMs in die zwei Unterschichten Medienzugriff (2a, MAC - Media Access Control) und logische Sicherung des Übertragungsabschnitts (2b, LLC - Logical Link Control) aufgeteilt [369, 370].

Die transportorientierte Komponente umfaßt die Schichten 2b bis 4 des OSI-RMs. Bedingt durch die Komplexität bei der Kopplung unterschiedlichster LANs [75] ergab sich auch für die Vermittlungsschicht 3 des OSI-RMs eine feinere Unterteilung.

Die Funktionalität der anwendungsorientierten Komponente des dreistufigen Modells wurde im OSI-RM auf 3 unabhängige Schichten verteilt, welche ihrerseits teilweise auch noch weiter untergliedert wurden.

Die jeweiligen Hauptfunktionen der 7 Schichten werden an dieser Stelle lediglich stichworthaft aufgeführt. Zur Vertiefung der Thematik steht dem interessierten Leser umfangreiche und detaillierte Literatur zur Verfügung (u.a. [12, 217, 35, 42]). Die relevanten Standards sind [333 - 370].

Die *Bitübertragungsschicht 1* ist für die Übertragung von Bitströmen über ein physikalisches Medium zuständig. Wahl der Kodierung, Einhaltung und Erkennung eines Taktes, elektrische bzw. optische Pegel, Wellenlängen, Dämpfung sowie mechanische Eigenschaften des Mediums und der angeschlossenen Stecker sind einige der Begriffe, die in der Bitübertragungsschicht bestimmend sind.

Die Hauptfunktion der *Sicherungsschicht 2* wird durch deren Bezeichnung deutlich: die gesicherte Übertragung von Bitströmen über einen Verbindungsabschnitt zwischen benachbarten Stationen. Desweiteren sollen die höheren Schichten vollständig von der Notwendigkeit des Wissens über Einzelheiten der Bitübertragungsschicht befreit werden. Im Bereich lokaler Netze fällt der gesamte Bereich des Medienzugriffs ebenfalls in den Aufgabenbereich der Sicherungsschicht (siehe Bild 2.9).

In der *Vermittlungsschicht 3* sind Wegewahl und Adressierung dominierende Begriffe. Besonders umfangreich wird die Funktionalität der Schicht 3 in Knoten, die zwischen den Kommunikationsteilnehmern angeordnet sind und u.U. verschiedene Teilnetze miteinander verbinden [359, 366, 75].

Die *Transportschicht 4* stellt den Benutzern eine Datenübertragung mit wählbaren Qualitätsmerkmalen zur Verfügung. Theoretisch ist das Erscheinungsbild der Schicht 4 völlig unabhängig von dem bzw. den tatsächlich darunter liegenden Netz bzw. Teilnetzen. Im Unterschied zur Sicherungsschicht bezieht sich die Schicht 4 direkt auf die Kommunikationsteilnehmer (Englisch: End-to-End) und nicht nur auf physikalisch benachbarte Stationen [355 - 358].

Das Hauptaugenmerk der *Kommunikationssteuerungsschicht 5* liegt auf der Synchronisation und Organisation des Dialogs zwischen Benutzern. Damit wird die Kontrolle, ob und wann - z.B. bei Halbduplexbetrieb - gesendet werden kann, eine wesentliche Funktion [353, 354].

Wie der Name der *Darstellungsschicht 6* vermuten läßt, ist diese Schicht für die Form und innere Struktur der übertragenen Daten zuständig. Mit der Frage der gewählten Zeichenkodierung (z.B. ASCII oder EBCDIC) fängt es an, die kryptografische Behandlung und eventuelle Komprimierung des Datenstromes stellen die zweite Funktionsgruppe dar, und schließlich muß auch die stark variierend ausführbare interne Anordnung von Datensätzen beachtet werden. Dazu muß den eigentlichen Nutzdaten allgemeinverständliche Information der Syntax mitgegeben werden. Bei den ISO/OSI-Standards ist hierfür die abstrakte Syntaxnotation ASN.1 eingeführt worden [349 - 352].

Die *Anwendungsschicht 7* ist nicht - wie oft fälschlicherweise interpretiert - die eigentliche Anwendung, sondern stellt nur vereinheitlichte Modelle für bestimmte Anwendungskategorien dar. So könnte z.B. ein ABS-Element der Schicht 7 ein Modell eines elektronischen Antiblockiersystems mit dessen abstrakten Hauptfunktionen Maximalverzögerung, Freigabe und eventuellen prozentualen Zwischenwerten repräsen-

tieren. Die jeweilige konkret vorliegende Anwendung - im Beispiel einerseits die Geberelektronik, andererseits die elektronische Ansteuerung des Bremssystems - muß nun lediglich an das abstrakte Modell angepaßt werden. Eine korrekte Funktion ist dann auch bei einer Kombination von Komponenten verschiedener Hersteller möglich.

Bestehende Diensttypen sind u.a. die Übertragung, Zugriffsregelung und Manipulation von Dateien, Verzeichnissen und Aufträgen, ein allgemeiner Nachrichtenaustausch, ein virtuelles Endgerät und Netzmanagementinteraktionen (file transfer, access and management - FTAM; directory service - DS; job transfer and manipulation - JTM; message handling service - MHS, virtual terminal - VT; common management information service - CMIS. [338 - 348, 382, Erläuterungen in 12].

2.3.2.2 Interaktion zwischen benachbarten Schichten

Neben der Festlegung der 7 Schichten und deren prinzipieller Funktionalität repräsentieren die Mechanismen zum Zusammenspiel der Schichten - sowohl auf demselben Kommunikationsknoten als auch zwischen den entfernten, an der Kommunikation beteiligten Knoten - das zentrale Element des OSI-RMs.

Dazu werden Dienste festgelegt, welche jede Schicht (N-1) der nächsthöheren Schicht (N) desselben Knotens zur Verfügung stellt [293]. Der Dienstnutzer (Schicht (N)) kann einen Dienst vom Diensterbringer ohne Kenntnis der Implementierung über abstrakt definierte Dienstzugangspunkte (SAPs, Service Access Points) in Anspruch nehmen. Das Ergebnis wird vom Dienstbenutzer (Schicht (N)) bearbeitet und anschließend der Schicht (N+1) als erweiterter Dienst angeboten (siehe Bild 2.7).

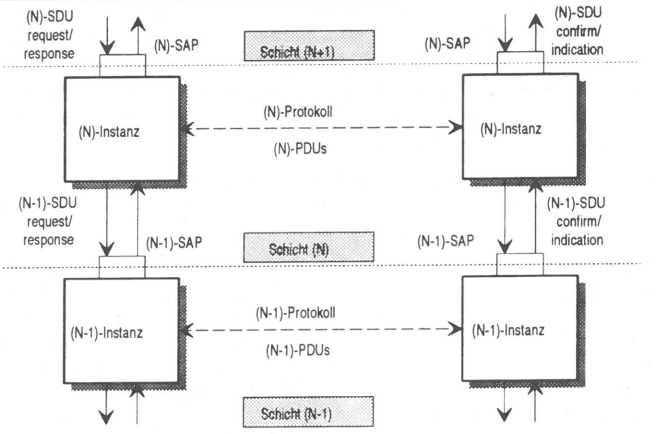


Bild 2.7: *Das zentrale Prinzip des ISO/OSI-Referenzmodells*

Die aktiven Elemente einer Schicht, welche für die Erweiterung des Dienstes in Empfangsrichtung bzw. die Nutzung darunterliegender Dienste in Senderichtung zuständig sind, werden im Deutschen als Instanzen bezeichnet (im Englischen dagegen nicht mit instances sondern mit entity). Zur Bereitstellung des vom Dienstbenutzer geforderten Dienstes tauschen die (N)-Instanzen der an der Kommunikation beteiligten Knoten Nachrichten untereinander aus (horizontale Schichtenkommunikation). Syntax und Semantik sind durch das (N)-Protokoll festgelegt. Die ausgetauschten Nachrichten werden daher auch als Protokolldateneinheiten (Protocol Data Units, PDUs) bezeichnet. Analog dazu werden die (vertikal) zwischen benachbarten Schichten eines Kommunikationsknotens ausgetauschten Nachrichten Dienstdateneinheit (Service Data Unit, SDU) genannt [336].

Die Generierung einer PDU geht folgendermaßen vor sich (Bild 2.8). Will ein Anwender den Dienst einer

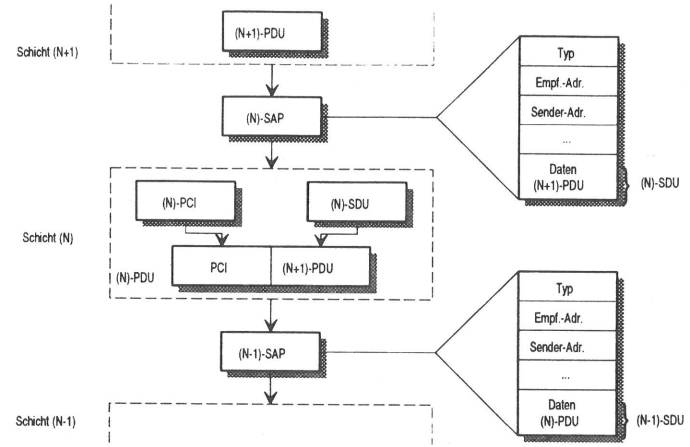


Bild 2.8: *Prinzip der Erzeugung von PDUs*

darunterliegenden Schicht in Anspruch nehmen, übergibt er die an den Kommunikationspartner zu versendenden Daten mittels einer SDU an die diensterbringende darunterliegende Schicht.

Diese verändert gegebenenfalls Form und Größe der Daten, erstellt eigene Protokollkontrollinformation (protocol control information, PCI) und setzt diese in Form von Steuerköpfen vor jeden entstandenen Datenteil. Die nun vollständigen PDUs dieser Schicht werden zur Übertragung bzw. Weiterverarbeitung als SDUs an die nächste Schicht übergeben. Der Vorgang wiederholt sich analog auf jeder weiteren Schicht, wobei die PDUs der höheren Schichten jeweils vollständig transparent als Nutzerdaten weitergeleitet werden. Bild 2.9 veranschaulicht diesen Sachverhalt.

Der Austausch von PDUs kann grundsätzlich auf zwei verschiedene Arten durchgeführt werden: verbin-

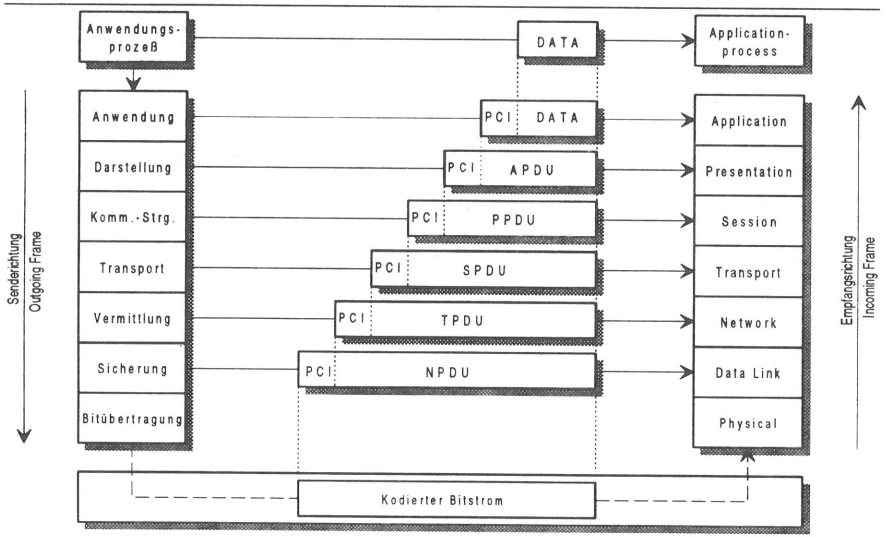


Bild 2.9: *Interaktion mehrerer Schichten*

dungsorientiert oder verbindungslos. Im ersten Fall wird zwischen den beteiligten Schichtinstanzen für die Dauer der Interaktion eine logische Beziehung - eben eine Verbindung - etabliert. Der Ablauf einer solchen Interaktion ist dabei in die 3 Phasen (Verbindungsaufbau, Datenübertragung und Verbindungsabbau) unterteilt. Der Vorteil dieser Kommunikationsform ist die Realisierbarkeit einer gesicherten Datenübertragung, da Maßnahmen zur Reihenfolgesicherung, zur Feststellung und Behebung von fehlerhaften oder fehlenden PDUs sowie eine Steuerung des Informationsflusses möglich sind (siehe auch 3.1.4).

Die verbindungslose Kommunikation ist durch die Übertragung voneinander unabhängiger PDUs charakterisiert. Da ein logischer Zusammenhang zwischen den einzelnen PDUs fehlt, muß jede PDU selbstbeschreibend sein, d.h. erforderliche Information zu deren Identifizierung und korrekter Übertragung vollständig selbst mitführen. Die verbindungslose Kommunikationsform zeichnet sich durch eine hohe Effizienz bei der Übertragung kurzer Nachrichten aus, weil die aufwendigen Phasen des Verbindungsauf- bzw. -abbaus entfallen können. Nachteilig ist die schwerer zu realisierende Behandlung von Fehlern.

Kapitel **3**

Problemanalyse

Bild 3.1 zeigt die heute auf den verschiedenen Schichten des OSI-RMs erreichbaren Durchsatzwerte [52, 92, 97, 167, 172, 173, 215, 248, 288, 292, 294]. Oberhalb der technologieabhängigen Schicht wird die Einheit Pakete (statt Bits) pro Sekunde benutzt, da der paketbezogene Aufwand dort klar dominiert.

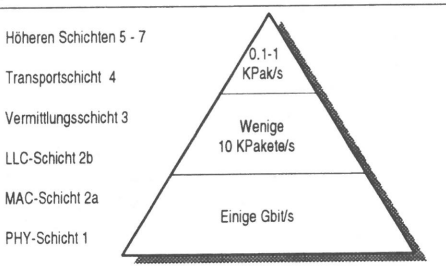


Bild 3.1: *Typische schichtenspezifische Leistungsfähigkeit*

Bevor einige Ursachen für den erheblichen Unterschied zwischen der Leistungsfähigkeit des physikalischen Netzes (technologieabhängige Komponente) und der erzielbaren Leistung oberhalb des Transport-Systems - im weiteren Verlauf der Arbeit mit TpS abgekürzt - aufgezeigt werden, folgt zunächst eine Beschreibung der grundsätzlichen Aufgaben eines TpS. Zusätzlich zu der abstrakten Sichtweise gemäß dem OSI-RM wird auch auf Funktionalitäten eingegangen, die zur Erbringung der abstrakten Anforderungen in einer realen Umgebung notwendig sind.

3.1 Logisch-funktionelle Aufgaben eines Transportsystems

Ein TpS umfaßt die Schichten 1 bis 4 des OSI-RMs, d.h. alle außer den anwendungsabhängigen Schichten. Das äußere Verhalten eines TpS entspricht daher der Schnittstelle zwischen den Schichten 4 und 5 des OSI-RMs. Daraus ergibt sich auch die wesentliche Eigenschaft eines TpS, nämlich die Existenz einer logischen Beziehung zwischen Kommunikationsendpunkten [267, 312]. Anzahl und spezifisches Verhalten von real zwischen den Endpunkten vorhandenen Netzabschnitten sind für Benutzer eines TpS völlig transparent. Der Begriff der Transparenz beinhaltet dabei sowohl, daß ein Benutzer des TpS keinerlei Wissen über die geographische Lage des Kommunikationspartners oder dazwischenliegende Netzwerke zu haben braucht, als auch, daß weder Form noch Inhalt der transportierten Information in irgendeiner Form beeinflußt werden.

Etwas allgemeiner kann ein ideales TpS durch die folgenden Merkmale charakterisiert werden:

- Zuverlässig; die transportierte Information darf also insbesondere weder gestört oder gar verloren werden noch mehrfach oder in falscher Reihenfolge beim Empfänger ankommen.
- Durchsatz und Verzögerung sollen akzeptabel und möglichst einheitlich sein.

- Die Unabhängigkeit von Kommunikationsströmen muß innerhalb des TpS gewahrt bleiben.
- Diese Forderungen sollen ohne Einschränkungen mit vertretbaren Kosten realisierbar sein. (Durch Optionen oder auch die direkte Angabe von Qualitätsmerkmalen durch den TpS-Benutzer werden in realen Systemen TpS-interne Entscheidungen, etwa bei notwendigen Anforderungen von Ressourcen, beeinflußt. Damit wird eine weitgehende Annäherung an einzelne, für den spezifischen Benutzer relevanten, Punkte des idealen Verhaltens erreicht.)

Eine erste Verfeinerung des angestrebten Verhaltens eines TpS führt zu den Teilbereichen: Adressierung, Fehlerbehandlung, Synchronisierung, Flußsteuerung und Maßnahmen zur Anpassung von Größen. Die grundsätzlichen Aspekte jedes einzelnen Teilbereiches werden in den folgenden Unterkapiteln erläutert.

3.1.1 Adressierung

Um Ressourcen sowohl zu identifizieren als auch zu lokalisieren, ist die Unterstützung eines globalen Adreßraumes eine der Funktionen eines TpS. Kann auf den unteren Schichten des OSI-RMs noch eine Zuordnung zwischen Adresse und physikalischem Objekt (Rechner, Drucker, Netzwerk-Controllerkarte, Gateway ...) bestehen, so werden ab der Schicht 4 logische Einheiten - also Prozesse, Dienste u.ä. - adressiert [103]. Die Benennung logischer Einheiten und damit von potentiellen Benutzern eines TpS sind jedoch von Rechner zu Rechner sehr verschieden.

Das im vorherigen Kapitel eingeführte OSI-RM mit dessen Kernprinzip der Dienstzugangs- und Verbindungsendpunkte (SAPs/CEPs) gibt hier keinerlei konkrete Anhaltspunkte. Und das, obwohl Länge und Struktur von Identifizierungen einen erheblichen Einfluß auf den notwendigen Aufwand und die Komplexität der Umsetzung in einer tatsächlichen Behandlung haben.

Für die unteren Schichten des OSI-RMs gibt es verschiedene - teilweise auch standardisierte - Empfehlungen über Adreßformate [110]. Das global eindeutige Schema der 48-Bit Adressen für den Medienzugang sei hier genannt. Auch bei den Dienstzugangspunkten zur Vermittlungsschicht bestehen zumindest einige vorgegebene Varianten sowie eine Vereinheitlichung von einigen Bitpositionen und deren Bedeutung: Authority and Format Identifier (AFI) innerhalb des Initial Domain Part (IDP), siehe z.B. [245].

3.1.2 Fehlerbehandlung

Für den Benutzer eines TpS ist die Forderung nach Zuverlässigkeit des Informationstransfers von herausragender Bedeutung. Fehlerfreiheit wiederum nimmt innerhalb des Begriffes Zuverlässigkeit einen breiten Raum ein. Berücksichtigt man, daß selbst modernste Technik nie auf Dauer völlig fehlerfrei sein kann, ergibt sich die Fehlerbehandlung als ein weiterer Funktionsbereich eines TpS.

Neben der Verfälschung von Dateneinheiten, bedingt etwa durch statistisch auftretende Störungen auf der physikalischen Übertragungsstrecke, können auch ganze Dateneinheiten verloren gehen oder vervielfacht werden. Weitere Fehlerkategorien sind z.B. Reihenfolgestörungen, Verletzung von Längen- und anderen formalen Vorgaben, Adressierungsfehler oder falsches zeitliches Verhalten.

Ein übliches Verfahren zur Erkennung - und teilweise auch zur Behebung - von Verfälschungen ist die Übersendung von zusätzlicher Information und das Ausnutzen der dadurch erreichten Redundanz. Prominente Vertreter solcher fehlererkennenden bzw. -korrigierenden Mechanismen sind Paritätsprüfungen und zyklische Prüfsummen (CRCs, cyclic redundancy checks) sowie redundante Übertragungs-(block-)codes. Daneben existieren vielfältige weitere Kategorien von Codes und Algorithmen mit stark unterschiedlichen Komplexitäten und Wirkungsweisen (z.B. [31, 41]).

Die meisten übrigen Fehlertypen lassen sich durch das Einführen von Sequenznummern einerseits sowie eines Mechanismus' zur Anforderung wiederholter Aussendung andererseits erkennen und beheben.

3.1.3 Synchronisation

Kommunikation ist nur dann sinnvoll möglich, wenn die beteiligten Kommunikationspartner sowohl eine einheitliche Sprache sprechen, d.h. sich zumindest auf einige grundsätzlichen Aspekte geeinigt haben, als auch ein gewisses Maß an Wissen über die aktuelle Situation des jeweils anderen Partners besitzen. Beide Gesichtspunkte gehören zum Begriff Synchronisation. Somit fallen die Größe der Datengrundeinheit, die eindeutige Markierung von Nachrichtenanfang und -ende oder die Unterscheidung von Nutz- und Kontrollinformation ebenso unter diesen Begriff wie Verbindungssteuerung und der Austausch jeglicher Art von Kontrollinformation zwischen den Kommunikationspartnern. Eine funktionierende Synchronisation wird damit zum wichtigsten Bestandteil des oben betrachteten Aufgabenbereichs der Fehlerbehandlung.

Um die Forderung nach Zuverlässigkeit erfüllen zu können, müssen beide Kommunikationspartner Wissen über den aktuellen Zustand besitzen. Das beinhaltet u.a. die Kenntnis darüber:

- welche Daten erfolgreich abgeschickt wurden,
- wann die nächsten Daten gesendet werden dürfen und
- welche Daten erfolgreich übertragen wurden.

Werden die zugehörigen Informationen explizit gespeichert und verwaltet, so ist dies gleichbedeutend damit, daß eine Verbindung zwischen den Kommunikationsendpunkten existiert (siehe nächsten Abschnitt). Man spricht dann von verbindungsspezifischen Daten. Natürlich können diese Daten auch bei der Verwaltung von Systemressourcen verwendet werden.

Die zentrale Problematik bei der Synchronisation stellt das nicht ideale Verhalten in Bezug auf endliche Übertragungszeit und Fehlerfreiheit des TpS dar. Es gilt, eine sinnvolle Abwägung zwischen benötigtem Aufwand, Effizienz und angestrebtem Grad an Zuverlässigkeit zu finden.

3.1.4 Verbindungsverwaltung

Wie schon im vorigen Abschnitt deutlich wurde, setzt eine zuverlässige Kommunikation bei den beteiligten Kommunikationspartnerinstanzen ein gewisses Maß an Wissen über den bisherigen Verlauf und den aktuellen Zustand der logischen Beziehung, eben eine Synchronisation, voraus. Eine naheliegende Implementierungsmöglichkeit besteht darin, daß jede Seite für sich über die ihr direkt zur Verfügung stehenden Informationen entsprechend Buch führt und den eigenen Zustand gelegentlich der Partnerinstanz mitteilt bzw. den eigenen Stand durch empfangene Zustandsinformation der Gegenseite aktualisiert. Wird für die Bereitstellung eines Dienstes diese Implementierungsform zur Unterstützung der Synchronisation eingesetzt, so wird von einem verbindungsorientiertem Dienst gesprochen. Die konträre Implementierungsform eines Dienstes der Schicht namens x, bei der unabhängige PDUs zwischen den Kommunikationsinstanzen ausgetauscht werden, heißt entsprechend verbindungsloser oder auch Datagramm-Dienst der Schicht x (connection oriented bzw. connectionless x-service, COxS / CLxS).

Ein Austausch von Verbindungsinformation setzt zunächst eindeutige Bezeichnungen voraus, mit der exakt die gewünschte Verbindung durch die jeweilige Partnerinstanz angesprochen werden kann. Dazu muß jede Seite der anderen den eigenen Bezeichner mitteilen. Dies wird mit speziell gekennzeichneten PDUs zu Beginn einer Verbindung, beim Verbindungsaufbau, durchgeführt. Über eine bestehende Verbindung können dann die eigentlichen Nutzdaten ausgetauscht werden. Sind keine Nutzdaten mehr zu übertragen, so können - müssen aber nicht - die verbindungsspezifischen Daten wieder gelöscht werden, man spricht dabei von Verbindungsabbau.

Das Verbindungskonzept kann für weit mehr als nur für Synchronisationszwecke benutzt werden. So kann etwa einer Verbindung ein fester Anteil der Systemressourcen fest zugewiesen werden, um bestimmte Dienstqualitäten (Qualities of Service, QoS) zu garantieren. Die Mechanismen zur Wegesuche können durch Verbindungen vereinfacht werden, indem den Verbindungen jeweils feste Wege

zugeordnet werden und damit eine zeitkritische Entscheidung über den einzuschlagenden Weg für jede einzelne PDU vermieden werden kann. Die Kommunikationspartner können "ihre" Verbindung darüber hinaus auch entsprechend den jeweiligen Bedingungen konfigurieren, indem die Benutzung oder das Weglassen bestimmter Optionen und Parameter der Verbindung zugeordnet werden.

Alle aufgeführten Eigenschaften von Verbindungen machen eine aufwendige Verhandlungsphase während des Verbindungsaufbaus notwendig. Desweiteren muß durch besondere Maßnahmen sichergestellt werden, daß Verbindungsauf- und -abbau sehr sicher und zuverlässig durchgeführt werden. Dies wird entweder explizit durch doppelte Quittierungen ("three-way handshake") oder implizit durch entsprechend aufeinander abgestimmte Zeitgeber realisiert [311].

Zur Verbindungsverwaltung gehört auch die Vergabe von Bezeichnern und das Sicherstellen von deren Eindeutigkeit. Auf diesen Punkt wird im Abschnitt über die Ressourcenverwaltung (3.2.7) eingegangen.

3.1.5 Datenflußsteuerung

Jegliche Interaktion erfordert bestimmte Systemressourcen. Bei einer Kommunikationsbeziehung muß beispielsweise Speicherplatz für abgehende und ankommende Informationsteile verfügbar sein, und die Protokollbearbeitung benötigt Rechenkapazität. Ressourcen in realen Systemen sind nie unbegrenzt verfügbar. Bedingt durch unterschiedliches Leistungsvermögen der an der Kommunikation beteiligten Systemkomponenten kann es daher zu Engpässen kommen. Mit Datenflußsteuerung sind alle Mechanismen bezeichnet, die den Umfang des Informationsflusses vom Sender durch das Netzwerk zum Empfänger so regulieren, daß der Empfänger diesen handhaben kann. Für diese Flußsteuerung zwischen Endpunkten der Kommunikation gibt es zwei Grundprinzipien mit jeweils einer Vielzahl möglicher Varianten:

1. Die absolute Quantität zu sendender Informationseinheiten wird reguliert oder
2. die erlaubte Informationsmenge pro Zeiteinheit (Rate) wird überwacht.

Mechanismen, die eine maximale oder durchschnittliche Rate des Informationsflusses als zu steuernde Größe gewählt haben, können in Umgebungen mit stark schwankenden Verkehrslasten zu Problemen führen. Zudem ist die korrekte Implementierung solcher Ratensteuerungsmechanismen nicht trivial, da u.a. das Netzwerk die beim Sender eingestellte Rate stark verändern kann. Nicht zuletzt deshalb werden in vielen heutigen Netzen quantitätsbasierte Verfahren zur Datenflußsteuerung eingesetzt.

Teilt der Empfänger dem Sender lediglich eine maximale Menge von Dateneinheiten mit, die gesendet werden dürfen, und wird keine explizite Folgenumerierung mitgeteilt (siehe z.B. Arpanet [5]) - können verlorengegangene oder duplizierte Nachrichten zu zweideutigen Situationen führen. Daher werden meist Mechanismen angewandt, die neben der Anzahl zu sendender PDUs explizit die Folgenummern derjenigen PDUs angeben, welche akzeptiert werden. Innerhalb dieser Verfahren dominieren kombinierte Fenster- und Kredit-Mechanismen, welche die folgende Funktionsweise besitzen:

Wie bei der Fehlerbehandlung erhalten alle PDUs eine eindeutige Reihenfolgenumerierung. PDUs, die den korrekten Empfang von Informationsteilen quittieren (ACK-PDUs), enthalten neben der Information zur Identifizierung der zuletzt empfangenen bzw. der als nächsten erwarteten PDU eine Angabe, wieviel PDUs noch gesendet werden dürfen (Kredite). Die Nummer des quittierten bzw. der nächsten erwarteten PDU legt die untere Begrenzung eines Fensters fest (Lower Window Edge, LWE), die zweite Angabe teilt dem Sender ein maximales Kontingent (also einen Kreditrahmen) für noch zu sendende PDUs zu und legt damit die Größe des aktuellen Fensters fest. Die obere Begrenzung des Fensters (Upper Window Edge, UWE) ergibt sich durch die Addition der LWE und der Anzahl der Kredite.

Konservative Protokolle schreiben vor, daß der Sender sich an das Fenster zu halten hat. Mit der Annahme, daß - bedingt durch Netzverzögerungen - die Fensterinformation bei ihrer Ankunft nicht mehr aktuell ist, ist es möglich, schon PDUs mit Sequenznummern oberhalb der UWE zu senden (optimistische Strategie). Dabei besteht allerdings die Gefahr, daß doch ein längerandauernder Ressourcenengpaß existiert,

was dann den Verlust von einigen der optimistisch versandten PDUs zur Folge haben kann [90].

Zwei Anmerkungen sind an dieser Stelle notwendig:

1. Fehlerbehandlung und Flußsteuerung basieren auf einer sequentiellen Numerierung der Dateneinheiten. Eine oft anzutreffende gemeinsame Implementierung der beiden Numerierungsschemata ist jedoch lediglich eine vereinfachende Implementierungsentscheidung und nicht logisch-funktionell begründet oder vom Standard her zwingend gefordert.
2. Neben der hier beschriebenen (horizontalen) Datenflußsteuerung zwischen Protokollinstanzen der gleichen Schicht bei den beteiligten Kommunikationspartnern nimmt die (vertikale) Flußsteuerung zwischen den benachbarten Schichten eines Protokollturmes, also innerhalb eines Kommunikationsknotens, einen nicht unerheblichen Stellenwert ein.

Beide Punkte werden im weiteren Verlauf der Arbeit noch genauer beschrieben.

3.1.6 Ressourcennutzung und Größenanpassung

Das wesentliche Merkmal von Kommunikationssystemen ist deren Heterogenität. Implementierungsbedingt ergeben sich z.B. Unterschiede bei maximalen PDU-Längen, der zulässigen Anzahl von Dienstzugangspunkten einer Schicht oder der Zahl gleichzeitig aktiver Verbindungen. Für die logische Funktionalität eines TpS ergeben sich daraus einige Anforderungen, die im folgenden getrennt nach daten- und verbindungsorientierten Mechanismen aufgeführt werden.

3.1.6.1 Datenspezifische Mechanismen

Entsprechend dem in Abschnitt 2.3 eingeführten Prinzip der Schichtung gibt es in Kommunikationssystemen zwei Typen von Dateneinheiten:

- Die zwischen gleichen Protokollschichten (horizontal) ausgetauschten Protokolldateneinheiten (PDUs) und
- die zwischen benachbarten Protokollschichten (vertikal) ausgetauschten Einheiten des in Anspruch genommenen Dienstes (SDUs).

Berücksichtigt man die unterschiedlichen Charakteristika heutiger Netze und Protokollschichtenimplementierungen in Bezug auf zulässige maximale Größen von Dateneinheiten, so lassen sich theoretisch vier Typen von Größenanpassungen konstruieren (jeweils nur in Senderichtung - also von einer Schicht (N) zur darunterliegenden Schicht (N-1) - aufgeführt, dualer Mechanismus in Empfangsrichtung):

1. Eine (N)-SDU wird in mehrere (N)-PDUs geteilt
2. Mehrere (N)-SDUs werden zu einer (N)-PDU zusammengefaßt
3. Eine (N)-PDU wird in mehrere (N-1)-SDUs umgesetzt und
4. Mehrere (N)-PDUs bilden eine (N-1)-SDU

Die Rahmenbedingungen des OSI-RMs erlauben 3 der 4 Kombinationen. Eine PDU ist als eigenständige Grundeinheit definiert, die mittels eines bestimmten Dienstes zur Partnerinstanz transferiert wird. Außerdem sind keine Verfahren zur Synchronisation unterschiedlicher empfangener SDUs vorgesehen. Der Standard sieht somit die Möglichkeit 3, also das Aufteilen einer PDU auf mehrere SDUs, nicht vor.

Der Fall 1 (Bild 3.2 b, a zeigt den Normalfall) beschreibt das bekannte Segmentieren von Daten der höheren Schicht in kleinere, den internen Schichterfordernissen angepaßte Einheiten. Die Teilstücke werden mit eindeutigen Sequenznummern und jeweils vollständigen Protokollsteuerköpfen der entsprechenden Schicht versehen. Die so erhaltenen SDUs werden nacheinander - über ein und denselben Dienstzugangspunkt - an die darunterliegende Schicht übergeben. Die Partnerinstanz auf der Empfangsseite hat die Aufgabe, diese Segmentierung vor dem Benutzer dieser Schicht zu verbergen. Dazu muß aus den unabhängig voneinander ankommenden und daher u.U. nicht mehr reihenfolgekorrekten Teilstücken wieder die

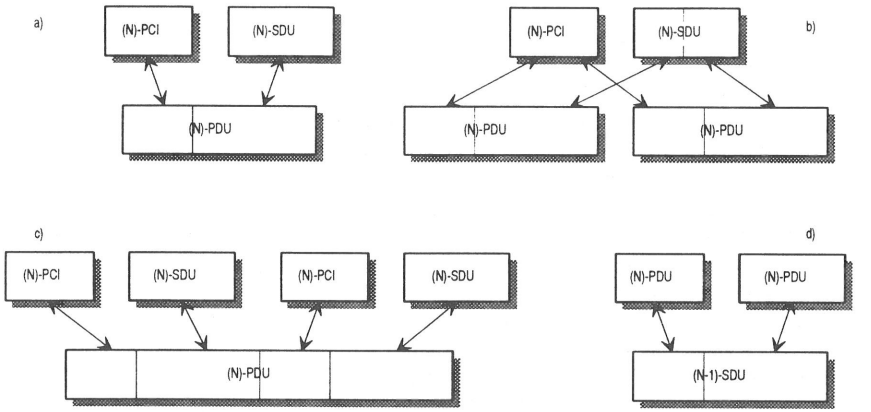


Bild 3.2: *Bidirektionale Abbildungsmechanismen von PDUs und SDUs*

ursprüngliche Benutzerdateneinheit gebildet werden. Die Entfernung der Protokollsteuerinformation, Umsortieren und Zwischenspeichern von Teilstücken sind dabei wesentliche Aspekte.

Bei der 2. Variante werden kleinere SDUs zu einer PDU zusammengefaßt (Bild 3.2 c), es wird also eine Blockbildung vorgenommen (Fachbegriffe: Blocken / Entblocken bzw. blocking / de-blocking). Klassisches Beispiel für diese Art der besseren Ausnützung von Ressourcen ist der Betrieb interaktiver Endgeräte wie Tastatur oder Bildschirm. Die zugehörigen anwendungsnahen Protokollschichten werden im allgemeinen für jeden eingegebenen Tastendruck eine Dienstanforderung und damit ein Dienstprimitiv an die darunterliegende Schicht übergeben. Besitzt diese sowohl eine gewisse Speicherfähigkeit als auch Wissen über die Eigenschaft der anfordernden Schicht, und verfügt sie zusätzlich über Mechanismen zur Sicherstellung der eindeutigen Kommunikationsbeziehung, können nun aufeinanderfolgende SDUs derselben logischen Verbindung in einer PDU übertragen werden.

Bei beiden angeführten Ausprägungen muß der erzielbare Nutzen durch Einsparungen von Steuerkopfinformation und damit von zugehöriger Verarbeitungs- und Übertragungskapazität dem Umfang des benötigten Aufwandes kritisch gegenübergestellt werden.

Für die 4. Variante (mehrere (N)-PDUs in eine (N-1)-SDU) gemäß (Bild 3.2 d) sind die Bezeichnungen Verkettung / Auftrennung bzw. concatenation / separation festgelegt worden. Es fällt zunächst nicht leicht, signifikante Unterschiede zur 2. Variante zu erkennen. Neben einer Verschiebung der Verantwortlichkeit unterscheiden sich die zwei Varianten im wesentlichen durch die Anzahl der (N)-Steuerköpfe: ist bei der Blockung grundsätzlich für jedes enthaltene Teil eigene (N)-Protokollsteuerinformation ((N)-PCI in Bild 3.2) notwendig, so reicht beim Zusammenfügen ein einziger (N)-Steuerkopf aus.

Da in der Praxis die Schichten eines Kommunikationssystems nicht unabhängig voneinander implementiert werden können und damit insbesondere auch die theoretisch gewünschte wahlfreie Kombination von Schichten verschiedener Hersteller nicht praktikabel ist, ist eine standardgemäße Unterscheidung in kommerziellen Produkten nicht anzutreffen.

3.1.6.2 Verbindungsspezifische Mechanismen

Im vorigen Abschnitt wurden die datenbezogenen Mechanismen zur Größenanpassung zwischen den Protokollschichten eingeführt. Betrachtet man statt Dateneinheiten nun logische Verbindungen, ergeben sich zwei grundsätzliche Möglichkeiten der Abbildung zwischen benachbarten Schichten (analog zum vorigen

Abschnitt wieder nur für die Senderichtung angegeben):

1. (N)-Verbindungen können zu einer (N-1)-Verbindung zusammengefaßt werden
2. eine (N)-Verbindung kann auf mehrere (N-1)-Verbindungen aufgeteilt werden.

Ausschließlich das unter 1. angeführte Zusammenfassen von Verbindungen wird von den ISO/OSI-Standards als Multiplexen bezeichnet (innerhalb der Kommunikationstechnik wird der Begriff Multiplexen allerdings mit vielerlei anderen Bedeutungen ebenfalls benutzt). Gründe für das Zusammenfassen von Verbindungen sind u.a. die Reduzierung von verbindungsbezogenen Kosten oder einfach die Beschränktheit vorhandener Ressourcen; keine Protokollimplementierung läßt eine unbeschränkte Zahl von aktiven Verbindungen zu. Bedingt durch die physikalisch vorhandenen Bausteine und/oder gewählten Datenstrukturen ergeben sich teilweise sehr niedrige Obergrenzen, so daß Multiplexen durchaus sinnvoll sein kann.

Das Aufspalten einer (N)-Verbindung auf mehrere (N-1)-Verbindungen, welches als splitting / recombination bekannt ist, kommt dann zur Anwendung, wenn dadurch eine erhöhte Gesamtleistungsfähigkeit der darunterliegenden Schicht(en) erreicht werden kann, und die zusätzlichen Kosten in Kauf genommen werden können. Eine Leistungssteigerung kann u.a. dann erwartet werden, wenn mehrere Wege zwischen Sender und Empfänger existieren oder eine verbindungsbezogene quantisierte Bedienstrategie vorliegt.

3.2 Physisch notwendige Aufgaben eines Transportsystems

Der vorige Abschnitt 3.1 machte deutlich, daß die ISO/OSI-Standards selbst innerhalb ihres eigentlichen Zuständigkeitsbereichs, d.h. bei den logischen Protokollfunktionalitäten, einen erheblichen Interpretationsspielraum beinhalten. Allein das Verstehen dieser Freiräume und die wissenschaftlichen Untersuchungen zu deren optimierter Verwendung stellen ein umfangreiches Forschungsgebiet dar. Allerdings wird dabei oft vergessen, daß jede logische Funktion in einem realen System auch in irgendeiner Form implementiert werden muß, und daß auch immer gewisse Einschränkungen durch die jeweilige Umgebung bestehen. Selbst wenn Realisierungsgesichtspunkte partiell mitberücksichtigt werden, wird deren Einfluß auf die Gesamtleistungsfähigkeit des Kommunikationssystems in der Regel deutlich unterbewertet.

Dieser Abschnitt stellt einige Implementierungs- und Betriebssystem-Teilaspekte vor, welche zur Erbringung der logischen Funktionen benötigt werden. Zusammengefaßt handelt es sich dabei um die Bereiche:

- Abbildung der Funktionen und Schichten auf Prozeduren oder Prozesse
- Realisierung von Schnittstellen zwischen Schichten bzw. Systemkomponenten
- Prozeßverwaltung
- Speicher- und Busanordnung
- Speicherverwaltung
- Zeitgeber (Timer)
- Aufbau und Verwaltung der protokollrelevanten Datenstrukturen
- Ressourcenverwaltung
- Unterstützung von Wartung und Diagnose

Jeder Bereich wird nachfolgend behandelt.

3.2.1 Abbildung logischer Funktionen

Typische Reaktionszeiten auf asynchron eintreffende Unterbrechungsanforderungen (interrupts) liegen bei heutigen Rechnerkernen in der Größenordnung von einigen Mikrosekunden (µs) (siehe z.B. [397 - 400, 402]). In der Regel beinhalten diese Zeiten lediglich den (maximalen) Zeitraum, der vom Eintreffen der Unterbrechungsanforderung bis zur Bereitschaft, den ersten Befehl der zugehörigen Unterbrechungs-

routine zu bearbeiten, benötigt wird. Der Aufwand für das meist unumgängliche Sichern von bestehenden Registerinhalten muß folglich noch hinzugerechnet werden. Je nach Registeranzahl und Speicherzyklusdauer ergeben sich schnell einige weitere µs. Bei einem modernen IBM RISC Arbeitsplatzrechner vom Typ RS/6000 werden beispielsweise 256 (!) Register bei einem Sicherungsbefehl bewegt [79]. Berücksichtigt man dann noch, daß beim Verlassen der Unterbrechungsbearbeitungsroutine der ursprüngliche Zustand der Register wieder hergestellt werden muß, so ergibt sich selbst für modernste Rechnertypen ein Verwaltungsaufwand für eine Unterbrechungsanforderung in der Größenordnung von 10 µs!

In derselben Größenordnung liegen auch Zeiten für einfachste systeminterne Funktionen, wie die 14 µs Messung eines GetProcessID-Aufrufes unter dem Betriebssystem AIX auf einer RS/6000 in [322] zeigt.

Je weiter man sich von der Hardware entfernt, desto länger werden die Zeiten, die für einen funktionellen Wechsel der bearbeiteten Aufgabe benötigt werden. Betrachtet man etwa einen Wechsel zwischen zwei unabhängigen Benutzerprozessen auf einem Mehrbenutzerbetriebssystem, sind Zeiten bis in den Millisekundenbereich hinein zu beobachten [26, 233, 406, 403]!

Es ist deshalb offensichtlich, daß die Art und Weise der Abbildung von Teilfunktionen oder auch ganzen Schichten auf Prozeduren und Prozesse allein durch die resultierenden Übergänge einen nicht unerheblichen Einfluß auf die sichtbare Systemleistungsfähigkeit haben. Diametral demgegenüber steht das Bestreben der Entwickler nach guter Strukturierung und größtmöglicher Modularität [104, 116, 247].

Es sei schon an dieser Stelle erwähnt, daß die ISO/OSI-Standards insbesondere nicht vorschreiben, daß die siebenschichtige OSI-Modellvorstellung eines Kommunikationssystems eins-zu-eins in eine Implementierung übernommen werden muß.

3.2.2 Schnittstellen

Vom rein logischen Standpunkt her gesehen sind in Kommunikationssystemen Schnittstellen lediglich zwischen benachbarten Schichten vorhanden. Die ISO/OSI-Standards haben für diesen Typ von Schnittstellen den abstrakten Begriff "Dienstzugangspunkt" (SAP, siehe 2.3.2) eingeführt. Allerdings werden keinerlei Hinweise zur realen Bedeutung bzw. dessen Implementierung gegeben. Es bleibt sogar unklar, ob diese SAPs überhaupt physikalisch in irgend einer Form - z.B. als echte, adressierbare Warteschlange, oder als Prozeduraufruf - vorhanden sein müssen. Andererseits sollte zumindest für die Überprüfbarkeit der Standardkonformität ein Zugang zu allen Diensten der einzelnen Schichten des OSI-RMs gewährleistet sein.

Unabhängig von einer eindeutigen Beantwortung der offenen Fragen und auch unabhängig von logischen Schichtungen bestehen in realen Systemen immer Schnittstellen zwischen einzelnen Komponenten:

Zunächst sind in jedem System ein oder mehrere Übergänge von Hard- auf Software vorhanden. Bei Kommunikationssystemen ist dies zumindest zwischen der physikalischen Netzankopplung und der Anwendungssoftware der Fall. Desweiteren sind sowohl Hard- als auch Softwarekomponenten in der Regel intern strukturiert, so daß weitere Schnittstellen entstehen. Bei der Hardware wird eine solche Unterteilung besonders deutlich, wenn die physikalische Netzankopplung (Netzadapterkarte) nicht fester Bestandteil des an der Kommunikation beteiligten Rechners ist. Typische Softwareunterbereiche sind etwa Treiber-, Betriebssystemkern- und Benutzerebene.

Zusammen mit den Modularitätsaspekten (3.2.1) ist einsichtig, daß die Art und Weise der Realisierung von Schnittstellen einen Einfluß auf das Leistungsvermögen eines Kommunikationssystems haben.

3.2.3 Prozeßverwaltung

Besitzt ein System nicht für jede Teilaufgabe einen Prozessor, so ergibt sich die Notwendigkeit, die unterschiedlichen Anforderungen und Aufgaben des Systems möglichst gerecht nacheinander zu bearbeiten.

Neben den kommunikationsorientierten Aufgaben, welche entsprechend 3.2.1 mehr oder weniger stark intern weiter unterteilt sein können, hat eine Systemumgebung immer auch andere Aufgaben zu erledigen. Selbst wenn die Kommunikation als ein nicht unterteilter Prozeß implementiert wäre und dieser Prozeß als einzige aktive Anwendung einer Rechenanlage erscheinen würde, laufen bei heutigen Multitasking-Betriebssystemen immer einige Systemaufgaben im Hintergrund ab [1, 5, 318].

Daraus ergibt sich die Notwendigkeit, aktive Prozesse geeignet zu deaktivieren, unter den Wartenden einen Prozeß auszuwählen und diesen zu aktivieren [4, 17, 36, 43]. Im Abschnitt 3.2.1 wurde schon auf den Zeitbedarf eines Prozeßwechsels eingegangen. Zusätzlich erfordert der Algorithmus zur Auswahl eines Prozesses (scheduling), insbesondere dann, wenn unterschiedliche Prioritäten zu beachten sind, einen nicht vernachlässigbaren funktionellen und damit zeitlichen Aufwand. Sind die unterschiedlichen Prozesse dazu nicht völlig unabhängig voneinander, so ergeben sich weitere Aufgaben, um die Synchronisation dieser Prozesse sicherzustellen und um Verklemmungssituationen zu vermeiden [71].

3.2.4 Speicher- und Busanordnung

Beeinflußt innerhalb der Prozessoren die gerade angedeutete Verwaltung der verschiedenen Prozesse die sichtbare Leistungsfähigkeit, so sind außerhalb des Rechnerkerns jene Komponenten, welche eine zentrale Funktionalität besitzen, als leistungshemmend anzunehmen. Solche Elemente mit zentralem Charakter sind nach [9, 10, 27] zum einen die Speicher, zum anderen auch die systeminternen Verbindungswege ("Busse"). Die rein logische Sichtweise konzentriert sich auf Semantik und Syntax von gespeicherter Information und vernachlässigt Überlegungen zum physikalisch notwendigen Zugriff. Analoges gilt für die Verbindungsstrukturen: logisch ist relevant, wer mit wem eine Kommunikationsbeziehung hat, Realisierungseinflüsse werden als unbedeutend und damit vernachlässigbar eingestuft.

Beim Umsetzen der - wegen ihrer logischen Einfachheit aus funktioneller Sicht beliebten - Modelle eines von allen Einheiten benutzbaren gemeinsamen Speichers bzw. eines universellen Busses zur Verbindung aller Komponenten treten schnell erhebliche, nicht erwartete Zeitverzögerungen auf. Diese sind größtenteils durch Zugriffskonflikte und Mechanismen zu deren eindeutiger Klärung bedingt. Zusätzlich muß beachtet werden, daß Speicherbausteine und Busse auch trotz modernster Technologie nach wie vor den leistungsschwächeren Systemteilen zugeordnet werden müssen. Einige Zahlen sollen das verdeutlichen:

- Der 16-bit breite in vielen Kleinrechnern eingesetzte AT-Bus hat ein theoretisches Durchsatzlimit von etwa 5 Millionen Oktetts pro Sekunde, d.h. 40 Mbit/s [393].
- Die *theoretische* Grenze für einen 32-bit breiten Microchannel, der in Kleinrechnern mit Intel 80386/486 Prozessoren überwiegend zum Einsatz kommt, liegt um etwa den Faktor 3.5 höher [416]. Die ausreichend erscheinende Größenordnung von deutlich über 100 Mbit/s bildet aber weiterhin für HS-Kommunikation einen Engpaß, wenn man berücksichtigt, daß ein Bus erstens nicht ausschließlich für einen Zweck benötigt wird, zweitens auf ein Datum mindestens zweimal - zum Einschreiben und Auslesen - zugegriffen wird und drittens die theoretischen Maximalwerte in der Praxis bei weitem nicht erreicht werden.
- Die leistungsfähigste Variante des VME-Busses hat einen rechnerischen Maximaldurchsatz von 35 Millionen Oktetts (280 Mbit/s) [209]. Es gilt auch hier das im vorherigen Abschnitt Gesagte.
- Ein Speichersystem, das aus DRAMs mit 100 ns Zykluszeit (was meist einer Zugriffszeit unter 50 ns entspricht!) aufgebaut ist, verfügt bei 16 Bit Breite über eine Speicherbandbreite von nur 160 Mbit/s. Allerdings muß jedes Datum zumindest eingeschrieben und einmal gelesen werden, was sofort zu einer (maximal) halbierten effektiven Speicherbandbreite führt.

Es empfiehlt sich daher, detaillierteres Wissen über Häufigkeit, Zweck, zeitliche Anforderungen und logische Beziehungen der durchzuführenden Zugriffe zu erwerben und durch geeignete Strukturen zu unterstützen. Informationen sollten nur dort vorgehalten werden, wo sie auch benötigt werden. Eine physikali-

sche Auftrennung der Speicher und auch Busse macht dann einen Sinn, wenn der Anteil lokaler Interaktion einen signifikanten Anteil am Gesamtaufkommen hat, und damit der zusätzliche, gelegentlich notwendige Aufwand zum Ausgleich der physikalischen Trennung in Kauf genommen werden kann.

Neben einer rein (flachen) Vervielfältigung von Speicherbereichen und Bussen bieten sich oft auch hierarchisch stukturierte Anordnungen an, die unter Umständen den Anforderungen und der zur Verfügung stehenden Technologie (kosten-)effizienter angepaßt werden können.

Das Konzept ausschließlich lokal benutzter Prozessorspeicher, gelegentlich auch mit einem eigenen Speicherbus kombiniert, ist ein weitverbreiteter Anwendungsfall dieser Überlegungen. Die bekannte Abstufung der Speichertechnologien von den kleinen, schnellen, aber teuren Prozessorregistern bis hin zu den sehr großen, aber dafür langsamen und billigeren Massenspeichermedien ist ein Beispiel für eine hierarchische Struktur bei Speichern [4, 9, 112].

3.2.5 Speicherverwaltung

Wo und wozu werden Speicher innerhalb eines Kommunikationssystems benötigt? Die grundsätzliche Aufgabe von Puffern ist das (Zwischen-)Speichern von (N)-SDUs, die über eine Dienstschnittstelle an eine (N)-Instanz übergeben wurden sowie die Erstellung bzw. Bearbeitung von abgehenden und ankommenden PDUs. Wie in Abschnitt 3.1.6 beschrieben, gibt es verschiedene Varianten, SDUs auf PDUs abzubilden und umgekehrt: Segmentierung innerhalb einer (N)-Instanz ist dann notwendig, wenn eine (N)-SDU zu groß für eine (N)-PDU ist, im umgekehrten Fall (blocking) passen mehrere (N)-SDUs in eine (N)-PDU, schließlich können auch mehrere (N)-PDUs zu einer (N-1)-SDU zusammengefaßt werden (concatenation). In allen drei Fällen sind jeweils vollständige (N)-PDUs in einer (N-1)-SDU enthalten. Jede Schicht braucht daher Puffer, die (N-1)-SDUs beinhalten können.
Aus diesen Vorüberlegungen ergeben sich drei Fragen an die Speicherverwaltung:

1. Wie können Pufferinhalte zwischen benachbarten Schichten übergeben werden?
2. Sollen zusammenhängende Pufferbereiche oder Strukturen, die aus mehreren Puffern bestehen, benutzt werden? Unmittelbar damit zusammenhängend stehen Fragen zur Größe der Elementarpuffer, die Vermeidung von Speicherverschnitt, die Bewertung des zusätzlich benötigten Verwaltungsaufwandes oder auch der Grad der Speichernutzung.
3. Wie kann die Gesamtheit aller Puffer verwaltet werden? Darin enthalten sind u.a. Zuteilungsstrategien von Speicher an die einzelnen Schichten sowie deren Rückführung nach Ende der aktiven Benutzung. Dieser Pufferkreislauf von Anfordern, Belegen, gegen Überschreiben Sichern und schlußendlich wieder Freigeben von Speichern hat eine signifikante Auswirkung auf die Systemleistungsfähigkeit und auf die effiziente Nutzung von Systemressourcen.

Weitere Aufgabenbereiche der Speicherverwaltung sind die physikalische Adressierung der Speicherbausteine, die Durchführung eventuell notwendiger Auffrischungszyklen (refresh cycles), die korrekte Beteiligung an Buszuteilung und -steuerung, die Auflösung von Konfliktsituationen - z.B. bei Mehrfachzugriffen - sowie die geeignete Interaktion mit anderen Systemkomponenten.

Es sei hier nochmals darauf hingewiesen, daß bei allen Überlegungen zu logischen Pufferstrukturen und zugehörigen Aufteilungen die physikalisch vorhandene Speicheranordnung entsprechend dem im vorigen Abschnitt Gesagten nicht vernachlässigt werden darf. Sind etwa alle Puffer über einen gemeinsamen Verbindungsweg (Bus) zu erreichen, so wird sich das Gesamtleistungsverhalten durch eine stark erhöhte Anzahl von tatsächlich stattfindenden Konflikten merklich verschlechtern.

Entsprechendes gilt für die Integration logisch verschiedener Pufferbereiche in einem physikalischen Speicherkomplex. Zum Umkopieren von einem Pufferbereich in den anderen muß zunächst von diesem Speicher gelesen werden (meist in eine Art Zwischenablage) und danach in einem zweiten Schritt in den

Zielbereich hinein geschrieben werden. Kann die Belastung der globalen Verbindungswege durch geeignete physikalische Strukturierung - z.B. durch die Einführung lokaler Busse - teilweise noch klein gehalten werden, so ist an der verringerten effektiven Speicherbandbreite nichts zu ändern.

3.2.6 Zeitgeber

Obwohl die Anwendung der Zeitglieder den logischen Protokollfunktionen zuzuordnen ist, fallen die Mechanismen zur Bereitstellung dieser Zeitgeber typischerweise in den Verantwortungsbereich von Betriebssystemen. Zeitgeber werden hier beschrieben, da deren protokollinterne Anwendung und Fragen zu deren optimaler Einstellung zunächst keine Beachtung finden, sondern die Implementierungsaspekte und deren Auswirkung auf das Gesamtsystem eingeführt werden.

Viele Protokolle benutzen das Mittel der Zeitüberwachung, um verschiedenartigste Fehlertypen zunächst einmal festzustellen und anschließend dann zugehörige Fehlerbehandlungsmaßnahmen einleiten zu können. Der Ablauf von Zeitgebern (time-out) kann beispielsweise auf eine vermißte Daten- oder Quittierungs-PDU aufmerksam machen oder verschiedene Inaktivitätsperioden einer Verbindung anzeigen. In [234] wird das Ablaufen eines Zeitgebers als Ereignis definiert, welches von einem zugehörigen Zeitgeber ausgelöst wurde und von einer spezifischen Protokollinstanz weiterverarbeitet werden kann. Der Zeitpunkt, wann ein Zeitgeber ein solches Ablaufereignis bewirken soll, kann durch absolute oder relative Beschreibungsformen vorgegeben werden. Es ist üblich, diese quantitative Angabe mit Wert des Zeitgebers (timer value) zu bezeichnen.

Jeder Transportverbindung sind gleichzeitig mehrere Zeitgeber zugeordnet. Die exakte Anzahl hängt u.a. von der gewählten Strategie zur wiederholten Aussendung von PDUs und weiteren Implementierungsentscheidungen bezüglich Wiederaufsetzen nach aufgetretenen Fehlern ab. Die vom US-amerikanischen nationalen Standardisierungsbüro NIST festgelegte TP4 Version [73] besitzt beispielsweise 12 verschiedene Zeitgeber, um die unterschiedlichsten Dinge zu überwachen. Die ISO-Spezifikation des Transportprotokolls [358] legt als Minimum fünf Zeitgeber mit um Größenordnungen divergierenden Auflösungen und unterschiedlichen Zuständigkeiten fest. Neben der Überwachung des korrekten Transfers einzelner PDUs durch den Wiederholungszeitgeber werden die Funktionalität des Fenstermechanismus, der Aktivitätszustand der aufgebauten Verbindung, der Verbindungsaufbau an sich und die Gültigkeit des Verbindungsbezeichners mit entsprechenden Zeitgebern kontrolliert. Die eingeführten englischen Bezeichnungen dieser Zeitgeber sind in der Folge ihrer obigen Nennung: retransmission timer, window timer, inactivity timer, give-up timer und reference timer.

Allerdings können in einer Implementierung auch deutlich mehr gleichzeitig aktive Zeitgeber eingesetzt werden. Im Extremfall - was der gängigen Auslegung des Standards entspricht - ist für jede noch nicht quittierte PDU ein eigener Wiederholungszeitgeber notwendig. Berücksichtigt man dann noch, daß größere Rechenanlagen bisweilen mehr als einhundert Verbindungen zu verwalten haben, nimmt die Anzahl der gleichzeitig benötigten Zeitgeber relativ große Werte an. Auf der anderen Seite resultieren die mit modernen Netzwerken heute erzielbaren hohen Bandbreiten und kurzen Verzögerungen in Zeitanforderungen bis in den Bereich weniger Millisekunden hinein. Die Variabilität der Zeitwerte, die benötigte Gesamtzahl und die geforderte hohe Granularität führen dazu, daß der Aufwand für das Setzen, Löschen und Verwalten der Zeitgeber - also Aspekte, die mit der logischen Funktion nichts zu tun haben - einen erheblichen Einfluß auf das Leistungsverhalten des Gesamtsystems hat und daher keinesfalls bei entsprechenden Leistungsprognosen vernachlässigt werden darf.

3.2.7 Ressourcenverwaltung

Es ist eine triviale, aber oft unberücksichtigte Tatsache, daß in realen Systemen alle Arten von Betriebsmitteln immer nur in endlicher Zahl vorhanden, also beschränkt sind. In der Regel wird - nicht zuletzt aus

wirtschaftlichen Gründen - angestrebt, die vorhandenen Ressourcen möglichst effizient zu nutzen. D.h. die Dimensionierung und die Verwaltung sollen so gewählt werden, daß im normalen durchschnittlichen Betriebsfall ein nicht allzu großer Ressourcenanteil ungenutzt bleibt, andererseits aber auch für die überwiegende Zahl von Ausnahmesituationen mit höherem Bedarf genügend Reserven vorhanden sind.

Einige Typen von Systemressourcen wurden in den vorherigen Abschnitten schon direkt oder indirekt angesprochen: Prozessorrechenzeit, Buszugriffsrecht und insbesondere die im System vorhandenen Speicher. Für die Verwaltung dieser Ressourcen ist neben der Anzahl auch die Leistungsfähigkeit - bei Speichern und Bussen z.B. die Bandbreite - ein bedeutender Parameter. Weitere Ressourcen in Kommunikationssystemen sind z.B. Bezeichner (identifier) für Verbindungen oder Zeitgebern. In beiden Fällen muß neben der durch die Implementierung beschränkten Anzahl auch die Eindeutigkeit berücksichtigt werden. Die Anzahl und die relativen Prioritäten von Unterbrechungsanforderungen sind ebenso nur in beschränktem Umfang verfügbar, wie spezielle Kommunikationskanäle zwischen Systemkomponenten.

Einige Parameter, die zur Verwaltung der aufgeführten beschränkten Betriebsmittel benötigt werden, sind unveränderliche Merkmale, die lediglich während der Dimensionierungsphase des Systementwurfes festgelegt werden müssen. Ein Großteil der Mechanismen zur Zuteilung der Ressourcen während des laufenden Betriebes und die Konfliktlösungsstrategien, die beim gleichzeitigen Zugriff auf gemeinsam genutzte Ressourcen eingreifen müssen, sind dagegen dynamische Prozesse. Unabhängig von deren funktionellem Verhalten beanspruchen diese Verwaltungsprozesse selbst natürlich auch gewisse Systemressourcen, welche wiederum den eigentlichen Produktionsabläufen fehlen und bei einer exakten Betrachtung des Systemleistungsverhaltens nicht vernachlässigt werden dürfen. Wichtiger ist allerdings die Berücksichtigung der aus den Verwaltungsmechanismen resultierenden (mittleren) Verzögerungen bei realistisch angenommenen Auftretehäufigkeiten.

Ein Hilfsmittel für die Ressourcenverwaltung sind angepaßte Datenstrukturen. Z.B. können alle noch unbelegten Einheiten einer bestimmten Betriebsmittelkategorie auf eine durch verzeigerte Listen realisierte Datenstruktur abgebildet werden, deren Abarbeitung nach dem Kellerspeicherprinzip (stack oder auch LIFO - Last In First Out) funktionieren könnte. Der nachfolgende Abschnitt beschäftigt sich genauer mit dem Einfluß von Datenstrukturen auf die Leistungsfähigkeit eines Kommunikationssystems.

3.2.8 Datenstrukturen

Innerhalb eines Kommunikationssystems müssen vielfältige Informationen gespeichert werden. Zu den bekannteren Vertretern benötigter Informationen gehören alle verbindungsspezifischen Daten (connection record), Umsetzungstabellen und Folgen von Puffern bzw. PDUs.

Um den selektiven Zugriff auf diese Daten zu ermöglichen, werden diese möglichst strukturiert im Speicher abgelegt. Häufig wird dabei lediglich auf eine logische Strukturierung geachtet, um etwa die Lesbarkeit von zugehörigen, in Hochsprachen geschriebenen, Programmquelltexten zu verbessern oder die Flexibilität zu erhöhen. Die Effizienz der physikalischen Zugriffs- und damit verbundenen Suchverfahren hat dagegen selten Einfluß auf die Wahl der Anordnung oder des prinzipiellen Typs der eingesetzten Datenstrukturen. Grundsätzliche Typen für die strukturierte Speicherung von Daten sind [24]:

- Felder (ein- oder mehrdimensional)
- Lineare Listen (geordnet oder ungeordnet)
- Baumartige Strukturen (überwiegend mit internem Ordnungsprinzip).

Felder haben den Vorteil des direkten indizierten Zugriffs, aber den Nachteil der festgelegten Größe. Für die anderen Grundtypen stellen die meisten Hochsprachen Mechanismen bereit, die eine - dem aktuellen Bedarf angepaßte - dynamische Variation des Umfangs der Datenstruktur unterstützen.

Bei linearen Listen mit n nichtsortierten Elementen ergibt sich ein mittlerer Zeitaufwand der Ord-

nung O(n/2) für das Lokalisieren eines zufällig gewählten Elementes. Dieser Aufwand läßt sich durch eine logische Unterteilung in mehrere kleinere geordnete Listen teilweise umgehen, allerdings steigt damit auch die Komplexität der zugehörigen Verwaltung [30].

Sind die Elemente einer dynamischen Liste nach einem Sortierkriterium angeordnet, so hängt die mittlere Suchzeit von der gewählten inneren Charakteristik der jeweiligen Suchwerte ab. Ein akademisch gewähltes Beispiel soll die Extreme verdeutlichen.

> Es soll eine Liste bestehend aus Elementen, die nur eine natürliche Zahl enthalten, betrachtet werden. Sortierkriterium sei der Wert der Zahl. Die Werte eines in aufsteigender Reihenfolge zählenden Bausteins sollen in diese Liste eingeschrieben werden. Ist die Liste selbst ebenfalls in aufsteigender Reihenfolge angeordnet, steht also das Element mit dem kleinsten Zahlenwert an vorderster Stelle der Liste, so muß immer die gesamte bestehende Liste durchsucht werden (Ordnung O(n)). Wurde zufällig oder bewußt die interne Listenanordnung nach absteigendem Zahlenwert festgelegt - das Listenelement mit der Zahl des größten Wertes steht an erster Stelle der Liste - endet die Durchsuchung der Liste jeweils schon beim ersten Listenelement. Selbst für beliebig große n ist der Suchvorgang hier von konstanter Ordnung O(1)!

Für Baumstrukturen mit internem Ordnungsprinzip können Suchverfahren eingesetzt werden, die lediglich einen Aufwand der Ordnung $O(\log_2(n))$ benötigen [30]. Baumstrukturen sind bei Softwareentwicklern nicht sehr populär, und es gibt auch gelegentlich prinzipielle Schwierigkeiten bei der Abbildung einer Datenmenge auf die Baumstruktur.

Die Anordnung und jeweilige Größe der Bestandteile eines Listenelementes - z.B. Elementbezeichner, verschiedene Parameter und Kontrollmarkierungen sowie Zeiger auf Vorgänger- und/oder Nachfolgerelement - können sich auf die Zahl der durchzuführenden elementaren Prozessorfunktionen auswirken. Eine problemangepaßte Berücksichtigung der Häufigkeitsverteilung des Zugriffs auf einzelne Elementbestandteile und eine daraus abgeleitete optimierte interne Anordnung kann etwa den Suchvorgang merklich verkürzen. Die Einhaltung von prozessortypischen Grundeinheiten, also z.B. Oktett, Wort oder Langwort (entsprechend 8, 16 bzw. 32 Bit), reduziert den Aufwand für Maskierung, Adreßkorrektur und Ausrichtung, der bei nicht angepaßten Elementen benötigt wird. Die erzielbare Effizienzsteigerung muß allerdings gegen den u.U. höheren Speicherbedarf abgewogen werden.

3.2.9 Unterstützung von Wartung und Diagnose

Mit steigendem Bedarf, bisher isolierte heterogene Netze zu verbinden, der daraus resultierenden erhöhten Teilnehmerzahl und der zwangsläufigen Einführung höherer Abstraktionsebenen nimmt die Komplexität moderner Kommunikationssysteme stetig zu. Um den Betrieb moderner Netze mit vertretbarem Aufwand zu ermöglichen, ist die gezielte Unterstützung eines intelligenten Netzwerkmanagements erforderlich.

Die ISO hat hierzu ein objektorientiertes Modell entwickelt (Bild 3.3), welches eine systemweit verteilte Datenbank von relevanten Daten (Management Information Base, MIB) vorsieht. Jede Protokollschicht soll zu dieser Datenbank mit schichtenspezifischen Daten beitragen, die in der ISO-Nomenkla-tur als Attribute der zu verwaltenden Objekte bezeichnet werden [381 - 391].

Eine Implementierung muß zunächst entsprechende Datenquellen vorsehen und zudem den Zugriff zu diesen ermöglichen. Beide Teilaspekte müssen schon zu Beginn eines Systementwurfs berücksichtigt werden [279], da ansonsten nur mit großem Aufwand für das Netzmanagement relevante Informationen bereitgestellt werden können. Bleibt bei der Implementierung die Leistungsfähigkeit nicht völlig unberücksichtigt, wirken sich Wartung und Diagnose für den *Normalbetrieb* nur geringfügig aus und können daher im Rahmen dieser Arbeit vernachlässigt werden.

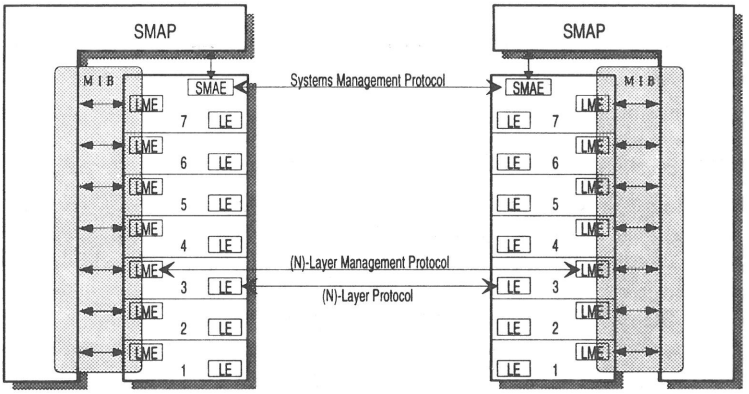


Bild 3.3: *Netzmanagementmodell der ISO*

3.3 Schwachpunkte heutiger Systeme

Die Gründe für die zu Beginn des Kapitels 3 gezeigte geringe erzielbare Leistungsfähigkeit der höheren Schichten des OSI-RMs sind auch im Zeitalter der Informationsverarbeitung noch nicht im Detail bekannt. Dies hat mehrere Gründe:

- Die Leistungsfähigkeit der Kommunikationssysteme war und ist für die überwiegende Zahl von Anwendungen ausreichend.
- Oftmals werden höhere Schichten überhaupt nicht benötigt und wenn, dann steht die funktionell korrekte Zusammenarbeit mit anderen Installationen im Vordergrund.
- Die Philosophie des OSI-RMs verleitet dazu, auch bei der Bewertung abgeschlossene Teilaspekte zu betrachten. Insbesondere gibt es viele Veröffentlichungen zu der Betrachtung einzelner Protokollmechanismen (u.a. [69, 80, 102, 142, 145, 158, 192, 195, 220, 276]). Die veröffentlichten Ergebnisse geben aber nur relative Ergebnisse im Vergleich zu anderen Versionen des untersuchten Teilaspekts an, Aussagen über den Einfluß des Vorschlages auf das Verhalten des gesamten Systems werden nicht gemacht bzw. können nicht gemacht werden.
- Durch die geforderte Implementierungsunabhängigkeit der ISO/OSI-Standards und die damit verbundene logische Abstraktion bleiben viele Systemaspekte völlig unerwähnt oder sehr unklar. Dies führt zwangsläufig zu falschen Annahmen bei der oben genannten isolierten Bewertung von Teilkomponenten und damit wiederum zu stark variierenden Resultaten.
- Die hohe Komplexität eines Kommunikationssystems macht es nahezu unmöglich, dieses komplett zu modellieren und/oder zu simulieren, ohne die Erfahrungen aus dem Entwurf einer vollständigen Implementierung zu haben.
- Andererseits sind nur wenige Implementierungen eines Gesamtsystems gemacht worden, und es gibt noch weniger entsprechende öffentlich zugängliche Beschreibungen oder Erfahrungsberichte.
- Bei den vorhandenen realen Implementierungen besteht entweder kein Bedarf an detaillierten Messungen, oder es fehlen die technischen Möglichkeiten, um genauere, also nicht nur pauschale Aussagen machen zu können.

Die Messungen (Literaturverweise siehe Beginn Kapitel 3) zeigen die Leistungsfähigkeit von Gesamtsystemen auf und liefern nur vereinzelt Hinweise auf mögliche Systemschwachpunkte. Die Messungen werden dazu überwiegend unter stark eingeschränkten und damit unrealistischen Systemkonfigurationen

durchgeführt. Anzuführen sind hier speziell die exklusive Nutzung sowohl des Netzes als auch der Rechenkapazitäten der beteiligten Kommunikationsknoten sowie geeignet gewählte Parameter des Verkehrs (PDU-Ankunftsabstände und -längen, Büschelförmigkeit, Last etc.) und anderer wichtiger Systemparameter wie Fehlerwahrscheinlichkeit, Zahl der Verbindungen oder Belegungszustand der Speicher.

Dieses Kapitel beschreibt exemplarisch einige Teilmechanismen, welche jeweils für sich genommen einen negativen Einfluß auf die Leistungsfähigkeit heutiger Transportsysteme haben. Allerdings können keine exakten Aussagen, sondern bestenfalls grobe Schätzungen über deren relative Gewichtung und damit über deren jeweilige Auswirkungen auf das *Gesamt*verhalten angegeben werden. Neben den durch Parameterwahl, Optionen und Protokollklassen möglichen internen Varianten von Protokollen wird auch die Zusammensetzung möglicher Protokolltürme betrachtet. Abweichend von der durch das OSI-RM für offene Kommunikationssysteme vorgegebenen Beschränkung auf die logisch abstrakte Sichtweise der reinen Protokollfunktionen, werden auch Implementierungsaspekte diskutiert.

3.3.1 Protokollparameter und -strategien

Es ist ein Merkmal von universellen Standards, daß bestimmte Parameter und Vorgehensweisen nicht eindeutig festgelegt sind, um die Implementierungen flexibel den jeweiligen Gegebenheiten anpassen zu können, ohne die Kompatibilität zum Standard einzubüßen. Prominenteste Vertreter sind die aktuell einzustellenden Werte der verschiedenen Zeitgeber [139, 193, 204, 326]. Weitere bekannte Beispiele sind Fenstergrößen und Zahl der Sendeberechtigungen [102, 158, 194, 276, 309].

Bei den nicht festgelegten Vorgehensweisen ist zuerst die Strategie zum Aussenden von Quittierungen zu nennen. Der Standard für das verbindungsorientierte ISO/OSI-Transportprotokoll [358] schreibt z.B. nicht vor, daß der Erhalt jeder PDU dem Sender explizit durch Aussenden einer Quittierungs-PDU bestätigt werden muß. Durch die - urprünglich für eine erhöhte Fehlertoleranz eingeführte - Festlegung, daß mit der Quittung für eine PDU mit der Folgenummer n immer gleichzeitig auch alle PDUs mit kleinerer Folgenummer implizit bestätigt werden, ergibt sich die Möglichkeit, bewußt nur den Erhalt jeder x-ten PDU zu bestätigen (kumulierte Quittung) und damit Netzressourcen einzusparen [223].

In analoger Weise schreibt der Standard nicht vor, daß PDUs mit nicht erwarteter Folgenummer sowie alle nachfolgenden korrekt empfangenen PDUs pauschal verworfen werden müssen. Es ist zulässig, alle empfangenen PDUs zwischenzuspeichern und intern wieder in die korrekte Reihenfolge zu bringen (bei PDU-Vertauschungen) oder nur exakt die fehlenden PDUs neu anzufordern (bei PDU-Verlusten).

Bei mehreren Möglichkeiten gibt es immer mehr oder weniger effiziente Varianten. In bestehenden Implementierungen werden die Möglichkeiten zur Parametermanipulation und damit zur Anpassung des Kommunikationssystems an die spezielle Konfiguration nur in Ausnahmefällen benutzt. Stattdessen werden die voreingestellten Defaultwerte benutzt. Diese müssen, entsprechend ihrem universellen Charakter, sehr pessimistisch dimensioniert sein. Einige markante Beispiele sollen diesen Sachverhalt verdeutlichen.

3.3.1.1 Fenstergröße

Wie schon in Kapitel 3.1.5 beschrieben wurde, müssen, bedingt durch endliche Systemressourcen und unterschiedliche aktuell verfügbare Leistungsfähigkeiten, bei den Kommunikationspartnern Mechanismen zur Steuerung des Informationsflusses vorgesehen werden. Neben der Überwachung der transferierten Datenmengen pro Zeiteinheit (Raten) werden hierzu bevorzugt Mechanismen eingesetzt, die auf der empfängerseitigen Zuteilung eindeutig gekennzeichneter zu sendender Datenmengen basieren. Bei diesen Fenstermechanismen können durch ungeschickte Wahl der Parameter erhebliche Reduzierungen der Effizienz bewirkt werden. Unter "Effizienz" ist hier das Verhältnis von tatsächlich erzielbarer und physikalisch vorhandener Übertragungsrate zu verstehen.

Das Prinzip der Effizienzreduktion soll zunächst bei einem Protokoll mit der Fenstergröße 1 und idealisierter vollständigen Fehlerfreiheit verdeutlicht werden. Nachdem der Sender eine PDU abgeschickt hat, muß er solange mit dem Versenden der nächsten PDU warten, bis er die Empfangsbestätigung und die darin enthaltene neue Sendeberechtigung erhält.
Um diesen Vorgang nicht nur qualitativ zu betrachten, werden folgende Größen eingeführt:

b : Übertragungsrate des Mediums in Bit pro Sekunde
e : Geografische Distanz der Kommunikationspartner in Metern
mit bekannter Ausbreitungsgeschwindigkeit folgt daraus:
d : Übertragungsverzögerung
n : Zahl der Nutzdatenbits einer PDU
h : Zahl der Steuerkopfbits einer PDU
l : Gesamtlänge einer Daten-PDU ($l = h + n$)
a : Gesamtlänge einer Quittierungs-PDU
v : Verarbeitungszeit für den Empfang einer PDU
W : Fenstergröße in PDUs (wird später gebraucht)

Multipliziert man die Verzögerung d mit der Bandbreite b, ergibt sich die - in der Literatur vielfach verwendete - Größe des Bandbreiten-Verzögerungs-(BV-)Produktes (bandwidth-delay product, BD-product). Diese Größe gibt die Entfernung in Bits und damit die Speicherfähigkeit des Mediums an.

Nach einer Zeit l/b ist das letzte Bit der zu sendenden PDU auf dem Medium. Die Übertragung und die anschließende Bearbeitung, während derer auch eine zugehörige Quittierungs-PDU erstellt wird, werden durch die Addition von d und v mitberücksichtigt: $l/b + d + v$. Vom Aussenden der Empfangsbestätigung bis zum Erkennen, daß der Sender nun die nächste PDU senden darf, vergeht die Zeit $a/b + d + v$. Innerhalb der Gesamtzeit $l/b + a/b + 2\bullet(d + v)$ hätten - bei der vorhandenen Bandbreite b - rechnerisch $l + a + 2\bullet b\bullet(d + v)$ Bits übertragen werden können. Tatsächlich wurden lediglich n Nutzdatenbits übertragen. Die effektive Ausnutzung E des Mediums läßt sich daher zu

$$E=\frac{n}{l+a+2b(d+v)}=\frac{n}{n+h+a+2b(d+v)} \tag{3.1}$$

angeben. Die Effizienz E kann also nur dann groß sein, d.h. möglichst nahe bei 1 liegen, wenn sowohl Steuerkopf- und Quittierungs-PDU-Längen vernachlässigbar sind, als auch eine Konfiguration mit kleinem BV-Produkt betrachtet wird. Werden Fehler mitberücksichtigt, verschlechtert sich die Effizienz durch die zusätzlichen Übertragungen weiter. In [42] wird die obige Ableitung um den Einfluß von Fehlern erweitert.

Wie kann diese Charakteristik positiv verändert werden? Die Antwort der Standards darauf war die Einführung größerer Fenstergrößen. Idealerweise soll das Fenster so groß gewählt werden, daß dem Sender immer schon vor Aufbrauchen seines Kontingentes neue Sendeberechtigungen vom Empfänger zugeteilt worden sind. Nimmt man weiter an, daß die Länge von Quittierungs-PDUs vernachlässigbar ist, z.B. weil diese den Nutzdaten-PDUs der Gegenrichtung mitgegeben wurden (Englisch: piggybacking) und daß sich die Zeiten für die Verarbeitungsphasen durch entsprechende Fließbandverarbeitung (Englisch: pipelining) ebenfalls nicht mehr auswirken, so kann die ideale Effizienz nahezu erreicht werden.

$$E=\frac{n}{n+h} \tag{3.2}$$

Wie groß muß die Fenstergröße W mindestens sein, damit der Sender ununterbrochen senden kann? Das Abschicken einer PDU benötigt die Zeit l/b. Hat der Sender zu Beginn ein Sendefenster der Größe W zugeteilt bekommen, so kann er also für eine Zeit $w\bullet l/b$ senden, bevor er sein ursprüngliches Kontingent an Sendeberechtigungen verbraucht hat. Die erste Quittung kann frühestens nach der Zeit $l/b + 2(d+v)$ eintreffen. Es ergibt sich die Ungleichung $l/b + 2(d+v) < W\bullet l/b$ bzw. nach W aufgelöst:

$$W > 1 + \frac{b}{l} 2 (d + v) \tag{3.3}$$

Schon bei heute anzutreffenden Werten, z.B. in einem FDDI-Netz (b = 100 Mbit/s, d = 0.5 ms, v < 0.1 ms) mit mittleren PDU-Längen von etwa l = 1000 Bit, können sich dreistellige Zahlenwerte für die Fenstergröße ergeben. Wird in einem solchen Fall mit der häufig empfohlenen bzw. voreingestellten Fenstergröße 7 gearbeitet, nähert sich das Verhalten stark dem anfangs aufgeführten Beispiel mit Fenstergröße 1. Nach dem schnellen Senden einer kleinen Folge von PDUs (burst) muß lange auf den Erhalt der nächsten Kredite gewartet werden. Der Unterschied zum Beispiel mit W = 1 ist im wesentlichen eine größere effektive PDU-Länge.

Erhöht man die Fenstergröße, muß der dadurch linear mitsteigende Bedarf an Pufferplätzen beim Sender berücksichtigt werden, da im schlechtesten Fall alle PDUs eines Fensters noch nicht quittiert worden sein können. Ist umgekehrt eine gewisse Zahl an Sendepuffern vorgegeben, ist damit auch die maximal mögliche Fenstergröße festgelegt.

Ein weiterer Faktor muß bei der Wahl der Fenstergröße miteinbezogen werden: Fehlerbehebungsverfahren, bei denen alle PDUs ab einer fehlerbehafteten erneut gesendet werden müssen (go-back-n), machen in vielen Fehlersituationen das erneute Aussenden eines gesamten Fensterinhaltes notwendig. In Netzen mit überdurchschnittlicher Fehlerhäufigkeit stellt die resultierende zusätzliche Netzlast als Folge der wiederholten Übertragungen eine signifikante Größe dar.

Abschließend sei auf die in vielen Protokollen vorhandene "natürliche" Beschränkung der maximalen Fenstergröße durch die teilweise sehr geringen für die Kreditvergabe vorgesehenen oder tatsächlich nutzbaren Bitpositionen in den Steuerköpfen hingewiesen. Die festgelegten Formate der ISO/OSI-Transportprotokolle sehen beispielsweise nur 4 Bit (im Normalformat) vor; die Fenstergröße ist somit auf maximal $2^4 - 1 = 15$ beschränkt. Etwas besser sieht es in dem weit verbreiteten Transportprotokoll TCP (Transmission Control Protocol, siehe z.B. [5, 260]) der Internetprotokollsäule aus: es sind hier 16 Bits vorgesehen. Die quantisierte Grundeinheit sind allerdings nicht ganze PDUs sondern einzelne Oktetts, so daß man umgerechnet auf eine Größenordnung von deutlich unter hundert PDUs kommt.

3.3.1.2 Werte von Zeitgebern

Der Verlust von PDUs kann in vielen Protokollen nur durch zugehörige Zeitüberwachungen festgestellt werden. Gleiches gilt für die Sicherstellung von zuverlässigen Verbindungsauf- und -abbauten sowie die Garantie der Eindeutigkeit von Verbindungsbezeichnern. Die Werte dieser Zeitgeber wirken sich teilweise direkt auf die sichtbare Systemleistungsfähigkeit aus. Die voreingestellten Defaultwerte müssen so konservativ gewählt werden, daß der überwiegende Anteil möglicher Umgebungen damit problemlos arbeiten kann. Eine Anpassung an spezifische Umgebungsbedingungen wird häufig nicht vorgenommen. Häufig sind die Einstellmöglichkeiten nicht bekannt oder aber es fehlen die technischen Möglichkeiten, die relevanten Eigenschaften der Umgebung mit ausreichender Genauigkeit bzw. überhaupt zu ermitteln. Einige Beispiele sollen die Problematik verdeutlichen.

Will man z.B. die Verzögerungen minimieren, die zur Resynchronisation nach dem Verlust einer Daten-PDU zwangsläufig benötigt werden, sollte der Wert des Wiederholungszeitgebers nicht viel größer als die Zeit sein, die für das eigentliche Senden der Daten-PDU, der Bearbeitung einer Daten-PDU auf Empfängerseite, der Erstellung einer zugehörigen Quittierungs-PDU sowie der Zeit zu deren Rücksendung, den senderseitigen Empfang derselben und der Erkennung benötigt wird. Diese kombinierte Daten-/ Quittungs-PDU-Umlaufverzögerung (roundtrip delay) enthält sowohl Wartezeiten der Sender- und Empfängerseite als auch netzbedingte Übertragungsverzögerungen beider Richtungen. Formaler ausgedrückt ergibt sich mit der Annahme einer mittleren reinen Übertragungszeit t_t, der Bearbeitungszeit t_c für die Auswertung einer empfangenen PDU und t_a für die Erstellung einer Quittierungs-PDU ein optimaler

Wert t_O für den Wiederholungszeitgeber von:

$$t_O = t_t + t_c + t_a + t_t + t_c = 2(t_t + t_c) + t_a \quad (3.4)$$

Dieser Ausdruck trat in ähnlicher Form auch schon bei den im vorigen Abschnitt gemachten Überlegungen zur Effizienz von Fenstermechanismen implizit auf.

Bei der Überwachung eines Übertragungsabschnittes auf der Schicht 2 bzw. 2b bei LANs sind die Werte durch die festgelegte Distanz und den relativ geringen Bearbeitungsaufwand nur wenig veränderlich. Im Gegensatz dazu sind all diese Werte auf der Schicht 4 und damit auch die Wahl optimaler Werte für die Zeitgeber stark von der augenblicklichen Situation des Netzwerks, also z.B. der Lastsituation oder der Wegewahl, abhängig. Die sichere Methode ist, den Wert des Zeitgebers so groß zu wählen, daß die Wahrscheinlichkeit für fälschlicherweise wiederholt ausgesandter PDUs, selbst in Hochlastsituationen, sehr klein gehalten werden kann. Allerdings verschlechtert sich dadurch, daß eine ausstehende Quittung erst wesentlich später festgestellt werden kann, die Effizienz entsprechend.

Die ideale Lösung eines sich dynamisch an die augenblicklichen Verhältnisse anpassenden Zeitgebers setzt andererseits sehr präzise und intelligente Meß- und Regelungswerkzeuge voraus. Auch Schätzungen der aktuellen Umlaufverzögerung aufgrund der genauen Beobachtung vorausgegangener PDUs bzw. deren Quittungen - wie u.a. in TCP [204] und auch einer Empfehlung des amerikanischen Standardisierungsinstituts NIST [73] vorgeschlagen - können zu falschen Werten führen. So ist im Falle des Erhalts einer Quittierung für eine wiederholt ausgesandte PDU nicht feststellbar, ob der Empfang des Originals oder der Wiederholungssendung die Bestätigungsmeldung ausgelöst hat. Darüberhinaus werden die Werte auch durch den eventuellen Einsatz von Mechanismen zur kumulierten Quittierung verfälscht.

Ein weiteres Beispiel für eine ineffiziente Parametereinstellung sind nicht aufeinander abgestimmte Werte für den Fensterzeitgeber auf Empfängerseite und dem Inaktivitätszeitgeber beim Sender. Ersterer gibt den Zeitraum an, nachdem spätestens wieder eine Quittierung ausgesandt werden muß. Ist der zugehörige Wert größer als der Wert der senderseitig den Abbau der Verbindung veranlaßt, so wird bestenfalls ein häufigeres Verbindungsauf- und -abbauen resultieren, was sich natürlich negativ auf das Leistungsverhalten auswirkt. Eine solche Wahl der Zeitgeber kann unter ungünstigen Randbedingungen allerdings auch zu logischen Verklemmungszuständen führen.

3.3.1.3 Quittierungs- und Wiederholstrategien

Wie schon angesprochen, wird bei vielen Protokollen ein Fenstermechanismus zur Datenflußsteuerung eingesetzt [146]. Ein Sender darf bis zu W Protokolldateneinheiten abschicken, bevor er eine Quittierung für die erste abgesandte PDU erhalten hat. Die Größe W wird dabei als Fenstergröße bezeichnet. Um mögliche Verklemmungssituationen als Folge gestörter oder verlorengegangener Quittierungsmeldungen zu vermeiden, wird nicht der Erhalt einer spezifischen PDU explizit bestätigt, sondern es wird dem Sender die Folgenummer der als nächstes erwarteten PDU mitgeteilt. Für den Empfänger ergibt sich daraus, daß er beim Erhalt einer PDU mit korrekter Folgenummer umgehend eine Bestätigung des Empfangs dieser PDU an den Sender schicken kann, er muß dies aber nicht tun, sondern kann noch bis zu maximal W PDUs abwarten. Der Einsparung an Aufwand und Übertragungsbandbreite beim Ansammeln von Quittierungen steht allerdings ein zusätzlich notwendiger - oftmals mit Zeitgebern realisierter - Mechanismus zur Sicherstellung der Funktionalität bei Sendepausen entgegen. Ferner ergibt sich ein erhöhter Aufwand für den Wiederholungspuffer beim Sender.

Im Fall des Erhalts von PDUs mit unerwarteten (aber innerhalb des zulässigen Empfangsfensters liegenden) Folgenummern können diese verworfen werden, die reihenfolgegestörten PDUs können aber auch zwischengespeichert werden [223]. In beiden Fällen kann sich das Aussenden einer Quittierungsmeldung mit den bis dahin korrekt empfangenen PDUs anschließen (muß es aber wiederum nicht).

Auf Senderseite bestehen auch einige Wahlmöglichkeiten: entweder wird der Erhalt von Empfangsbestätigungen und die darin enthaltene Angabe zur als nächstes erwarteten PDU ausgenutzt, um Entscheidungen für Wiederholungssendungen zu beeinflussen, oder die Wiederholungsaktivität wird ausschließlich auf der Basis von Zeitgebern durchgeführt. In beiden Fällen kann jeweils nur genau das vermißte Teil erneut losgeschickt werden oder aber zusätzlich alle PDUs mit höheren Folgenummern.

Die vielfältig kombinierbaren Sender- und Empfängermechanismen resultieren in einem weiten Spektrum erreichbarer Effizienz. Es ist einsichtig, daß z.B. die Kombination eines Verwerfens von inkorrekten PDUs auf Empfängerseite mit einer Wiederholung nur der PDU, für die der Zeitgeber abgelaufen ist, zwar einfach zu implementieren ist, dafür aber eine schlechte Effizienz ergibt. Wenn andererseits von einer geringen PDU-Verlust- bzw. Reihenfolgestörungsrate ausgegangen werden kann, ist die Anwendung des Verfahrens zur Aussendung aller PDUs ab der fehlerhaften (go-back-n) ebenso ineffizient. Besonders in Netzen mit großem BV-Produkt ergeben sich ungünstige Leistungswerte [67].

Durch das Fehlen einer Möglichkeit zur expliziten Anforderung einer bestimmten PDU (selective repeat) kann ein Empfänger, der inkorrekte PDUs zwischengespeichert hat, nur durch die Aussendung einer Quittierungsmeldung (mit der tatsächlich bei ihm als nächstes erwarteten Folgenummer als Inhalt) sofort nach dem Erhalt der wiederholt ausgesandten PDU den Sender dazu bringen, die schon begonnene Aussendung aller folgender PDUs wieder abzubrechen.

In diesem Abschnitt wurde mit einigen Beispielen aus dem Bereich der Quittierungs- und Wiederholstrategien aufgezeigt, daß sich die in den Standards enthaltenen Freiheiten und die daraus resultierenden Kombinationen von Implementierungsvarianten negativ auswirken können. Eine Festlegung kann in speziellen Fällen zwar eventuell zu einer Effizienzeinbuße führen, trotzdem erscheint eine gezielte Beschränkung der Benutzungs- und Implementierungsfreiheiten angebracht. Nicht nur die mittlere Leistungsfähigkeit könnte davon profitieren, sondern auch der prinzipielle Umgang mit den Standards, deren Implementierung und deren alltäglicher Betrieb würden erleichtert werden.

3.3.1.4 Verbindungsverwaltung

Ein kritischer Aspekt des OSI-Referenzmodells ist die Notwendigkeit, auf jeder verbindungsorientiert arbeitenden Schicht jeweils eine eigene Verbindung zu verwalten. Typischerweise werden pro Verbindungsaufbau eine Verbindungsanforderung (Connect Request, CR) und eine zugehörige Bestätigung (Connect Confirm, CC) zwischen den beteiligten Instanzen ausgetauscht. Zusätzlich muß bei unzuverlässigen Netzen der korrekte Empfang der CC-Meldung zur Verhinderung ansonsten möglicher Mehrdeutigkeiten explizit bestätigt werden, so daß für den Verbindungsaufbau einer Schicht sogar bis zu 3 Nachrichten durch das Netzwerk transportiert werden müssen (three-way handshake), bevor die eigentlichen Nutzdaten transportiert werden können. Nimmt man eine Konstellation an, bei der die Schichten 2b, 3 und 4 verbindungsorientiert arbeiten, müssen mindestens $3 \cdot 2 = 6$ Meldungen über das Netz ausgetauscht werden, bevor das erste Nutzdatum übertragen werden kann (siehe Bild 3.4). Diese Zahl läßt sich reduzieren, indem der Verbindungsaufbau mehrerer Schichten zusammengefaßt wird, was allerdings ein gewisses Maß an Wissen über das innere Verhalten von Nachbarschichten bzw. einen entsprechenden Informationsaustausch zwischen benachbarten Schichten, beispielsweise indirekt über das Netzmanagement, unbedingt erforderlich macht.

War das Verhältnis zwischen dem zum Verbindungsauf- und -abbau (einer Schicht) benötigten Zeitaufwand und der Zeit, die für die reine Nutzdatenübertragung benötigt wurde, bei relativ langsamen Netzen noch akzeptabel, können sich bei Netzen mit einem hohen BV-Produkt die Verhältnisse dramatisch ändern. Quantitativ läßt sich dieser Sachverhalt unter Benutzung der in Abschnitt 3.3.1.1 eingeführten Symbolik sowie einiger zusätzlicher Festlegungen leicht aufzeigen. Drei weitere Größen kommen hinzu: die Nettozeiten für die Bearbeitung eines Verbindungsaufbaus auf Sender- und Empfängerseite t_{CS}

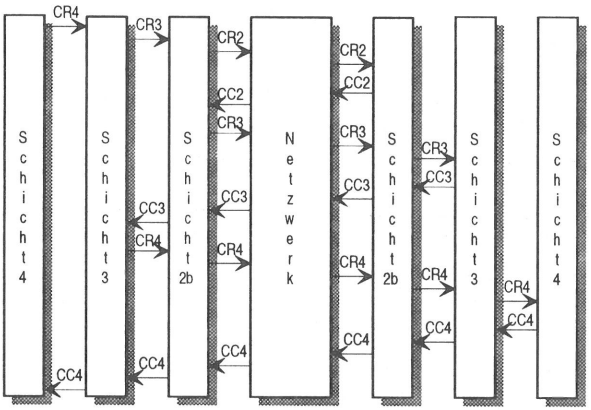


Bild 3.4: *Verbindungsaufbau über mehrere Schichten*

und t_{CR} (genau genommen besteht t_{CS} aus zwei Teilen, es wird Zeit zum Erstellen von CR und zum Auswerten von CC benötigt) sowie die Anzahl i der PDUs, in die eine SDU mit n Nutzbits aufgeteilt (segmentiert) werden muß. Es sei angenommen, daß sowohl die zum Verbindungsaufbau benötigten CR und CC-PDUs als auch die Quittierungs-PDUs jeweils nur aus einem Steuerkopf der Länge h Bits bestehen und daß der Erhalt jeder PDU sofort einzeln bestätigt wird. Die Übertragung laufe fehlerfrei ab. Für den Verbindungsaufbau ergibt sich die Zeit t_{set} zu:

$$t_{set} = t_{CS} + \frac{h}{b} + d + v + t_{CR} + \frac{h}{b} + d + v = t_{CS} + t_{CR} + 2(\frac{h}{b} + d + v) \tag{3.5}$$

Die Transferzeit t_{dat} für die Nutzdaten und deren Quittungen ist:

$$t_{dat} = i\frac{h + \frac{n}{i}}{b} + d + v + i(\frac{h}{b} + d + v) = \frac{n}{b} + 2i(\frac{h}{b} + d + v) \tag{3.6}$$

Die Effizienz läßt sich durch das Verhältnis t_{set}/t_{dat} angeben. Für Abschätzungen reicht das Verhältnis der gesamten Verbindungsaufbauzeit $(t_{CS} + t_{CR})$ zum Zeitaufwand zur Übertragung der Nutzdaten (n/b) aus.

Geht man von realistischen Gesamtzeiten von 100 ms für einen Verbindungsaufbau aus, so ergeben sich bei einer Bandbreite b von 64 Kbit/s schon bei weniger als 1000 Oktetts ($n < 8000$ Bits) ähnliche Grössenordnungen für t_{set} und t_{dat}. Mit zunehmender SDU-Länge wird der Anteil an der Verbindungsverwaltung kontinuierlich kleiner. Nimmt man ein modernes Hochgeschwindigkeitsnetzwerk mit einer Bandbreite von 100 Mbit/s an, würde erst ab SDU-Längen über einer Million Oktetts (ca. 10 Mbit) die Zeit für die Nutzdatenübertragung in etwa gleich der zur Verbindungsverwaltung benötigten Zeit werden!

In modernen Netzwerken ist es ineffizient, für jede Datenübertragung den vom Protokoll her vorgegebenen Ablauf Verbindungsaufbau - Nutzdatenübertragung - Verbindungsabbau durchzuführen. Extrem schlecht wird diese Vorgehensweise bei transaktionsorientierten Anwendungen, bei denen typischerweise jeweils nur wenige Daten während einer logischen Kommunikationsbeziehung ausgetauscht werden.

Andererseits erfordert die Mehrfachnutzung von Verbindungen einen erheblichen Koordinierungs- und Verwaltungsaufwand, welcher in bestehenden Systemen nur selten entsprechend erbracht wird. Aus OSI-Sicht fällt die zugehörige Funktionalität überwiegend in die Verantwortung der Kommunikationssteuerungsschicht, welche ohnedies zu den bisher am wenigsten beachteten Schichten gezählt werden muß.

Darüber hinaus hat die Gestaltung der Tarifierung einen Einfluß: solange der Benutzer nicht für die tatsächlich benötigte Belegung von Netzressourcen bezahlen muß, sondern lediglich die Zeitdauer zwischen

abgeschlossenem Verbindungsauf- und -abbau pauschal berechnet wird, ist es unökonomisch, eine Verbindung länger als unbedingt notwendig bestehen zu lassen [222].

Obwohl der Einfluß inhaltlicher Aspekte des Verbindungsaufbaus, wie gesehen, deutlich vom reinen Zeitaufwand für die Verbindungsverwaltung dominiert wird, sollen einige diesbezügliche Schwachpunkte angesprochen werden. Die ISO/OSI-Standards müssen bekanntlich so ausgelegt sein, daß grundsätzlich jegliche Kombination aus Benutzervorstellung und realer Netzkonfiguration abgedeckt werden. Als Konsequenz daraus wurden umfangreiche Optionen definiert. Deren Benutzung und die Entscheidung über einzusetzende spezifische Optionsparameter sowie deren Position und Format müssen im Einzelfall während des Verbindungsaufbaus verhandelt und festgelegt werden. Da jede Abweichung von einem erwarteten Vorgehen aufwendig ist, sollten diese Optionen nur in begründeten Fällen überhaupt eingesetzt werden und nicht - wie häufig anzutreffen - pauschal, etwa mit Defaultwerten belegt, angewendet werden.

Zusätzlich ergeben sich in aller Regel durch die Umgebung, speziell die vorhandenen Möglichkeiten der darunterliegenden Schichten, erhebliche reale Einschränkungen. Entgegen dem OSI-Prinzip der möglichen absoluten Unwissenheit über Details der darunterliegenden Schichten sind in der Praxis viele dieser Einschränkungen doch bekannt. Ist die Verwendung von Optionen und die Aushandlung von Verbindungsparametern unvermeidbar, so sollte dieses Wissen auch verwendet werden.

Wenn beispielsweise in darunterliegenden Schichten keine Prioritätsbehandlung vorgesehen ist, macht die u.U. aufwendige Aushandlung einer bevorzugten Datenklasse keinen Sinn. Wenn nur ein verbindungsloser Dienst der Vermittlungsschicht implementiert ist, sind Anforderungen zu Dienstqualitäten, die eine Verbindung zwingend voraussetzen würden, ebenso sinnlos.

Wirken sich die genannten Beispiele hauptsächlich auf Umfang und Dauer des Verbindungsaufbaus aus, so wirkt sich die Vereinbarung der maximal zulässigen PDU-Größe auch während der Nutzdatenübertragung aus: bei günstiger - dem unterliegenden Netzwerk angepaßten - Wahl kann mehrmaliges effizienzminderndes Segmentieren innerhalb des gesamten Schichtengebildes verhindert werden.

3.3.1.5 Prüfsummenrealisierung

Die Prüfsummenberechnung der ISO-Transportschicht ist eine der Funktionen, welche aufgrund ihrer leichten logischen Verständlichkeit und dem fast nicht vorhandenen Spielraum durch mögliche Parameteränderungen oder Implementierungsvarianten bei Protokollstudien selten eingehender betrachtet wurde. Verschiedene Messungen und Erfahrungsberichte [92, 108, 294] zeigen allerdings den signifikanten Einfluß der optionalen Prüfsummenberechnung beim Einsatz des OSI-Transportprotokolls der Klasse 4.

Schon in herkömmlichen 10 Mbit/s LANs und den damit verbundenen längeren zur Verfügung stehenden Zeiträumen pro übertragenem Bit, führt das Einschalten der Prüfsummenberechnung zu einer dramatischen Leistungsverringerung von bis zu 60% (siehe z.B. [215])!

Als maßgebliche für diesen enormen Einfluß verantwortliche Gegebenheit wird bestenfalls das durch die Anordnung der Prüfsummenoktetts im Steuerkopf einer TPDU bedingte zusätzliche Kopieren angeführt. Dagegen wird dem zugrunde liegenden Algorithmus grundsätzlich keine bedeutende Aufmerksamkeit zuteil. Um beliebig schnell die Prüfsummenoktetts zu berechnen, werden eine maschinennahe Programmierung, Eingriffe in den Mikrocode oder der Einsatz von spezieller Hardware empfohlen. Implizit wird dabei von der logischen Einfachheit auf die ebenso leichte Implementierbarkeit geschlossen.

Allerdings stellt sich bei genauerer Betrachtung heraus, daß die Bewegung der Daten absolut gesehen mit einigen 100 ns pro Oktett zwar tatsächlich einen erheblichen Zeitaufwand bewirkt, daß jedoch dieser im Vergleich zu dem gesamten benötigten Aufwand der Prüfsummenberechnung von bis zu einigen 100 µs (pro Oktett!) vernachlässigbar ist.

Zunächst wird die Definition [358] der OSI-TP4 Berechnung nach Fletcher [153] angegeben:

Setze zwei Oktetts so, daß

1. die Summe aller Oktetts Null ergibt (3.7) und
2. die Summe aller Produkte eines Oktetts mit seiner absoluten Position innerhalb des zu überprüfenden Oktettstromes ebenfalls zu Null wird (3.8).

$$\sum_{i=1}^{L} a_i = 0 \qquad (3.7) \qquad\qquad \sum_{i=1}^{L} i\,a_i = 0 \qquad (3.8)$$

Dabei ist zu beachten, daß alle arithmetischen Operationen zum Modul m = 255 auszuführen sind.

Für die Verifizierung der Prüfsumme in Empfangsrichtung, also mit vom Sender gesetzten Prüfoktetts, erscheint es trivial, die zwei Bestandteile der Fletcherprüfsumme direkt eins-zu-eins in ein sehr kurzes und damit vermeintlich schnelles Hochsprachenprogramm etwa wie folgt umzusetzen:

```
PROGRAM TestChecksum;

VAR
   c0, c1, i, Wert    : INTEGER;

BEGIN  {TestChecksum}
   c0  := 0;
   c1  := 0;
   i   := 1
   READ (Input, Wert);
   WHILE (NOT(EndOfInput)) DO
   BEGIN
      c0 := (c0 + Wert)   MOD 255;
      c1 := (c1 + i*Wert) MOD 255
      i  := i + 1;
      READ (Input, Wert);
   END;  {While}
   IF (c0 <> 0) OR (c1 <> 0)THEN ChecksumError
                            ELSE  ChecksumOK
END;
```

Mit dieser Version wurden Versuche auf verschiedenen Maschinen durchgeführt und gemessen. Um trotz der beschränkten Auflösungen der Systemuhren von teilweise lediglich 2/100 Sekunden aussagekräftige Resultate zu erhalten, wurden einige hundert PDUs mit jeweils 1024 Oktetts simuliert. Der sich durch die Verwaltung der äußeren PDU-Zählschleife ergebende Fehler wurde durch Vergleichsläufe ohne Prüfsummenberechnung ermittelt und entsprechend berücksichtigt. Tabelle 1 enthält einige Werte.

Die überraschend hohen Werte, die bis in den 100µs-Bereich (pro Oktett!) hineinreichen, waren nur ein unerfreulicher Aspekt. Überprüfungen von Hand deckten zudem auch fehlerhaftes Verhalten auf!
Das Problem ließ sich schlußendlich auf Überlaufsituationen von Variablen eingrenzen. Benutzt der Rechner z.B. 16 Bit für die Darstellung einer vorzeichenbehafteten Zahl vom Typ INTEGER, so ergibt sich schon ab (Zwischen-)Ergebnissen, die größer als $2^{(16-1)}$, also 32.768 sind, eine vom Rechner tolerierte und daher nicht explizit angemahnte Überlaufsituation. Da das höchstwertige Bit, welches für die Anzeige des Vorzeichens reserviert sein sollte, bei Werten größer als 32.768 eine "1" enthält, wird das

Tabelle 3.1: *Zeiten für die Prüfsummenverifikation (in µs)*

PC mit i80486DX, 33 MHz	Turbo Pascal 6.0	15
VAXstation 3200	VAX-Pascal	25
VAXstation 2000	VAX-Pascal	68
PC mit i80386DX, 25 MHz	Turbo Pascal 5.5	95
PC mit i80386SX, 16 MHz	Turbo Pascal 5.5	209

Resultat als negative Zahl interpretiert. Die darauffolgende Modulorechnung liefert fälschlicherweise ein negatives Ergebnis, das sich - bedingt durch die Konventionen zur Darstellung negativer Zahlen - zudem im Betrag auch noch um 1 vom korrekten Wert unterscheidet!

Offensichtlich kommt nur der Term *i•Wert* für eine solche Bereichsüberschreitung in Frage. Bei Werten im mittleren Bereich reichen schon wenige hundert Oktetts lange PDUs aus, um den Grenzwert zu überschreiten. Aus logisch funktioneller Sicht lassen sich wiederum relativ leicht Kontrollabfragen und angepaßte Korrekturmaßnahmen einfügen. Allerdings müssen diese naturgemäß in der innersten Schleife eingebracht werden, müssen also bei jedem betrachteten Oktett durchlaufen werden. Als Folge davon wirkt sich jede einzelne zusätzliche Befehlszeile gleich merklich auf den benötigten Zeitbedarf aus.

Für die Generierung der zwei Prüfsummenoktetts in Senderichtung werden zunächst die Prüfoktetts x und y, die sich an der Stelle k und k+1 der zu prüfenden PDU befinden, mit dem Wert Null angenommen. Über die somit vollständige PDU wird analog zum Prüfsummentest auf Empfangsseite die Berechnung der Hilfsgrößen $c0$ und $c1$ durchgeführt. Nach dem Durchlauf der PDU ergeben sich für die zwei noch unbekannten Prüfoktetts x und y die zwei unabhängigen Gleichungen:

$$(c0+x+y)\,MOD\,255=0 \qquad (3.9) \qquad\qquad (c1+k\,x+(k+1)\,y)\,MOD\,255=0 \qquad (3.10)$$

aus denen sich x und y wie folgt berechnen lassen:

$$x=(c1-(k+1)\,c0)\,MOD\,255 \qquad (3.11) \qquad\qquad y=(k\,c0-c1)\,MOD\,255 \qquad (3.12)$$

Die erhaltenen Werte für x und y können jetzt an den Positionen k bzw. k+1 der PDU eingefügt werden, und die PDU kann abgesandt werden. Der Aufwand für die Berechnungen, insbesondere die explizite Behandlung von Modulosubtraktionen und Modulokorrektur bei negativen Zahlen, ist zwar deutlich größer als der Aufwand für einen Durchlauf der inneren Schleife, fällt jedoch schon ab kleinen PDU-Längen für den Gesamtaufwand kaum mehr ins Gewicht.

Einige erste Begründungen für den überaus hohen Zeitbedarf für die Berechnung der Prüfsumme nach dem Fletcherverfahren werden nachfolgend aufgeführt:

- Es handelt sich bei der Fletcherprüfsumme nicht um eine für Hardwareimplementierung geeignete zyklische Redundanzprüfung durch Polynomdivision wie sie von den CRCs (Cyclic Redundancy Checks) der mediennahen Schichten bekannt ist.
- Stattdessen gehört der Fletcheralgorithmus in die Kategorie der arithmetischen Codes, die Additionen, Subtraktionen und auch aufwendige Multiplikationen benutzen.
- Der Algorithmus ist durch das explizite Vorhandensein der Oktettposition strikt auf Oktetts ausgerichtet. Selbst wenn Maschinen über größere Wortbreiten verfügen, muß auf Oktettbasis gerechnet werden.
  (In [284] wird ein Verfahren vorgestellt, daß auch mit größeren Wortbreiten arbeiten kann. Allerdings ist der Aufwand für die Korrekturrechnung nicht vernachlässigbar, und es wird eine vom originalen Fletcheralgorithmus abweichende Berechnungsweise eingesetzt).
- Die Absolutheit der Positionsangabe erschwert zudem mögliche Parallelisierung oder Fließbandverarbeitung erheblich.
- Die gewählte Moduloarithmetik bzw. die zugehörige Moduloumrechnung nach einer ohne Moduloarithmetik durchgeführten "normalen" Operation ist alles andere als trivial und kann oft nur durch sukzessive Subtraktion des Moduls durchgeführt werden.

  (Anmerkung: im Gegensatz zur Modulo 256-Arithmetik können Berechnungen zum Modul 255 nicht direkt durch Oktettbetrachtungen durchgeführt werden.)

3.3.2 Wahl des Protokollstapels

Im vorherigen Abschnitt 3.3.1 wurden einige Schwachpunkte angesprochen, die allgemein innerhalb einer Protokollschicht und speziell bei der OSI-Transportschicht anzutreffen sind. Für das Gesamtverhalten eines kompletten Kommunikationssystems müssen aber auch schichtenübergreifende Aspekte beachtet werden, wobei der Zusammensetzung des Protokollstapels (protocol stack) aus der Vielzahl der vorhandenen Schichtenvarianten eine besondere Bedeutung zukommt.

Obwohl durch die Einführung von Anpassungs- und Übersetzungsfunktionen die Kombination von Protokollschichten aus verschiedenen Protokollfamilien nicht grundsätzlich unmöglich ist (und in vielen realen Implementierungen auch tatsächlich anzutreffen ist [260, 295]), soll in diesem Abschnitt überwiegend von den Wahlmöglichkeiten innerhalb einer Protokollfamilie die Rede sein.

Bis einschließlich zur Schicht 3 sind verschiedene Protokollwelten - teilweise schon lange bevor die Arbeiten am OSI-RM begannen - eingeführt. Neben der OSI-konformen Schichtung und der im Bereich der Computerkommunikation besonders weitverbreiteten (D)ARPA-Internetzprotokollfamilie ((Defense) Advanced Research Projects Agency) sei hier insbesondere die im Bereich der öffentlichen Netze vielfach eingesetzte CCITT X-Serie (X.21, X.25, HDLC, LAPB etc.) erwähnt.

Oberhalb der noch stark netzorientierten Vermittlungsschicht sind dagegen außer den OSI-geprägten Protokollen und - mit Einschränkungen - der Internetzfamilie keine weiteren firmenunabhängigen Protokollwelten vorhanden. Im Gegensatz zum OSI-Ansatz der Verborgenheit von Details darunterliegender Schichten ist die Internetzfamilie, trotz der vorhandenen Schichtung, relativ durchgängig und mit Wissen über die jeweils anderen Schichten aufgebaut. Auch hat der Benutzer nur sehr wenig Wahlmöglichkeiten bei der Zusammenstellung eines Protokollstapels aus Protokollen der Internetzfamilie.

Ganz anders bei OSI. Um das OSI-RM möglichst flexibel sowohl den jeweiligen Anforderungen als auch den Eigenschaften der vorhandenen Umgebung anpassen zu können, wurden für die einzelnen Schichten jeweils unterschiedliche Dienste festgelegt, aus denen der Dienstnutzer, in der Regel entspricht das der nächsthöheren Schicht, den für ihn am besten geeigneten Dienst auswählen kann. Die einzelnen Dienste unterscheiden sich u.a. in ihrem grundsätzlichen Charakter, d.h. sie können verbindungslos oder verbindungsorientiert sein. Weitere Unterscheidungsmerkmale sind die Verfügbarkeit und die Leistungsfähigkeit bestimmter Protokollmechanismen wie etwa Fehlererkennung und -behebung, Flußsteuerung oder Mehrfachnutzung von Ressourcen (Multiplexen). Die unterschiedlichen Dienste einer Schicht werden durch die Einführung von Klassen der jeweiligen Schicht repräsentiert.

Jede dieser Klassen liegt in der Praxis als eigenständige Implementierung vor. In einer realen Umgebung sind selten alle Klassen einer Schicht verfügbar. Im Normalfall wird der Systemverwalter versuchen, genau eine Klasse pro Schicht zu unterstützen. Die Auswahl richtet sich dabei in erster Linie nach der relativen Leistungsfähigkeit der Klassen einer Schicht, und weniger nach Eigenschaften anderer Schichten, oder gar nach Gesichtspunkten zur Leistungscharakteristik des Gesamtsystems. Dies führt zu Protokolltürmen mit funktioneller Redundanz mit entsprechend reduzierter Gesamtleistungsfähigkeit.

Ein Beispiel einer nicht optimierten Schichtenwahl ist die Kombination der verbindungsorientierten Typklasse 2 der logischen Sicherungsschicht mit der ebenfalls leistungsfähigsten Klasse 4 der Transportschicht: Die Klasse LLC Typ 2 bietet einen Dienst an, der die reihenfolgerichtige Übertragung von Nutzinformation der höheren Schicht garantiert. Fehlererkennung, Quittierungs- und Wiederholungsstrategien sind also Bestandteil dieses Dienstes. Ebenso sind sowohl eine Verbindungsverwaltung als auch Flußsteuerungsmechanismen zur Erbringung des LLC Typ 2 Dienstes Voraussetzung.

TP4, als leistungsfähigste Klasse der Transportschicht, ist vom Standard [355], wie in Tabelle 3.2 aufgeführt, für Netzumgebungen der Kategorie C vorgesehen. Kategorie C bedeutet dabei ein unzuverlässiges, fehlerbehaftetes Netz, in dem Information durchaus auch verloren oder verdoppelt werden kann [42].

Darüber hinaus ist es in einem Netz der Kategorie C zulässig, wenn die Netzwerkverbindung gelegentlich unterbrochen wird. (Typ A Netze sind perfekt, bei Netzen des Typs B kann die Netzverbindung zwischenzeitlich abgebaut bzw. zurückgesetzt worden sein).

Tabelle 3.2: *Elemente der 5 ISO/OSI-Transportprotokollklassen*

Klasse	0	1	2	3	4
Verbindungsaufbau	x	x	x	x	x
Verbindungsablehnung	x	x	x	x	x
Zuordnung zu Schicht 3 Verbindung	x	x	x	x	x
Segmentieren von TSDUs in TPDUs	x	x	x	x	x
Zuordnung von TPDUs zu Verbindungen	x	x	x	x	x
TPDU Übertragung	x	x	x	x	x
Behandlung von Protokollfehlern	x	x	x	x	x
Regulärer Verbindungsabbau	x	x	x	x	x
Zusammensetzen von TPDUs		x	x	x	x
Verbindungsabbau im Fehlerfall	x		x		
Priorisierter TPDU-Transfer (Expedited data)		o	o	x	x
TPDU-Folgenumerierung		x	o	x	x
Steuerung des Informationsflusses			o	x	x
Wiederaufsetzen nach RESET		x		x	x
Speichern von TPDUs bis zum Erhalt einer Quittung		x		x	x
Neuzuordnung nach Abbau der Schicht 3		x		x	x
Stabile (eingefrorene) Referenzen		x		x	x
Multiplexen von Verbindungen			x	x	x
Nutzen mehrerer Schicht 3 Verbindungen (Splitting)					x
Wiederholungssendung nach Ablauf eines Zeitgebers					x
Herstellen der korrekten TPDU-Reihenfolge					x
Zeitgeber zur Überwachung der Verbindungsaktivität					x
Schicht 4 Prüfsumme					o
Netzkategorie	A	B	A	B	C

x - vorhanden o - optional

Um die geforderte Dienstgüte und alle Dienstmerkmale erbringen zu können, muß TP4 offensichtlich alle Funktionalitäten der LLC-Schicht vom Typ 2 beinhalten. Unabhängig davon, ob die exakt gleichen Mechanismen benutzt werden, ist eine nicht optimale funktionelle Redundanz vorhanden.

Ein weiteres Beispiel für die Notwendigkeit der Abstimmung zwischen den Schichten eines Protokollturmes ist das Segmentieren von Nachrichten. Die ISO/OSI Standards sehen die Möglichkeit für das Aufteilen von großen SDUs in der Kommunikationssteuerungss-, Transport- und Vermittlungsschicht vor. Alleine der Aufwand für das unnötige mehrfache Ausführen dieser Aktion wirkt sich schon negativ auf die Leistungscharakteristik aus. Zusätzlich kann durch nicht optimale Wahl der jeweiligen Teilungsgrößen (keine ganzzahligen Vielfache) eine Verschlechterung des Verhältnisses der übertragenen Nutz-

bits zur Gesamtzahl aller übertragenen Bits induziert werden. Ein willkürlich gewähltes Zahlenbeispiel soll dies verdeutlichen:

Zu übertragen seien 9200 Oktetts, die maximalen Segmentgrößen der Schichten 5, 4 und 3 seien 4500, 1024 bzw. 576 Oktetts. Bild 3.5 zeigt die nacheinander entstehenden Segmente.

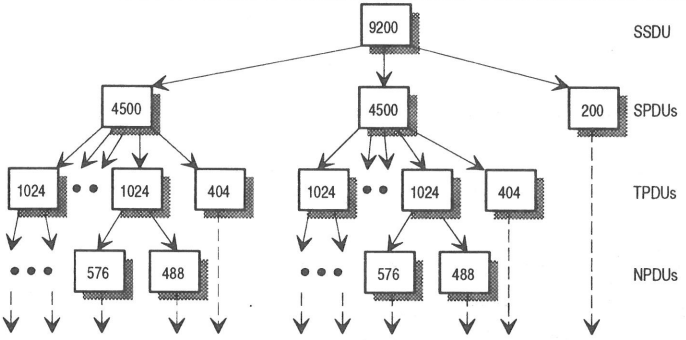


Bild 3.5: *Auswirkung von Mehrfachsegmentierung (Baum).*

Die Segmentierung auf Schicht 5 ergibt 2 TSDUs der Länge von 4500 Oktetts und ein Reststück mit 200 Oktetts. Die Schicht 4 kann letzteres unverändert an die Schicht 3 weitergeben; die großen Blöcke werden jeweils in 4 NSDUs à 1024 Oktetts und eine 404 Oktett lange PDU zerlegt. Schicht 3 kann sowohl das 200 Oktett lange Reststück der Schicht 5-Segmentierung, als auch die zwei 404 Oktett langen Fragmente, die bei der Schicht 4-Teilung entstanden sind, unverändert weiterleiten. Die 8 maximal langen NSDUs werden jeweils in zwei Teile der Längen 576 bzw. 448 zerteilt. Insgesamt ergeben sich:

- je 8 NPDUs mit 576 bzw 448 Oktetts
- 2 NPDUs mit 404 Oktetts und
- 1 NPDU mit 200 Oktetts.

Das sind 19 PDUs der Schicht 3, von denen jede natürlich einen eigenen Steuerkopf besitzt, was gleichbedeutend mit der Übertragung von Bits ist, die keine Nutzinformation darstellen. Zum Vergleich: hätte man die 9200 zu übertragenden Nutzoktetts gleich auf die kleinste maximale Länge von je 576 Oktetts segmentiert, so wären nur 16 NSDUs (15 maximaler Länge, eines mit 560 Oktetts) notwendig gewesen.

Neben der Wahl der Protokollklassen kann das Verhalten des Kommunikationssystems durch die Auswahl von Optionen und deren Parameter innerhalb der einzelnen Schichten den Wünschen und Gegebenheiten angepaßt werden. Zur Verdeutlichung sind einige der TP4-Optionen nachfolgend aufgeführt:

- Maximal zulässige TPDU-Länge
- Information über mit den Instanzen verbundene Dienstzugangspunkte
- Versionsnummer
- Alternative akzeptable Protokollklassen
- Prüfsummenverwendung
- Maximale Verzögerung bis zum Aussenden der nächsten Quittierung
- Minimal akzeptierbarer und mittlerer erwarteter Durchsatz
- Maximal akzeptierbare und mittlere erwünschte Restfehlerrate
- Prioritätsfestlegung
- Maximal akzeptierbare und mittlere gewünschte Übertragungsverzögerung
- Toleranzgrenze für Zahl der Netzverbindungsstörungen oder Störungszeit

Einige der Parameter sind für die entfernte Instanz von Interesse, andere dagegen sind direkt für die Interaktion mit dem unterliegenden Diensterbringer bestimmt. Die ISO-Standards führen allerdings für viele Optionen weder eine genauere Syntax, d.h. Format- und Wertefestlegung, ein, noch wird angegeben, was und wie bei der Verwendung der Optionen ausgeführt werden kann oder soll. Darüber hinaus setzt die Verwendung von bestimmten Optionen entgegen dem ISO-Grundprinzip eine genaue Kenntnis der unterliegenden Schichten voraus. Ohne dieses Wissen über Eigenschaften der unterliegenden Schichten bzw. des zugrunde liegenden Netzes können überflüssige und/oder unsinnige Optionen bzw. Parameter von Optionen gewählt werden, deren Bearbeitung u.U. erhebliche Leistungseinbußen bewirken können.

3.3.3 Aspekte der Implementierung

Bei der Einführung der nicht protokollspezifischen Unterstützungsfunktionen in Abschnitt 3.2 wurde schon angedeutet, welch eminenten Einfluß diese umgebungs- und implementierungsabhängigen Funktionalitäten auf das Verhalten und die Leistungsfähigkeit eines Kommunikationssystems haben. Einige Teilaspekte, welche einen besonders großen Anteil zum von außen sichtbaren Verhalten des Systems beitragen und damit bei ungeeignetem Einsatz die Leistungsfähigkeit des Kommunikationssystems erheblich negativ beeinträchtigen können, werden in diesem Kapitel detaillierter betrachtet.

3.3.3.1 Abbildung logischer Funktionen

Bei der Umsetzung logischer Funktionalität in reale Systemkomponenten muß prinzipiell eine Aufteilung in Hard- und Softwarebereiche vorgenommen werden. In heutigen Kommunikationssystemen existiert lediglich für die technologieabhängigen und mediennahen Schichten 1 und 2a Hardwareunterstützung in Form von speziellen Netzwerkadapterkarten, die oft auch über einen (beschränkten) Pufferspeicher verfügen. Es ergibt sich die in Bild 3.6 gezeigte typische Konfiguration [38].

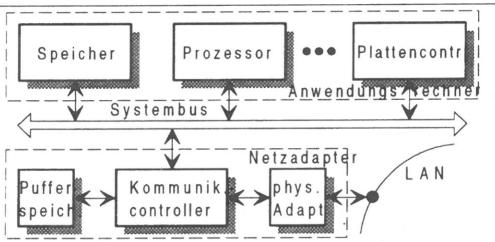


Bild 3.6: *Typischer Aufbau eines Kommunikationssystems.*

Ab Schicht 2b des OSI-RMs werden die Protokollfunktionen in Software auf dem AR ausgeführt. Bei einigen Implementierungen wird die Schicht 2b schon auf der Adapterkarte ausgeführt, wobei nicht ausschließlich Software Verwendung findet. Hardwareunterstützung für Teilfunktionen der höheren Schichten ist in heutigen Systemen nicht anzutreffen, deshalb wird im folgenden das Augenmerk auf die Abbildung von logischen Funktionen auf Softwarestrukturen gelegt.

Die Implementierung von Software zur Unterstützung und den Betrieb eines Kommunikationssystems wird naturgemäß stärker von Flexibilitäts- und Portierungsaspekten bestimmt, als die Erstellung nicht kommunikationsorientierter Programme. Aus diesem Grunde werden Protokollfunktionalitäten möglichst maschinenunabhängig in entsprechende Software umgesetzt, was in der Regel auch eine Einbindung in den Systemkernbereich eines Betriebssystems (operating system kernel) ausschließt. Stattdessen wird von einer Einbindung in den regulären Benutzerbereich ausgegangen.

Die Aufteilung auf unabhängige Prozesse oder Subprozesse, im Englischen als Tasks oder Threads be-

kannt, orientiert sich dabei sehr stark an dem OSI-RM und dessen ausgeprägter Unterteilung in Schichten. Fast zwangsläufig überträgt sich die rein logisch abstrakte OSI-Schichtung eins-zu-eins auf eine erste Aufteilung aller Aufgaben eines Protokollstapels auf unabhängige Prozesse: für jede zu implementierende Schicht wird (zumindest) ein eigener Prozeß vorgesehen. Dieser hat zu den Prozessen, die benachbarte Schichten des OSI-RMs realisieren, jeweils explizite Ein- und Ausgänge (InterProcess Communication, IPC), auf deren Realisierungsform noch gesondert eingegangen werden wird. Wie schon in Abschnitt 3.2.1 erwähnt wurde, ergibt sich aus der Verwendung mehrerer Prozesse die Notwendigkeit für Wechsel des gerade aktiven Prozesses. Diese Prozeßwechsel werden in der DV-spezifischen Literatur (z.B. [4, 27, 36, 43]) mit Kontextwechsel bezeichnet. Der Kontext eines Prozesses umfaßt dabei:

- den Namen des Prozesses
- die Bezeichnung der gerade bearbeiteten Teilaufgabe
- Angaben zur augenblicklichen Stellung in der Prozeßhierarchie
- den Zustand des Rechnerkerns
- Angaben zum Arbeits- und Alarmzustand des Prozesses
- bestehende Rechte bzw. Beschränkungen
- Informationen über Belegung, Reservation und aktuellen Zustand von Betriebsmitteln
- hinreichende Beschreibung des Programmadreßraumes

All diese kontextspezifische Information muß bei einem Wechsel zunächst für den abzulösenden Prozeß gespeichert werden und dann für den "neuen" Prozeß entsprechend wieder eingerichtet werden. Dazu müssen neben dem eigentlichen Informationstransfer durch physikalisches Bewegen von Daten auch Abfragen, Berechnungen, Reinitialisierungen, Konsistenzprüfungen und ähnliche Aktionen durchgeführt werden. Zudem muß die Zeit für den Auswahlmechanismus (scheduling) und andere Funktionen der Prozeßverwaltung miteinbezogen werden. So können sich die jeweils relativ kurzen Einzelzeitbedarfe zu Gesamtzeiten bis in den Millisekundenbereich hinein aufaddieren. Zur besseren Einordnung des Zahlenwerts und seiner gravierenden Auswirkungen für Kommunikationssysteme im Hochgeschwindigkeitsbereich dient folgende Betrachtung:

Gegeben sei eine PDU mit der maximal zulässigen Internetzgröße von 576 Oktetts. Zur Übertragung der 4608 Bits dieser PDU werden bei einem Medium mit 1 Mbit/s Bandbreite etwa 4.6 ms benötigt. Soll das System Echtzeiteigenschaften besitzen, steht maximal genau diese Zeit zur kompletten Bearbeitung der PDU zur Verfügung. Soll ein Durchsatz von 100 Mbit/s erreicht werden, ergeben sich für die Bearbeitung der PDU nur noch etwa 46 µs [196]! Während der Zeit, die für einen einzigen Kontextwechsel (hier: zwischen Prozessen im Benutzerbereich) benötigt wird,

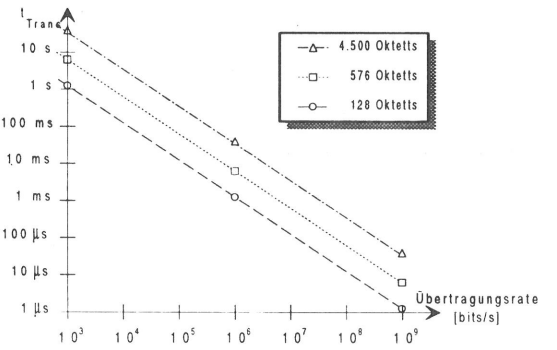


Bild 3.7: *Maximale Bearbeitungszeiten für unterschiedliche PDU-Längen*

müßten daher schon weit über zehn PDUs dieser Länge vollständig bearbeitet sein! In Bild 3.7 sind die Übertragungszeiten t_{Trans} für weitere PDU-Längen über der Bandbreite aufgetragen.

Bedingt durch die grundsätzlich positive Tendenz innerhalb der Systementwurfstechnik zu einer hochmodularen Strukturierung [277] hört die Modularisierung bei weitem nicht bei den Schichten auf. Beliebt sind Verfeinerungen, die sich u.U. auch wieder in eigenen Prozessen oder zumindest Subprozessen niederschlagen, in Sende- und Empfangsteil, das Aufteilen nach Funktionen (Steuerkopfauswertung, Fehlerbehandlung, Flußsteuerung, Interaktion zur unteren / oberen Nachbarschicht, Segmentieren ...) oder die Einführung von je einem Prozeß pro Verbindung einer Schicht.

3.3.3.2 Schnittstellen

Die Frage der Realisierung von Schnittstellen in Kommunikationssystemen ist eng mit dem im vorigen Abschnitt Gesagten verknüpft. Stand dort der für Prozeßwechsel benötigte Aufwand im Vordergrund, so werden hier Schwachpunkte der Art und Weise von Schnittstellenimplementierungen angesprochen.

Der sicherste aber zugleich ineffizienteste Ablauf der Zusammenarbeit zweier unabhängiger Komponenten setzt sich aus folgenden Teilschritten zusammen [293, 322, 323]:

- Zunächst wird eine (physikalische) Kopie aller benötigten Daten angefertigt.
- Die gewünschte Zieleinheit wird dann über das Vorhandensein von für sie bestimmten Daten und deren genaue Lokation informiert.
  (Wird statt der aktiven Information der gewünschten Zielkomponente mittels direktem Aufruf oder Unterbrechungssignal durch den Kommunikationsinitiator ein periodisches Überprüfen des gemeinsamen Speicherbereiches von Seiten der Zielkomponente eingesetzt, so ergeben sich Ineffizienzen aus den nutzlos verbrachten Wartezeiten.)
- Die eigentliche Übergabe der Daten kann nun stattfinden.
- Die sendende Komponente wartet auf den Erhalt einer expliziten Bestätigung über den korrekten Empfang oder das Ergebnis der mit den übergebenen Daten ausgeführten Aktion.

Dieses Verfahren setzt außer der Existenz eines von beiden Komponenten zugänglichen Speicherbereiches keine weitergehenden Kenntnisse über den inneren Aufbau der jeweiligen anderen Komponente voraus und begünstigt damit die Entwicklung und den Einsatz unabhängiger Systemkomponenten.

Betrachtet man beispielhaft einen Dienstzugangspunkt (SAP) als eine ausgewählte Schnittstelle, wird einsichtig, daß der Zeitbedarf für die Erstellung der physikalischen Kopie der zu übergebenden Daten schnell nicht vernachlässigbare Größen annehmen kann. Über den SAP findet ja definitionsgemäß nicht nur eine Signalisierung mittels kurzen Nachrichten statt, sondern es werden auch die kompletten Dateneinheiten der rufenden Schicht übergeben (call by value).

Handelt es sich um Prozesse auf verschiedenen Rechnern, wird durch den beschriebenen strikt sequentiellen Ablauf auch das Potential zur möglichen Parallelverarbeitung der Komponenten zunichte gemacht.

Die enge Auslegung des OSI-RMs führt zusammen mit dem Bestreben nach größtmöglicher Modularität in heutigen Kommunikationssystemen zu vielen explizit sichtbaren Schnittstellen. So werden die logisch abstrakten Dienstzugangspunkte des OSI-RMs prinzipiell in eine real vorhandene und von außen her zugängliche Form umgesetzt. Begründet wird dies mit der Forderung zur Teilnahme an Konformitätstests.
(Anmerkung: Da in der Praxis eine Kombination von einzelnen Schichten verschiedener Hersteller an der unterschiedlichen Interpretation der in den Standards enthaltenen Freiheiten scheitert, werden nicht Eigenschaften einer isolierten Schicht überprüft, sondern Anordnungen von mehreren Protokollschichten. Dabei stehen dann die beobachtbaren Dienste und deren Merkmale im Vordergrund und nicht die Syntax und Semantik von Schnittstellen.)
Die Fähigkeit zur externen Zugänglichkeit erfordert allerdings nicht nur einen entsprechenden Anpas-

sungs- und Verarbeitungsaufwand, sondern führt auch zu zusätzlichen internen Schnittstellen, wenn z.B. dadurch ein Wechsel zwischen Kern- und Benutzermodus eines Betriebssystems erforderlich wird.

Die bisher gemachten Aussagen wurden jeweils an Beispielen von Schnittstellen zwischen Softwarekomponenten verdeutlicht. Für Schnittstellen zur Hardware oder auch für hardwareinterne Übergänge gelten die gemachten Aussagen jedoch in gleicher Weise. Zusätzliche Qualitäten ergeben sich durch die notwendigen Maßnahmen zum Ausgleich von physikalischen Geschwindigkeitsunterschieden der beteiligten Komponenten sowie durch die beim Verlassen von Systemkomponenten grundsätzlich höhere Wahrscheinlichkeit für Ressourcenkonflikte.

Insgesamt dominiert auch bei der Betrachtung von Schnittstellen die im vorigen Abschnitt gemachte Aussage, daß die Anzahl der Schnittstellen zwischen unabhängigen Systemkomponenten und damit der Modularitätsgrad umgekehrt proportional zu der erzielbaren Effizienz ist.

3.3.3.3 Speicherverwaltung

In der Diskussion der Modularität von Kommunikationssystemen in den zwei vorigen Abschnitten wurde schon implizit die Notwendigkeit von jeweils spezifisch zu den einzelnen Komponenten zugeordneten Speicherbereichen deutlich. Speicherplatz wird sowohl komponentenintern zur Wahrnehmung der rein logischen Funktionalität benötigt als auch als Hilfsmittel zur Realisierung von Schnittstellen zwischen unabhängigen Systemkomponenten. In realen Systemen hat nicht jede Komponente einen physikalisch exklusiv ihr zugeordneten Speicher zur Verfügung. Ebensowenig ist jeweils eine komponentenspezifische Speicherverwaltung pro unabhängiger Komponente anzutreffen. Stattdessen werden die individuell benötigte Speicherkapazität und die zu deren separater Verwaltung unumgängliche Funktionalität mittels eines physikalischen Speichers und einer zentralen Speicherverwaltung durchgeführt.

Eine Variante der Speicherverwaltung ergibt sich durch das Bereitstellen jeweils eigener Pufferbereiche für jede Systemkomponente (Bild 3.8). Nachfolgend wird vereinfachend der Fall betrachtet, bei der eine Schicht eine unabhängige Komponente mit jeweils eigenem zugeordneten Speicherbereich repräsentiert.

Eine (N)-SDU wird in diesem Fall aus dem (N+1)-Bereich in den (N)-Bereich kopiert, dort in (N)-PDUs umgearbeitet und schlußendlich als (N-1)-SDU in den (N-1)-Pufferbereich kopiert (entsprechendes gilt auch für die Empfangsrichtung). Diese Variante hat zwei gewichtige Vorteile:

1. Alle geforderten Abbildungsmechanismen von SDUs auf PDUs und umgekehrt können direkt umgesetzt werden.
2. Die einzelnen Schichten sind völlig unabhängig voneinander, was eine Voraussetzung für Portabilität der einzelnen Schichten ist.

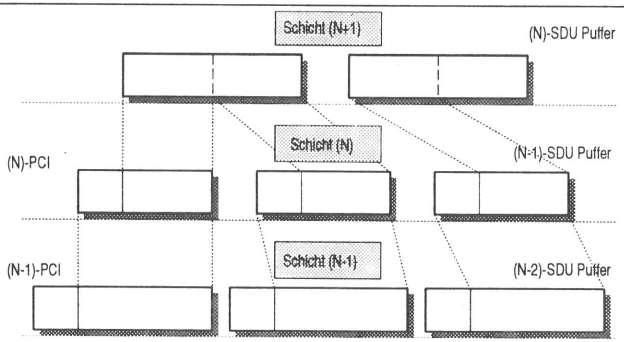


Bild 3.8: *Schichteninteraktion mittels physikalischem Kopieren*

Betrachtet man andererseits die für das Kopieren benötigten Zeiten, wird deutlich, daß die vom logischen Gesichtspunkt her triviale Datenbewegung in der Realität einen erheblichen Anteil der Gesamtbearbeitungszeit einer PDU ausmacht. Werden beispielsweise wieder die schon in Abschnitt 3.2.4 erwähnten modernen dynamischen RAM-Bausteine mit 100 ns Zykluszeit sowie 16 bit breite Zugänge vorausgesetzt, ergibt sich für die einmalige Bewegung einer 1000 Oktett langen PDU eine Zeit von 51.2 µs. Nimmt man einen mit 25 MHz getakteten RISC-Prozessor an, der im Mittel pro Taktzyklus einen Befehl abarbeiten kann, so können in dieser Zeit 1280 Maschinenbefehle ausgeführt werden! (In [108, 196] und auch im eigenen TpS werden etwa 400 Befehle für die Transportprotokollbearbeitung benötigt).

Die Größe der zu speichernden Information und auch deren Aufenthaltsdauer im Speicher variieren stark. Um trotzdem einen akzeptablen Nutzungsgrad des Speichers zu erreichen, wird dieser in Stücke gleicher Größe, sogenannte (Speicher-)Seitenrahmen oder Elementarpuffer EP, aufgeteilt. Will ein Benutzer Information in einen so strukturierten Speicher einschreiben, so wird zunächst ein freier Rahmen gesucht und für das Einschreiben bereitgestellt. Übersteigt die Länge der Nutzerinformation die Länge eines Rahmens, so wiederholt sich die obige Prozedur so lange, bis das letzte Teilstück auf einen Rahmen paßt. Natürlich muß die Reihenfolge der zu einem Benutzer gehörenden Rahmen und deren physikalische Adressierung sichergestellt sein. Die Freigabe von Speicherrahmen und das Auffinden von unbelegten Rahmen sind weitere offensichtliche Aufgaben der Verwaltung eines so strukturierten Speichers. Da der Verwaltungsaufwand umgekehrt proportional der Rahmengröße ist und da bei nicht kommunikationsspezifischen Aufgaben kaum kurze Informationseinheiten auftreten, ist die Rahmengröße in heutigen Betriebssystemen mindestens auf 1024 Oktetts (oft sogar deutlich darüber) festgelegt.

3.3.3.4 Zeitgeber

In den Abschnitten 3.1.2, 3.2.6 und 3.3.1.2 wurde die Notwendigkeit von Zeitgebern in Kommunikationssystemen und die Wahl von deren Werten schon angesprochen. Wie können Zeitgeber realisiert werden? Die einfachste Variante ist es, eine genügend große Anzahl von Zeitgebern physikalisch (als Hardware) vorhanden zu haben, um jeder neuen Anforderung exakt einen davon exklusiv zuordnen zu können. Der Verwaltungsaufwand würde sich in diesem Fall auf das einmalige Auswählen und Setzen sowie das Freigeben (Löschen) eines Zeitgebers beschränken. Der benötigte Hardwareaufwand ist allerdings selbst schon bei kleiner Anzahl indiskutabel hoch. Ein Verfahren zur Reduktion der Anzahl physikalisch notwendiger Zeitgeber ist die Einführung einer zusätzlichen Komponente zur Zeitgeberverwaltung. Durch geeignet gewählte interne Verarbeitung stellt diese Komponente dem Benutzer eine große Anzahl logisch unterschiedlicher Zeitgeber zur Verfügung. Das äußere, für den Benutzer sichtbare, Verhalten der gesamten Zeitgeberkomponente unterscheidet sich dabei nur unwesentlich von der obigen Variante.

Die einfachste und häufig anzutreffende Vorgehensweise basiert auf einem periodisch ausgesandten Signal des Zeitgebers. Diese Ticks lösen Unterbrechungsanforderungen aus, auf die dann die Verwaltungskomponente reagiert. Die Periode der Ticks muß der maximal geforderten Zeitgeberauflösung angepaßt sein, wodurch die Zahl der Unterbrechungsanforderungen und der resultierende Verarbeitungsaufwand rasch signifikant werden können. Da - prinzipiell mögliche - intelligentere Tick-Generierungsstrategien zur Reduktion der Anzahl der Unterbrechungsanforderungen kaum eingesetzt werden, wird im folgenden nur die periodische Aussendung von Ticks zugrunde gelegt. Bevor auf die zwei Formen der Zeitgeberverwaltung eingegangen wird, ist die Einführung von vier Unterfunktionen notwendig [305, 322]:

- STARTE_ZEITGEBER(Identifikation, Zeitwert, Ablaufreaktion):
  Mit dieser Routine kann der Benutzer des Zeitgebersystems das Anlaufen eines Zeitgebers initiieren. Der Zeitwert kann relativ (Ablauf nach x Zeiteinheiten von jetzt ab) oder absolut (Ablauf zum Zeitpunkt y bzw. Zustand z des Zeitgeberbausteins) erfolgen. Neben dem Zeitwert, nach dem bzw. zu dem der Zeitgeber ablaufen soll, muß eine eindeutige Bezeichnung und eine Charakterisierung des erwarteten Verhaltens im Falle des Ablaufs des Zeitgebers mitübergeben werden. Die Routine

kann so ausgelegt werden, daß die Generierung eines neuen Zeitgebers mitenthalten ist.

- STOPPE_ZEITGEBER(Identifikation):
  Diese Routine benutzt die übergebene Identifikation, um den zugehörigen Zeitgeber zu lokalisieren und anzuhalten. Gleichzeitig wird der Zeitgebereintrag meist auch gelöscht. Das Anhalten und eventuell spätere erneute Weiterlaufen kann in diesen Fällen nur durch die Generierung eines neuen Zeitgebers mit allen benötigten Parametern nachgebildet werden.
- PRO_TICK_ROUTINE:
  Setzt man eine Granularität von T Einheiten (z.B. 1 ms) voraus, so muß diese Routine alle T Einheiten überprüfen, ob einer der aktiven Zeitgeber abgelaufen ist. Ist dies der Fall, wird die nachfolgend beschriebene Routine ABLAUF_EREIGNIS aufgerufen.
- ABLAUF_EREIGNIS:
  Innerhalb dieser Routine wird die Ablaufreaktion, die mit STARTE-ZEITGEBER übergeben wurde, durchgeführt. Zusätzlich wird STOPPE_ZEITGEBER für den abgelaufenen Zeitgeber aufgerufen, um Mehrdeutigkeiten zu vermeiden.

Die erste Funktion wird vom Benutzer aufgerufen, die zweite kann sowohl vom Benutzer als auch intern aufgerufen werden. Die zwei anderen Routinen werden direkt oder indirekt von periodischen Zeitsignalen, den Ticks, ausgelöst, welche von speziellen Hardware (Uhren)-Bausteinen erzeugt werden.

Zum Vergleich von Zeitgeberimplementierungen werden drei Kriterien eingeführt [305, 322]:

1. Speicherbedarf, der für die notwendigen Datenstrukturen und deren Verwaltung innerhalb des Zeitgebermoduls benötigt wird.
2. Verzögerungszeit, die vom Aufruf einer Routine bis zu deren Beendigung vergeht.
3. Komplexität der Routinen.

Alle drei Bewertungskriterien hängen von der Anzahl von gerade aktiven Zeitgebern ab. Der Parameter n repräsentiert diese Anzahl, wobei wahlweise Mittel- oder auch Maximalwert eingesetzt werden können.

Geht man von ungeordneten Zeitgeber- und Speicherstrukturen aus, so ergibt sich eine einfache Zeitgeberimplementierung. Wird ein neuer Zeitgeber angefordert, so weist die START_ZEITGEBER Routine irgendeinen freien Speicherbereich zu, in dem die Parameter des neuen Zeitgebers abgelegt werden können. Bei jedem Tick muß die PRO_TICK_ROUTINE die Zeitwerte aller aktiven Zeitgeber nacheinander auf das Ablaufen des zugehörigen Zeitgebers überprüfen. Bei relativer Angabe der Zeitwerte muß innerhalb der PRO_TICK_ROUTINE zusätzlich ein Dekrementieren jedes aktiven Zeitgebers durchgeführt werden. Es ergibt sich somit ein Aufwand der Komplexitätsordnung $O(n)$.
Beim Starten muß ein freier Speicherbereich bereitgestellt werden, zum Stoppen muß der Speicherbereich lokalisiert werden. Dazu sind jeweils im Mittel $n/2$ Suchschritte notwendig, wobei n die maximal zulässige Anzahl von Zeitgebern darstellt.

Vergibt man die Identifikationen so, daß diese direkt als Adresse für den zugehörigen Speicherbereich benutzt werden können, so entfällt zwar der lineare Aufwand zur Suche des entsprechenden Speicherbereichs beim Starten und Stoppen eines Zeitgebers, dafür ist aber eine Verwaltung der Identifikationen und eine zusätzliche Kommunikation zwischen Benutzer und Zeitgeberkomponente zur Anforderung bzw. Zuteilung der Identifikation notwendig. Will der Benutzer seine eigenen Identifizierer benutzen, wird eine Umsetzung externer Bezeichner in interne erforderlich. Der Aufwand für die PRO_TICK-ROUTINE hängt nicht von der inneren Struktur der Speicherbereiche ab und bleibt daher unverändert.

Der hohe Aufwand für die PRO_TICK_ROUTINE führt dazu, daß diese Implementierungsvarianten nur dann sinnvoll eingesetzt werden können, wenn:

- die Zahl n der Zeitgeber relativ klein ist,
- Zeitgeber überwiegend nach wenigen Ticks ablaufen oder

- die PRO_TICK_ROUTINE z.B. durch Hardwareunterstützung genügend leistungsfähig ist.

Eine in heutigen Betriebssystemen häufig anzutreffende Möglichkeit zur Verringerung des Pro-Tick-Aufwandes besteht darin, die Zeitgeber entsprechend ihren Zeitwerten relativ zum Vorgänger zu ordnen. Als Folge der Sortierung braucht die PRO_TICK_ROUTINE nur noch auf den Zeitgeber zuzugreifen, der als nächstes ablaufen wird (next-due timer).

Offensichtlich erhöht sich jedoch der Aufwand speziell der Funktion START_ZEITGEBER erheblich. Der richtige Platz innerhalb der bestehenden Zeitgeber muß gefunden werden. Je nach gewähltem Mechanismus zur Realisierung der inneren Ordnung müssen die zugrunde liegenden Datenstrukturen umstrukturiert werden. Bei relativer Angabe der Zeitwerte muß zusätzlich eine Umrechnung in Relation zu allen vorher ablaufenden Zeitgebern durchgeführt werden, denn der eigene Zeitwert entspricht ja immer nur dem Zeitraum bis zum Ablaufen des direkten Vorgängers.

Das Suchen der reihenfolgerichtigen Position eines neuen Zeitgebers ist unabhängig von der gewählten Datenstruktur und unabhängig vom eingesetzten Suchalgorithmus im schlechtesten Fall von der Ordnung $O(n)$. In diesem Fall müssen alle bestehenden Zeitgeberwerte betrachtet werden, bevor die endgültige Position gefunden wurde. Die mittlere Verzögerungszeit für die Funktion START_ZEITGEBER hängt stark von zwei Charakteristika der Anwendung ab. Dies sind:

1. Die Verteilungsfunktion der angeforderten Zeitintervallgrößen und
2. Die Verteilungsfunktion des Ankunftsprozesses von Aufrufen der Funktion.

Eine Modellierung durch unendlich viele Bedieneinheiten ist möglich, weil logisch gesehen jeder Zeitgeber bei jedem Tick dekrementiert (bedient) wird. In [259] wird gezeigt, daß man mittels des Kriteriums von Little die durchschnittliche Warteschlangenlänge dieses Systems bestimmen kann. Aus der ebenfalls bestimmbaren Verteilung der verbleibenden Zeit aller gerade bearbeiteter Zeitgeberelemente kann die Dichte der Restsystemverweilzeit der Zeitintervallverteilung abgeleitet werden.

Wird ein Poisson-Ankunftsprozeß gewählt, wird die Liste von vorne her durchsucht und nimmt man für Lesen und Schreiben jeweils Kosten von genau einer Einheit an, dann ergeben sich nach [259] folgende mittleren Kostenordnungen für zwei ausgewählte Verteilungen der Intervallgrößen:

$O(2+n \cdot 2/3)$	für eine negativ-exponentielle Verteilung
$O(2+n \cdot 1/2)$	für eine Gleichverteilung

Für die negativ-exponentielle Verteilung kann die Kostenordnung auf $O(2+n \cdot 1/3)$ reduziert werden, wenn die Liste vom Ende her durchsucht wird. Der Suchbeginn am Ende der Liste, also von den Zeitgebern mit den am weitesten in der Zukunft liegenden Zeiten her, ist für Umgebungen geeignet, in denen häufig Zeitgeber mit ähnlichen Werten benötigt werden. In Betriebssystemen ist dies der Fall, bei Protokollen nicht.

In traditionellen UNIX-Systemen ist die sortierte Anordnung der Zeitgeber (callout table) als Feld implementiert [1, 322]. Felder (arrays) haben den Vorteil des direkten Zugriffs und einer impliziten Ordnung, aber den Nachteil eines nicht angepaßten statischen Speicherbedarfs. Wird die Tabelle häufig benutzt, was bei Protokollen der Fall ist, entsteht durch die lineare Komplexität ($O(n)$) zur Restrukturierung des Feldes sowohl beim Einsortieren als auch beim Austragen ein hoher Aufwand.

Im Berkeley UNIX wird eine einfache verkettete Liste als Grundstruktur eingesetzt [26]. Diese ist beim Einfügen und Löschen günstiger, da keine bestehenden Einträge verschoben werden müssen. Der lineare Suchaufwand für Starten und Stoppen bleibt jedoch bestehen.

Bei doppelt verketteten Listen und gesonderter Speicherung der Lokation eines Zeitgebers (in Relation mit dessen Identifikation) kann die Routine STOPP_ZEITGEBER unmittelbar, d.h. mit von n unabhängigen Kosten der Ordnung $O(1)$, auf den Wert dieses Zeitgebers innerhalb der Liste zugreifen und ihn - durch entsprechendes Umschreiben der Zeiger von Vorgänger und Nachfolger - austragen, ohne wiederum die ganze Liste durchsuchen zu müssen.

Allgemein bekannte nachteilige Aspekte von Listenstrukturen sind die Verwaltung der Zeiger und der durch die Zeiger erhöhte Speicherbedarf pro Zeitgeberelement. Demgegenüber steht die optimale dynamische Anpassung des Speicherbedarfs an die aktuellen Bedingungen.

Werden absolute Zeitwerte eingesetzt, entfällt gleichfalls die Forderung nach impliziter relativer Anordnung der benutzten Datenstrukturen. Es können daher auch allgemeinere Datenstrukturen wie zum Beispiel der in [60] beschriebene *Heap* eingesetzt werden.

Ein Heap ist ein (binärer) Baum, in dem der Schlüssel eines Eintrags kleiner oder gleich der seiner Nachfolger (Kinder) ist [213]. Ein Heap kann u.a. auch in einem Feld abgelegt werden. Durch geeignete interne Anordnung können dabei sogar die expliziten Verweise auf die jeweiligen Kinder entfallen.

Durch die nach wie vor bestehende innere Ordnung bleibt auch der Aufwand für die PRO_TICK_ROUTINE auf die Bearbeitung genau eines Zeitgeberelements beschränkt. Die beim Einfügen und Austragen benötigte Umstrukturierung eines binären Baumes, also das Auf- und Abwärtsverschieben von Einträgen, hat jedoch allgemein lediglich logarithmische Komplexität. Für die einfachen Start- und Stoppfunktionen ist ein binärer Baum damit insgesamt effizienter. Werden die bei Protokollen typischen Paare von gleichzeitig auszuführenden Start- und Stoppoperationen betrachtet, so weist Wolf in [322] nach, daß geeignet implementierte Verfahren basierend auf doppelt verketteten Listen noch effizienter sind.

Allen angeführten Verfahren von Zeitgeberrealisierungen ist gemeinsam:

1. Sie sind auf Softwareimplementierungen ausgerichtet.
2. Durch die notwendigen Pro-Tick-Unterbrechungsanforderungen ergibt sich entweder eine geringe Zeitgebergranularität oder eine Belastung des Prozessors (siehe 3.2.1).
3. Die Anzahl der Zeitgeber und die Häufigkeit von Änderungen, d.h. Starten und Stoppen, schlagen sich linear in der Prozessorbelastung nieder und sind daher zu beschränken.
4. Es handelt sich bei allen Varianten um allgemeine oder für Betriebssystem- (nicht aber für Kommunikations-) Anwendungen konzipierte Verfahren. Allein durch die in Tabelle 3.3 zusammengefaßten unterschiedlichen Anforderungen ergeben sich zwangsläufig Ineffizienzen für Zeitgeber in Kommunikationssystemen.

Tabelle 3.3: *Unterschiede zwischen betriebssystem- und kommunikationsorientierten Zeitgebern.*

	BS-Zeitgeber	Komm.-Zeitgeber
Anzahl	< 10	> 100
Auflösung	µs	ms
Spanne	s	min
Auftreten	überwiegend linear	völlig ungeordnet
Typisches Verhalten	Ablaufen	Nicht Ablaufen
Primitive	Setzen, Löschen	zus.: Rücksetzen, Neustarten

3.3.3.5 Datenstrukturen

Im vorigen Abschnitt wurde der Einfluß von Datenstrukturen sowie deren Zugriffs- und Verwaltungsverfahren schon ersichtlich. Zeitgeber sind nur ein Beispiel für die Notwendigkeit der strukturierten Organisation von Daten in Kommunikationssystemen. Verschiedene Informationen, die logisch in einem Zusammenhang stehen, werden mittels entsprechend gewählter Datenstrukturen auch physikalisch zusammen abgelegt. Es kann dann eine hierarchische Adressierung vorgenommen werden, zunächst wird das Kriterium des inneren Zusammenhangs adressiert, danach die jeweilige Unterkomponente.

Besonders wichtig wird der Einsatz von Datenstrukturen bei mehrfach vorhandenen Informationen gleicher interner Organisation und Bedeutung. Dies ist insbesondere bei der Verwaltung von Ressourcen aller Art der Fall. Einige Beispiele hierfür sind:

- Elemente einer beliebigen Warteschlange und die Warteschlangen selbst
- Zusammenhängende Speicherbereiche und daraus gebildete größere Strukturen
- Identifikationen und Adressen
- Prozesse, Prozeduren und deren Zustandsinformation
- Medienbandbreite
- Prioritäten

Eine besondere Bedeutung in Kommunikationssystemen haben die Informationen zur Verwaltung von Verbindungen und den aktuellen Zuständen der beteiligten Kommunikationspartner.

In Abschnit 3.2.8 wurden die Realisierungsformen für Datenstrukturen und deren Hauptmerkmale schon eingeführt. Bei den Entwicklern von Software sind dynamische Datenstrukturen, wie z.B. verkettete Listen, beliebt. Deren Vorteil ist die der aktuellen Situation angepaßte Speichernutzung. Die schwierige Frage der optimalen Dimensionierung, welche bei der Verwendung statischer Datenstrukturen während der Programmentwicklung zu beantworten ist, entfällt bei dynamischen Datenstrukturen weitgehend. Zu deren Aufbau, Zugriff und Verwaltung existieren Bibliotheksroutinen. Entscheidendes Kriterium dieser Bibliotheksroutinen ist deren korrekte Funktionalität. Den auf den Inhalt einer Datenstruktur konzentrierten Entwickler interessiert die interne Leistungsfähigkeit solcher Routinen dagegen kaum.

Auf verschiedenen Rechnern durchgeführte Tests haben allerdings gezeigt, daß der (mittlere) Zeitbedarf für das Durchsuchen einer Liste schon bei einer Liste mit weniger als 100 Elementen eine Größenordnung von 100 Mikrosekunden erreichen kann (Bild 3.9)!

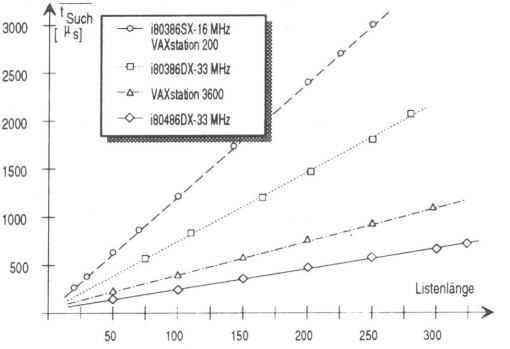


Bild 3.9: *Mittlere Suchzeiten in Abhängigkeit der Listenlängen*

Berücksichtigt man, daß auf einige Datenstrukturen wie z.B. die Verbindungsdaten pro PDU sogar mehrmals zugegriffen werden muß, ergeben sich für eine Betrachtung der gesamten Systemleistungsfähigkeit relevante Größenordnungen! Anders herum gesagt können sich durch eine rein funktionelle "Black-Box"-Betrachtung der entsprechenden Bibliotheksroutinen unerwartete Leistungsengpässe ergeben. Die tatsächlich bestehenden Möglichkeiten zur Leistungssteigerung werden durch die Konzentration auf das korrekte äußere funktionelle Verhalten einer Teilkomponente überhaupt nicht mehr erkannt.

Kapitel **4**

Lösungsansätze

In Kapitel 3 wurden die vielfältigen Funktionen eines Transportsystems sowie einige der Hauptschwachpunkte heutiger Realisierungen beschrieben. Zu jeder einzelnen Funktionalität eines Transportsystems gibt es Vorschläge zu deren Optimierung. Die Ansätze zur Behebung bzw. Minderung der verschiedenen aufgezeigten Schwächen sind jeweils Gegenstand eines der Unterabschnitte dieses Kapitels. Analog zu der Anordnung im vorherigen Kapitel wird auch hier eine strukturierte Unterteilung der Lösungsansätze nach Protokoll- und Implementierungsorientierung vorgenommen.

4.1 Protokollorientierte Ansätze

Arbeiten zu Modellierung und Bewertung sowie daraus resultierender Optimierung von Protokollen bilden einen klassischen Schwerpunkt der Kommunikationstechnik. Neben der Konzentration auf Einzelaspekte bestehender Protokolle war die Entwicklung neuer Protokolle immer schon ein Bereich, dem große Aufmerksamkeit zuteil wurde. Die gesonderte Behandlung von Details bestehender Protokolle einerseits und neuen Protokollvorschlägen andererseits trägt der funktionellen Differenzierung Rechnung.

4.1.1 Protokollinterne Optimierungen

Bereiche, an denen Arbeiten zur Verbesserung von bestehenden Protokollen ansetzen können, sind:

1. Auswahl und Benutzung von Optionen
2. Fixierung von Protokollparametern und
3. Festlegung von einzusetzenden Strategien.

Obwohl die Bereiche eine Beschränkung auf einzelne Schichten assoziieren, sollen auch schichtenübergreifende Betrachtungen der Kategorie der protokollinternen Optimierungen zugerechnet werden. Die Problematik bei der Kombination eines Protokollstapels aus unabhängigen Klassenimplementierungen der einzelnen Protokollschichten wurde schon in Abschnitt 3.3.2 verdeutlicht. Erste Lösungsansätze auf diesem Bereich sind verbindliche Empfehlungen für die zu verwendenden Kombinationen, sogenannte Protokollprofile, von Seiten verschiedener Institutionen (z.B. [148, 302, 304]).

Basierend auf diesen Empfehlungen der nationalen Gremien entstanden weitere verfeinerte Festlegungen für spezifische Aufgabenbereiche. Die zwei bekanntesten Vorschläge für solche Protokolltürme kommen aus den Bereichen Fertigungs- und Büroautomatisierung: MAP (Manufacturing Automation Protocols) von General Motors und TOP (Technical Office Protocols) von Boeing [21]. Beide legen komplette Protokolltürme fest, d.h. es werden sowohl die unteren Schichten als auch Charakteristika der anwendungsbezogenen Schichten oberhalb der Transportschicht festgelegt.

Danthine führt in [121] weitere Vereinigungen und Standardisierungsgremien an, die ebenfalls Empfehlungen für die Kombination von Protokollklassen und -optionen erarbeiten:

- OSITOP, als Übermenge von TOP

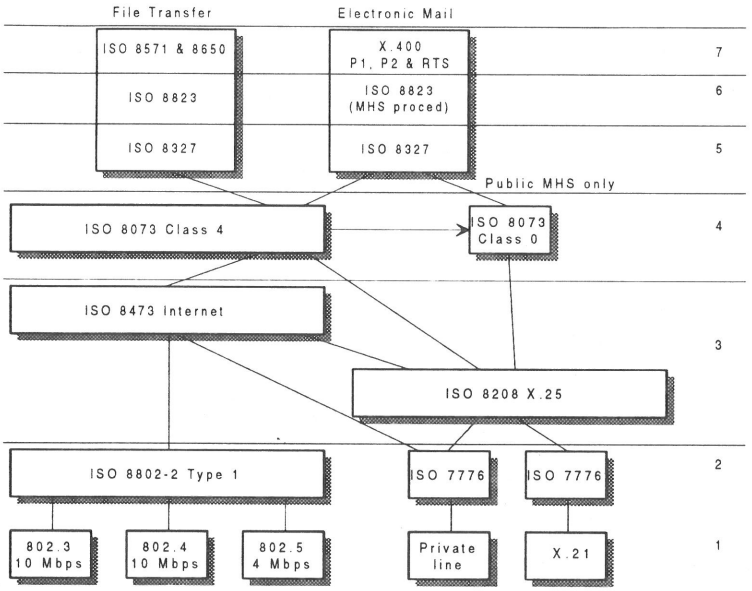


Bild 4.1: *Schichtenkombinationen gemäß COS*

- die Herstellervereinigung SPAG (Standard Promotion and Application Group)
- die nordamerikanische Initiative COS (Corporation for Open Systems), welche Schichtenkombinationen für Dateienübertragung und elektronische Postdienste gemäß Bild 4.1 vorgeschlagen hat
- die europäische CEN-Vereinigung (Centre Européen Normative), deren Ziel europaeinheitliche Standards sind und
- deren japanisches Pendant die POSI-Gruppe (Profile-definitions for OSI-systems).

Alle aufgeführten Gremien streben die Erstellung von sogenannten *funktionellen* Standards an. Gemeinsame Ziele dieser funktionellen Standards sind:

1. Eine Reduzierung der Vielfalt der Kombinationsmöglichkeiten sowie eine Anpassung an bestimmte Randbedingungen.
2. Die frühzeitige kontrollierte Einsatzmöglichkeit von stabilen Standardisierungsvorschlägen.
3. Sowohl Anwendern als auch Protokollentwicklern soll ein sinnvolles Referenzmodell für den möglichen Einsatz der unterschiedlichen Standards gegeben werden.

Die funktionellen Standards sollen eine Form haben, die folgendem Schema folgt: Wenn die Funktionalität X vorhanden sein soll, dann wähle die Standards A,B,C ... mit der angeführten (eingeschränkten und festgelegten) Syntax und Semantik und kombiniere sie in der ebenfalls angeführten Art und Weise.

Bei den unteren 4 Schichten im LAN/MAN-Bereich haben sich nationale Gremien u.a. von Finnland [148], Großbritannien [302] und den Vereinigten Staaten [304] für das folgende OSI-Profil entschieden:

- Die technologieabhängigen Schichten 1 und 2a des OSI-RMs bleiben weitgehend wählbar. Es muß lediglich sichergestellt sein, daß eine Schnittstelle zur darüberliegenden logischen Sicherungsschicht 2b gemäß IEEE 802.1 LLC Standardisierung [369, 370] eingehalten wird.
- Auf der Transportschicht 4 wurde die Benutzung der leistungsfähigsten Protokollklasse TP4 gewählt. Damit werden zum einen die unteren Schichten funktionell entlastet und können daher

(eher) mit den hohen nutzbaren Bandbreiten der physikalischen Schicht Schritt halten. Zum anderen wird man durch diese Wahl unabhängig von den Eigenschaften und der Topologie des unterliegenden Netzwerkes, braucht also weder exaktes Wissen darüber zu haben, noch müssen Vermutungen und Annahmen getroffen werden.

- Als Folge der Verwendung von TP4 kann für die Sicherungsschicht 2b die Protokollklasse vom Typ 1 empfohlen werden, welche lediglich einen unquittierten und verbindungslosen Datagrammdienst realisiert.
- Auch für die Vermittlungsschicht kann durch die Verwendung von TP4 auf den aufwendigeren verbindungsorientierten Dienst gemäß [359] verzichtet werden. Stattdessen ist der verbindungslose Dienst [360] (English: ConnectionLess Network Service, CLNS) Bestandteil der Profile.

Trotz der Festlegung solcher Profile bleiben immer noch Spielräume innerhalb der beteiligten Protokollschichten. Einige der schichteninternen Parameter- und Strategieoptimierungen werden nachfolgend getrennt nach Schicht behandelt, wobei die sieben Schichten des OSI-RMs gemäß dem in Abschnitt 2.3.1 eingeführten dreischichtigen Modell zusammengefaßt werden.

4.1.1.1 Technologieabhängige Schichten

Innerhalb der Nachrichten- und Kommunikationstechnik wurde und wird traditionell den unteren Schichten des OSI-RMs, bedingt durch ihre enge Verknüpfung mit der Technologie, überproportional starke Aufmerksamkeit gewidmet. Das stetige Verschieben der technologischen Machbarkeitsgrenze war dabei lange Zeit einziges Ziel. Heute bestimmen zunehmend auch wirtschaftliche Betrachtungen die Forschungsaktivitäten auf diesem Gebiet. Bei den derzeitigen Hochtechnologieprojekten sind die Entwicklungen von öffentlichen wie privaten Netzwerken mit Bandbreiten über 1 Gbit/s zu nennen. Herausragende Vertreter im privaten Bereich sind die innerhalb der ANSI Arbeitsgruppe FFOL (FDDI Follow On LAN, [231, 262]) diskutierten Vorschläge:

- ATMR (ATM-Ring) der NTT Japan [55, 189]
- CRMA bzw. CRMA-II (Cyclic Reservation Multiple Access) des Forschungslaboratoriums der IBM in Rüschlikon, Schweiz [53, 332] und
- MetaRing / -Net des IBM-Forschungszentrums Yorktown Heights, USA [99].

Der japanische ATM-Ring wurde, ebenso wie CRMA, bereits als lauffähiger Prototyp der Öffentlichkeit präsentiert. Für MetaRing ist ein solcher Prototyp angekündigt. Im Rahmen dieser Arbeit wird nicht näher auf Eigenschaften der drei Netztypen eingegangen, entsprechende Informationen und Literaturhinweise sind z.B. in [281] enthalten. Festzuhalten ist, daß alle drei Vorschläge auf dem, von heutigen lokalen Netzwerken her bekannten, Prinzip des gemeinsam genutzten Mediums basieren. Es ergibt sich damit der Nachteil, daß jede Anschlußstelle eines dieser Netze mit der schnellen und damit teuren Netzankopplungstechnik ausgestattet sein muß, unabhängig davon, ob diese Station tatsächlich die volle Medienbandbreite benötigt oder nicht.

Im öffentlichen Bereich dominieren die Begriffe B-ISDN und ATM [13, 32]. Um Erfahrungen mit der Komplexität von Netzen hoher Bandbreite, den dafür geeigneten Anwendungen und dem Benutzerverhalten zu erhalten, werden international Feldversuche durchgeführt. Ein Überblick der US-amerikanischen Gigabit-Testprojekte Aurora, Blanca, Casa, Nectar und Vistanet ist in [160] enthalten. Es soll nicht unerwähnt bleiben, daß die Einführung der weltweit einheitlichen synchronen digitalen Hierarchie SDH (Erläuterungen siehe [275]) sowie die Verfügbarkeit entsprechend leistungsfähiger (optischer) Übertragungstechnik [72] die Grundvoraussetzung aller Aktivitäten im öffentlichen Bereich ist.

Neben den Arbeiten an Netzen der übernächsten Generation werden auch die in Abschnitt 2.1.2 aufgeführten heutigen HSLANs und MANs weiterentwickelt und optimiert. Möglichkeiten zur Optimierung der Medienzugriffsprotokolle ergeben sich u.a. durch Aushandeln bzw. Festlegung von Protokollpara-

metern, Ausnutzung von teilweise vorhandenen unterschiedlichen Prioritätsmechanismen oder die Wahl der benutzen Verkehrsklasse, falls mehrere vorhanden sind. Bei FDDI sind z.B. sowohl mehrere Prioritäten und Betriebsmodi als auch unterschiedlich zu behandelnde Verkehrsklassen vorgesehen (siehe FDDI-Standards [375 - 380]). Darüber hinaus kann durch die Wahl von - für das Protokoll relevanten - Zeitgeberwerten die Dienstgüte und das Leistungsverhalten beeinflußt werden. Umfangreiche simulative und analytische Untersuchungen zu Optimierungen bei FDDI und anderen aktuellen Medienzugangsprotokollen aus dem HS-LAN/MAN-Bereich wurden u.a. von Tangemann [297] gemacht.

Aus der Vielzahl von Arbeiten auf dem Gebiet der MAC-internen Optimierungen soll hier nur die Integration von Rechnerunterstützung bei den notwendigen - teilweise adaptiven - Entscheidungsprozessen besonders erwähnt werden. Dazu werden die grundlegenden Erkenntnisse über Auswirkungen von bestimmten Umgebungsbedingungen, welche durch die vielfältigen Protokolluntersuchungen erworben wurden, in Regeln für Expertensysteme [19] umgesetzt. Je nach Auslegung gibt ein solches Expertensystem dem Benutzer lediglich unverbindliche Ratschläge zur Wahl optimierter Parameter und Verfahren, oder das System kann selbst aktiv Änderungen veranlassen. Zwangsläufige Voraussetzungen für eine dynamische Anpassung an geänderte Bedingungen sind ausreichend präzise Informationsgeber [317, 318] und intelligente Diagnosewerkzeuge [273] mit umfassendem Regelwerk. Eine Interaktion mit dem Netzwerkmanagement und den Schichtenmanagementinstanzen ist ebenfalls erforderlich.

In [285] werden die wesentlichen Merkmale von Protokollen für die Bitübertragungsschicht angegeben. Neben der Leitungsbitrate sind insbesondere wichtig (Tabelle 4.1):

- die Leitungscodierung (gute Taktrückgewinnung mit wenig Redundanz),
- die Art der Takt- und Rahmensynchronisation,
- die kleinste manipulierbare Einheit sowie
- Fehlertoleranzeigenschaften.

Lösungsansätze, welche - abweichend von dem üblichen Bestreben nach maximaler Leistungsfähigkeit - auch wirtschaftliche Gesichtspunkte berücksichtigen, gewinnen zunehmend an Bedeutung. Hauptansatzpunkt sind die hohen Kosten für eine Ankopplung an Netze mit gemeinsamem Medium und großer Bandbreite. Diese Kosten fallen bekanntermaßen unabhängig von den tatsächlichen Bedürfnissen des oder der

Tabelle 4.1: *Wichtige Merkmale von Protokollen der Bitübertragungsschicht*

	Medienbitrate Leitungscode	Taktsynchronisation	Rahmensynchronisation	Bitmanipulation	Fehlertoleranz
FDDI [261]	125 Mbit/s 4B5B NRZI	plesiochron	eindeutige 10-Bit Kennung	Symbolpaare je 4 (5) Bit	Doppelring "Wrapping"
DQDB (SONET) [126]	155 Mbit/s Scrambling	plesiochron	nicht eindeut. 48-Bit Kennung	Oktett	Doppelbus (phys. Ring)
CBN [165, 183]	600 Mbit/s 4B5B NRZI	plesiochron	eindeutige 5-Bit Kennung	Oktett	Umgehung von Fehlern (Bypass)
Metrocore [48]	150 Mbit/s 5B6B PMSI	synchron	eindeutige 4-Bit Kennung	einzelne Bits	Doppelbus (phys. Ring)
LION [51]	636 Mbit/s 8B1C	plesiochron	8 eindeutige 27-Bit Kennungen	keine	Doppelbus (phys. Ring)
TDM-Loop [226]	100 Mbit/s 9B1C Scramble	synchron	eindeutige 10-Bit Kennung	keine	Doppelring "Wrapping"
Hyperchannel [154]	100 Mbit/s 4B5B NRZI	plesiochron	eindeutige 10-Bit Kennung	keine	Doppelbus (phys. Ring)

an dieser Station angeschlossenen Endbenutzer(s) an.

Durch die Fortschritte der Übertragungstechnologien verlieren einige der ursprünglichen Argumente für ein gemeinsam benutztes Medium an Bedeutung. So sind beispielsweise die Kosten für Hochgeschwindigkeitsübertragungsstrecken dramatisch gefallen, während gleichzeitig die erreichbaren Distanz-Bandbreitenprodukte moderner Techniken, welche ein kombiniertes Maß der Nutzbandbreite und der Dämpfung darstellt, stark gestiegen sind [72]. Interessanterweise führt dies teilweise zu Vorschlägen, die auch schon früher einmal aktuell waren [281]:

1. Punkt-zu-Punkt-Verbindung der Teilnehmer durch echte Vollvermaschung oder Teilvermaschung in Kombination mit indirekten Multi-Hop Wegen vom Sender zum Empfänger.
2. Kommunikation über einen zentralen - z.B. ATM basierten - Vermittlungsknoten mit hoher kumulierter Bandbreite, an den die Teilnehmer jeweils über Punkt-zu-Punkt-Leitungen sternförmig angeschlossen sind.
3. Vervielfältigung und entsprechende Parallelisierung bestehender Netze [137, 205], was u.U. jedoch Auswirkungen auch auf höhere Schichten nach sich ziehen kann.
4. Hierarchische Netze, deren innere Strukturen nicht durch räumliche Gegebenheiten vorgegeben sind, sondern durch das jeweils optimale Kosten-Nutzen-Verhältnis des Teilnehmeranschlusses (dies kann dazu führen, daß statt eines direkten Anschlusses an das im Nebenraum zugängliche Hochgeschwindigkeitsnetz eine indirekte Ankopplung über einen räumlich entfernten Konzentrator und ein dazwischenliegendes MAN erfolgen kann).
5. Der gestiegene Integrationsgrad läßt auch programmierbare Bausteine für mehrere verschiedene technologieabhängige Schichten zu (z.B. [98, 237]). Natürlich nur dann, wenn man die aktuellen Hochtechnologieprojekte von der Universalität ausnimmt und sich auf die Netze der darunterliegenden Leistungsklasse konzentriert.

Alle angeführten Aktivitäten aus dem Bereich der technologieabhängigen Schichten 1 und 2a des OSI-RMs machen deutlich, daß dieser Bereich auch in absehbarer Zukunft keine Begrenzung für das Leistungsverhalten eines Kommunikationssystems sein wird. Man kann sogar mit einer gewissen Zuversicht davon ausgehen, daß leistungsfähige Bausteine für die unteren Schichten jeweils rechtzeitig bei entsprechend gestiegenem Leistungsvermögen der höheren Schichten zu akzeptablen Preisen allgemein verfügbar sind bzw. auch in Zukunft sein werden. Eine detaillierte Betrachtung der angeführten Lösungsansätze kann daher im Rahmen dieser Arbeit entfallen.

4.1.1.2 Transportorientierte Schichten

Klammert man das umfangreiche und daher weitgehend eigenständige Teilgebiet der Wegewahl und alle damit zusammenhängenden Fragen einmal aus, so ergibt sich die folgende Liste von Teilbereichen, welche Bestandteil von Optimierungsbestrebungen sind:

- Überwachung des Datenflusses
- Fehlererkennung und -behebung
- Auf- und Abbau sowie Verwaltung von Verbindungen
- Einsatz verschiedener Protokollmechanismen und -optionen

In diesem Abschnitt werden nur solche Lösungsansätze berücksichtigt, welche im Rahmen bestehender OSI-Standards ausführbar sind. Darüberhinausgehende Ansätze, die entweder akademischen Charakter haben oder zu völlig neuen Protokollen führen, werden in 4.1.2 aufgeführt.

Die Standards für die logische Sicherungsschicht 2b im Bereich von LANs [369, 370] beschreiben zwei grundsätzliche Klassen von möglichen Diensten:

1. einen verbindungslosen Datagrammdienst und
2. einen verbindungsorientierten Dienst.

Beim verbindungslosen Dienst fällt naturgemäß der Bereich der Verbindungsverwaltung weg. Die Überwachung des Datenflusses und eine Fehlerkorrektur durch wiederholtes Aussenden von Dateneinheiten setzen ebenfalls das Bestehen einer Verbindung voraus. Durch eine weitgehende Festlegung der PDU-Formate und Steuerkopfinhalte sind beim verbindunglosen Dienst der Schicht 2b somit keine nennenswerten (funktionsbezogenen) Optimierungen möglich.

Innerhalb des verbindungsorientierten Dienstes können dagegen zahlreiche Freiheiten zur Optimierung ausgenutzt werden. Ein Großteil der möglichen Ansätze ist allerdings mit denen der Transportschicht nahezu identisch, so daß auf eine gesonderte Betrachtung verzichtet werden kann.

Auch in der Vermittlungsschicht gibt es die zwei - vom notwendigen Protokollaufwand her gesehen - grundsätzlich verschiedenen Formen des verbindungslosen CLNS [360] und verbindungsorientierten Dienstes CONS [359] . Allerdings ergeben sich im Unterschied zum verbindungslosen Dienst der Sicherungsschicht beim CLNS durch Optionen Möglichkeiten für Optimierungsüberlegungen. Ein Blick auf den Aufbau eines Steuerkopfes der Vermittlungsschicht (Bild 4.2) verdeutlicht diesen Sachverhalt.

Zusätzlich zu verschiedenen möglichen Adreßformaten und fest vorgegebenen Teilen des Schicht-3-Steuerkopfes ist auch ein optionaler Teil variabler Länge vorgesehen. Ein Großteil der erlaubten Optionen ist allerdings im Umfeld eines verbindungslosen Dienstes wenig sinnvoll - dies gilt speziell für Optionen zur Dienstgüte - oder ist in ihrer Bedeutung und Nutzung nicht ausreichend festgelegt. Klammert man zusätzlich Fragen der Leitwegbestimmung und Segmentierung (siehe Abschnitt 5.2.1.3) aus, verringert sich die Zahl der Optionen so weit, daß Überlegungen zu deren optimierter Behandlung obsolet werden.

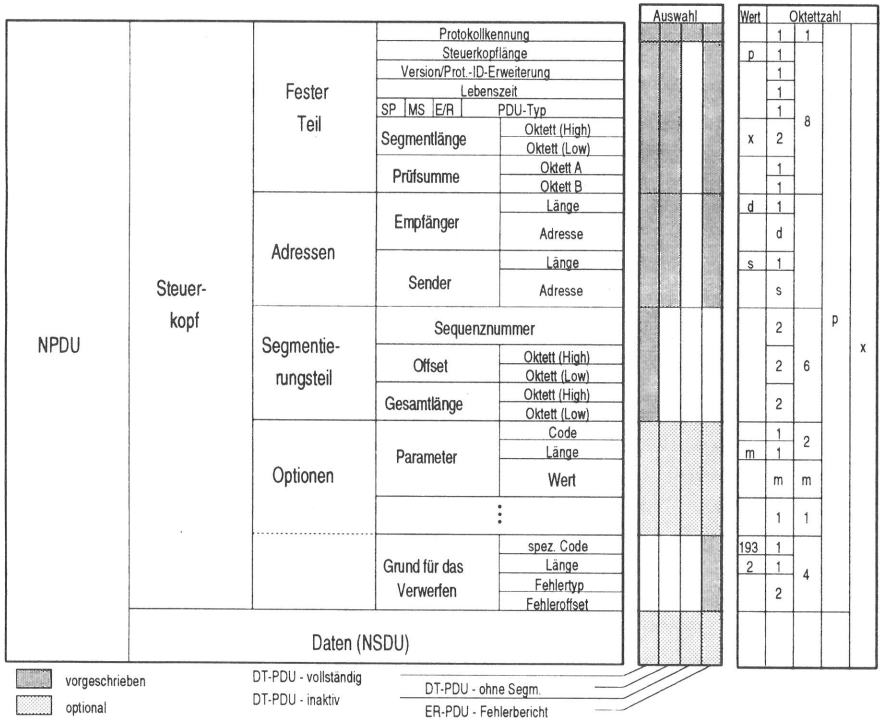


Bild 4.2: *Aufbau des Steuerkopfes der Vermittlungsschicht*

Als wesentliche Aufgabe der Schicht 3 verbleibt neben der Realisierung des im Standard [360] beschriebenen Dienstes und der Kommunikation mit den Nachbarschichten die Erkennung, Behandlung und eventuelle Umsetzung von Adressen. Durch eine Implementierung sowohl in der Hochsprache C als auch auf Maschinenebene [258] wurde nachgewiesen, daß die eigentliche CLNP-Protokollfunktionalität eine sehr geringe Komplexität besitzt, welche bei Weitem vom Aufwand für das Absuchen von Adreßlisten überwogen wird. Auf eine eingehende Beschreibung der Schicht-3-Funktionen wird daher verzichtet.

Durch die Fokussierung auf Endsysteme kann die umfangreiche Problematik der Kopplung von Netzen (z.B. [75]) im Rahmen dieser Arbeit ebenfalls unberücksichtigt bleiben. Analoges gilt - bei entsprechend gewählten Szenarien mit intelligenten Koppelstationen - auch für die Mechanismen zur Wegewahl.

Die OSI-Standards für die Transportschicht [358, 355] legen die zu benutzenden Verfahren für Verbindungsauf- und -abbau fest: um Problemen mit noch im Netzwerk vorhandenen Irrläufersegmenten früherer Verbindungen weitgehend ausschließen zu können, gilt eine Verbindung erst dann als erfolgreich aufgebaut, wenn der Erhalt der Quittierung des Verbindungsaufbauwunsches beim Zielpartner ankommt. Bevor eine Verbindung besteht, müssen also drei Meldungen über das Netzwerk ausgetauscht werden:

1. Verbindungsaufbauwunsch von A zu B. Als Information sind neben den notwendigen Adressen und Identifikationen auch Vorschläge zu angestrebten Dienstklassen, Optionen und Leistungsmerkmalen darin enthalten.
2. Quittierung des Verbindungsaufbauwunsches mit u.U. reduzierten Optionen oder Dienstgüteforderungen von B zurück an A.
3. Bestätigung, daß tatsächlich ein Verbindungswunsch von A wieder zu B besteht. In dieser dritten Meldung sind die nun endgültigen Verbindungsparameter enthalten.

Der von Watson schon 1981 gemachte Vorschlag - der durch zusätzliche Zeitüberwachungsmechanismen die Effizienz steigert [311] - kann daher in OSI-Systemen nicht angewandt werden. In einigen alternativen Protokollen wird er allerdings berücksichtigt und auch weiterentwickelt [100].

Möglich sind Überlegungen zu langlebigen Verbindungen, die nicht unmittelbar nach Beendigung eines u.U. kurzen Datentransfers abgebaut bzw. vor jeder Datenübertragung aufgebaut werden müssen. Der Aufwand für das Aufrechterhalten von nicht mehr aktiven Verbindungen und die zusätzlich notwendigen Systemressourcen müssen dabei dem erzielbaren Zeitgewinn und den eingesparten Verbindungsauf- und abbauten gegenübergestellt werden. Die Tarifierung kann u.U. auch eine Rolle spielen (siehe 3.3.1.4)

Die Wahl von Optionen und Protokollmechanismen kann die Leistungsfähigkeit einer Protokollinstanz zwar beeinträchtigen, wurde jedoch bisher nur vereinzelt wissenschaftlich untersucht. Eine Ausnahme bildet die Arbeit von Colella, Aronoff und Mills [113], die den Einsatz des Mechanismus Bevorzugte Datenübertragung (expedited data transfer) untersucht und mögliche Optimierungen aufzeigt.

Betrachtet man bestehende Protokollimplementierungen (im LAN-Umfeld) genauer und berücksichtigt die oben angeführten funktionellen Standards, so ergibt sich, daß nur ein sehr kleiner Anteil der von den OSI-Standards vorgesehenen Optionen und Mechanismen sinnvoll angewandt werden kann bzw. tatsächlich eingesetzt wird. Durch eine Einbeziehung der Funktionalitäten der Nachbarschichten und der jeweiligen Randbedingungen verlieren bestimmte Optionen und Mechanismen der einzelnen Schichten ihren Sinn und können daher, ohne funktionelle Einschränkungen zu bewirken, unberücksichtigt bleiben.

Einige Protokollparameter bieten sich dagegen geradezu für Untersuchungen zu deren Optimierung an. Besonders beliebt sind Werte von Zeitgebern (etwa in [139, 193, 194, 204, 326]) und Fenstergröße [309], wobei sich die Fenstergröße weniger für eine isolierte Betrachtung eignet (siehe nächste Abschnitte).

Die Bereiche Erkennen und Beheben von Fehlern sowie Datenflußsteuerung dominieren, wie auch aus der Anzahl der Veröffentlichungen mit zugehörigen Titeln zu entnehmen ist (u.a. [, 146, 192, 197, 276], klar. In diesen Bereichen lassen die Standards auch einen Spielraum.

Die hier im Wesentlichen interessierende Protokollklasse 4 der OSI-Standards [358, 223] für die Transportschicht setzt einen Fenstermechanismus zur Datenflußsteuerung ein. Ein Sender darf bis zu W - wobei W der Fenstergröße entspricht - Transport-PDUs (TPDUs) abschicken, ohne auf Quittierungen warten zu müssen. Der Empfänger schickt dem Sender spezielle Quittierungs-PDUs (ACK-TPDUs), die sowohl die Folgenummer der ersten ausstehenden Daten-TPDU als auch eine Information zur Aktualisierung des Sendefensters (Credits) enthalten. Zur Erhöhung der Fehlertoleranz ist bewußt nicht gefordert, daß der Empfang jeder TPDU unverzüglich mit einer eigenen ACK-TPDU quittiert werden muß. Für den Empfänger ergeben sich daraus die in Tabelle 4.2 aufgeführten Empfangsstrategien.

Tabelle 4.2: *Vorgehensweisen beim Erhalt einer Daten-TPDU*

A) TPDU ist reihenfolgerichtig:
1. TPDU wird gespeichert, Quittung wird sofort abgeschickt
2. TPDU wird gespeichert, Quittung wird erst nach n TPDUs gesendet

B) TPDU ist reihenfolgegestört
1. TPDU wird verworfen und
   nichts weiter wird unternommen oder
   eine Quittung der letzten korrekten TPDU wird abgeschickt
2. TPDU wird (zwischen-)gespeichert und
   nichts weiter wird unternommen oder
   eine Quittung der letzten korrekten TPDU wird abgeschickt

Desweiteren enthält TP4 (gemäß dem ursprünglichen ISO-Standard [358], Erweiterungen sind jedoch in Vorbereitung [113, 122, 358]) im Gegensatz zu anderen Protokollen keine Mechanismen zum Anfordern ausstehender oder fehlerhafter TPDUs. Erneute Aussendungen von TPDUs können ausschließlich von ablaufenden Zeitgebern - den lokalen Wiederholungszeitgebern - auf der Sendeseite ausgelöst werden. Der Standard [358] verlangt nicht explizit das Vorhandensein eines Zeitgebers pro TPDU. Ist die Zeit zwischen zwei aufeinanderfolgenden TPDUs kürzer als die doppelte Laufzeit (Laufzeit = Zeit um eine Meldung vom Sender zum Empfänger zu übertragen), so ist in einem Fehlerfall das Ablaufen von Zeitgebern folgender u.U. korrekt übertragener TPDUs nicht zu verhindern. Die Reduktion der Zeitgeberanzahl bis herunter zu einem einzigen Wiederholungszeitgeber ist daher in schnellen Netzen durchaus ohne größere funktionelle Einbußen möglich. Aus dem Gesagten lassen sich die in Tabelle 4.3 zusammengefaßten Möglichkeiten ableiten.

Tabelle 4.3: *Senderverhalten beim Ablauf eines Zeitgebers*

A) Ein einzelner Zeitgeber:
1. Exakt eine TPDU wird wiederholt, Zeitgeber zurücksetzen
2. Wiederholung aller noch unquittierten TPDUs, Zeitgeber zurücksetzen

B) Ein Zeitgeber pro TPDU
1. Wiederholung der zugehörigen TPDU, nur deren Zeitgeber wird zurückgesetzt
2. Wiederholung der zugehörigen TPDU, Zurücksetzen der Zeitgeber aller TPDUs

In heterogenen Kommunikationssystemen kann prinzipiell jede beliebige Kombination der Empfangs- und Sendestrategien auftreten. Meister berichtet in [223] über Simulationsergebnisse einiger der möglichen Kombinationen. Randbedingungen der Simulationen sind eine relativ hohe TPDU-Verlustwahrscheinlichkeit bis zu 0.05, Bearbeitungszeiten des Transportsystems (Schichten 1 - 4) von knapp 10 ms, Übertragungsverzögerungen von 20 ms (entspricht etwa 4000 km Entfernung) und Verbindungsstrecken mit Bandbreiten von 64 Kbit/s bzw. 4 Mbit/s. Die alleinig interessierende Größe ist der erzielbare effektive (Nutzdaten-)Durchsatz, der oberhalb der Transportschicht sichtbar ist.

Obwohl sich einige Überlegungen von Meisters Untersuchungen auf Systeme mit anderen Randbedingungen übertragen lassen, ergab sich durch die doch erheblich abweichenden Systemparameter die Notwendigkeit, eigene Untersuchungen durchzuführen. Diese werden nachfolgend detailliert beschrieben.

Wahl der Sende- und Empfangsstrategie

Bei der Wahl der Strategie werden folgende Ziele verfolgt:

Z1) Zu jeder in Frage kommenden Transportbitrate ist ein Sendefenster zu ermöglichen, das im Falle fehlerfreier Übertragung einen kontinuierlichen Sendefluß garantiert.

Z2) Bei Verlusten von Daten-PDUs müssen die Sende- und die Empfangsstrategie verhindern, daß ein zu hoher Anteil der übertragenen PDUs wiederholt wird.

Z3) Bei PDU-Verlusten darf keine übermäßige Verlangsamung der Kommunikation eintreten.

Z4) Die nötigen Vorkehrungen beim Sender und Empfänger dürfen für den Fall fehlerfreier Übertragung nur einen unwesentlichen Mehraufwand bewirken.

Der Untersuchung liegen folgende Festlegungen und Dimensionierungsannahmen zugrunde:

A1) Es sind Transportbitraten von bis zu 100 Mbit/s erlaubt. Bei TPDUs von ca. 1000 Oktetts Länge entspricht dies einer TPDU-Rate von $R_T = 12500$ TPDUs pro Sekunde.

A2) Es wird eine TPDU-Verlustwahrscheinlichkeit p_L von $5 \bullet 10^{-4}$ angenommen. TPDU-Verluste aufgrund von fehlerhaft ankommenden TPDUs seien in p_L berücksichtigt. Da Pufferüberläufe die Hauptursache für TPDU-Verluste darstellen, wird p_L als von der TPDU-Länge unabhängig angenommen.

A3) Die mittlere Antwortzeit t_r zwischen Senden einer TPDU und dem Erhalt einer Quittung für diese TPDU vom Empfänger wird zu 40 ms angenommen

A4) Die Größe des Sendefensters ist durch die Zeit t_{rmax} bestimmt. Die Wahl von t_{rmax} hat sicherzustellen, daß für die zuletzt gesendete Daten-TPDU eine Quittung innerhalb dieser Zeit mit ausreichend hoher Wahrscheinlichkeit beim Sender ankommt. t_{rmax} ist durch t_r nach unten begrenzt. Über den Unterschied von t_{rmax} und t_r entscheidet das Verhalten des Netzes. Nur für ein Netz, das leicht schwankende Antwortzeiten garantiert, wird der Unterschied gering sein. Für die folgenden Betrachtungen wird $t_{rmax} = 1.5 \bullet t_r = 60$ ms angenommen.

A5) TPDU-Verluste in TP4 stellt der Sender mittels einer Zeitüberwachung fest. Der Zeitwert t_O ist immer durch die Zeit t_r und in der Regel durch die Zeit t_{rmax} nach unten begrenzt.

Bewertung verschiedener Strategien

Zur Abschätzung des Anteils O der TPDU-Wiederholungen an der Gesamtzahl übertragener TPDUs (zusätzlicher Übertragungsaufwand) werden 6 Szenarien mit zugehörigen Strategien betrachtet. Aussagen werden durch Mittelwertbetrachtungen gewonnen. Hierzu wird folgendes angenommen:

- Mehrfachwiederholungen sind unwahrscheinlich. In den Fällen, in denen dies nicht realistisch erscheint, werden die Ergebnisse als untere Grenze für den zusätzlichen Übertragungsaufwand interpretiert.
- Der Transportfluß wird als gleichmäßig vorausgesetzt, sofern nicht ausdrücklich eine abweichende Annahme gemacht wird.
- TPDU-Verluste in den Szenarien 1 bis 5 sind unabhängig voneinander. Büschelförmige (bursty) Fehler werden in Szenario 6 berücksichtigt.

Folgende Überlegungen gelten im Falle unabhängiger TPDU-Verluste:

Nach Auftreten eines TPDU-Verlusts muß der Sender die normale Sendesequenz unterbrechen und TPDUs wiederholen. Eine solche Rückwärtsbewegung in der Sendesequenz wird als Stockung bezeichnet. Da nicht jeder Verlust eine Stockung verursacht, ist die Zahl der Stockungen nicht identisch mit der Zahl verlorener TPDUs. Vielmehr können mehrere verlorene TPDUs im Verlauf einer Stockung erneut übertragen und vom Empfänger richtig empfangen werden.

Sei N die Zahl der ursprünglich zu übertragenden TPDUs, $V = N \bullet p_L$ die Zahl der TPDU-Verluste, S die Zahl der Stockungen sowie R die Zahl der pro Stockung durch Wiederübertragung revidierten TPDU-

Verluste. Folgender Zusammenhang ist dann gültig:

$$S \bullet R = V \qquad (4.1) \qquad\qquad S = \frac{p_L N}{R} \qquad (4.2)$$

Strategie 1) Empfangsfenster $W_R = 1$, Sendefenster $W_S = R_T \bullet t_{rmax}$. Das Sendefenster wird durch den Empfänger nie verkleinert. In diesem Fall werden beim Empfänger *alle* TPDUs verworfen, die nicht in der richtigen Sequenz ankommen. Der Sender wird über notwendige Wiederübertragung durch das Ablaufen eines Zeitgebers informiert, zu einem Zeitpunkt, zu dem bereits alle TPDUs innerhalb des bestehenden Sendefensters W_S ausgesandt wurden. Die Wiederholung dieser TPDUs bestimmt den zusätzlich notwendigen Übertragungsaufwand. Quantitativ läßt er sich wie folgt abschätzen: Sei L die Anzahl verlorener TPDUs innerhalb einer Gruppe von W_S TPDUs, die mit einer verlorenen TPDU beginnt:

$$L = 1 + p_L(W_S - 1) \approx 1 + p_L W_S \qquad \text{für } W_S \text{ groß}, p_L \text{ klein} \qquad (4.3)$$

Dann folgt für S mit (4.2) daraus:

$$S = \frac{p_L N}{1 + p_L W_S} \qquad (4.4)$$

Da jede Stockung die Wiederübertragung von G TPDUs erforderlich macht, ergibt sich der durch die ein zweites Mal zu übertragenden TPDUs normierte Zusatzaufwand O zu:

$$O = \frac{S W_S}{N} = \frac{p_L W_S}{1 + p_L W_S} \qquad (4.5)$$

Für das gegebene Dimensionierungsbeispiel erhält man: $O = 0.27$.
Dieser Wert ist erheblich, so daß die Annahme von vernachlässigbaren Mehrfachwiederholungen hier unrealistisch ist. Der Wert kann nur als untere Grenze für den zusätzlichen Übertragungsaufwand benutzt werden. Diese Grenze sagt aus, daß für die gegebene Dimensionierung die Strategie 1 ungeeignet ist.

Es sei angemerkt, daß heutige Transportsysteme häufig mit einem weit geringeren Produkt $p_L \bullet W_S$ zurechtkommen müssen. Für diese Systeme liegt der Wert W entsprechend niedriger. Die einfache Sende- und Empfangsprozedur dieser Strategie kann daher in solchen Fällen ausreichen.

Strategie 2) Empfangsfenster W_R = Sendefenster $W_S = R_T \bullet t_{rmax}$. Der Sender unterhält zu jeder gesendeten Daten-TPDU einen Zeitgeber, bei dessen Ablauf nur die zugehörige TPDU wiederholt wird. Das Sendefenster wird nie verkleinert. In diesem Fall verwirft der Empfänger keine richtig empfangenen TPDUs. Der Zusatzaufwand kommt vielmehr dadurch zustande, daß nach dem Ablauf eines Zeitgebers für eine verlorengegangene TPDU die Zeitgeber unmittelbar folgender TPDUs ebenfalls ablaufen. Erst nach Wiederübertragung der verlorenen TPDU und der Zeit t_r erhält der Sender eine Quittung, die das unnötige Ablaufen weiterer Zeitgeber verhindert. Bild 4.3 zeigt den beschriebenen Fall:

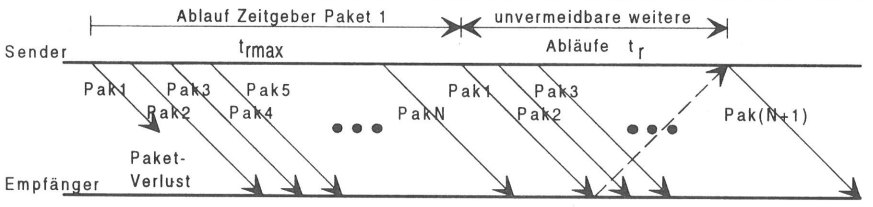


Bild 4.3: *Ablaufdiagramm der 2. Strategie*

Im Fehlerfall (Stockung) wiederholt der Sender $R_T \bullet t_r$ TPDUs. Pro Stockung werden $R = 1 + p_L \bullet R_T \bullet t_r$ zuvor eingetretene TPDU-Verluste durch Wiederübertragung revidiert. Damit gilt:

$$S = \frac{p_L N}{1 + p_L R_T t_r} \qquad (4.6)$$

Der Anteil des Zusatzaufwandes O ist gegeben durch:

$$O = \frac{S t_r R_T}{N} = \frac{p_L R_T t_r}{1 + p_L R_T t_r} \tag{4.7}$$

Anders als in Strategie 1 sind weitere Wiederholungen unwahrscheinlich. Der Empfänger hat bis auf $p_L \bullet N$ TPDUs alle anderen gespeichert. Der Sender sendet alle diese TPDUs erneut. Die Anzahl der TPDUs, die weder bei der ersten noch bei der zweiten Übertragung richtig beim Empfänger ankommen, beläuft sich im Mittel auf $p_L \bullet p_L \bullet N$. Bei der gegebenen Dimensionierung ist dieser Wert und damit die Auswirkung von Mehrfachwiederholungen vernachlässigbar. Der zusätzlich verursachte Übertragungsaufwand ist im wesentlichen durch W entsprechend oben angeführter Gleichung gegeben. Worin

$$p_L R_T t_r = 5 \bullet 10^{-4} \bullet 12.500 \frac{\text{TPDU}}{\text{s}} \bullet 40\,\text{ms} = 0.25 \tag{4.8}$$

ist und O mit (4.7) zu 0.25/1.25 = 0.2 wird; ein Wert, der als nicht akzeptabel erscheint.

Strategie 3) Strategie 2 wird betrachtet mit dem Unterschied, daß nur ein Sendezeitgeber (wie [223]) vorhanden ist, bei dessen Ablauf alle Daten-TPDUs wiederholt werden. Zwei Fälle werden unterschieden:

S3A) Der Transportfluß bei Wiederholungen ist gleichmäßig, d.h in der Regel werden TPDUs nicht büschelförmig (bursty) übertragen. In diesem Fall ist der Zusatzaufwand durch den in Strategie 2 berechneten Ausdruck gegeben. Die Verwendung eines Zeitgebers pro gesendeter TPDU bringt hier keinen Vorteil.

S3B) Die Wiederholung von TPDUs findet häufig büschelförmig statt. In diesem Fall gelten die Überlegungen von Strategie 2 mit dem Unterschied, daß im ungünstigsten Fall S Gruppen von je W_S (statt L) TPDUs erneut übertragen werden müssen. Dies führt zu:

$$S = \frac{p_L N}{1 + p_L W_S} \tag{4.9}$$

$$O = \frac{p_L W_S}{1 + p_L W_S} = 0.27 \tag{4.10}$$

Bemerkenswert ist, daß die Verwendung von einem einzigen Sendezeitgeber pro TPDU im Fall S3A keine und im Fall S3B maximal eine Ersparnis von (127-120)/120=5.8% bei der Anzahl der zu übertragenden TPDUs bringt. Allgemein ist das Ausmaß der möglichen Ersparnis vom Unterschied der Zeiten t_r und t_{rmax} abhängig. Nur bei einem erheblichen Unterschied kann sich die Verwendung eines Sendezeitgebers pro TPDU merklich auswirken!

Strategie 4) $W_R = R_T \bullet t_{rmax}$. Die Größe des Sendefensters wird vom Empfänger gesteuert. Im Normalfall ist $W_S = R_T \bullet t_{rmax}$. Nach Auftreten einer Lücke im Empfangsstrom reduziert der Empfänger das Sendefenster auf die Größe der Empfangslücke. Es wird nur ein Sendezeitgeber verwendet.

Der Sender stellt nach Erhalt der Quittung zur Verkleinerung des Sendefensters das Senden weiterer TPDUs ein. Nach Ablauf des Sendezeitgebers überträgt er die TPDUs, die innerhalb des verkleinerten Sendefensters liegen, erneut. Der Empfänger empfängt diese und sendet eine Quittung für alle in Sequenz empfangenen TPDUs. Das Sendefenster wird durch die Quittung auf die ursprüngliche Größe erweitert, falls die Empfangslücke vollständig geschlossen wurde. Dieses Verfahren ist einem selektiven Wiederholungsmechanismus ähnlich, erfordert jedoch einen Zeitgeber, der die Wiederübertragung initiiert. Im folgenden wird das Verfahren als "selektive Zeitüberwachung" bezeichnet. Im Bild 4.4 verkleinert die erste Quittung das Sendefenster auf eins. Die zweite stellt die ursprüngliche Größe wieder her.

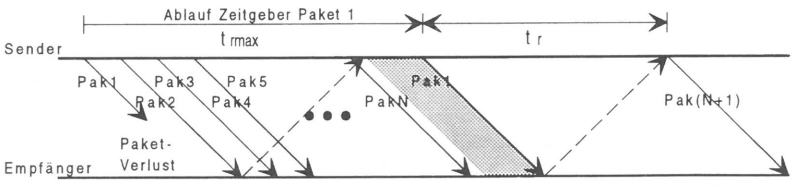


Bild 4.4: *Ablaufdiagramm der 4. Strategie*

Der Sender erhält die Aufforderung zur Verkleinerung des Sendefensters, bevor ein Zeitgeber abläuft. Die selektive Zeitüberwachung stellt sicher, daß nur die TPDUs erneut übertragen werden, die verlorengegangen sind. Der Anteil O der ein zweites Mal zu übertragenden TPDUs ist auf $p_L=5\bullet10^{-4}$ begrenzt.
Der Aufwand für den Sender ist gering: er muß nur in der Lage sein, sein Sendefenster anzupassen. Der Empfänger dagegen muß in der Lage sein, bis zu W_S TPDUs zwischenzuspeichern und diese nach Wiederherstellung der Empfangssequenz anzusprechen und an den Transportbenutzer weiterzuleiten. Der hierfür notwendige Aufwand wird mit der nächsten Strategie minimiert.

Strategie 5) Es wird die oben beschriebene selektive Zeitüberwachung verwendet, mit folgender Einschränkung: bei Auftreten einer Empfangslücke akzeptiert der Empfänger nur TPDUs, die einen zusammenhängenden Block bilden. TPDUs, die weitere Lücken im Empfangsstrom anzeigen, werden verworfen. Im Bild 4.5 sind dies die TPDUs 6 und 7:

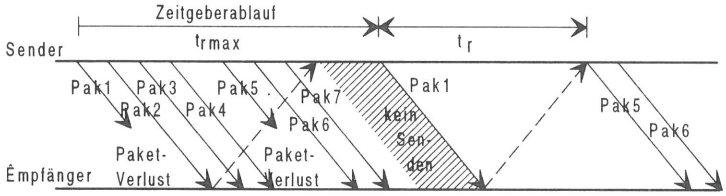


Bild 4.5: *Ablaufdiagramm der 5.Strategie*

Folgende Rechnung weist nach, daß der verursachte Zusatzaufwand gering ist: Pro Stockung werden $R = 1+p_L\bullet R_T\bullet t_r$ TPDU-Verluste durch Wiederübertragung revidiert. Die Stockungen berechnen sich zu:

$$S=\frac{p_L N}{1+p_L R_T t_r} \tag{4.11}$$

Anders als in Strategie 2 müssen pro Stockung im Mittel weniger als $t_r\bullet R_T$ TPDUs erneut übertragen werden. Der Empfänger hat ja per Definition die Möglichkeit, eine Lücke im Empfangsstrom zuzulassen, ohne alle danach folgenden TPDUs verwerfen zu müssen. Die Wahrscheinlichkeit p_A, daß nach einem aufgetretenen Sequenz-Fehler der Empfänger die folgenden $R_T\bullet t_r$-1 TPDUs alle empfängt, ist wie folgt:

$$p_A=(1-p_L)^{R_T t_r-1}=0.78 \tag{4.12}$$

Im Fall, daß von diesen $R_T\bullet t_r$ - 1 TPDUs weitere verlorengehen, muß der Empfänger die nach dem ersten zusätzlichen Sequenzfehler eintreffenden TPDUs verwerfen. Die Zahl der nicht verworfenen TPDUs ist durch den mittleren Abstand zwischen zwei aufeinanderfolgenden Verlusten festgelegt. Mit

$$\begin{aligned} p(\text{Abstand}=1) &= p_L \\ p(\text{Abstand}=i) &= (1-p_L)^{i-1}\bullet p_L \qquad \text{für } i \le R_T t_r - 1 \\ p(\text{Abstand}\ge R_T t_r-1) &= (1-p_L)^{R_T t_r-1} \end{aligned}$$

erhält man für die Zahl NV der im Mittel nicht verworfenen TPDUs:

$$NV=0\bullet p_L+1\bullet(1-p_L)p_L+\ldots+i(1-p_L)^i p_L+\ldots$$

$$\ldots+(R_T t_r-2)(1-p_L)^{R_T t_r-2}p_L+(R_T t_r-1)p_A \tag{4.13}$$

Nach einigen Umformungen erhält man für NV:

$$NV=(R_T t_r-1)p_A+\frac{1-p_L}{p_L}(1+R_T t_r-2)(1-p_L)^{R_T t_r-1}-(R_T t_r-1)(1-p_L)^{R_T t_r-2} \tag{4.14}$$

Für die erwünschte Dimensionierung erhält man, daß für jede Stockung im Fluß von je $R_T\bullet t_r$=500 gesendeten TPDUs NV=441 nicht verworfen werden. Verworfen werden damit nur (500-441)/500 = 11% der TPDUs. Der Anteil O der erneut zu übertragenden TPDUs ergibt sich zu:

$$O=\frac{0.11\bullet S R_T t_r}{N}=\frac{0.11\bullet p_L R_T t_r}{1+p_L R_T t_r}=0.022 \tag{4.15}$$

Dieses Ergebnis zeigt, daß die selektive Zeitüberwachung gepaart mit einer Empfangsstrategie, die nach einer Lücke im Empfangsstrom TPDUs akzeptiert, TPDUs nach einer weiteren Lücke jedoch verwirft, für das verfolgte Dimensionierungsbeispiel nur einen zusätzlichen Übertragungsaufwand von etwas mehr als 2% erfordert. Die Verwaltung eines einzigen zusammenhängenden Blocks von empfangenen TPDUs ist leicht in Form einer Kettenverwaltung zu implementieren.

Strategie 6) Die Strategien 4 und 5 wurden für nicht büschelförmige Verluste hergeleitet. Ist dies nicht der Fall, kann S4 zu einer Verlangsamung der Kommunikation führen, da das Schließen jeder der dicht beieinander liegenden Empfangslücken durch eine Quittung angestoßen werden muß (Lock-Step). Die Strategie 5 kann sogar völlig versagen. Folgende Erweiterung von S5 löst das Problem:

> Bei Feststellen einer Empfangslücke verringert der Empfänger das Sendefenster nicht unbedingt auf die Größe der festgestellten Lücke. Statt dessen wird es höchstens auf einen Wert reduziert, der der bekannten (erwarteten) Anzahl von TPDUs in einem Verlustbüschel entspricht. Ist die Empfangslücke größer als dieser Wert, so wird ihre Größe verwendet.
>
> TPDUs, die innerhalb dieses Bereichs (verringertes Sendefenster) liegen, werden vom Sender nach dem Ablauf eines Zeitgebers wiederholt. Der Empfänger kann sie daher beim ersten Übertragungsversuch verwerfen. Beim zweiten Übertragungsversuch erreichen sie in der richtigen Sequenz den Empfänger und werden von ihm akzeptiert. TPDUs, die außerhalb des verringerten Sendefensters liegen und dennoch beim Empfänger ankommen, werden von diesem akzeptiert, sofern sie einen zusammenhängenden Block (ohne Lücken) bilden.

Im folgenden Beispiel (Bild 4.6) verringert der Empfänger nach Feststellen des Verlusts von TPDU 1 das Sendefenster auf 3. Er verwirft daher die TPDUs 2 und 3, speichert aber die TPDUs 4, 5, 6 und 7. Nach Erhalt der TPDUs 1, 2 und 3 in Sequenz vergrößert er das Sendefenster auf die ursprüngliche Größe.

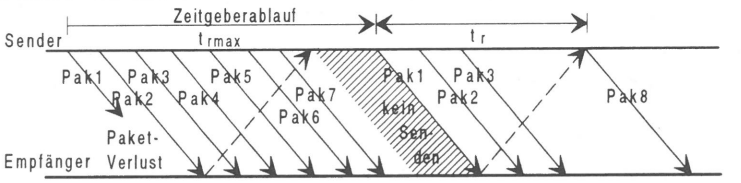


Bild 4.6: *Ablaufdiagramm für die 6. Strategie*

Diese Strategie wird beiden Verlustarten (unabhängig bzw. burstartig) gerecht. Außerdem vermeidet sie den Übergang in einen lock-step-ähnlichen Austausch von Daten-TPDUs und Quittungen.

Voraussetzung für diese Strategie ist eine Schätzung der Länge der Verlustbüschel. Mit der Wahl dieser Länge zu 1 geht die Strategie in die von S5 über. Die Festlegung der erwarteten Büschellänge zu $W_S = R_T \bullet t_{rmax}$ macht die Strategie identisch mit der von S1. Ist keine genaue Schätzung möglich, so ist ein Wert zu wählen, der diese Extreme vermeidet (etwa 1/8 von $R_T \bullet t_{rmax}$).

4.1.1.3 Anwendungsorientierte Schichten

Im Bereich der lokalen Netze dominieren nach wie vor Kommunikationsformen, die direkt auf den unteren Schichten des OSI-RMs aufsetzen. Selbst für eine Kommunikation über WANs reicht häufig allein die Transportfunktionalität aus oder es wird bestenfalls noch die Unterstützung eines bestimmten Anwendungstypus, d.h. Schicht-7-Elemente wie z.B. die ISO Protokolle FTAM und ROSE [339, 341] oder die DoD Internet Varianten FTP und Telnet [5], in Anspruch genommen. Implementierungen mit einem vollständigen 7-schichtigen OSI-Protokollstapel sind daher nur schwer zu finden. Erfahrungsberichte und Messungen an solchen Systemen sind naturgemäß noch rarer. In [167] wird über entsprechende Arbeiten

am schwedischen Institut für Informatik SICS berichtet. Die Vermutung, daß mit steigender Schicht die Leistungsfähigkeit, also etwa der Nutzdurchsatz, stetig abnimmt, wird in dem Bericht bestätigt.

Betrachtet man die Funktionen der anwendungsorientierten Schichten 5 bis 7 [338 - 354] des OSI-RMs unter Leistungsgesichtspunkten, fallen schnell zwei mögliche Problembereiche auf:

1. Ver- und Entschlüsselung und
2. Darstellungswandlungen.

Beide Funktionen gehören zur Schicht 6 (Darstellung). Mit den Betrachtungen von möglichen Schwachpunkten heutiger Kommunikationssysteme in Kapitel 3 wird das unumgängliche Behandeln jedes einzelnen Bits der ausgetauschten Information als das offensichtliche Problem beider Funktionen erkannt.

Von Clark und Tennenhouse [109] durchgeführte Untersuchungen zeigen z.B. auf, daß selbst mit maschinennahen handkodierten Routinen auf einem RISC-Prozessor des Typs MIPS R200 maximal 28 Mbit/s bei einer einfachen Umwandlung eines Ganzzahlenfeldes in die abstrakte Syntax Notation ASN.1 [351, 11, 162] erreicht werden können. Werden die höheren Schichten der prototypischen ISO-Entwurfsumgebung ISODE [191] in Verbindung mit einem herkömmlichen TCP/IP-Transportsystem verwendet, so wird der Einfluß der Darstellungskonvertierung noch deutlicher: bis zu 97% des gesamten Protokollaufwandes waren schon für einfache Umwandlungen notwendig! In Fällen mit höheren Anforderungen an die Konversionsmechanismen verschlechterten sich die Verhältnisse sogar noch.

Um trotz des hohen Aufwandes, der für die Wandlung der Darstellungsform benötigt wird, eine akzeptable Systemleistungsfähigkeit zu erzielen, können zwei Wege beschritten werden:

1. Die möglichen Kommunikationspartner einigen sich auf eine Darstellungsform, womit dann diese Teilfunktion der Präsentationsschicht weggelassen werden kann.
2. Es muß intensiv an Methoden gearbeitet werden, die zu einer gravierenden Leistungssteigerung der Formatwandlungsfunktion führen.

Der erste Weg wird tatsächlich in vielen heutigen Systemen gewählt. Die Anwendungen tauschen ihre Daten dabei in einer unstrukturierten rein binären Form (raw data oder auch image data) aus. Auch die funktionellen Standardempfehlungen MAP und TOP gehen von einer quasi leeren Schicht 6 aus (siehe Abschnitt 4.1.1). Der unvermeidbare resultierende Verlust an Allgemeinheit kann jedoch nicht grundsätzlich für alle möglichen Kommunikationsszenarien akzeptiert werden.

Im Gegensatz zu der Vielzahl an Arbeiten im Bereich der unteren 4 Schichten sind erst einige wenige Arbeiten zur Optimierung der höheren Schichten im Gange. Erwähnenswert sind:

- ein französischer Ansatz [186], der eine nicht ISO-konforme Syntax vorschlägt, welche insgesamt wesentlich einfachere Konstrukte enthält (lightweight transfer syntax) und
- ein Vorschlag aus den AT&T Bell Laboratorien [168], der ein Kommunikationssystem nicht in vertikale Schichten sondern horizontal, nach - möglichst unabhängigen - Funktionalitäten unterteilt. Mit dieser grundsätzlich anderen Sichtweise sind auch für Funktionen der höheren Schichten leistungsfähige Realisierungen, sowohl in Software auf geeignet konfigurierten Multiprozessorsystemen als auch hardwareunterstützt, möglich (siehe auch Abschnitt 4.1.4).

In Arbeiten, welche eine Leistungssteigerung durch Fließbandverarbeitung der einzelnen Schichten propagieren (Abschnitt 4.1.1.4), wird vom Einsatz eines Prozessorelements pro Schicht ausgegangen, das entweder für jede Schicht einheitlich sein kann oder u.U. gesondert für jede Schicht entwickelt wurde. Dietsch und Ulrich schlagen in [129] dagegen eine massive Parallelisierung gleichartiger Prozessoren innerhalb der höheren Schichten als Mittel zu einer Leistungssteigerung vor. Bedingt durch Abhängigkeiten innerhalb einer übertragenen Dateneinheit kommt allerdings nur eine Parallelisierung auf der Ebene von Benutzerdateneinheiten (ADUs, Application Data Units) oder auf Verbindungsebene in Frage.

Bei der Implementierung von Routinen zur Umwandlung von Darstellungsformen sollte der Einfluß von

verlorenen und in falscher Reihenfolge empfangenen Dateneinheiten nicht vergessen werden [109]. Die Konversion kann nur auf vollständig empfangene und in der richtigen Reihenfolge komplett zusammengesetzten Dateneinheiten angewandt werden. Bei Fehlern in den unteren Schichten wird die Darstellungsschicht demzufolge solange nicht aktiv sein können, bis der Fehler - z.B. durch wiederholtes Senden der fehlenden oder falschen Teile - behoben wurde. Selbst unter der Annahme, daß die Umwandlung so optimiert wurde, daß sie gerade leistungsfähig genug ist, um eine angekommene Dateneinheit zu bearbeiten, bevor die nächste Dateneinheit angeliefert wird, kann die Darstellungsschicht die zwangsweise inaktiv verbrachte Zeitspanne nicht ausgleichen. Clark und Tennenhouse schlagen in [109] zur Verbesserung dieser Situation die Einführung von Synchronisationspunkten vor, ab denen Manipulationsaktionen schon beginnen können, selbst wenn vorangegangene Datenteile noch ausstehen.

Eine Unterstützung der Darstellungskonvertierung durch Hardware wäre wünschenswert, ist jedoch bedingt durch die komplizierten Regeln und die große Zahl von Variationen innerhalb der ASN.1-Spezifikation derzeit nur mit weitreichenden Einschränkungen machbar. Gegen eine Hardwareunterstützung spricht auch die enge inhaltliche Verknüpfung mit der Anwendung: die umgewandelten Daten werden in der Regel nicht einfach sequentiell im Benutzerspeicher abgelegt, wie das etwa im Falle einer Dateiübertragung der Fall ist, sondern entsprechend der Struktur von Hochsprachenvariablen. Bei Aufrufen entfernter Prozeduren (remote procedure calls) müssen die Daten u.U. in einer definierten Reihenfolge auf dem Stapel (Stack) der zugehörigen Anwendung abgelegt werden.

Mit Hardwareunterstützung sieht es bei den kryptographischen Funktionen [219] ganz anders aus: für Verfahren unterschiedlichster Komplexität, Handhabung und Dekodiersicherheit sind integrierte Bausteine verfügbar [418, 306]. Deren Leistungsfähigkeit liegt je nach angestrebter Sicherheit und eingesetztem Verfahren zwischen einigen wenigen bis zu knapp 100 Mbit/s. Durch die Wechsel zwischen Software und Hardware und wieder zurück entstehen allerdings die in 3.2.2 angesprochenen Schnittstellenprobleme und die Notwendigkeit, alle Bits einer Dateneinheit ein weiteres Mal zu bewegen!

Im bisherigen Verlauf der Arbeit, u.a. in Abschnitt 3.2.4, wurde schon deutlich, daß das physikalische Bewegen von Daten eine zeitintensive Funktion ist. Grundsätzlich sollen Kopiervorgänge vermieden werden. Im günstigsten Fall könnten die drei Aktionen, welche jedes übertragene Bit behandeln müssen, also Prüfsummenbildung, Wandlung der Darstellungsform und Kryptographie, so kombiniert werden, daß die Daten nur ein einziges Mal bewegt werden müssen. Wäre die Kombination der drei Funktionen echtzeitfähig, d.h. könnte die komplette Bearbeitung eines Oktetts durch alle drei Funktionsmechanismen schneller abgeschlossen werden als das nächste Oktett eintreffen würde, so könnten diese während des unumgänglichen Kopierens der Daten bei der Übernahme vom Netzadapter an den AR in Empfangsrichtung bzw. in umgekehrter Richtung "nebenbei" (on-the-fly) miterledigt werden.

4.1.2 Neue Protokolle

Die isolierte Betrachtung einzelner Aspekte von Protokollen und Maßnahmen zu deren Optimierung, so wie sie im Abschnitt 4.1.1 beschrieben wurde, resultiert häufig nur in relativ geringen Leistungssteigerungen. Durch die enge Verknüpfung der einzelnen Mechanismen wird die Verbesserung des Verhaltens an einer Stelle oft nur durch eine damit einhergehende Verschlechterung an anderer Stelle erreicht werden. Zudem schreiben die bestehenden Protokolle grundsätzliche Vorgehensweisen und Verfahren zwingend vor, womit der Spielraum für mögliche Änderungen von vornherein erheblich eingeschränkt wird.

Aufbauend auf den Untersuchungen einzelner Protokollaspekte ist der Schritt zur Entwicklung eines neuen - wegen der reduzierten Komplexität auch als leichtgewichtig bezeichneten - Protokolls naheliegend. Bei dieser Vorgehensweise brauchen keinerlei Rücksichten auf bestehende Randbedingungen genommen werden. Außerdem können problemlos einschränkende Annahmen über die relevanten Bedingungen der Umgebung - hier ist als wichtigstes Merkmal die Charakteristik der zu unterstützenden

Kommunikationsform zu nennen - getroffen werden, welche sich dann wiederum in einer spezifischen Optimierung einzelner Mechanismen des neu zu entwerfenden Protokolls niederschlagen.

Eine Übersicht einiger der bedeutenderen leichtgewichtigen Transport-Protokolle TPe findet man in [136, 270] zusammen mit einem Vergleich mit herkömmlichen TPen wie TCP oder OSI TP4 vorgestellt. Gemeinsames Merkmal aller leichtgewichtigen TPe sind entsprechend diesen Referenzen:

- Grundsätzlich einfachere Protokollalgorithmen;
- Feste Formate für Steuerköpfe und -nachläufe (Header und Trailer),
- Der Protokollkern wird durch wenige Felder im Steuerkopf realisiert.

Eine umfangreiche Beschreibung aller neuen TPe würde den Rahmen dieser Arbeit sprengen. Daher werden in diesem Abschnitt nur ausgewählte TPe und deren wichtigsten Innovationen gegenüber bereits davor entwickelten beschrieben. Ziel ist, die wesentlichen Unterschiede zwischen den TPen herauszuarbeiten. Um einen Vergleich zu ermöglichen, werden zunächst die Eigenschaften der zwei Standard-TPen TCP und OSI TP4 zusammengefaßt. Anschließend an eine Übersicht von 12 leichtgewichtigen TPen, wird ein neues TP beschrieben, das in Zusammenarbeit zwischen dem Autor und dem europäischen Netzwerk Zentrum (ENC) der IBM in Heidelberg entwickelt und implementiert wurde.

4.1.2.1 Zum Vergleich: Standardtransportprotokolle

Trotz Absichtserklärungen internationaler und nationaler Gremien sowie vieler großer Firmen der Kommunikationstechnik, die OSI-Standards mittelfristig zu unterstützen, dominieren in der Praxis nach wie vor unangefochten die Protokolle der Internet-Familie. Auf Transportebene ist TCP (Englisch für Transmission Control Protocol) das relevante Protokoll. Auf Grund der Dominanz wird neben OSI TP4 nachfolgend auch auf den in vielen Eigenschaften dazu ähnlichen Defacto Standard TCP eingegangen.

OSI/TP4

OSI/TP4 ist das Klasse 4 TP der ISO. Die durch die ISO alternativ bereitgestellten Dienstklassen 0 bis 3 [355] enthalten jeweils Untermengen der in der Klasse 4 enthaltenen Funktionalität (Abschnitt 3.3.2).

Verbindungsauf- und -abbau erfolgen über 3- bzw. 2-Wege-Ablauf (3/2-way-handshake). Der Dienstbenutzer kann während des Verbindungsaufbaues Dienstgüteparameter mit dem Diensterbringer aushandeln. Die Überwachung des Datenflusses erfolgt mit Hilfe von Fenstertechnik. Die Funktionen zur Fehlerbehandlung sind senderorientiert: TPDU-Verluste werden grundsätzlich durch den Ablauf eines Zeitgebers auf Senderseite erkannt und durch erneute Übertragung der TPDU durch den Sender behoben.

Weitere Merkmale dieses TPs sind Mechanismen zur Aufspreizung einer Transportverbindung auf eine Vermittlungsverbindung (splitting) sowie zur Übertragung mehrerer TPDUs in einer NSDU durch Verkettung (concatenation).

TCP

TCP [253] ist ein bereits in den 70-er Jahren entstandenes TP. Es hat sich auf Grund seiner Verbreitung zu einem Defacto-Standard entwickelt. Der Aufbau der wichtigsten Protokollfunktionen, wie Verbindungsmanagement, Datenflußregelung und Fehlerbehandlung entspricht weitgehend dem von OSI TP4:

- Verbindungsauf- und -abbau erfolgen explizit über 3- bzw. 2-fachen Meldungsaustausch
- Die Datenflußregelung basiert auf Fenstertechnik.
- Fehler werden durch den Mechanismus des Aufsetzens beim ersten nichtquittierten Segment (go-back-n) behoben.

TCP setzt auf dem verbindungslosen IP-Dienst (Internet Protocol, [251, 252, 5]) auf.

4.1.2.2 Fremdentwicklungen

Aus der stetig wachsenden Menge neuer TPe werden im folgenden 12 Vertreter näher betrachtet. Bewußt wird keine Konzentration auf die vier bekanntesten TPe (Delta-t, NETBLT, VMTP, XTP) vorgenommen, um die breite Streuung auf dem Gebiet neuer Protokolle zu verdeutlichen.

ALTP-OT

ALTP-OT (Application-oriented Lightweight Transport Protocol for Object Transfer, [290, 291]) ist ein leichtgewichtiges, anwendungsorientiertes TP innerhalb der AXON-Kommunikationsarchitektur für verteilte Systeme. Ziel beim Entwurf von ALTP-OT war, eine durchsatzeffiziente Datenübertragung bei Verwendung zukünftiger breitbandiger Netze sicherzustellen, wie sie beispielsweise zur Unterstützung von Anwendungen mit häufigem Austausch von Bilddaten erforderlich ist.

Dieses Protokoll arbeitet über einem verbindungsorientierten Vermittlungsdienst, dessen Datentransferphase nicht gesichert ist, der Mehrpunktkommunikation (multicasting) unterstützt und dem Benutzer über Mechanismen der Betriebsmittelverwaltung Dienstgütegarantien zur Verfügung stellt (Multipoint Congram oriented High Performance Internetwork Protocol; MCHIP, siehe [221, 244]).

Da der Vermittlungsdienst in der Lage ist, definierte Dienstgüten zu garantieren, kann die Datenflußregelung sehr effizient mit Hilfe von Ratenüberwachung realisiert werden.

Die Fehlerbehandlung umfaßt Mechanismen zur Behandlung von Verfälschungen, TPDU-Duplikaten, TPDU-Verlusten und Reihenfolgevertauschungen. Die für die Fehlererkennung benötigten Funktionen wurden, soweit möglich, in den Protokollinstanzen auf Empfängerseite realisiert. Fehler können bei diesem empfangsorientierten Ansatz, der zuerst in NETBLT eingesetzt wurde, schneller angezeigt werden als bei herkömmlichen Protokollen. Die Fehlerbehandlung erfolgt auf Basis von TPDU-Gruppen. Damit wird ein Kompromiß eingegangen zwischen den beiden Extremen kürzester Verzögerungszeit bei Fehlerbehandlung auf TPDU-Basis gegenüber minimalem Zusatzaufwand bei der Fehlerbehandlung auf Datenblockbasis, der speziell auf die durch dieses Protokoll zu unterstützenden Anwendungen zugeschnitten ist. Die Fehler selbst werden dann durch Übertragungswiederholung der verlorenen Daten behoben.

Datakit

Bei Datakit [155, 156] werden die Daten zwischen den Transportinstanzen in Form eines Bytestromes übertragen. Es werden 9-bit-Bytes verwendet, wobei das 9. Bit anzeigt, ob das Byte Benutzer- oder Kontrollinformationen enthält. Die Übertragung des Bytestromes in den darunter liegenden Schichten kann jeweils mit der für statistisches Multiplexen optimalen Größe erfolgen.

Es wird zwischen Sender- und Empfängerinstanz unterschieden. Auf der Empfangsseite befindet sich eine Protokollmaschine - der sogenannte universelle Empfänger (Universal Receiver) - die durch Kommandos des Senders (Kontrolloktett) gesteuert wird. Jedes Kontrolloktett wird als eigenständiger Befehl vom universellen Empfänger abgearbeitet. Grund für diese Protokollstruktur ist die Vermutung, daß in herkömmlichen TPen häufig die Empfangsseite aufgrund höherer Komplexität einen Engpaß bildet. Ein Ziel ist daher, einen Teil der Funktionalität von der Empfänger- zur Senderseite zu verlagern. Die Empfängerinstanz kann je nach geforderter Dienstqualität in 2 Betriebsweisen betrieben werden:

1. Im Blockmodus werden die Daten in numerierten Blöcken übertragen, was Fehlerbehandlung durch Wiederholungssendung auf Blockbasis ermöglicht.
2. Im Zeichenmodus (character mode) erfolgt keine Fehlerkorrektur, der Verwaltungsaufwand ist entsprechend kleiner.

In beiden Fällen wird die Datenflußregelung über einen Fenstermechanismus realisiert.

Delta-t

Delta-t [314] ist ein allgemein einsetzbares TP, das sowohl transaktions- als auch datenstromorientierte Kommunikationsformen unterstützt. Delta-t setzt auf einer IP-Vermittlungsschicht auf. Wesentliche Neuentwicklung von Delta-t ist die implizite zeitgebergesteuerte Verbindungsverwaltung.

Bei Delta-t werden Verbindungen implizit bei Erhalt der ersten TPDU einer Verbindung aufgebaut. Beim Verbindungsaufbau wird auf beiden Seiten der Verbindung ein Zeitgeber gestartet. Auf Empfängerseite wird der Zeitgeber bei Erhalt einer neuen PDU zurückgesetzt. Bei Ablauf des Zeitgebers wird die Verbindung abgebaut, indem die zugehörigen Zustandsinformationen gelöscht werden. Der Zeitwert wird so festgesetzt, daß alle Wiederholungen durch den Sender und alle Duplikate erkannt werden können.

Auf Senderseite wird der Zeitgeber beim Senden einer PDU zurückgesetzt. Der Zeitwert wird so festgelegt, daß die geforderten Wiederholungen möglich sind und Bestätigungen für gesendete PDUs empfangen werden können.

Der Verbindungsabbau erfolgt auf beiden Seiten implizit bei Ablauf des Zeitgebers. Dabei werden die gespeicherten Zustandsinformationen gelöscht. Wichtig ist, daß die Zeitwerte so bestimmt werden, daß die Verbindung zuerst auf Empfänger- und dann erst auf Senderseite abgebaut wird. Damit wird sichergestellt, daß PDUs stets mit der richtigen Folgenummer gesendet werden. Würde die Verbindung auf der Senderseite zwischenzeitlich ab- und wieder aufgebaut, würden die PDUs auf der Empfängerseite auf Grund falscher Folgenummern verworfen. Voraussetzung für implizite Verbindungsverwaltung ist eine maximale Lebensdauer (Maximal Packet Lifetime, MPL), um bei der Bestimmung der Zeitwerte einen Maximalwert für den Zeitraum festzulegen, in dem Duplikate auftreten können.

GAM-T-103

GAM-T-103 [227] ist ein TP, das Dienste zur Übertragung zeitkritischer Informationen zur Verfügung stellt. Ziel bei der Entwicklung ist, eine besonders effiziente Dienstschnittstelle zur Verfügung zu stellen. Es werden zwei alternative Formen der Schnittstellenverwaltung verwendet: Neben dem in ISO verwendeten Typ "flow type", der sich als (FIFO-)Warteschlange modellieren läßt und bei dem bei Überlauf der Warteschlange die jeweils jüngsten SDUs verworfen werden, wird ein "sampling type" eingeführt, bei dem im Fall des Überlaufes die jeweils älteste SDU verworfen wird.

Außer den üblichen Primitiven: Anforderung, Anzeige, Antwort und Bestätigung (request, indication, response, confirm), werden weitere eingeführt. Mit Hilfe dieser Instruktionsprimitive kann der Dienstbenutzer dem Diensterbringer Anweisungen über dessen Verhalten beim Auftreten vorhersehbarer Ereignisse übermitteln. Beispielsweise können vom Dienstbenutzer Aufgaben, wie die Generierung einer Verbindungsaufbaubestätigung als Reaktion auf den Erhalt eines Verbindungsaufbauwunsches oder die Bestätigung empfangener Benutzerdaten, an den Diensterbringer delegiert werden.

Die Primitive sind daher ein wichtiges Hilfsmittel zur Minimierung der Anzahl der zur Übertragung erforderlichen Dienstaufrufe. Man unterscheidet zwischen permanenten und temporären Primitiven:

Permanente Instruktionsprimitive bleiben solange gültig, bis sie durch darauffolgende Instruktionsprimitive ersetzt werden;

Temporäre Instruktionsprimitive verlieren ihre Gültigkeit nach einmaliger Ausführung.

GAM-T-103 setzt direkt auf der Sicherungsschicht auf. Mehrpunktverbindungen werden so realisiert, daß eventuelle Mehrfachverteiler- (Multi-/Broadcast-) Eigenschaften des Übertragungsmediums ausgenutzt werden können. Den Verbindungen können unterschiedliche Dienstqualitäten zugeordnet werden, je nachdem, ob Duplikate oder Reihenfolgefehler vom Benutzer toleriert werden können und ob eine Segmentierung von SDUs erforderlich ist.

NETBLT

NETBLT (NETwork BLock Transfer) [106, 107] ist ein TP aus dem IP-Umfeld, das für Massendatenübertragung optimiert wurde. Ziel beim Entwurf war die Maximierung des Durchsatzes.

Der Verbindungsaufbau erfolgt bei NETBLT über 2-Wege-Ablauf (two-way handshake). Dabei können die für die Ratenüberwachung benötigten Parameter ausgehandelt werden. Der Verbindungsabbau erfolgt implizit über Zeitgeber, der Wert der Zeitgeber wird beim Aufbau ausgehandelt.

Die zu übertragenden Daten werden in Blöcken zusammengefaßt. Signalisierung und Verwaltung der Zustandsgrößen erfolgen jeweils auf Blockbasis. Der Sender signalisiert zunächst dem Empfänger, daß er einen Datenblock zur Übertragung bereithält. Der Empfänger reserviert Speicher der entsprechenden Größe und meldet dem Sender mit Hilfe einer GO-Nachricht, daß mit der Übertragung des Blockes begonnen werden kann. Der Daten-Block wird zur Übertragung in PDUs fragmentiert. Verlorengegangene PDUs werden vom Empfänger selektiv neu angefordert.

NETBLT war das erste Protokoll mit Datenflußregelung über Steuerung des zeitlichen Abstandes zwischen PDUs (Ratenüberwachung, rate control). Durch Steuerung der Rate, mit der PDUs oder kleinere Gruppen von PDUs (bursts) gesendet werden, kann die Übertragungsrate an die Bearbeitungsgeschwindigkeit der an der Übertragung beteiligten Instanzen angepaßt werden. PDU-Verluste durch Pufferüberläufe können so weitgehend vermieden werden.

PROMPT

PROMPT [59] ist ein empfangsorientiertes TP, das für die Übertragung auf Hochgeschwindigkeitsnetzen entwickelt wurde. Im Gegensatz zu den herkömmlichen sendeorientierten TPen wird die Datenübertragung bei PROMPT durch den Empfänger eingeleitet und auch kontrolliert.

Verbindungsauf- und -abbau erfolgen über 2-Wege Ablauf. Beim Aufbau werden die Parameter für Senderate und Wiederholungszähler ausgehandelt.

Zur Datenübertragung sendet die Instanz der Datensenke zunächst eine Anforderung zur Instanz der Quelle. Die Anforderung spezifiziert, welche Daten in welcher Form übertragen werden sollen. Die Übertragung wird durch den Empfänger kontrolliert. PDU-Verluste werden durch Ablauf von Zeitgebern oder Reihenfolgefehler erkannt. Nicht empfangene Datenteile werden selektiv erneut angefordert.

Bei der Übertragung zeitkritischer Daten wird die Wahrscheinlichkeit für PDU-Verluste und die Wiederholungsanforderung durch mehrmaliges Senden der gleichen PDUs zusätzlich vermindert.

RRDP

RRDP (Rhodos Reliable Datagramm Protocol) [164] ist ein anwendungsorientiertes TP, das speziell zur Unterstützung von Interprozeßkommunikation in verteilten Systemen konzipiert wurde.

Die Übertragung ist gesichert. Bei beiden Kommunikationspartnern muß dazu Zustandsinformation verwaltet werden. Allokation und Deallokation der Information erfolgt implizit bzw. zeitgebergesteuert. Für das Verbindungsmanagement werden prinzipiell die gleichen Mechanismen eingesetzt wie in Delta-t.

Größere SDUs können segmentiert werden. Zur Datenflußregelung wird ein erweiterter Zurück-bis-zum-n-ten-Segment (go-back-n) Mechanismus verwendet. Zur Übertragung werden die Daten auf Gruppen verteilt. Zu jedem Zeitpunkt wird nur eine Gruppe mit fest definierter TPDU-Anzahl übermittelt. Alle TPDUs einer Gruppe werden vom Empfänger gemeinsam mit Hilfe einer Quittung bestätigt. Diese Bestätigung kann auch Informationen über noch nicht empfangene TPDUs der Gruppe enthalten (NACK). In diesem Fall werden (nur) diese TPDUs erneut übertragen. Nach Bestätigung des korrekten Empfangs einer TPDU-Gruppe durch den Empfänger kann die nächste Gruppe übertragen werden.

SNR

In diesem von Netravali und Sabnani entwickelten TP [238] werden Kontroll- und Zustandsinformationen - im Gegensatz zu herkömmlichen TPs - nicht ereignisgesteuert, sondern periodisch ausgetauscht[1]. In jeder Kontrollnachricht sind dabei jeweils alle zu übertragenden Kontrollinformationen enthalten, alle Kontrollnachrichten sind daher gleich strukturiert.

Ziel dieses Mechanismus ist, die Komplexität zu verringern, indem die Übertragung von Kontrolldaten von der Übertragung von Nutzdaten entkoppelt wird, was die Parallelisierung der Protokollabläufe vereinfacht. Andererseits können durch die eingesetzte Redundanz Fehler durch verlorengegangene Quittungen (ACKs) vermieden werden, die sonst leicht zu unnötigen Wiederholungen führen könnten. Der erhöhte Bedarf an Bandbreite für Kontrollinformationen kann im Umfeld von Hochgeschwindigkeitsnetzen in Kauf genommen werden. Dabei wird die Strategie verfolgt, beim Entwurf die Komplexität im Zweifelsfall zu Lasten der benötigten Bandbreite zu verringern.

Um diesen Nachteil zu vermindern, werden gleichzeitig zur Erhöhung der Durchsatzeffizienz bei der Fehlerbehandlung Mechanismen für selektive Wiederholung verlorengegangener TPDUs eingesetzt.

TP++

TP++ [67, 70, 147] ist zur Zeit noch in der Entwicklung. Daher liegt bisher weder eine vollständige Protokollspezifikation noch eine komplette Implementierung vor. Dennoch sind einige wichtige Protokollmechanismen bereits fertiggestellt.

TP++ stellt Dienste für zeitkritische Informationsübertragung, transaktionsorientierte Kommunikation, sowie Massendatenübertragung zur Verfügung. Zur Realisierung dieser Dienste werden jeweils die am besten geeigneten Mechanismen bestehender TPs zusammengestellt:

Das Verbindungsmanagement ist zur Vermeidung großer Verzögerungszeiten beim Verbindungsaufbau implizit, also zeitgebergesteuert. Es wird davon ausgegangen, daß Daten unterhalb der Transportschicht in möglichst kleinen PDUs übertragen werden, um Verzögerungszeiten und Schwankungen derselben (jitter) bei der Übertragung zu minimieren (Ausrichtung hin zu ATM). Daher sind Mechanismen vorgesehen, um TPDUs analog zu Datakit zu segmentieren und in mehreren PDUs zu übertragen. Multiplexen mehrerer Schicht-4-Verbindungen auf eine Schicht-3-Verbindung ist nicht vorgesehen. Prinzipiell sollte Multiplexen aus Effizienzgründen nur auf einer Schicht unterstützt werden [145, 299]. Es wird davon ausgegangen, daß bereits die Schicht-3 zur Bereitstellung von Dienstgütegarantien Multiplexmechanismen enthält. TPDU-Verluste werden auf Senderseite erkannt. Die Fehlerbehandlung erfolgt zur Reduzierung des Bearbeitungsaufwandes auf Basis von TPDU-Gruppen. Bei zeitkritischem Informationsaustausch wird die Fehlerbehandlung unter Inkaufnahme größeren Verwaltungsaufwandes durch redundante fehlerkorrigierende Kodierung (Forward Error Control; FEC, [56]) zusätzlich beschleunigt. Durch weitgehende Entkopplung einzelner Protokollalgorithmen - wie beispielsweise der Datenflußsteuerung von der Fehlerbehandlung - wird die Protokollkomplexität minimiert und gleichzeitig die Parallelisierbarkeit bei der Protokollimplementierung erhöht.

TPR

TPR (Transport Protocol for Real time services) [49] ist ein TP, für isochrone Daten. Es setzt auf den Diensten der FDDI MAC Schicht auf. Fast die gesamte Protokollfunktionalität mit Ausnahme der TPDU-Erstellung befindet sich auf Empfängerseite: TPDU-Verluste werden von TRP durch wiederholte Übergabe der vorhergehenden TPDU an den Dienstbenutzer behandelt. Diese Form der Fehlerbehandlung ist für die Übertragung von Audio- und Video-Datenströmen geeignet. Schwankungen bei der Verzögerung

[1] In ähnlicher Form ist dieser Mechanismus in einem Vorschlag für ein Schicht-2-Protokoll enthalten[87]

werden durch Zwischenspeicherung auf Empfangsseite ausgeglichen. Dabei werden alle TPDUs solange im Zwischenspeicher zurückbehalten, bis sie schließlich alle die gleiche Gesamtverzögerung aufweisen, die in etwa der maximalen Ende-zu-Ende-Verzögerung des Netzes entspricht. Dieser Mechanismus setzt voraus, daß TSDUs vom Benutzer auf Senderseite mit genau der gleichen Frequenz vom Netz angenommen werden, wie sie auf der Empfangsseite an den Benutzer abgegeben werden können. Unterschiedliche Datenraten (drift) werden durch Anwachsen/Abfallen der Anzahl von TPDUs im Zwischenspeicher des Empfängers erkannt und durch Verwerfen von TPDUs bzw. wiederholte Übergabe von SDUs an den Benutzer ausgeglichen.

VMTP

VMTP (Versatile Message Transfer Protocol) [93, 218, 241, 321] ist ein Anfrage-/Antwort (request/response) Protokoll, das in verteilten Systemen über einem verbindungslosen IP-Dienst realisiert wurde. Ziel ist eine Unterstützung transaktionsorientierter Kommunikationsformen.

Es handelt sich um ein verbindungsorientiertes TP. Der Verbindungsaufbau kann sowohl implizit über Zeitgeber als auch explizit über 3-Wege-Verfahren erfolgen, je nachdem, welche Zeitdauer seit der Initialisierung des Systems vergangen ist. Der Verbindungsabbau erfolgt stets implizit über Zeitgeber. Zur Minimierung des für die Zustandsvariablen benötigten Speicherplatzes kann zwischen einer anfordernden Instanz (client) und einer spezifischen Bedieneinheit (server) zu einem Zeitpunkt stets nur maximal eine Verbindung aufgebaut werden.

Neben herkömmlichen Adressen werden solche eingeführt, deren Zuordnung zu einer Instanz nur über einen begrenzten Zeitraum gültig bleibt (T-stable identifier). Danach können diese Adressen auch anderen Instanzen zugeordnet werden. Um sicherzustellen, daß die Änderung der Zuordnung von allen beteiligten Instanzen erkannt wird, müssen diese Adressen über einen festgelegten Zeitraum T in einen Zwischenzustand gebracht werden, in dem sie keiner Instanz zugeordnet werden dürfen. Ziel dieses Mechanismus ist, bei begrenzter Adreßlänge einen möglichst großen Bereich adressierbarer Instanzen zur Verfügung zu stellen, indem nur den wirklich aktiven Instanzen eine Adresse zugeordnet wird.

VMTP verwendet zwei verschiedene PDU-Formate: je eine pro Richtung. Die resultierenden Anfrage- bzw. Antwort-Nachrichten werden in einer oder mehreren TPDUs übertragen und können bis zu 14 Millionen Oktetts lang sein. Im Flußmodus (streaming mode) können auch größere Datenmengen durch Einsatz von Ratenüberwachung und selektiver Wiederholung effizient übertragen werden.

VMTP bietet eine Reihe von Funktionen, die bei der Unterstützung zukünftiger Anwendungen immer wichtiger werden dürften wie z.B. Mehrpunktdienste oder Weiterleiten von Nachrichten (multicasting bzw. forwarding). Das Protokoll ist rekursiv strukturiert:

> Teile des Managements und der Fehlerbehandlung werden durch den Aufruf entfernter Prozeduren (remote procedure calls, RPCs) realisiert, die selbst wieder auf VMTP aufbauen.

XTP

Die in HS-Netzen geforderte Leistungsfähigkeit des Transportsystems kann teilweise durch eine Realisierung der Protokollfunktionalitäten in Hardware erzielt werden. Eines der wichtigsten Entwurfsziele bei XTP (eXpress Transfer Protocol, [396, 94 - 96, 268, 282, 320] war daher, eine Implementierbarkeit des Protokolls in VLSI-Technik [94, 274] zu ermöglichen bzw. zu erleichtern (siehe Abschnitt 4.1.3).

XTP eignet sich auch für Softwarerealisierungen, die - im Gegensatz zur Hardwarevariante - Bestandteil vieler Forschungsprojekte ist (z.B. [78, 133, 225, 330]) Bedingt durch die weite Verbreitung hat XTP mittlerweile auch Einzug in verschiedene internationale (Vor-)Standardisierungsprojekte und -gremien gefunden, die sich mit Protokollen für Hochgeschwindigkeitskommunikation befassen:
Die Arbeitsgruppe X3S3.3 des amerikanischen Standardisierungsinstituts ANSI Empfehlungen für einen

Transportdienst und ein zugehöriges Transportprotokoll [394] für hohe Geschwindigkeiten herausgegeben, die zwar die Bezeichnung XTP nicht explizit enthalten, aber von ihren Inhalten her - und auch von der personellen Verantwortlichkeit innerhalb der Gruppe - stark auf XTP ausgerichtet sind.

Die europäische Initiative OSI95 [122] beschäftigt sich neben der Definition einer neuen Klasse der Sicherungsschicht (LLC Typ 4), die - analog zur Anpassungsschicht AAL in ATM-Netzwerken - Transportfunktionalitäten besitzt, auch mit Vorschlägen zu einer weiteren für hohe Geschwindigkeiten geeigneten Transportklasse, welche ebenfalls auf XTP aufbauen soll.

Parallel zu OSI 95 laufen weitere europäische Forschungsvorhaben, die mit derselben Thematik beschäft sind. Bei dem RACE-Projekt R2060 Coordination, Implementation and Organisation of Multimediasystems CIO [101] soll z.B. ein Transportdienst plus TP-Protokoll entwickelt und implementiert werden, das für Multimedia-Anwendungen geeignet ist, was sofort die Forderung nach hoher Geschwindigkeit impliziert. Aus Effizienzgründen wird derzeit wiederum eine XTP-basierte Lösung favorisiert.

In XTP werden Transport- und Vermittlungsschicht zu einer Transferschicht zusammengefaßt. Durch wenige feste TPDU-Formate soll die Bearbeitung der Kontrollfelder vereinfacht werden. Durch Ausrichten der Felder auf 64-Bit Grenzen wird das Verschieben von Daten auf Wortgrenzen minimiert. Ein vereinfachter Prüfsummenalgorithmus wird verwendet, der sich effizient in Hardware implementieren läßt. Ein Teil der Funktionalität ist von Empfänger zum Sender verschoben; Quittungen werden dazu beispielsweise nur auf Anforderung vom Sender generiert.

Die Schicht 3 verwendet implizites zeitgebergesteuertes Verbindungsmanagement. Zur Vermeidung von Pufferüberläufen wird der Mechanismus der Ratenüberwachung eingesetzt. Schicht 4 verwendet eine Mischform von implizitem und herkömmlichem, explizitem Verbindungsmanagement. Der Verbindungsaufbau erfolgt implizit mit der ersten übertragenen TPDU. Auf diese Weise ist auch eine effiziente Übertragung von Datagrammen möglich. Der Verbindungsabbau erfolgt über den bekannten 3-fachen Meldungsaustausch, um den für Zustandsinformationen benötigten Speicherplatz zu minimieren. Gegebenenfalls auftretende Duplikate können bei diesem Verfahren erkannt werden, da bereits die Schicht 3 die Reihenfolge der übertragenen TPDUs garantiert.

4.1.2.3 Eigenentwicklung

In zukünftigen Multimedia-Systemen werden transaktionsorientierte Kommunikationsvorgänge und zeitkritische Informationsübertragungen einen dominierenden Anteil an der gesamten Kommunikation darstellen [175, 249]. Zur Unterstützung dieser Kommunikationsformen scheinen sich die bestehenden TPe nicht oder nur bedingt zu eignen. Mittels des Entwurfs und der Implementierung eines neuen TPs sollte zunächst aufgezeigt werden, welche Mechanismen notwendig sind, um transaktionsorientierte Kommunikationsvorgänge optimal zu unterstützen.

Aus einer genauen Betrachtung der Randbedingungen, die durch die Multimediakommunikation auftreten, ergab sich die Forderung nach Benutzung eines verbindungsorientierten Vermittlungsdienstes. Nur mit dem Konstrukt der logischen Beziehung zwischen Sender und Empfänger können Systemressourcen reserviert werden, was wiederum eine unabdingbare Voraussetzung für die Garantie einer geforderten Dienstgüte ist. Die Kombination mit einem verbindungsorientierten Dienst der Vermittlungsschicht erlaubt prinzipiell auch die Unterstützung zeitkritischer Kommunikation. Die mögliche Unterstützung der dritten Hauptform von Kommunikation, der Massendatenübertragung, mit dem neu entwickelten Protokoll sollte ebenfalls untersucht werden.

Die Funktionsweise eines solchen asymmetrischen Anfrage-Antwort-Protokolls ist wie folgt:

Zunächst sendet die Transportinstanz des Auftraggebers (client) eine Anforderung mittels des vereinbarten Anfrage-Antwort-Protokolls zum Diensterbringer (server).

Der Diensterbringer bearbeitet die Anfrage und teilt dem Auftraggeber mit einer entsprechenden Antwortnachricht das Ergebnis der Bearbeitung mit. Mit dem Erhalt der Antwort wird implizit der Erhalt der Anfrage bestätigt, so daß keine gesonderten Kontrollmeldungen erforderlich sind.

Werden auch für Verbindungsmanagement und Quittierung des Erhalts erfolgreich übertragener TPDUs implizite Verfahren eingesetzt, kann jegliche Signalisierung zwischen den beteiligten Instanzen auf die Grundtypen Anfrage bzw. Antwort abgebildet werden. Damit wird mit jeder Meldung ein Maximum an parallel ausführbarer Funktionalität beim Kommunikationspartner angestoßen, und die Verzögerungen für Verbindungsauf- und -abbau sowie die Anzahl auszutauschender Nachrichten werden minimiert.

Der Entwurfsablauf wurde gemäß der ISO-Vorstellung in einen Dienst- und einen Protokollteil getrennt. Im Grundlagenteil (Abschnitt 2.3.2.2) wurden schon die grundsätzlichen Gesichtspunkte von Protokoll und Dienst eingeführt: während Protokolle den Ablauf der Kommunikation zwischen Instanzen der gleichen Schicht bei den - möglicherweise räumlich voneinander entfernten - Kommunikationspartnern bestimmen, legen Dienste die Kommunikation zwischen benachbarten Schichten innerhalb eines lokalen Kommunikationspartners fest. Die Effizienz der Kommunikation kann dabei sehr stark von der Ausprägung der Dienstelemente und Parameter abhängen. Durch eine ungeschickte Wahl ist es möglich, daß nützliche Funktionen, die auf unteren Schichten vorhanden sind, verdeckt werden und dem Benutzer nicht zugänglich sind [269]. Vissers und Logrippo [307] geben die allgemeinen Anforderungen an einen Dienst so an:

1. Der volle Umfang der Möglichkeiten der unterlegten Schichten soll dem Dienstnutzer, d.h. der nächsthöheren Schicht, verfügbar gemacht werden (bottom-up).
2. Die durch das Kommunikationssystem angebotenen Dienste müssen die Anforderungen heutiger und auch zukünftiger Anwendungen erfüllen (top-down).

Der entwickelte Dienst läßt sich aus den vier ISO-üblichen Primitiven aufbauen (Bild 4.7):

- T-TRANSACTION.request
- T-TRANSACTION.indication
- T-TRANSACTION.response
- T-TRANSACTION.confirm

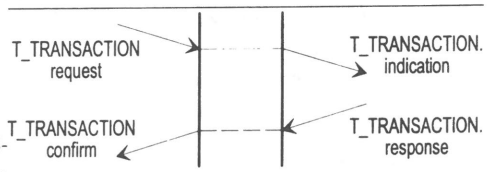


Bild 4.7: *Primitive des Anfrage-Antwort-Dienstes*

Zunächst wird mit den Primitiven T_TRANSACTION.request bzw. T_TRANSACTION.indication eine Anfrage vom Auftraggeber zum Diensterbringer übertragen. Die Antwort wird anschließend mit T_TRANSACTION.response bzw. T_TRANSACTION.confirm in umgekehrter Richtung übermittelt. Der Dienst ist zuverlässig, d.h. es wird vom Diensterbringer *garantiert*, daß jede Nachricht auch bei Übertragungsfehlern der unteren Protokollschichten exakt einmal empfangen wird.

Zur Anzeige nicht behebbarer Fehler muß das Primitiv T-ABORT.indication eingeführt werden.
Dem Benutzer des transaktionsorientierten Basisdienstes werden einige Parameter zur Spezifikation gewünschter Dienstgüte zur Verfügung gestellt. In Tabelle 4.4 sind die fünf Primitive des Basisdienstes mit den jeweils zulässigen Parametern aufgeführt.

Zur Unterstützung zeitkritischer Kommunikationsformen und Massendatenübertragung sind einige Erweiterungen des Basisdienstes erforderlich. Unter anderem werden auch Antwortmeldungen erlaubt, die nicht mehr mit einer einzigen TPDU übertragen werden können. Diese Erweiterung bedingt eine Vielzahl von zusätzlichen Mechanismen - es muß beispielsweise ein Verbindungskontext gespeichert werden - und Dienstparametern. Für eine detaillierte Beschreibung wird auf [270] verwiesen. Dort finden sich außerdem Beschreibungen des Zusammenspiels der einzelnen Primitive sowie Erläuterungen zur Realisierung von Mehrpunktkommunikation.

Tabelle 4.4: *Dienstprimitive und Parameter*
X: vorgeschrieben, (=): Wert entspricht dem Wert des vorhergehenden Parameters

Parameter	Primitiv				
	T-TRANSACTION				T-ABORT
	request	indication	response	confirm	indication
Dienstnutzeradresse	X	X(=)	X(=)	X(=)	X(=)
Diensterbringeradresse	X	X(=)			
Max. Übertragungszeit	X	X			
Restfehlerwahrscheinl.	X	X			
Priorität	X	X			
Übertragungssicherheit	X	X			
Maximale Bedienzeit	X	X(=)			
Nutzerdaten	X	X(=)	X	X(=)	
Fehlercode					X

Wesentliche Merkmale des realisierten Protokolls sind:

- Ein impliziter zeitgebergesteuerter Verbindungsaufbau wird eingesetzt. Wird die dafür notwendige maximale PDU-Lebensdauer nicht vom benutzten Vermittlungsdienst bereitgestellt, kann sie auf Transportebene durch synchronisierte Uhren realisiert werden.
- Das Problem der Eindeutigkeit von Verbindungsnummern - insbesondere auch nach kurzzeitigen Systemzusammenbrüchen - wird durch den Einsatz von Permanentspeichern und zugehörige Zeitüberwachungsmechanismen sichergestellt.
- Bei der Adressierung wird auf das hierarchische ISO-Schema [335] und entsprechende Umsetzungstabellen (directories) zurückgegriffen.
- Es werden keine Quittierungsverfahren benötigt. Geht eine Meldung verloren, so kann dies der auf eine Antwort wartende Klient durch Ablauf eines Zeitgebers definitiv feststellen. Die Löschung der für eine Wiederholung gespeicherten Antwort beim Diensterbringer wird entweder ebenfalls durch Zeitüberwachung ausgelöst oder durch den Erhalt der nächsten Anfrage vom selben Kunden (plus Vereinbarung, daß nur eine Anfrage gestartet werden darf!).
- Es werden 32-Bit Sequenznummern benutzt, um Duplikatsprobleme auszuschließen.
- Fenstertechnik und Ratenüberwachung werden zur Steuerung des Datenflusses benutzt.
- In der implementierten Version ist keine Prüfsumme enthalten. Es wird davon ausgegangen, daß Fehler durch untere Schichten erkannt werden und zu Verlusten "umgewandelt" werden.
- Mehrpunktkommunikation und Weiterleiten von Anfragen werden effizient unterstützt.

Für die detaillierten Überlegungen zur Dimensionierung der notwendigen Zeitwerte wird auf [269, 270] verwiesen, ebenso für Zustandstabellen und Ablaufdiagramme. Dienst und Protokoll wurden in die Umgebung des Heidelberger Transportsystems HeiTs [174 - 177] integriert (Bild 4.8).

Neben dem Vorhandensein von Netzanschlüssen für Token Ring, FDDI und B-ISDN fällt der explizit unterteilte Block für Unterstützungsfunktionen - hier: Zeitgeber, Puffer- und Prozeßverwaltung sowie ein eigener Block zur Verwaltung von Systemressourcen - auf. Durchgeführte Messungen zwischen zwei mit 33 MHz getakteten IBM PS/2 Model 95 Rechnern mit Intel 80486DX Prozessor ergaben mittlere Zeiten von etwa 1.25 ms für das Senden einer Anfragemeldung bzw. knapp 1 ms für das Empfangen. Die Mittelwerte der Bearbeitungszeiten für die einzelnen Aktionen der Transportinstanz (ohne Aufrufe des Vermittlungsdienstes) sind in Tabelle 4.5 aufgeführt.

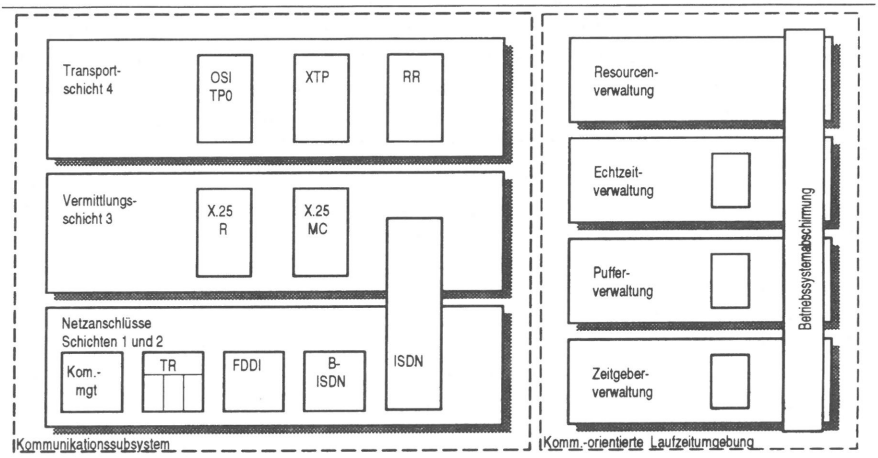


Bild 4.8: *Umgebende Systemarchitektur des entwickelten Anfrage-Antwort-Protokolls*

Die Gesamtzeit einer vollständigen Transaktion aus Sicht des Auftraggebers ergab sich zu 5.5 ms. Damit entfallen 4 ms auf tiefere Protokollschichten, Betriebssystemaktivitäten und Übertragungszeiten. Die Transportschicht bildet also offensichtlich hier nicht (mehr) den Engpaß!

Tabelle 4.5: *Zeitwerte für die einzelnen Primitive*

T_TRANSACTION.request	0.13 ms
T_TRANSACTION.indication	0.43 ms
T_TRANSACTION.response	0.23 ms
T_TRANSACTION.confirm	0.17 ms

Andererseits sind Zeiten im Millisekundenbereich im HS-Bereich immer noch um mindestens eine Grössenordnung zu langsam! Da für diese Implementierung von beinahe idealen äußeren Umständen ausgegangen werden konnte, ist anzunehmen, daß auch andere neu entwickelte Protokolle nicht wesentlich bessere Werte erbringen können, wenn sie tatsächlich in eine reale Systemumgebung integriert werden. Eine Optimierung lediglich der durch Protokolle festgelegten logischen Funktionalität von Kommunikationssystemen resultiert zwar in einer gewissen Verbesserung des Gesamtverhaltens des Systems im direkten Vergleich mit Standardprotokollen, jedoch ist allein mit neuen Protokollen eine HS-Kommunikation nicht erzielbar. Da Heterogenität ein herausragendes Merkmal von Kommunikationssystemen darstellt (und auch in Zukunft stellen wird), ist es fraglich, ob die Entwicklung und der Einsatz nicht standardisierter Lösungen sinnvoll ist bzw. grundsätzliche Marktchancen haben wird.

4.1.3 Hardwareimplementierung von Protokollfunktionen

Bei den technologieabhängigen Schichten 1 und 2a des OSI-RMs wird Protokollfunktionalität durch Zusatzhardware unterstützt oder vollständig durch entsprechende Bausteine realisiert. Nur so können die harten zeitlichen Anforderungen moderner Übertragungssysteme überhaupt befriedigt werden. Das Spektrum der Komplexität der Hardwarekomponenten reicht dabei von einfachsten Teilfunktionen, wie dem Einfügen und Entfernen von Füllbits (bit stuffing) oder dem Erkennen von bestimmten Bitmustern innerhalb des Datenstromes, bis hin zum selbstständigen Abwickeln ganzer Protokollabläufe.

Für Übertragungsprotokolle und Medienzugangsverfahren gibt es integrierte Bausteine oder komplette Baugruppen [166, 211]. Als Beispiel seien die Ethernet-Controller [63, 135, 287, 308] und darauf

aufbauende Einschubkarten für die verschiedensten Rechenanlagen genannt. Analoges gilt für Token Ring [111] und Token Bus Systeme [58, 134, 188] nach IEEE 802.5 bzw. IEEE 802.4 oder für FDDI [149, 178, 413, 411] im privaten Bereich, sowie die unteren Protokollschichten in öffentlichen Netzen (z.B. X.25 [143], LABD [214]) und der Nebenstellentechnik [157]. Krishnakumar und Sabnani geben in [211] einen Überblick über existierende Protokollcontroller und stellen mögliche Vergleichskriterien auf.

Ein schwerwiegendes Problem mit reinen Hardwareimplementierungen von Protokollfunktionen ist durch den dynamischen Charakter von Protokollen gegeben. So werden auch heute noch - mehr als 10 Jahre nach der internationalen Verbreitung "stabiler" Versionen der Internetprotokolle IP und TCP durch weltweit zugängliche elektronische Datenbanken [251 - 253] - Veränderungen an Einzelaspekten vorgenommen. Diese für alle Protokolle typische kontinuierliche Veränderungen sind einerseits auf detailliertere Erfahrungen, die eben erst im Verlauf eines mehrjährigen Betriebs gesammelt werden können, und andererseits auf geänderte äußere Anforderungen zurückzuführen. Als Folge davon können

- entweder nur "stabile" Teilfunktionen in Hardware umgesetzt werden
- oder die Hardware wird programmierbar ausgelegt.

Ersteres trifft fast nur für Funktionen geringer Komplexität zu. Neben den logisch wenig komplexen aber technologisch aufwendigen Funktionen der physikalischen Schicht wie Taktrückgewinnung, Leitungskodierung und -dekodierung, Bitsynchronisation, Basissignalisierung mittels Kontrollbits und ähnlichem (siehe 4.1.1.1), sind alle Funktionen zur Sicherstellung der korrekten Teilnahme am jeweils eingesetzten Medienzugriffsprotokoll, u.a. also Adreßerkennung, Prioritätsmechanismen, Zeitüberwachungsfunktionen, Rahmenerkennung und -bildung sowie einfache Fehlerbehandlungsroutinen, zu nennen.

Auf höheren Protokollschichten kommt die Prüfsummengenerierung und -überprüfung sowie die Identifizierung bestimmter Teile des Steuerkopfes und deren schnelle Zuordnung zu bereits vorhandenen Informationen für eine Hardwarerealisierung in Frage. Auch kann die Ausführung einiger Protokollmechanismen wie z.B. die Folgenumerierung oder priorisierte Behandlung durchaus durch (passive) Hardware unterstützt werden. Das Anstoßen der entsprechenden Aktionen erfordert allerdings oft soviel Intelligenz, daß eine direkte Umsetzung in Hardware kaum oder gar nicht möglich ist, oder es gibt zwar relativ einfach in Hardware zu realisierende Mechanismen, aber eine zu große Variantenzahl davon.

Die im zweiten Punkt genannte Programmierfähigkeit kann in letzter Konsequenz bis zu eigenständigen (Spezial-)Prozessorsystemen führen, welche u.U. sogar verschiedene Protokolle unterstützen können. Der Übergang zu den im folgenden Abschnitt 4.1.4 beschriebenen Multiprozessorsystemen wird damit sehr fließend. Bei realen Implementierungen muß zwangsläufig auch die rein logische Sichtweise verlassen werden, so daß die nachfolgend beschriebenen hardwareunterstützten Ansätze nicht ausschließlich Protokollfunktionalitäten abdecken, sondern auch wesentliche sonstige Systemgesichtspunkte, z.B. zur Speicher- oder Schnittstellenverwaltung enthalten.

Obwohl Hardwarelösungen für einzelne Protokollaspekte teilweise erwähnenswerte Ideen enthalten, werden in diesem Abschnitts lediglich Ansätze detaillierter betrachtet, die jeweils die Funktionalität ganzer Protokollschichten umfassen. Exemplarisch werden dabei folgende Arbeiten vorgestellt:

- Eine generische Architektur für die Schichten 1 und 2a von HS-LANs.
- Eine Realisierung der verbindungsorientierten LLC-Protokollklasse 2.
- Das Projekt "Protocol Engine", das die Entwicklung von Bausteinen zur Implementierung der Schichten 3 und 4 - mit Ausrichtung auf XTP - zum Ziel hat.
- Eine universelle - allerdings akademische - Controller-Architektur zur Unterstützung verschiedener mit formalen Beschreibungssprachen spezifizierten Protokollschichten.

4.1.3.1 Die technologieabhängigen Schichten

Eine Gruppe von der TH Dänemark setzt die verbesserte Leistungsfähigkeit der Technologie nicht dazu ein, um ein spezifisches Protokoll der technologieabhängigen Schichten in immer höhere Leistungsbereiche hinein zu skalieren. Stattdessen wird die Leistungsreserve moderner Technologien benutzt, um Flexibilität, d.h. äußere Einstellbarkeit von parametrisierten Funktionen, zu ermöglichen [98, 198, 237, 285]. Die Festlegung von gemeinsamen Funktionen, die entweder unverändert in jedem Protokoll vorkommen oder sich nur durch Parameter unterscheiden, machte eine umfassende Untersuchung bestehender Protokolle (siehe Abschnitt 4.1.1.1) und entsprechender Hardware erforderlich.

Die Untersuchungen führten schließlich zu einer funktionellen Architektur eines generischen physikalischen Spezialprozessors, bestehend aus getrennten Pfaden für Sende- und Empfangsrichtung. Eine von den Autoren durchgeführte Machbarkeitsstudie ergab eine Gesamtkomplexität von ungefähr 4000 Gattern. Bei Verwendung verfügbarer und schneller bipolarer Silizium- oder Galliumarsenid (GaAs)-Standardzellen-Technologie kann die gefundene Lösung für alle gängigen Hochgeschwindigkeitsnetze bis hin zum 600 Mbit/s LION [51] eingesetzt werden.

Analoge Überlegungen wurden für 10 bedeutendere Medienzugriffsprotokolle aus dem HS-Bereich durchgeführt. Mechanismen zur Steuerung des Übertragungsbeginns und -endes, zur Behandlung von Prioritäten, die Art der Adressierung, Datensicherheit und Fehlerbehandlungsverfahren waren dabei wichtige Kriterien. Durch die vielfältigen Varianten und großen Unterschiede zwischen den einzelnen Protokollen der Medienzugriffsschicht war hier keine reine Hardwarelösung, bei der lediglich bestimmte Parameter von außen kontrollierbar sind bzw. Teilfunktionen wahlfrei zu- und abgeschaltet werden können, möglich. Stattdessen wurde eine programmgesteuerte Steuerung gewählt, die durch einige generische Hardwarekomponenten (u.a. Adreßvergleich, zyklische Prüfsummenerstellung und -verifikation sowie Zähler und Zeitgeber) unterstützt wird. Wiederum wurde eine Aufteilung in Sende- und Empfangsrichtung eingeführt. Die Interaktion mit der Bitübertragungsschicht wird jeweils asynchron über eigene Warteschlangen realisiert. Kommunikation mit höheren Schichten wird über einen gemeinsamen Mehrtorspeicher mit eigener (komplexer) Steuerung abgewickelt. Verschiedene auf RISC-Prozessoren basierende Abschätzungen weisen die Machbarkeit bis hin zu Protokollen mit Bandbreiten nahe bei 1 Gbit/s nach.

4.1.3.2 Die logische Sicherungsschicht

Das Ziel eines Mitte der 80-er Jahre im T.J. Watson Forschungslabor der IBM durchgeführten Projekts war es, die bei FDDI zur Verfügung stehende Medienbandbreite von 100 Mbit/s oberhalb der logischen Sicherungsschicht 2b des OSI-RMs zur Verfügung zu stellen. Vorüberlegungen für die umfangreichste Protokollklasse 2 zeigten, daß selbst maschinennah programmierte Softwarelösungen im Verbund mit einer regulären Betriebssystemumgebung weit von dem angestrebten Ziel entfernt bleiben würden! So war eine Umsetzung in eine eigenständige Hardware unumgänglich [298].

Die große Zahl an möglichen internen Pfadvarianten - z.B. durch unterschiedliche Syntax und Semantik bei Adressen - erschwerte eine komplette Umsetzung in festverdrahtete Hardware so sehr, daß schlußendlich der Entwurf eines (programmierbaren) Spezialprozessorkerns plus umgebender Unterstützungskomponenten gewählt wurde. In einem ersten Prototypentwurf wurde dann schließlich statt des eigentlich geplanten anwendungsspezifischen Prozessorkerns einer der damals leistungsfähigsten und gerade erst verfügbaren Universalrechnerbausteine, der Intel80386 [397], eingesetzt.

Besonders herauszuheben ist die dominierende Bedeutung von protokollunabhängigen Faktoren wie Pufferverwaltung, Vorgehensweise bei den reinen Datenbewegungen und "sonstiger" Betriebssystemeinfluß. Die Autoren sahen keine Möglichkeit, mit einem (beliebigen) Betriebssystem das gesteckte Ziel von 100 Mbit/s oberhalb der Schicht 2b zu erreichen!

4.1.3.3 Die Transferschichten 3 und 4

Der bestimmende Gedanke beim Entwurf der Standardprotokolle war die Robustheit und die Universalität. Leistungsgesichtspunkte und Implementierbarkeit fanden dagegen kaum Beachtung. Ganz anders bei der Entwicklung von XTP (siehe 4.1.2.2): eine effiziente Implementierung in Hardware war von Anfang an eines der Hauptziele. Feste Steuerkopfformate, einfacher hardwaregeeigneter Prüfsummenalgorithmus, generelle Ausrichtung auf Vielfache von 64 Bit, Maßnahmen zum Ausgleich des unterschiedlichen Aufwands beim Senden bzw. Empfangen und eine Darstellung der Einzelfunktionen in einer Form, die direkt die implizite Parallelität sichtbar macht, sind einige der resultierenden Eigenschaften von XTP. Schon bald nach der Veröffentlichung erster XTP-Versionen wurde von Greg Chesson das Projekt einer zugehörigen Protokollmaschine (Protocol Engine, PE) begonnen [94, 274]. Die beteiligten Firmen SGI (Silicon Graphics, Incorporated, Mountain View, CA, USA) und PEI (Protocol Engine, Incorporated, Santa Barbara, CA, USA) entwarfen das in Bild 4.9 schematisch dargestellte Konzept.

Es fällt auf, daß weder explizite Schichten zu erkennen sind noch sonstige kommunikationsspezifische Baugruppen enthalten sind. Ein Großteil der Protokollmechanismen wird tatsächlich nicht durch eigene Hardwarekomponenten, sondern durch Software realisiert. Jede der vier Bausteine enthält bis zu zwei speziell entworfene RISC-Prozessorkerne mit jeweils zugeordneten optimierten vollkundenspezifischen Datenpfaden. Da die wesentlichen Funktionen des PE-Chipsatzes in ähnlicher Form auch in anderen Kommunikationssystemen notwendig sind und diese besonders gut den relativ geringen Anteil der reinen protokollbezogenen Aufgaben verdeutlichen, werden die Aufgaben der vier Chips explizit aufgelistet.

Der zentrale Speicherverwaltungsbaustein BCTL (Buffer Controller) hat die Funktionen:

- Speicherverwaltung (Reservieren, Belegen, Freigeben, Adressieren, falls notwendig regelmäßiges Auffrischen, Auffinden von unbelegten Speicherbereichen usw.)
- Sicherstellung und Durchführung des Zugriffverfahrens auf den Datenbus (Dbus)
- Verkettung von einzelnen Puffern zu logischen Pufferwarteschlangen
- Korrekte Behandlung von reihenfolgegestörten PDUs
- Gleichzeitige korrekte Verwaltung von logischen Ketten verschiedener Priorität

Der Schnittstellenbaustein zum Anwendungsrechner HPORT ist u.a. verantwortlich für:

- Das korrekte Segmentieren der zu übertragenden Benutzerdaten in Echtzeit inklusive
- Prüfsummenberechnung und -einfügung und Folgenummergenerierung
- Selbständige Verwaltung von Speicherblöcken des Anwendungsrechners
- Eigenständige Kontrolle des Hostbusses und bündelförmiger direkter Speicherzugriff (burst mode DMA)
- Korrektur der Oktett-internen Bitanordnung (big-/little-endian swapping)

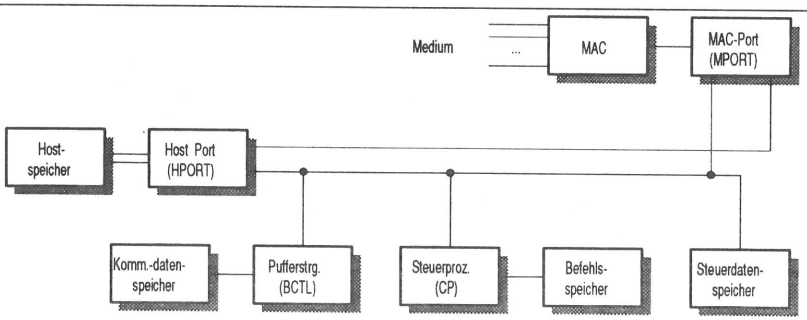


Bild 4.9: *Die Struktur des PE-Chipsatz-basierten Netzwerkcontrollers*

Die Aufgaben des Medienport Bausteins MPORT sind sehr vielfältig:

- Anpassung an die technologieabhängigen Schichten (beliebige Typen sind möglich)
- Kontrolliertes Empfangen verschiedener PDU-Formate, Prüfsummenverifikation
- Absuchen des Steuerkopfes mittels programmierbaren endlichen Automaten u.a. zur Prüfung der korrekten Syntax oder auch der semantischen Konsistenz
- Identifizierung des zugehörigen Kontexts
- Fehlererkennung
- Prioritätserkennung
- Überprüfung von Folgenummern
- Überwachung sowohl der Sende- als auch der Empfangsrate
- Anpassung der Bitanordnung innerhalb von Oktetts
- Adreßerkennung und -verarbeitung in Echtzeit
- Steuerkopfbearbeitung (in Echtzeit, d.h. vor Ankunft der nächsten PDU beendet!)
- Bereitstellung von ausreichend Zwischenspeicherkapazität zur Unterstützung der gebündelten Übertragung von Daten (burst mode transfer).

Der Verwaltungsprozessor CP hat neben schichtenspezifischen Managementfunktionen auch einige protokollorientierte Aufgaben. Im wesentlichen sind dies:

- Verbindungsauf- und -abbau (u.U. auf mehreren Schichten)
- Verwaltung der Verbindungsinformationen
- Erzeugen von Befehlsmeldungen auf dem Daten- und dem Kommandobus
- Erstellen von Quittungen und anderer notwendiger Meldungstypen.

Durch den weitgehenden Verzicht von festverdrahteten auf XTP ausgerichteten Hardwarekomponenten soll der PE-Chipsatz - mit gewissen Einschränkungen - inzwischen auch für andere Protokolle nutzbar sein. Eine interessante Eigenschaft der Protokollmaschine ist ihre Eignung sowohl für Endsysteme (in der beschriebenen Konfiguration, also mit mindestens einem MPORT und einem HPORT) als auch für Zwischensysteme wie Brücken oder Router. Im zweiten Fall werden statt eines HPORTs ein oder mehrere weitere MPORT-Bausteine an das interne Bussystem angekoppelt; die Firmware des Speicherverwaltungschips muß entsprechend geändert werden.

Weiterhin ist bemerkenswert, daß die Protokollmaschine konsequent auf die Optimierung der Anzahl abgearbeiteter Dateneinheiten ausgelegt ist und bei der Bewertung der Leistungsfähigkeit auf die Angabe irreführender Duchsatzwerte verzichtet wird, die nur mit Dateneinheiten großer Länge (8 KB und mehr) erreicht werden. Greg Chesson geht von überwiegend sehr kurzen Nachrichten in zukünftigen Kommunikationsnetzen aus. Zwischen 60 und 80% sind schon in heutigen Netzen ein üblicher Wert für den durchschnittlichen Anteil der kurzen Nachrichten (<128 Oktetts, [318]), durch vermehrte Dezentralisierung und neue Anwendungstypen wird sich nach Chessons Meinung dieser Anteil noch erhöhen!

4.1.3.4 Ein generisches Hardwarekonzept

Der Vorschlag [210] von Krishnakumar et. al. von den AT&T Bell Laboratorien in Murray Hill, NJ, USA, will dem Nachteil der Inflexibilität von Hardwarelösungen für Protokolle dadurch entgegenwirken, daß eine noch weitergehende Loslösung von den eigentlichen Protokolldetails vorgenommen wird:
Die Kernfunktionalität eines beliebigen Protokolls soll als eine Ansammlung endlicher Automaten (Finite State Machines, FSMs) modelliert und formal spezifiziert werden, welche untereinander mit synchronen Nachrichtenkonzepten kommunizieren. Dabei sollen Details wie Nachrichtenformate oder Zeitgeberfunktionen in dieser Modellierung nicht betrachtet werden, sondern parallel mit einer eigenständigen kontextfreien Syntax beschrieben sein. Das so entstandene Modell repräsentiert die Gesamtkoordination zwischen den verschiedenen beteiligten Protokollinstanzen.

Die grundlegende Idee ist nun, daß ein VLSI-System entworfen werden kann, welches in der Lage ist, die durch FSMs spezifizierten Kernfunktionen verschiedener Protokolle in effizienter Weise auszuführen. Zwangsläufige Unterschiede bei Details und Funktionalitäten niedriger logischer Abstraktionsebenen sollen durch äußere Unterstützungsbausteine - sogenannte Satelliten - behandelt werden. Für die Umsetzung der formal spezifizierten Automatenmodelle ist eine weitgehende Rechnerunterstützung (silicon compiler) vorgesehen. Die wesentlichen Blöcke der generischen PROVE-Architektur (PROgrammable Protocol VLSI Engine) sind in Bild 4.10 aufgeführt.

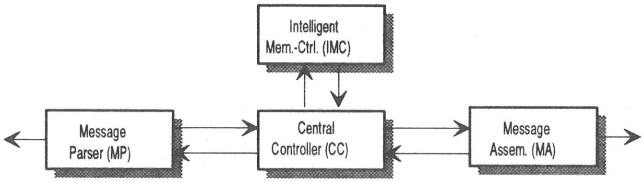


Bild 4.10: *Funktionsblöcke von PROVE*

Die zentrale Steuereinheit CCU (Central Controller Unit) ist umgeben von Blöcken zum Untersuchen empfangener Dateneinheiten (Message Parser, MP) und deren Zusammenstellung in Senderichtung (Message Assembler, MA) sowie einer intelligenten Speicher- und Schnittstellenverwaltung IMC (Intelligent Memory controller).

Während der Untersuchungen empfangener PDUs bzw. hauptsächlich deren Steuerköpfe durch den Block MP werden jeweils bei Erkennung bestimmter Merkmale Startmarkierungen (tokens) generiert, die in der CCU entweder direkt zugehörige Übergänge in den einzelnen Automaten auslösen können oder aber vom MP gesammelt werden, bis die Korrektheit der kompletten PDU festgestellt wurde. Der Block MA hat die umgekehrte Aufgabe zu erfüllen, er bekommt von der CCU Markierungen übergeben, die ihm gestatten, die zu sendende Dateneinheit schrittweise weiter zu vervollständigen.

Geht man von der - in vielen Protokollen gültigen - Annahme kontextfreier Formate aus, ergeben sich für MP und MA effiziente Architekturen, die als spezialisierte FSMs oder auch als Einfachstprozessor bezeichnet werden können und sich durch eine geringe Komplexität auszeichnen. Kontextfreiheit impliziert, daß ein kontinuierliches Abarbeiten eines Eingangsdatenstromes ausschließlich mit dem Wissen über den derzeitigen Zustand und der nächsten Eingabe möglich ist, also kein Verweis auf schon empfangene Daten - und eine damit verbundene Zwischenspeicherung - notwendig ist. Durch die Forderung nach Flexibilität, d.h. Parametrisierbarkeit, ergibt sich die Notwendigkeit von Mikrocodespeichern einerseits sowie entsprechenden Schaltern (meist durch Multiplexer zu realisieren) und Registern andererseits.

In Bild 4.11 aus [211] sind die FSMs und die Verbindungswege innerhalb der CCU zu erkennen. Der Meldungsaustausch zwischen den FSMs oder zwischen CCU und den Satelliten und die Auflösung eines dabei u.U. entstehenden Ressourcenkonflikts wird von internen Zuteilungseinheiten ARB gesteuert. Die grundsätzliche Funktionsfähigkeit des PROVE-Ansatzes wurde durch die Realisierung einiger Schicht 2- und Schicht 3-Protokolle überprüft. Es ergaben sich zwei Erkenntnisse:

1. Aspekte der Speicherverwaltung bekommen in schneller werdenden Kommunikationssystemen eine immer stärkere Bedeutung. Ihnen muß mindestens die gleiche Aufmerksamkeit zuteil werden, wie den eigentlichen Protokollaspekten.
2. Die Stärken dieses Lösungsansatzes liegen in der großen Flexibilität, den zwangsweise vorhandenen formalen Spezifikationen und der (teil-)automatisierten Umsetzung in VLSI. Die inherente erhöhte Leistungssteigerung durch den Einsatz von Hardware wird allerdings weitgehend durch die Maßnahmen zur Ermöglichung der Flexibilität ausgeglichen.

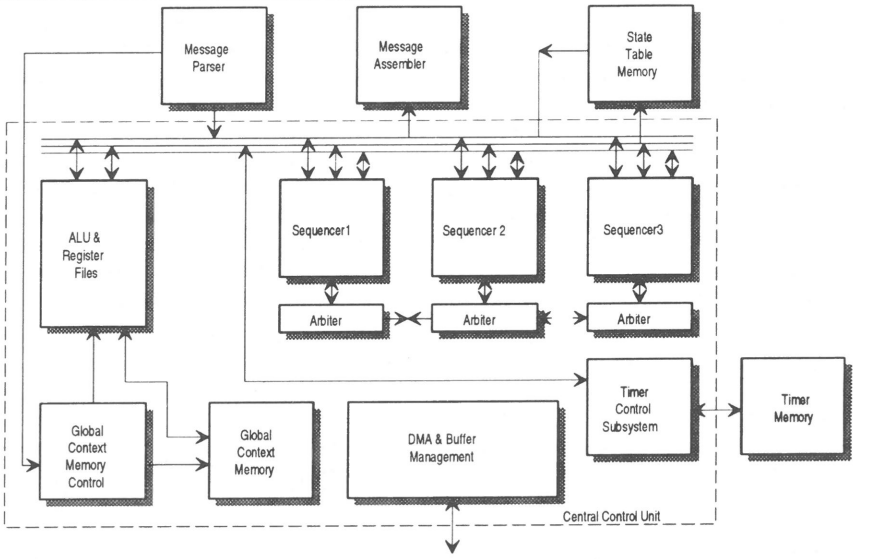


Bild 4.11: *Aufbau der zentralen PROVE-Steuereinheit*

Wegen der wesentlich stärkeren Nutzung von Parallelität und dem Einsatz von vielen Prozessoren wird die PROVE-Vorschlag im nächsten Abschnitt nochmals behandelt.
Die Synthese von Hardware aus formalen Protokollbeschreibungen ist auch Inhalt von [47, 254].

4.1.4 Multiprozessorsysteme für Protokollbearbeitung

Ein zentrales Prinzip von Systemen mit mehreren Verarbeitungseinheiten (VE) lautet:

> Der Einsatz von mehreren VE resultiert nur dann in einem Gewinn an Leistungsfähigkeit des Gesamtsystems, wenn die zu bearbeitende Aufgabe inherente Parallelitäten aufweist.

Das Lokalisieren von nutzbarer Parallelität steht daher vor der Umsetzung in Multiprozessorsysteme.
Zitterbart hat in ihrer Dissertation [329] grundsätzliche Überlegungen zur Thematik der Parallelität in Kommunikationssystemen durchgeführt und deren mögliche Nutzung anhand einer exemplarischen Implementierung der Schichten 3 und 4 des OSI-RMs auf einem Transputernetzwerk aufgezeigt.

Bevor einige ausgewählte kommunikationsorientierte Multiprozessorsysteme beschrieben werden, werden zunächst wesentliche allgemeingültige Aussagen und Begriffe zusammengefaßt.

4.1.4.1 Strukturelle Parallelität

Die hierarchische, geschichtete Strukturierung von Kommunikationssystemen legt direkt eine Form der zeitlichen Parallelität nahe: wenn jeder Schicht eine unabhängige Verarbeitungs-Einheit VE zugeordnet wird, kann in jeder Schicht zur gleichen Zeit jeweils eine PDU bearbeitet werden. Nach Fertigstellung der schichtenspezifischen Funktionalität wird die PDU an die Nachbarschicht weitergegeben. Aufgrund der direkten Ableitung dieser Form der Parallelität aus der vorhandenen Struktur führt Zitterbart in [329, 331] die Bezeichnung strukturelle Parallelität ein. Die Zuordnung einer VE zu einer Protokollschicht ist nicht bindend festgelegt, eine Erweiterung auf Unterschichten des OSI-RMs ist durchaus möglich.

Eine Eigenschaft dieser Fließbandverarbeitung der Schichten (layer pipeline), die sich in realen Systemen

oft negativer auswirkt als zunächst erwartet, ist, daß die Leistungsfähigkeit des gesamten Systems, vom "schwächsten Glied der Kette" abhängt. Denn die Zeitdauer t_{next}, nach der alle n PDUs jeweils an die nächste Fließbandstation weitergegeben werden können bzw. die erste Stufe eine neue PDU aufnehmen kann, bestimmt sich zu :

$$t_{next} = MAX(t_{Schicht1}, t_{Schicht2}, \ldots t_{Schichtn}) \tag{4.16}$$

wobei $t_{Schicht\ i}$ die Verarbeitungszeiten der einzelnen Fließbandstationen sind. In der Regel wird dabei nicht ausgenutzt, daß t_{next} abhängig von den internen Arbeitsschritten selbst variieren kann. Statt eines lokalen Maximums wird für t_{next} das absolute Maximum aller möglichen Varianten gewählt, und das Weiterschalten läuft dann periodisch immer nach der Zeitdauer t_{next} ab ($\Rightarrow$ Schichtentakt). Nur bei ausgeglichenen, d.h. nicht zu unterschiedlichen, Zeiten der einzelnen Schichten ist ein akzeptabler Ausnutzungsgrad aller VEs und eine daraus resultierende Leistungssteigerung des Gesamtsystems möglich.

In der bisher beschriebenen Form durchlaufen PDUs das Multiprozessorsystem in beiden Richtungen. Um dadurch zwangsläufig entstehende Leistungseinbußen zu verhindern, ist es naheliegend, jeweils eine eigene Fließbandverarbeitung für Sende- und Empfangsrichtung einzuführen. Bekanntlich sind die beiden Richtungen bei verbindungsorientierten Protokollen nicht vollständig unabhängig voneinander. Der Empfang von Kontrollinformationen, z.B. Quittungen und Parameter zur Regulierung des Datenflusses, kann das Verhalten der Sendeseite erheblich bestimmen. Innerhalb einer Zwischeninstanz müssen sogar komplette PDUs zwischen dem Empfangs- und Sendeteil zumindest einer Schicht transportiert werden. Allgemein ergibt sich eine logische Sichtweise gemäß Bild 4.12

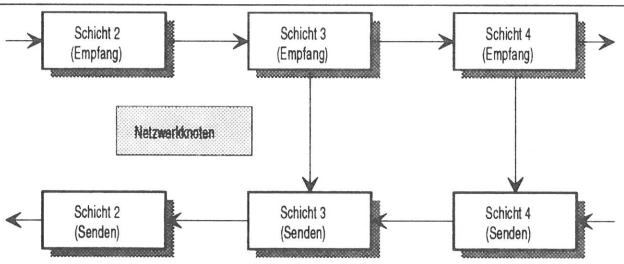


Bild 4.12: *Strukturelle Parallelität*

Drei Beispiele für Arbeiten, welche die geschilderte Form der strukturellen Parallelität einsetzen oder zumindest als Ausgangspunkt für Erweiterungen heranziehen, seien erwähnt.

Jensen und Skov schlagen in [198] den Einsatz identischer Prozessoren mit einer konventionellen Architektur für alle technologieunabhängigen Schichten, d.h. 2b - 7, als Ergänzung zu ihren Spezialprozessoren für die unteren Schichten (siehe voriges Abschnitt) vor. Eine abstrakte Modellierung einer beliebigen OSI-Schicht wird aufgestellt, welche aus fünf generischen Prozessen besteht: SAP nach oben und unten, Ereignisverwaltung, Zeitüberwachung sowie dem eigentlichen Schichtenkern. Die prinzipielle Notwendigkeit von speziellen, mit den Verbindungskanälen eines Transputers [401] vergleichbaren, Kommunikationskanälen wird ebenso abgeleitet wie eine Liste von grundsätzlichen Anforderungen an die Prozessoren der höheren Schichten. Zur Vermeidung aufwendiger Datentransfers zwischen den Schichten ist ein gemeinsamer Speicher mit entsprechend vielen (logischen) Zugängen vorgesehen, welcher durch Zeitmultiplextechnik und einer intelligenten Priorisierung mit herkömmlichen RAM-Bausteinen realisiert werden soll. Im Gegensatz zu den fundierten Aussagen zu ihren schon implementierten Bausteinen lassen Jensen und Skov bei den höheren Schichten viele Aspekte unberücksichtigt.

Die Vorteile der Wirtschaftlichkeit, der einheitlichen Entwicklungsumgebung und wesentlich kürzerer Entwicklungszeiten beim Einsatz gleichartiger Bausteine wird bei einer japanischen Arbeitsgruppe von der resultierenden Leistungseinbuße vollständig ausgeglichen. In [300] schlägt dieses Team deshalb den

Einsatz von jeweils für jede Schicht speziell entwickelten Bausteinen vor. Eine exemplarische Realisierung der Schichten 2b bis 4 sowie erste vergleichende Aussagen zu deren Leistungsfähigkeit werden beschrieben. Wegen den umfassenden, nicht nur auf Protokollverarbeitung ausgerichteten, Eigenschaften dieser Arbeit wird sie ausführlicher in Abschnitt 4.2.2.4 behandelt.

Ein italienisches Team aus Catania erwähnt in [91] ebenfalls eine vollständige, alle 7 Schichten des OSI-RMs umfassende Architektur, bei der die Daten in einem für alle Schichten zugänglichen gemeinsamen Speicher abgelegt werden. Tatsächlich wird dann lediglich eine interne (räumliche) Parallelisierung der Sicherungsschicht 2b durch Einführung gleichartiger Einheiten zur Verarbeitung jeweils einer Verbindung beschrieben und modelliert. Im Vordergrund steht die Modellierungstechnik durch eine Unterform der Petri-Netze (TTPN, Timed Token Petri Nets [2]) und der Einsatz eines eigenen Werkzeuges.

Auch Ulrich und Dietsch von der Universität Erlangen behalten die OSI-Schichtung bei. Zum Ausgleich der jeweiligen Schichtenkomplexitäten führen sie in [129, 303] analog dem italienischen Vorschlag schichteninterne Vervielfachungen von VEs - hier:Transputer vom Typ T414 - ein. Neben einer rein verbindungsorientierten Zuordnung der einzelnen Prozessoren und einer teilweisen Auftrennung in Sende- und Empfangsrichtung werden auch Einzelfunktionen wie Verbindungsverwaltung oder Prüfsummenberechnung an explizite Prozessoren vergeben. Darüberhinaus wird über erste Ansätze zur extensiveren Nutzung schichteninterner Parallelität (welche nachfolgend eingeführt wird) innerhalb der Schichten 6 und 7 berichtet. Zur Entlastung des gemeinsamen Speichers werden die transputereigenen Kommunikationskanäle (links) intensiv für die Interaktion zwischen benachbarten Schichten genutzt.

Es gibt weitere Arbeiten, die auf dem Prinzip der Fließbandverarbeitung von einzelnen OSI-Schichten basieren. Einige davon werden wegen ihres deutlich über reine Protokollfunktionen hinausgehenden Charakters im Abschnitt über Gesamtbetrachtung von Kommunikationssystemen (4.2.2) behandelt.

4.1.4.2 Schichteninterne Parallelität

Alle Ansätze zur Nutzung struktureller Parallelität machen zwei wesentliche Problembereiche deutlich:

1. Die Schichten des OSI-RMs umfassen stark unterschiedliche Komplexitäten. Das Ausbalancieren der Unterschiede in den Bearbeitungseinheiten ist damit ein erhebliches Problem.
2. Bei der Umsetzung von struktureller Parallelität muß den Mechanismen zur Übergabe von Daten und Kontrollnachrichten zwischen benachbarten Blöcken besondere Beachtung gewidmet werden. Nur wenn die Bearbeitungszeit der Stufen deutlich größer als die zur Übergabe benötigte Zeit ist, resultiert aus dieser Form der Parallelisierung eine merkliche Leistungsverbesserung.

Der Punkt 1 läßt sich durch eine feinere Granularität verbessern. Die Schichten können dazu in weitere Unterschichten aufgeteilt werden. Schließlich kommt man zu Funktionen, die nicht weiter unterteilt werden können. Diese atomaren Funktionen werden sequentiell nacheinander abgearbeitet. Durch Untersuchungen zu deren tatsächlichen Abhängigkeiten können voneinander unabhängige Nebenläufigkeiten festgestellt und durch entsprechende parallele Verarbeitung zur Leistungssteigerung ausgenutzt werden. In der Praxis können sich komplizierte Präzedenzgraphen mit sequentiellen und nebenläufigen Zweigen, aber auch mit Verzweigungen ergeben (siehe Bild 4.13).

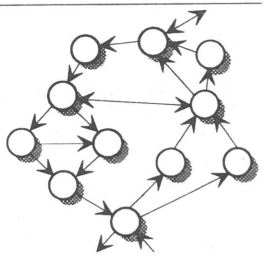


Bild 4.13: *Präzedenzgraph einer Protokollschicht*

Die Koordinierung bzw. Synchronisierung der einzelnen Zweige eines solchen Graphen sind schon auf rein logischer Ebene nichttriviale Aufgabe und daher von akademischem Interesse. In einer praktischen Realisierung würden darüberhinaus noch weitere Probleme resultieren. Nicht nur Zeitpunkt und Form von Synchronisationsereignissen spielen eine Rolle, sondern Fragen zur eventuell notwendigen physikalischen Zwischenspei-

cherung, zu den real einsetzbaren Kommunikationsmechanismen oder auch zur Umsetzung von theoretischen Zeitvorgaben in eine reale Implementierung.

Bezieht man zusätzlich die Möglichkeit der *räumlichen* Parallelisierung, also das Vervielfachen ein und derselben Teilfunktion einer Schicht, mit ein, so kann man zwar immer näher an eine theoretisch optimierte Ausbalancierung der Auslastung einzelner VE herankommen, das Bestimmen einer solchen Konfiguration kann jedoch schwierig oder, bei Berücksichtigung dynamischer Abläufe, sogar ganz unmöglich werden. In der Praxis wird daher nicht die Ebene der atomaren Funktionen zur Konfiguration von Multiprozessorsystemen herangezogen, sondern Funktionensgruppen. Gute Beispiele sind:

- Empfang von Daten der Nachbarschicht
- (Teil-)Auswertung des Steuerkopfes
- Protokollbearbeitung einer Dateneinheit
- Übergabe an die nächste Schicht.

Genau diese Ebene der Granularität wird z.B. von einer Forschergruppe des IBM-Forschungslabors Yorktown ausgenutzt. In [47] berichten sie neben einem Werkzeug zum Umsetzen formaler Beschreibungen in VLSI-Bausteine auch über die interne Struktur ihres Multiprozessorsystems. Die Bearbeitung innerhalb einer Protokollschicht wird dabei in drei Schritte aufgeteilt (für Empfang aufgeführt):

- Der Kopfprozessor verarbeitet den Steuerkopf soweit, bis die grundsätzliche Konsistenz und die zugehörige Verbindung festgestellt wurden. Interessanterweise werden - zumindest theoretisch - auch Steuerköpfe variabler Länge unterstützt.
- Zur eigentlichen Protokollverarbeitung wird die Dateneinheit danach an einen Verbindungsprozessor weitergeleitet, wobei pro aktiver Verbindung ein eigener Prozessor vorhanden ist.
- Der dritte Prozessortyp ist für die Übergabe der fertig bearbeiteten Dateneinheit an die darüberliegende Schicht zuständig.

Professor Schwartz und zwei Assistenten von der New Yorker Columbia Universität kommen bei einer Untersuchung der Optionen für die Ausnutzung von Parallelität in Protokollen [196] im allgemeinen und OSI TP4 im besonderen auf eine Lösung fast gleicher Granularität. Wesentlicher Unterschied zum Vorschlag von Abu-Amara und seinen Kollegen ist die Aufhebung der Zuordnung zu einer bestimmten Verbindung. Jede empfangene PDU wird unabhängig von ihrem Inhalt einer beliebigen freien VE zugeteilt. Statt eines Kopfprozessors wird der erste Prozessor "lediglich" zur schnellen Verteilung auf die einzelnen PDU-Prozessoren benötigt. Der enorme Vorteil der deterministischeren Auslastung einzelner VEs und der Unabhängigkeit von der dynamischen Anzahl gerade aktiver Verbindungen muß allerdings mit einem Aufwand für die zusätzlich erforderlichen Mechanismen zum Sicherstellen der Konsistenz der Verbindungsparameter sowie der Reihenfolgesicherung innerhalb einer Verbindung bezahlt werden. Bemerkenswert an diesem Ansatz sind zum einen die verblüffend niedrige Anzahl von Instruktionen, die für die TP4-Implementierung gebraucht wurden, zum anderen die hohen im fehlerfreien Fall erreichbaren Gesamtdurchsätze von bis zu einem Gbit/s beim Einsatz von nur 8 einfachen VE!

4.1.4.3 Schichtenübergreifende Parallelität

Löst man sich von der durch das OSI-RM vorgegebenen Sichtweise der Schichtung, erschließt sich eine weitere Dimension für die Nutzung von Parallelität. Die Überlegungen zur optimalen Konfiguration eines Multiprozessorsystems müssen dann die Gesamtheit aller Teilfunktionen des gesamten Kommunikationssystems miteinbeziehen. Zitterbart hat für diese Form den Begriff funktionelle Parallelität oder auch funktionsbezogene Parallelität geprägt. In diesem Bereich liegt auch der Schwerpunkt ihrer Arbeit.

Einige Arbeiten seien genannt, die auf dem Prinzip der schichtenübergreifenden Denkweise beruhen: Rupprecht et al. schlagen eine Architektur vor, die aus gleichartigen VE, den Aktionseinheiten AE, besteht, welche flexibel für Aufgaben jeder Schicht eingesetzt werden können [179, 264]. Ein regulärer

Durchlauf einer PDU durch den gesamten Protokollturm eines Kommunikationsteilnehmers, d.h. die reihenfolgerichtige Zuweisung sowohl von Funktionen zu AE als auch von AE zu PDUs, wird durch *Entscheidungs*einheiten vorgenommen. Die Kommunikation zwischen den zwei Modultypen wird über zwei Warteschlangen, eine für Aufträge, eine für Rückmeldungen (siehe Bild 4.14). Analog zu den für die Kategorisierung von Rechnerarchitekturen gebräuchlichen Abkürzungen SISD bis MIMD (siehe z.B. [4, 9, 10, 27, 43] gibt Rupprecht der Architektur die Bezeichnung MDMA (Multiple Decision - Multiple Action, mehrfache Entscheidungen - mehrfache Aktionen).

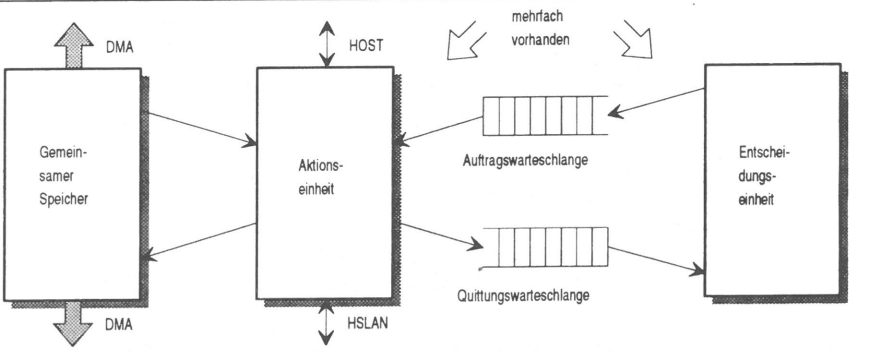


Bild 4.14: *Grundprinzip der MDMA-Architektur aus [264]*

Eine AE besteht dabei aus einem autonomen Prozessor mit eigenem Programmspeicher. In einer Prototypversion kam ein gemeinsamer Speicher zum Einsatz. Einen zentralen Stellenwert in Rupprechts Arbeiten nimmt die formale Spezifikation der wichtigsten Abläufe innerhalb des MDMA-Systems mittels Petrinetzen dar. Darauf aufbauend werden Simulationen und abschätzende Berechnungen durchgeführt. Berichte über konkrete Implementierungen von mehrschichtigen Protokolltürmen oder Aussagen zur Anwendbarkeit der Architektur auf OSI-Protokolle - und dabei insbesondere Protokolle der Schichten 2b und höher - werden nicht gegeben, so daß einige Fragen unbeantwortet bleiben.

Der im Abschnitt über leistungssteigernde Maßnahmen in den anwendungsbezogenen Schichten schon erwähnte Vorschlag von Haas [168, 169] geht noch einen Schritt weiter. Bild 4.15 zeigt, daß ein Kommunikationssystem bei Haas aus drei Blöcken besteht :

1. Netzwerk-Zugangskontrolle NAC (Network Access Control) für die OSI-Schichten 1 bis 3.
2. Ein Block, den Haas als Kommunikationsschnittstelle CI bezeichnet (Communication Interface) und der die OSI-Schichten 4 bis 6 umfaßt.
3. Die anwendungsspezifische Schicht 7.

Innerhalb des Blockes CI werden alle Funktionen der Schichten 4 - 6 eines Kommunikationssystems gemeinsam betrachtet und horizontal, ausschließlich nach - möglichst unabhängigen - Funktionalitäten, untergliedert. Jede Funktion wird individuell so optimal implementiert wie möglich. Haas schließt dabei anwendungsspezifische Hardware nicht aus. Als mögliche Blöcke der Empfängerseite seiner HOPS-Architektur (Horizontally Oriented Protocol Structure) gibt er an:

- Fehlerüberwachung
- Behandeln und Initiieren wiederholter Übertragungen
- Verbindungsverwaltung (falls benötigt)
- Reihenfolgesicherung

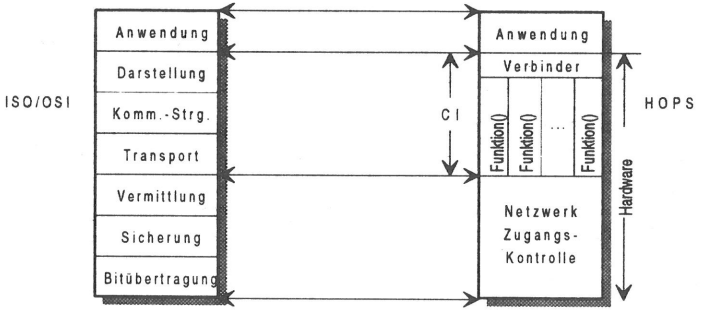


Bild 4.15: *Die Schichtung bei HOPS*

- Steuerung des Informationsflusses
- Adressierung (aktiv und passiv)
- Darstellungsform
- Verwaltung von Sitzungen (session control)
- Überwachung und Behebung von Blockierungssituationen (congestion control)

Selbst wenn eine Funktion vom Resultat einer anderen abhängt, kann sie zunächst schon ausgeführt werden, wartet allerdings dann auf die Freigabe. Sollte diese nicht eintreffen, muß der ursprüngliche Zustand wieder hergestellt werden, d.h. das Ergebnis ist solange lokal abgelegt, bis die Freigabe durch die andere Funktion erfolgt. Typisches Beispiel hierzu sind Maßnahmen zur Reihenfolgesicherung, welche natürlich erst dann wirksam werden dürfen, wenn der Empfang einer fehlerfreien Dateneinheit von der Fehlerüberprüfungseinheit gemeldet wird. Die Ausführung kann jedoch auch schon vorher stattfinden.

Der herausragende Vorteil der Architektur ist der hohe Grad an nutzbarer Parallelität, der dazu führt, daß der Zeitbedarf für die Gesamtheit aller Funktionen nicht mehr der Summe der Einzelzeiten entspricht, sondern durch die langsamste Einzelfunktion bestimmt ist. Weitere Vorteile sieht Haas in:

- der Reduktion funktionaler Redundanz
  (Die OSI-Schichtung führt zu vielen mehrfach ausgeführten Funktionen),
- der Möglichkeit, nur genau die Funktionen zu benutzen, die wirklich gebraucht werden,
- dem weitgehenden Wegfall von schichteninterner Zwischenspeicherung und Meldungsaustauschen zwischen Schichten.

Analog zu den Varianten innerhalb von vertikalen Protokollschichten können die Wahlmöglichkeiten durch die Einführung von Varianten innerhalb eines Funktionsblockes noch zusätzlich erweitert werden. Beim Block zur Auslösung von Wiederholungssendungen können beispielsweise verschiedene selektive Anforderungsverfahren angeboten werden. Durch die freie Zusammenstellung von Funktionen und Verfahren, welche auch unter der Bezeichnung Protokollmenü bekannt ist, kann eine optimale Anpassung an die Anforderungen verschiedener Anwendungen erreicht werden. Stattet man den Netzzugangscontroller mit eigener Intelligenz aus, so kann die Wahl u.U. dynamisch den sich ändernden Umgebungsbedingungen angepaßt werden, ohne daß die höheren Schichten oder der Benutzer darin involviert sein müssen.

Das gesamte Modell erscheint einleuchtend und zukunftsweisend. Es bleiben jedoch viele implementierungsbezogene Probleme wie Synchronisation der Einzelfunktionen, Verwaltung gemeinsamer Ressourcen oder Interaktion mit dem Anwendungsrechner und dessen Betriebssystem offen. Von großer Bedeutung für das HOPS-Konzept ist die eingesetzte Verwaltung und die Aufteilung des Speichers. Leider finden sich auch zu diesem Bereich keine detaillierteren Angaben in [168, 169].

Da XTP-Funktionen der Schichten 3 und 4 abdeckt, sind auch zwei XTP-Arbeiten in diesem Abschnitt über schichtenübergreifende Parallelität zu erwähnen. Zur Spezifikation von XTP [396] wurden voneinander unabhängige Automaten eingesetzt. Sowohl der Franzose Diot [133] als auch eine Karlsruher Gruppe um Braun [78] versuchen, die resultierenden Automaten direkt auf eigenständige Prozesse oder in letzter Konsequenz auf jeweils eigene Prozessoren abzubilden. Beide setzen Transputer ein.

4.2 Berücksichtigung der Umgebung

Sobald die Protokollfunktionalitäten in eine reale Implementierung umgesetzt werden, ergeben sich neue Problembereiche. Auf dem Gebiet der nicht protokollorientierten Funktionen von Kommunikationssystemen sind allerdings bei weitem nicht soviele Arbeiten im Gange, wie bei den rein logischen Protokollfunktionen. Neben wenigen theoretischen Arbeiten zu Einzelaspekten [54, 105, 116, 305, 323, 327] sind einige Berichte über die Implementierungen kompletter Kommunikationssysteme zu nennen [64, 81, 130, 151, 187, 322]. Letztere umfassen natürlich auch die logische Protokollfunktionalität.

Zunächst werden Verbesserungen von Teilfunktionen behandelt, anschließend werden einige Komplettsysteme mit einer Fokussierung auf protokoll*un*abhängige Unterstützungsfunktionen beschrieben.

4.2.1 Betrachtung einzelner Aspekte

Bereiche mit großem Einfluß auf die gesamte Leistungsfähigkeit eines Kommunikationssystems sind:

- Die Wahl der Speichertypen, sowie deren Anordnung und Verwaltung
- Das Abbilden von Funktionen auf Prozesse und die Behandlung von Prozeßwechseln
- Die Realisierung von Zeitgebern
- Die Komplexität von Schnittstellen

Lösungsansätze für jeden der genannten Bereiche werden im folgenden vorgestellt.

4.2.1.1 Speicher- und Busanordnungen

Im Kapitel 3 wurde der Einfluß der Bus- und Speicheranordnung schon mit Zahlenbeispielen verdeutlicht. Viele Komponenten brauchen "eigenen" Speicher. Nicht nur zum gegebenenfalls schichtenspezifischen Ablegen der zu übertragenen Nutzdaten, auch für die Interaktion mit anderen Komponenten, als lokaler Speicher für Programme und Konfigurationsdaten, sowie als Basis für die diversen Tabellen und Listen. Eine Lösung ohne jegliche Zugriffskonflikte wäre, entsprechend viele Busse und Speicher vorzusehen. Trotz dem grundsätzlichen Bestreben der Minimierung von funktionell notwendigen Speichern würde dieses Vorgehen jedoch zu einer unakzeptablen Zahl von Bussen und Speicherbausteinen führen.

Das andere Extrem ist die Einführung eines gemeinsamen Speichers, wie es auch von vielen Forschergruppen propagiert wird [78, 91, 129, 133, 159, 198, 264, 300]. Aus logischer Sicht stellt ein gemeinsamer Speicher die ideale Lösung zur Reduktion der aufwendigen Datenübergabe zwischen den Schichten dar. Es gibt nur wenige Protokollfunktionen, die einen Zugriff auf die Nutzdaten erfordern. Könnten diese gesondert bearbeitet werden, bräuchten die Daten nur einmal eingelesen und ausgelesen werden. In Empfangsrichtung würden Daten von der Medienzugriffsschicht in den zentralen Speicher geschrieben werden und idealerweise erst von der Anwendungsschicht wieder ausgelesen werden. Die Datenübergabe zwischen den Schichten könnte dann auf den - meist ebenfalls über den gemeinsamen Speicher ausgeführten - Austausch von Referenzen auf die relevanten Datenbereiche reduziert werden.

Aber selbst wenn ausschließlich die minimal notwendigen zwei Zugriffe pro Oktett einer Dateneinheit betrachtet werden, also keinerlei Zugriffe auf Steuerköpfe oder zu anderen Zwecken eingeplant werden, kommt man trotzdem schnell an die Leistungsgrenze heutiger Speicherbausteine. Die folgende Überle-

gung für eine der beiden Kommunikationsrichtungen soll diesen Sachverhalt verdeutlichen:

Es sei n die PDU-Länge in Oktetts, w die Breite des Speichers (auch in Oktetts) und B die (Nutz-)Bandbreite des Mediums (in bit/s). Für die Anzahl A von notwendigen Speicherzugriffen für eine n Oktett lange PDU und für die Übertragungszeit t_T der PDU gilt mit den gemachten Vereinfachungen:

$$A = \frac{2 \bullet n}{w} \qquad (4.17) \qquad t_T = \frac{8 \bullet n}{B} \qquad (4.18)$$

Die maximal zulässige Zykluszeit t_z des Speichers ergibt sich damit zu:

$$t_z = \frac{t_T}{A} = \frac{4 \bullet w}{B} \qquad (4.19)$$

Bild 4.16 zeigt die maximal zulässigen Zykluszeiten über der Medienbandbreite (Parameter w). Für 100 Mbit/s müßten bei oktettweisem Speicherzugriff schon sehr schnelle Bausteine mit Zykluszeiten t_Z unter 40 ns eingesetzt werden. Selbst bei 16 bit Speicherbreite ergeben sich immer noch Anforderungen von 80 ns. Die gemachten Überlegungen gelten wie gesagt nur für einen Halbduplex-Betrieb.

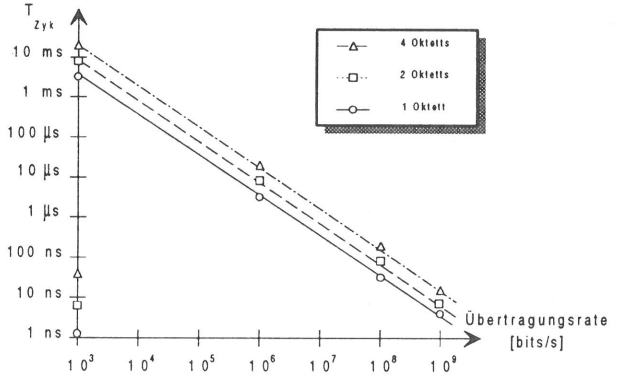


Bild 4.16: *Maximal zulässige Zykluszeiten in Abhängigkeit von der Bandbreite*

Schon diese - unter idealisierten Annahmen - notwendigen Zeiten verdeutlichen die dringende Notwendigkeit von verbesserten Speicherkonzepten. Möglichkeiten sind das Vergrößern der Speicherbreite oder die Einführung von Speicherverschränkung (address interleaving oder bank switching), bei dem aufeinanderfolgende Bitblöcke der Breite w - im folgenden *Wort* genannt - zyklisch in mehrere (z.B. k) physikalische Speicher eingeschrieben werden. Wird Wort i in den Speicher 3 eingeschrieben, so wird das Wort i+1 in den 4., Wort i+2 in den 5., das Wort i+k-2 in den 1. und das Wort i+k wieder in den 3. Speicher eingeschrieben. Die Bandbreiten- und Zykluszeitanforderung jedes einzelnen Speichers werden um den Faktor k reduziert. Die wahlfreie Adressierung spezifischer Worte wird jedoch aufwendiger.

Eine weitere Möglichkeit besteht durch die Verwendung von Speicherbausteinen mit mindestens zwei unabhängigen physikalischen Zugängen (dual/multi ported memories). Solche Bausteine sind sowohl nur mit relativ kleinen Größen verfügbar, als auch deutlich teurer als Speicher mit nur einem Eingang. Zur Reduzierung der notwendigen Busbandbreiten und zur einfacheren Zugriffssteuerung sind Mehrtorspeicher trotzdem Bestandteil vieler Kommunikationssysteme. Die genannten Nachteile werden dadurch umgangen, daß intern zwar Eintorspeicher eingesetzt werden, aber durch Zeitmultiplex extern der Anschein eines - mindestens k-fach langsameren - k-Torspeichers erweckt wird. Die Zugriffssteuerung ist alles andere als trivial, denn die einfache zyklische Zuteilung - pro Umlauf bekommt jedes Tor das Zugriffsrecht für genau eine vereinbarte Datenmenge - wird vielen zeitkritischen Abläufen nicht gerecht, so daß Prioritätsmechanismen und variierende Zuteilungszeiten erforderlich werden.

Kommunikationssysteme sind durch überwiegend lineare Datenflüsse gekennzeichnet. Die einzelnen PDUs laufen sequentiell von einer Komponente zur nächsten. Ideale Bausteine für solche sequentiellen Abläufe sind FIFO-Speicher (First In - First Out). Da für die FIFO-Bausteine ähnliches gilt wie bei den Mehrtorspeichern, also geringe Größe und teuer, werden FIFOs bestenfalls für Teilbereiche, z.B. Warteschlangen für den Kommandoaustausch zwischen Komponenten, Ausgleichspuffer u.ä., jedoch nicht für die Speicherung der Nutzdaten eingesetzt.

Ein Kompromiß zwischen der Größe und Wirtschaftlichkeit herkömmlicher Eintorspeicher und der Flexibilität sowie höheren Gesamtbandbreite der Speicher mit zwei unabhängigen wahlfreien Zugängen bilden Videospeicher. Diese VRAM-Bausteine bieten gleichzeitig eine serielle und eine parallele Zugriffsmöglichkeit [114, 255]. Der serielle Zugang wird beispielsweise in Empfangsrichtung benutzt, um die Daten von der Medienzugriffseinheit in den Datenpufferspeicher einzuschreiben. Dazu werden zunächst einige Oktetts in ein Schieberegister geschoben und sobald dies voll ist, in einem einzigen Speicherzyklus - welcher dem parallelen Zugang vorenthalten wird - in eine Speicherspalte übertragen. Der parallele und wahlfreie Zugang wird für Zugriffe auf einzelne Oktetts verwendet. Beim seriellen Zugang entfällt der beträchtliche Aufwand für das Adressieren und Dekodieren der einzelnen Oktetts, so daß leicht - auf einzelne Oktetts umgerechnete - Zykluszeiten unter 50 ns - statt etwa 200 ns am parallelen Port - erreicht werden können. Durch Kaskadierung solcher VRAM-Bausteine kann die Schieberegisterlänge an äußere Gegebenheiten, z.B. feste bzw. quantisierte PDU-Längen oder optimale Blockgröße für einen bündelförmigen direkten Speicherzugriff beim AR (block-/burst-mode DMA), angepaßt werden.

Das bisher Gesagte läßt sich analog auf Bussysteme übertragen. Insgesamt sollte genau überlegt werden, ob, und falls ja, wo Speicherzugriffe stattfinden sollen:

- Eine gewisse Auftrennung funktionell unabhängiger Speicher- und Busbereiche erscheint trotz des damit verbundenen Verdrahtungs- und Verwaltungsaufwandes in Hochgeschwindigkeitssystemen unumgänglich zu sein.
- Sind mehrere Prozessoren vorhanden, so sollte jeder einen lokalen Speicher für eigene Programme und Daten besitzen.
- Eine physikalische Aufteilung in Sende- und Empfangsrichtung ist ebenfalls angeraten.
- Wenn eine Auftrennung in Nutzdaten- und Kontrollfluß möglich ist, sollte diese auch vorgenommen werden.
- Gemeinsame Speicher nur aus Bequemlichkeit bzw. zur Vereinfachung der logischen Sichtweise sind zu vermeiden. Es sollten nur die funktionell unumgänglichen Zugriffe auf gemeinsame Ressourcen erfolgen, und Daten sollten - soweit als möglich - nur genau dort (lokal) gespeichert werden, wo sie auch benötigt werden.

4.2.1.2 Pufferverwaltung

Unabhängig von der Speicheranordnung müssen die Speicher intern verwaltet werden. Wesentliche Aufgaben hierbei sind (siehe auch Abschnitt 3.2.5):

- Anfordern von Speicherbereichen
- Belegen von Speicherbereichen (und vor Überschreiben sichern)
- Selbständiges Verlagern von Bereichen
- Lokalisieren von Informationen
- Logisches Verbinden von physikalisch nicht zusammenhängenden Speicherbereichen, z.B. mehrere Nutzdatenteile oder Steuerkopfinformationen und Nutzdaten
- Freigeben und Rückgabe, Bereitstellen von freien Bereichen
- Information über den aktuellen Belegungszustand jedes einzelnen Speicherbereichs
- Füllstandsüberwachung
- Realisierung des physikalischen Zugriffs

Eine übliche Realisierung ist das Anlegen von logisch unabhängigen Speicherbereichen pro Schicht (pro Verbindung), welche aus verketteten Elementarpuffern (EP) einheitlicher Größe aufgebaut sind [293, 323, 327]. Entsprechend der OSI-Sicht gibt es in jeder Schicht einen solchen EP-Pool für Eingangs- und Ausgangsrichtung. Es ergeben sich folgende logische Schritte bei der Bearbeitung einer Dateneinheit:

1. Die Schicht erhält ein Signal über eine PDU im Empfangspuffer plus Adreßinformation.
2. Die Schicht kann jetzt auf die neuen Daten zugreifen und alle Funktionen ausführen. Darin enthalten ist die Manipulation von Form und Inhalt der Steuerkopfinformation sowie die in Abschnitt 3.1.6 beschriebenen Varianten des Segmentierens bzw. Zusammenfügens.
3. Die vollständig bearbeitete PDU wird in den Ausgangspufferbereich kopiert.
4. Beide Nachbarschichten werden informiert: der vorherigen Schicht wird lediglich die Beendigung der Bearbeitung gemeldet, der nächsten Schicht wird das Vorhandensein einer Dateneinheit und zugehörige Adreßinformation mitgeteilt.

Das Freigeben der Ausgangspuffer ist entweder von impliziten Protokollmechanismen (Ende-zu-Ende Quittungen) abhängig oder kann durch die Fertigmeldung der Nachbarschicht initiiert werden.
Bei einer direkten Umsetzung der logischen Schritte in eine Implementierung gemäß Abschnitt 3.3.3.3 können problemlos alle Protokollmechanismen behandelt werden. Die Längen sowohl der Steuerköpfe als auch der PDUs auf den einzelnen Schichten unterliegen keinen Einschränkungen, die Flexibilität ist daher groß. Gravierender Nachteil der direkten Umsetzung ist der immense zeitliche Aufwand für das häufige Bewegen von Daten und die Verwaltung der großen Zahl von Puffern.

Eine Verringerung der Kopiervorgänge bietet sich dadurch an, daß sich nur wenige Protokollmechanismen auf die einzelnen Datenoktetts beziehen oder gar deren Form verändern. Die eigentlichen Nutzdaten brauchen daher eventuell nicht kopiert werden. Für die Behandlung der Steuerköpfe gibt es zwei Verfahren, die sich in Speichereffizienz und dem notwendigen Verwaltungsaufwand unterscheiden [293, 323]:

> Entweder werden direkt die Steuerköpfe selbst logisch verbunden (Bild 4.17),
> oder für jeden Steuerkopf wird ein *eigener* Puffer bereitgestellt, und diese Puffer werden miteinander verkettet (Bild 4.18).

Bei der 1. Variante wird in Senderichtung ein oder - falls benötigt - mehrere Puffer mit Steuerköpfen der einzelnen Schichten aufgefüllt. Würden die Steuerköpfe eine feste Länge besitzen, wäre jeder Schicht die genaue Position ihres Steuerkopfes bekannt, was besonders auf der Empfangsseite zu Vereinfachungen führen könnte. Bei - real dominierenden - Steuerköpfen mit variierenden Längen erhält die Schicht (N) von der Schicht (N+1) einen Adreßzeiger auf den Beginn des Schicht (N+1) Steuerkopfs, von dem aus der eigene Startpunkt bestimmt werden muß. Eine Segmentierung einer TSDU in zwei TPDUs mit diesem Verfahren ist schematisch in Bild 4.17 dargestellt. Dabei stehen AH, PH, SH und TH für die

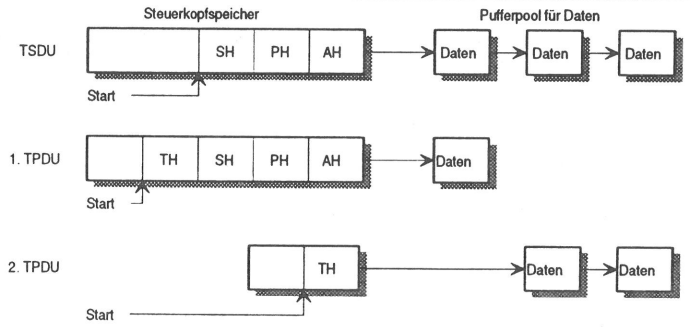


Bild 4.17: *Segmentierung mit gemeinsamem Puffer für die Steuerköpfe*

Steuerköpfe der Schichten 7 bis 4, Blöcke, die ein D enthalten, repräsentieren Nutzdaten.

Die 2. Variante vermeidet das sequentielle Auffüllen des Steuerkopfpuffers durch eigene Puffer für jede Schicht. Der prinzipielle Verlauf des Segmentierens ist in Bild 4.18 zu sehen.

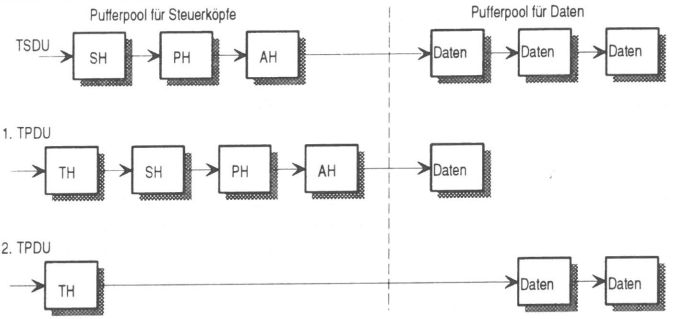


Bild 4.18: *Segmentierung mit schichtenspezifischen Steuerkopfpuffern*

Der Gewinn an Flexibilität und Parallelisierungsgrad muß allerdings mit aufwendigen Maßnahmen zur Verwaltung der zusätzlichen Puffer, zu deren Verkettung und zu pufferinternen Endebestimmungen (bei Steuerköpfen variabler Länge) "bezahlt" werden.

Das Weiterreichen von Puffern zwischen den Schichten wirft einige Problemstellungen auf:

- Das Prinzip der unabhängigen Schichten kann nicht mehr beibehalten werden. Z.B. sind ein durchgängiges Verständnis von Adressen und eine Zugriffsmöglichkeit auf gemeinsame Ressourcen Voraussetzung für eine korrekte Funktion.
- Es gibt Schichten, die nur Puffer anfordern und andere Schichten, die ausschließlich Puffer freigeben. Schichtenspezifische Pufferverwaltungen scheiden daher aus. Stattdessen ist ein globales Puffermanagement notwendig. Protokollmechanismen, die direkt von der Verfügbarkeit von Pufferplatz beeinflußt werden (Informationsflußsteuerung) und bestimmte Maßnahmen zur Sicherstellung von verbindungsspezifischen Dienstgütemerkmalen werden durch diese Änderung erschwert.
- Wahlfreie Segmentierung an beliebigen Punkten von PDUs sowie auf mehreren Schichten wird durch die Pufferstrukturen erschwert bzw. weitgehend ausgeschlossen. Eigenschaften des Netzes sollten bei der Festlegung der Elementarpuffergröße berücksichtigt werden. Nur wenn das Minimum der maximalen PDU-Größen aller Schichten größer als die Elementarpuffergröße ist, lassen sich ineffiziente Segmentierungen vermeiden.

Woodside und Montealegre berichten in [323] über Untersuchungen verschiedener Verwaltungsstrategien. Neben einer Betrachtung der reinen Kopiervorgänge berücksichtigen sie explizit auch die Kosten von Unterstützungsmechanismen, die "normalerweise" bei solchen Untersuchungen vernachlässigt werden. Funktionen zur Anforderung und Freigabe von Puffern sind ebenso aufgeführt, wie die Erstellung und Bearbeitung von Steuerköpfen, die Prüfsummenerstellung oder die Verkettung von Puffern.

> *Durch eine bloße Änderung der Pufferverwaltungsstrategie ergeben sich Leistungsunterschiede bis zu einem Faktor von 5!*

Das Bereitstellen eines unbelegten EPs kann durch Absuchen, z.B. vom Speicheranfang / -ende her oder ab dem letzten belegten EP, erreicht werden. Die zugehörige Kennung kann entweder Bestandteil jedes einzelnen Puffers sein oder zentral in einer gesonderten Tabelle abgelegt sein. Das zeitaufwendige sequentielle Durchsuchen kann bei geschickter Realisierung der Kennungen u.U. durch Hardware unter-

stützt werden. Mit dem Verfahren der Verkettung von unbelegten Puffern kann ein wesentlich besseres Verhalten bei vielen Puffern bzw hoher Speicherauslastung erreicht werden. Bei geringer Pufferzahl bzw. geringem Speicherfüllgrad ist dafür aber ein höherer Aufwand erforderlich. Bei der Freigabe von Puffern werden diese - bildlich gesprochen - auf einen Stapel freier Puffer zurückgegeben. Bei einer Neuanforderung werden EPs von diesem Stapel zugeteilt. Voraussetzung für eine korrekte Funktion ist natürlich, daß sich durch Initialisierungsmaßnahmen im jungfräulichen Zustand alle EPs auf diesem Stapel befinden, und daß bei Belegung und Freigabe von Puffern entsprechende Bearbeitungsschritte vorgenommen werden. Sinnvoll ist zusätzlich eine ständig nachgeführte Füllstandsanzeige.

4.2.1.3 Zeitgeber

Im Abschnitt 3.3.3.4 wurden einige Varianten heutiger Zeitgeber und deren wesentliche Eigenschaften betrachtet. Wie können die dort aufgeführten Belastungen für den Anwendungsrechner reduziert werden? Neben einer Auslagerung der Zeitgeberfunktionalität in Hardware sind auch durch Änderungen der zugrundeliegenden Strukturen und Verfahren noch Verbesserungen erreichbar. Vertreter beider Vorgehensweisen werden nachfolgend beschrieben.

Eine von Varghese und Lauck in [305] vorgeschlagene optimierte (Software-) Methode kann dann angewendet werden, wenn sichergestellt ist, daß die Zeitwerte aller Einträge kleiner als eine vorgegebene Schranke MaxIntervall sind. Es wird ein zirkulärer Puffer mit MaxIntervall Einträgen benutzt (timing wheel, Zeitrad). Jeder Eintrag enthält eine Liste mit Elementen gleicher Ablaufzeit. Ein Zeiger verweist auf den Eintrag, der der momentanen Zeit entspricht (siehe Bild 4.19).

Bei jedem Ereignis eines Taktgebers (Tick) wird der Zeiger modulo MaxIntervall inkrementiert, was - bildlich gesprochen - dem Weiterrücken auf die nächste Position des Rades entspricht. Ist die zugehörige Liste nicht leer, wird sie von vorne her abgearbeitet, d.h. die jeweils in den Listenelementen als Parameter enthaltenen vordefinierten Funktionen werden ausgeführt. Das Setzen eines Zeitgebers erfolgt durch einen Eintrag in die Liste des Feldes, das um die Anzahl der gewünschten Ticks hinter dem momentanen Wert liegt. Formaler ausgedrückt entspricht dies:

Zielelementnummer = (Aktuelles Element + gewünschte Tickzahl) MODULO MaxIntervall.

Werden die Listen doppelt verkettet, ist der Aufwand für Setzen und Löschen von konstanter Ordnung O(1). Der Pro-Tick-Aufwand variiert entsprechend der Listenlänge L_i des aktuellen Listenelements i. Berücksichtigt man allerdings die minimal geforderte Auflösung (ms) und den maximal möglichen Wertebereich (0.5 Stunde) von Zeitgebern in Kommunikationssystemen, können sich Räder mit über einer Million Elementen ergeben. Selbst mit einer Beschränkung auf lediglich zwei Adreßoktetts pro Element resultiert daraus ein Speicherbedarf, dessen Verwaltung in heutigen Rechnersystemen zu unakzeptablen

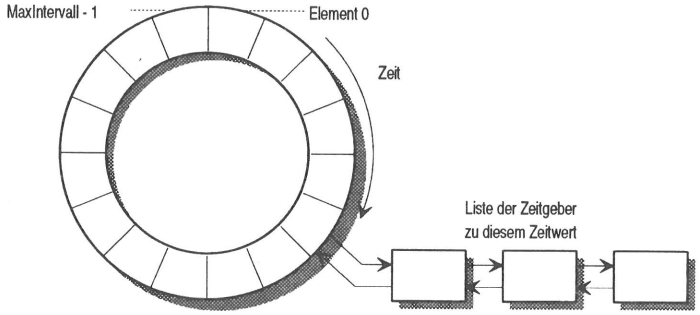


Bild 4.19: *Ringpuffer als Basis für eine optimierte Zeitgeberimplementierung*

Kosten führen würde - man denke nur an den Zeitbedarf für Ein- und Auslagern eines solchen Speicherbereiches (verwandte englische Begriffe: paging und swapping). Durch die drastisch gesunkenen Preise pro Speicherzelle wäre eine externe Hardware für das eigentliche "Rad" dagegen wirtschaftlich realisierbar (siehe Abschnitt über Hardwareunterstützung 4.2.2.3).

Werden Zeitgeber mit vergleichbaren Anforderungen an Auflösung und Maximalwert zu Klassen zusammengefaßt, reduziert sich die Anzahl der Radpositionen, und der Einsatz des Verfahrens wird wieder sinnvoll. So ist ein Zeitrad ausschließlich für die Sendewiederholzeitgeber durchaus denkbar.

Eine Möglichkeit zur Reduktion der Elementzahl ohne die angeführte Beschränkung ist die Verwendung von Komprimierungsverfahren (hashing). Zur Ermittlung des Zielelements wird dem in Ticks angegebenen Zeitwert mittels definierter Abbildungsvorschrift ein Wert kleiner als MaxIntervall zugewiesen. Mit dem komprimierten Wert wird die Bestimmung des Zielelements danach wie oben durchgeführt. Zur eindeutigen Zuordnung muß zusätzlich der unkomprimierte Wert oder eine signifikante Ergänzung als Parameter abgespeichert werden. Der (Zähl-)Mechanismus zum Weiterschalten, d.h. Adressieren der momentanen Radposition, muß den kompletten Bereich der unkomprimierten Werte überdecken. Zur Adressierung des realen wesentlich kleineren Rades muß bei jedem Tick die vorgeschriebene Abbildung durchgeführt werden. Nachteile des Komprimierungsverfahrens sind:

- Längere Elementlisten und damit erhöhter Pro-Tick-Aufwand
- Zusätzlicher Speicherbedarf für den unkomprimierten Wert
- Statt linearem Abarbeiten der Liste, muß jeweils der Wert geprüft werden
- Der Aufwand für das eigentliche Komprimieren von Zeitwert und Tick-Generator.

Einfache Beispiele für Abbildungsvorschriften sind beliebige Divisionen oder die Modulorechnung. Bei Divisionen kann je nach gewähltem Divisor entweder der Rest oder der ganzzahlige Anteil des Ergebnisses als komprimierter Wert benutzt werden. Ist MaxIntervall eine n-te Potenz von 2, so entspricht eine Division durch MaxIntervall dem Abschneiden der niederwertigen n Bit bzw. die niederwertigen n Bit stellen direkt den Rest der Division dar. Ganzzahliger Anteil der Division und Rest ergänzen sich in diesem Falle einfach zum originalen Wert.

Das Ausnutzen von Hierarchie ist eine weitere Möglichkeit, die Gesamtanzahl von Elementen zu reduzieren. Um die Zahl 1.000.000 darzustellen, werden nicht eine Million Ziffern benötigt, sondern nur 7, die jeweils Einheiten von aufsteigenden 10-er Potenzen repräsentieren. Analog braucht man für die Darstellung aller möglichen 32-Bit Zeitwerte ein Zeitrad bzw. ein Feld mit 2^{32} Elementen. Stattdessen können einige wesentlich kleinere Felder mit entsprechend unterschiedlicher Granularität benutzt werden. Beispielsweise können vier Felder wie folgt gewählt werden:

- ein Feld mit 60 Elementen, die Minuten darstellen,
- ein zweites Feld mit ebenfalls 60 Elementen für die Sekunden,
- ein kleines Feld mit nur 20 Elementen entsprechend 50 ms und
- ein 50-elementiges Feld für Millisekunden

Mit nur 60 + 60 + 20 + 50 = 190 Elementen können somit 60•60•20•50 = 3.6 Millionen Zeitwerte im Bereich bis zu 60 Minuten mit einer Auflösung von 1 ms dargestellt werden. Allerdings wird sowohl die Berechnung des Zeitwertes als auch der Mechanismus zum Weiterschalten etwas aufwendiger. Der prinzipielle Ablauf wird anhand eines Beispiels erläutert:

Die aktuelle Zeit sei 4 Minuten und 31.274 Sekunden, was bei den gewählten Einheiten den Elementen 4, 31, 5 und 24 entspricht. es soll ein Zeitgeber mit 2.56 Sekunden gesetzt werden. Die Berechnung ergibt in den gewählten Einheiten die Werte 4, 33, 16 und 34. Im Minutenfeld unterscheidet sich der Zielwert nicht von dem aktuellen Wert. Der Zeitgeber wird demnach in die Liste eingetragen, die zu dem 33. Element - also 2 Felder vom aktuellen Feld entfernt - des Sekundenfeldes gehört. Dort werden auch die zwei restlichen Werte 16 und 34 des Zeitgebers abgespeichert.

Das Weiterschalten im Millisekundenfeld läuft wie gewohnt ab. Beim Übergang vom Element 49 auf das Element 0 wird die Position des 50 ms Feldes inkrementiert. Analoges geschieht, wenn Element 19 des 50 ms Feldes bzw. Element 59 des Sekundenfeldes inkrementiert werden. Nachdem das Millisekundenfeld 35 mal übergelaufen ist, wird im Sekundenfeld das Element 33 und damit den Listeneintrag des Beispielzählers erreicht (Zählerstand zu diesem Zeitpunkt 4, 33, 0, 0). Der Eintrag wird aus der Liste entfernt. Der Zeitgeber wird zusammen mit dem verbleibenden Wert 34 in die Liste des 16. Elements des 50 ms Feldes neu eingetragen. Nach 16 weiteren Überläufen des ms-Feldes (Zählerstand 4, 33, 16, 0) wiederholt sich der Vorgang des Umtragens des Zeitgebers. Diesmal wird der Zeitgeber ohne weitere Parameter in die Liste des 34. Elements des ms-Feldes neu eingetragen. 34 Ticks später, zum gewünschten Zeitpunkt 4 Minuten und 33.834 Sekunden (4, 33, 16, 34), wird die zugehörige vom Benutzer vereinbarte Ablaufreaktion ausgelöst.

Das Umtragen eines Zeitgebers kann entfallen, wenn die Granularität mit steigender Hierarchieebene abnehmen darf. Im obigen Beispiel könnte man den Zeitwert von 2.56 Sekunden um 166 ms auf 2.726 Sekunden aufrunden (Zielzählerstand 4, 34, 0, 0 !) und vereinbaren, daß beim Erreichen des zugehörigen Elements 34 des Sekundenfeldes direkt die vom Benutzer gewünschte Ablaufreaktion ausgelöst wird. Offensichtlich kann der Verlust an Genauigkeit bis zu 50% der Auflösung des höchsten relevanten Feldes betragen (im Beispiel bis zu 0.5 Sekunden). Um den Präzisionsverlust auszugleichen, kann alternativ z.B. genau ein Umtragen zugelassen und dann erst gerundet werden. Im Beispiel würde sich ein um 16 ms zu hoher Zeitwert von 2.576 (Zielzählerstand 4, 33, 17, 0) ergeben.

Die Betrachtung von Mechanismen zur Realisierung von Zeitgebern in diesem Abschnitt sollte verdeutlichen, daß selbst mit optimierten zugrundeliegenden Strukturen ein nicht vernachlässigbarer Aufwand notwendig ist. Bei einer größeren Zahl von Zeitgebern mit stark variierenden Zeitwerten und unterschiedlichen Forderungen an die Auflösung, wie es für Kommunikationssysteme typisch ist, ist der Entwurf einer gesonderten, möglichst hardwareunterstützten, Zeitgebereinheit unumgänglich.

4.2.1.4 Innere Strukturierung der Software

Unabhängig von Fragen nach der Aufteilung eines Kommunikationssystems in Hardware und Software sowie des Verflechtungsgrads der Software mit dem Betriebssystem, welcher im nächsten Abschnitt behandelt wird, bestimmt die innere Organisation der Software und damit die Abbildung von Funktionen auf Prozesse und Prozeduren die Leistungsfähigkeit erheblich (siehe auch 3.2.1). Die naheliegendste Möglichkeit der Abbildung einer Schicht auf einen oder mehrere Prozesse und Warteschlangen zwischen den Schichten (Bild 4.20) erfordert einigen Laufzeitaufwand (siehe 3.3.3.1 und 3.3.3.2) wegen:

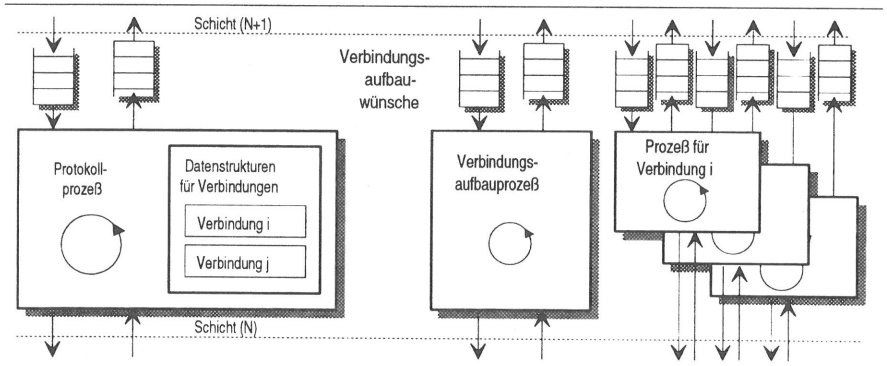


Bild 4.20: *Abbildung einer Schicht auf Prozesse*

1. Kontextwechsel und 2. Kopieren von Daten

Neuere Betriebssysteme wie z.B. MACH [301] verringern die Zeiten, die in herkömmlichen Betriebssystemen wie UNIX für Kontextwechsel benötigt werden, durch die Einführung von "leichtgewichtigen" Prozessen [322]. Diese einfachere Form von Prozessen besitzt nicht jeweils einen eigenen, geschützten Adreßraum, sondern mehrere Prozesse benutzen einen gemeinsamen Adreßraum, wodurch sich der Aufwand für das Umschalten zwischen diesen entsprechend verringert. Um diese "leichteren" Prozesse auch mit anderen Betriebssystemen nutzen zu können, wurden verschiedene Softwarepakete - z.B. PRESTO [64] und μSystem [81] - entwickelt. Solche Pakete stellen dann in einem oder mehreren herkömmlichen ("heavy-weight") Prozessen eine Vielzahl von Aktivitätsträgern zur Verfügung (tasks oder threads)[2]. Durch den gemeinsam benutzten Adreßraum können neben den Kosten für Prozeßwechsel auch die Kosten des Nachrichtenaustausches reduziert werden. Die Verwaltung einer Warteschlange zur Speicherung der ausstehenden Aufträge ist jedoch unvermeidbar.

Um diese Probleme zu umgehen, wird schlägt Clark in [105] eine andere Strukturierungsmöglichkeit vor. Zwischen den Schichten oder allgemeiner zwischen den einzelnen Prozessen soll keine asynchrone Kommunikation durch das Versenden von Nachrichten stattfinden, sondern die einzelnen Schichten bzw. Prozesse sollen durch (synchrone) Prozeduraufrufe miteinander kommunizieren.

Für den abwärts verlaufenden Datenfluß vom Benutzer zum physikalischen Medium ergibt sich diese Vorgehensweise in natürlicher Weise aus der Sichtweise des OSI-RMs mit dessen Dienstnutzern und -erbringern (Kommunikationsaufträge), und sie entspricht den üblichen Programmstrukturen. Auch für Kommunikationsanzeigen (indications) ist das Schema möglich. Die Eigenschaft, daß die Abfolge der Prozeduraufrufe dem Steuer- und Datenfluß entspricht, bezeichnet Clark als *Upcall* unabhängig von der tatsächlichen Richtung, die auf-, ab- oder seitwärts sein kann.

Eine Schicht besteht aus einer Ansammlung von Unterroutinen, die in mehreren Aktivitätsträgern existieren können und in diesem Fall als Multitask-Module bezeichnet werden (siehe Bild 4.21). Die Tasks kooperieren innerhalb eines Moduls mittels gemeinsamer Zustandsvariablen. Pro zu verarbeitender Daten-PDU kann z.B. ein Aktivitätsträger zugeordnet sein.

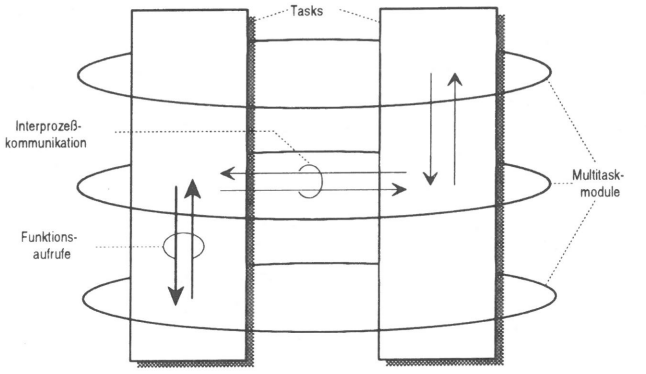


Bild 4.21: *Das Modell der Aktivitätsträger*

Senden und Empfangen kann durch die Upcall-Methode gut strukturiert werden. So wird z.B. die Entscheidung des Sendezeitpunkts nicht ausschließlich vom Benutzer getroffen, sondern auch durch Kon-

[2] Die Notation für Aktivitätsträger wird in der Literatur uneinheitlich gehandhabt.
In diesem Abschnitt werden die von Clark in [105] verwendeten Bezeichnungen benutzt.

trollmechanismen der unteren Schicht(en) mitbestimmt. Dazu wird vor der Datenübergabe ein abwärts verlaufender Aufruf (downcall) durchgeführt, der anzeigt, daß Daten zum Senden bereitstehen. Sobald die untere Schicht dazu bereit ist, fordert sie mittels eines Aufrufes in Gegenrichtung (upcall) die Daten an. Das Wissen über die aufzurufende Prozedur wurde beim abwärtsgerichteten Aufruf als Parameter mitgegeben und in einer Prozedurvariable zwischengespeichert.

Vorteile der Upcall-Methode sind [54, 322]:

- Effizienz
  - Prozeduraufrufe sind schneller als herkömmliche IPC-Mechanismen
  - Ein Umkopieren von Daten und die Verwaltung deren Zwischenspeicherung ist durch den gemeinsamen Adreßraum nicht notwendig
- Einfachere Implementierung
  - Es wird kein Code für die Pufferung an Schichtgrenzen benötigt (dafür sind Überlegungen zur Koordination mit anderen Aktivitätsträgern notwendig)
  - Anfragen über protokollbeeinflussende aktuelle Eigenschaften der höheren Schicht sind möglich. Das Wissen über noch zum Senden anstehende Daten kann etwa einen Einfluß auf die Datenflußkontrolle oder die Quittierungsstrategie haben.

Neben den Problemen bei der Programmierung paralleler Systeme (Datenabhängigkeiten, Synchronisierung, Blockierungen usw.) gibt es einige spezifische Probleme mit der Upcall-Methode:

- Aufrufe von unteren zu oberen Schichten sind gefährlich, falls die obere Schicht einen Fehler begeht. Denn dann kann die untere Schicht in einen inkonsistenten Zustand geraten. Speziell die Verwendung gemeinsamer Ressourcen wirkt sich dann kritisch aus. Eine fehlertolerante Behandlung folgender Ressourcen ist daher notwendig:
  1. Daten in unteren Schichten; globale nicht taskspezifische Daten müssen vor einem Aufwärtsruf in einen konsistenten Zustand gebracht werden und dürfen nicht gesperrt werden.
  2. Aktivitätsträger; sind diese erneuerbar, können sie im Fehlerfall abgebrochen und neu erzeugt werden, ansonsten sind Maßnahmen zur Wiederherstellung vorzusehen.
- Indirekt rekursive Aufrufe können - da unvorhersehbar - zu problematischen Zustandsänderungen führen. Zur Verhinderung sind einige Kontrollmechanismen vorzusehen:
  1. Vor einem aufwärtsgerichteten Aufruf wird immer ein konsistenter Zustand hergestellt, nach der Rückkehr wird der Zustand erneut ausgewertet.
  2. Während eines Aufwärtsrufs werden keine Aufrufe in die Gegenrichtung zugelassen.
  3. Abwärtsgerichtete Aufrufe dürfen nur Markierungen setzen; die zugehörigen Aktionen werden von anderen Unterroutinen ausgeführt.
- Änderungen am globalen Zustand eines Moduls können schwierig sein, da sich eine Reihe von Aktivitätsträgern in diesem befinden können. Die Einführung eines zugehörigen globalen Signals, das an alle relevanten Tasks gesendet wird, kann hier Abhilfe schaffen.

Das Verfahren verringert zwar den Aufwand für das Wechseln von Prozessen und für die Kommunikation zwischen Prozessen deutlich, aber der Aufwand für das Anlegen und Schließen unterscheidet sich nicht wesentlich von dem herkömmlicher Prozesse. Wolf schlägt hierfür in [322] eine Vorgehensweise analog zu den bekannten Konzepten zur Verwaltung von dynamischen Speicherstrukturen vor: bei der Initialisierung wird schon gleich eine gewisse Anzahl von universell einsetzbaren Aktivitätsträgern auf Vorrat eingerichtet (thread pool). Statt dem aufwendigen Anlegen und Schließen reicht ein einfaches Belegen schon existierender Aktivitätsträger und ein rein *logisches* Freigeben aus.

Die Australier Zhang und Seneviratne [327] erweitern Clarks Methode. Um die eins-zu-eins Zuordnung zwischen rufender und gerufener Routine abzuschwächen und damit die Abhängigkeit zwischen den Rou-

tinen zu verringern bzw. die gewünschte Modularität zu erhöhen, führen sie Schnittstellenmodule ein, die die Aufrufe verteilen und koordinieren. Die ursprüngliche direkte Zuordnung zwischen den Routinen wird durch den parametrisierten Aufruf des Schnittstellenmoduls ersetzt. Statt des Aufrufs genau einer bestimmten Routine, ist somit der vereinheitliche Aufruf von m verschiedenen Routinen mit einer einzigen Form möglich. Daher wird auch der Name m-map Upcall für dieses Verfahren eingeführt.

4.2.1.5 Integration in ein Gesamtsystem

An welcher Stelle kann ein Kommunikationssystem in eine bestehende Rechenanlage eingefügt werden? Nach Clark [104] gibt es dafür drei Möglichkeiten (Bild 4.22, von links nach rechts):

1. als unabhängige separate (Prozessor-)Baugruppe
2. innerhalb des Betriebssystems
3. als Benutzerprozeß bzw. -prozesse

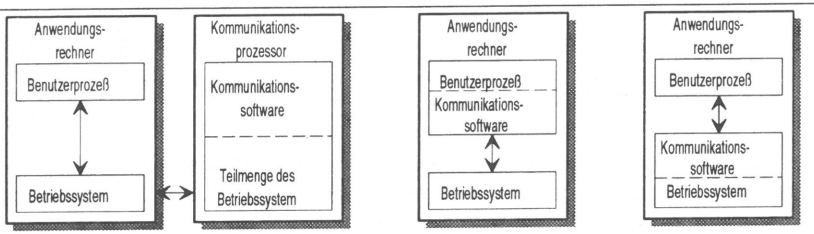


Bild 4.22: *Varianten für die Integration in ein Gesamtsystem*

Bevor die für die Arbeit relevanten Ansätze zur Reduzierung des Aufwandes beim leistungsfähigen Übergang von einem unabhängigen Kommunikationssubsystem in den Anwendungsrechner AR aufgezeigt werden, werden zunächst Grundzüge der zwei verbleibenden Möglichkeiten aufgezeigt.

Die Implementierung im Benutzerbereich ist der offensichtlichste und einfachste Ansatz. Hardwareunabhängigkeit, leichte Portierbarkeit und geringes erforderliches Systemwissen sind wesentliche Vorteile einer Implementierung ausschließlich im Benutzerbereich. Allerdings ist die Leistungsfähigkeit, bedingt durch die hohen Zeitaufwände für Prozeßwechsel, für HS-Systeme unakzeptabel.

Zur Abarbeitung eines asynchron eintreffenden und durch Hardware-Unterbrechungsanforderung gemeldeten Pakets stehen auch einige Varianten zur Wahl (Bild 4.23). Entweder reagiert ein vollständiger Protokollprozeß - regulär nur im Kern möglich, mit geänderten Betriebsystemfunktionen auch direkt im Benutzerbereich - oder die Reaktion auf die Unterbrechungsanforderung wird gestaffelt [322]: in einem ersten hochprioren und nichtunterbrechbaren Schritt wird nur der empfangene Rahmen von der MAC-

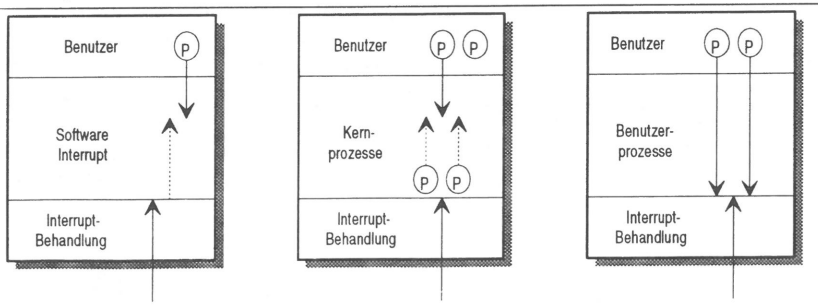


Bild 4.23: *Varianten für die Behandlung von Unterbrechungsanforderungen*

Schicht in einen Pufferbereich übertragen und in eine zugehörige Auftragswarteschlange eingereiht. Gleichzeitig wird eine niederpriorisierte und unterbrechbare (Software-) Unterbrechungsanforderung generiert, welche die eigentliche Protokollbearbeitung - im Kern- oder Benutzerbereich - anstößt.

Die Einführung eines Paketfilters [232] im Kernbereich des Betriebssystems ist eine Möglichkeit, die Leistungsfähigkeit zu verbessern. Der Paketfilter verteilt ankommende Pakete auf die zugehörigen Benutzerprozesse, die dann die eigentliche Protokollbearbeitung durchführen (siehe Bild 4.24).

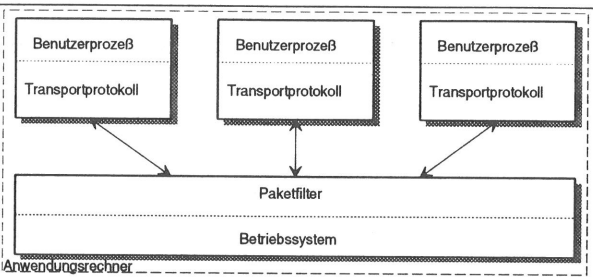


Bild 4.24: *Prinzip des Paketfilters nach [232]*

Das im Kern durchgeführte Demultiplexen reduziert die Anzahl benötigter Systemaufrufe und (Benutzer-)Prozeßwechsel erheblich. Wolf [322] schätzt den Gewinn etwa auf einen Faktor 4 gegenüber einer vollständig im Benutzerbereich implementierten Version.

Eine Kernlösung ist durch die direkte Nutzung von Systemressourcen und einer weiteren Verringerung der Systemaufrufe und Prozeßwechsel um den Faktor 2 schneller. Einen höheren Grad an Systemwissen und ein Verlust an Systemunabhängigkeit sowie Änderungsunfreundlichkeit sind Nachteile.

Insgesamt betrachtet ergeben sich durch die Einbindung in eine bestehende Betriebssystemumgebung viele leistungsbestimmende Aspekte, die mit der rein logischen Funktion nichts zu tun haben. Allerdings ist es schwer - wenn nicht unmöglich - deren Einfluß quantitativ anzugeben. Denn es fehlen entweder die aussagekräftigen Meßverfahren oder die Zahl der beeinflussenden Faktoren ist zu groß.

Selbst wenn der Einfluß aller aufgeführten Betriebssystemmechanismen auf ein Kommunikationssystem vernachlässigbar wäre, ergeben sich beim Einbinden in ein Gesamtsystem weitere funktionsunabhängige Leistungsbegrenzungen. Besonders die Basismechanismen zur Interaktion zwischen Kommunikations- und Anwendungssystem (siehe auch die Abschnitt 3.2.2 und 3.3.3.2 über Schnittstellen) und zum bloßen Bewegen von Daten sind hier zu nennen. Es werden nachfolgend einige Vorschläge gemacht, die sich mit den Basismechanismen Unterbrechungsanforderung und Datenbewegung befassen.

Die hohen Zeitbedarfe durch Wechsel des gerade bearbeiteten Arbeitsablaufes wurden schon mehrfach angesprochen. Selbst die Reaktion auf Unterbrechungsanforderungen (interrupts) direkt auf Hardwareebene benötigt schon einige µs. Die einzige Lösungsmöglichkeit ist das Verringern der Anzahl der Unterbrechungsanforderungen. Dazu sollten Routineabläufe nicht über Unterbrechungsanforderungen abgehandelt werden. Beispiele sind das Mitteilen von zeitunkritischen Informationen und das detaillierte Berichten über die reguläre Durchführung von gerade durchgeführten Abläufen, etwa der Beginn und das Ende des Aussendens einer zur Übertragung anstehenden Dateneinheit, das Belegen oder Freiwerden eines jeden Datenpuffers und ähnliches.

Bei der Übergabe von Aufträgen und Rückfragen müssen Unterbrechungen nicht unbedingt eingesetzt werden. Rückfragen können durch eine vorausschauende Strukturierung in der 1. Antwort enthaltenen Information vermieden werden. Statt z.B. nur einem freien Speicherbereich kann gleich eine Liste von freien Bereichen oder ein Verweis auf eine solche Liste mitübergeben werden [202, 203].

Die Einführung einer Auftragswarteschlange kann in vielen Fällen die Benutzung von Unterbrechungsanforderungen ersetzen. Speziell dann, wenn die beteiligten Systeme sich wiederholende Abläufe durchführen - die Übergabe von empfangenen Dateneinheiten ist solch ein sich stetig wiederholender Ablauf - kann die Entkopplung durch eine Warteschlange die Gesamtleistungsfähigkeit positiv beeinflußen, ohne die mittleren Wartezeiten von Aufträgen in der Warteschlange unangemessen anwachsen zu lassen.

Nur für zeitkritische Ereignisse ist die Verwendung einer Unterbrechungsanforderung gerechtfertigt. Neben Pufferbelegungen, -freigaben und -überläufen wird in Kommunikationssystemen der Empfang von Dateneinheiten mittels Unterbrechungsanforderung dem AR gemeldet. Aber selbst, wenn ausschließlich der Datenempfang mit diesem Verfahren signalisiert würde, wäre die maximale Anzahl bei Kommunikationssystemen für den HS-Bereich schnell überschritten. Diese maximale Anzahl beträgt nämlich selbst beim Supercomputer Cray 2 lediglich 5000 Unterbrechungen pro Sekunde [62, 265]!

Ein plausibler Ansatz zur weiteren Verringerung (u.a. in [79]), der jedoch einige Anforderungen an Speichergröße, Signalisierung und verfügbare Systemintelligenz stellt, lautet:

> Statt den Empfang jeder Dateneinheit einzeln zu melden, kann die Verfügbarkeit ganzer Ketten von Dateneinheiten dem Anwendungsrechner signalisiert werden.

Dies ist nur bei Kommunikationsformen möglich, die keine harten Anforderungen an die Verzögerung stellen. Da die Fähigkeit zum Sammeln von PDUs Einfluß auf die Dienstgüte hat, ist die Integration in das Verfahren zum Aushandeln der Dienstgüte bzw. eine Erweiterung des Verfahrens erforderlich.

Eine andere Möglichkeit zur Verringerung von Unterbrechungen ist das passive Anzeigen von Ereignissen durch Setzen von Markierungen in allgemein zugänglichen Meldungsbriefkästen (mailboxes), welche regelmäßig vom AR abgeprüft werden (polling, angedeutet in [124]). Voraussetzung für das Funktionieren dieses Vorschlages ist, daß die mittlere Zeit zwischen zwei Abfragen des ARs kleiner ist als die mittlere Ankunftszeit zwischen zwei aufeinanderfolgenden Ereignissen. Eine Zwischenspeicherung von relevanten Ereignissen ist zur Vermeidung von Verlusten bei kurzzeitigen Schwankungen zwischen den beiden Zeiten unbedingt vorzusehen.

Das physikalische Bewegen von Daten belastet die Zentraleinheit des ARs sowohl direkt als auch indirekt. Eine direkte Belastung ergibt sich durch die notwendige Adressierung und Signalisierung. Diese direkte Belastung läßt sich durch den von außen gesteuerten Speicherzugriff (Direct Memory Access, DMA) stark reduzieren. Während des Transfers ist bei vielen Rechnerarchitekturen jedoch der Verbindungsweg zwischen Zentraleinheit und Speicher belegt. Die Zentraleinheit kann daher nur solange die "gewonnene Zeit" nutzen, bis der erste Speicherzugriff notwendig wird ($\Rightarrow$ indirekte Belastung).
Die Reduzierung der indirekten Belastung geht nur über Veränderungen der Architektur. Einführen zusätzlicher Verbindungswege und die Verwendung von Mehrtorspeichern sind Maßnahmen.

Die großen Unterschiede bei der Implementierung eines nicht von der Zentraleinheit gesteuerten externen Speicherzugriffs und die verallgemeinerte Benutzung des Begriffs DMA erfordern einige erläuternde Anmerkungen. Zunächst wird der Ablauf eines direkten Speicherzugriffs mit Hilfe eines DMA-Controllers in Erinnerung gerufen. In einfachen Rechnern ist eine Anordnung gemäß Bild 4.25 anzutreffen.

- An einem gemeinsamen Bus sind alle Komponenten angeschlossen: Zentraleinheit, Speicher, Kommunikationssubsystem und DMA-Steuereinheit.
- Die CPU initialisiert den DMA-Controller mit einer Speicherstartadresse, der Anzahl zu übertragender Dateneinheiten (je nach Busbreite in Vielfachen von 8, 16 oder 32 Bits), einer Identifikation des involvierten Peripheriegerätes und eine Kennung für die Übertragungsrichtung ("1" bedeute hier Schreiben vom Netzwerk in den Speicher).

Signalisiert der LAN-Controller seine Bereitschaft zum vereinbarten Transfer, hat er also Daten

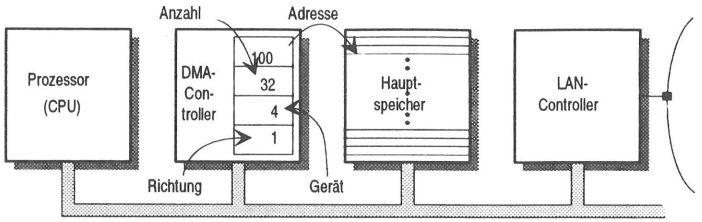


Bild 4.25: *Typische Architektur kleinerer Rechner*

empfangen, fordert der DMA-Controller zunächst die alleinige Kontrolle über den Bus an. Nachdem die CPU ihm die Kontrolle über den Bus übertragen hat, wird:

- ein Datenregister des Kommunikationssystems adressiert und in ein Hilfsregister - im DMA-Controller oder der Speicherverwaltungseinheit - gelesen.
- In einem zweiten Schritt wird die erste Speicheradresse vom DMA-Controller auf den Bus gelegt und der Inhalt des Hilfsregisters dorthin geschrieben.
- Inkrementierung der Speicheradresse, Dekrementieren des Zählers und Prüfung des Abbruchkriteriums folgen. Wenn dieses nicht erfüllt ist, wiederholen sich die zwei Schritte "Lesen vom LAN-Controller" und "Schreiben in den Speicher".

Offensichtliche Nachteile dieser Anordnung sind:

- Die vorgegebenen DMA-Parameter (DMA-Kanal) erfordern ein zusätzliches Kopieren innerhalb des Speichers oder ein ständiges Umprogrammieren des DMA-Kanals von der CPU.
- Aus gleichem Grunde können nur Dateneinheiten fester Länge übertragen werden.
- Der DMA-Controller belegt den Bus vollständig (als Bus-Master), d.h. die CPU wird normalerweise indirekt wieder gesperrt werden.
- DMA benötigt (mindestens) zwei komplette Buszyklen für die Übertragung einer Datengrundeinheit. Die sowieso schon relativ geringen theoretischen Busbandbreiten (siehe Abschnitt 3.2.4) werden von vorn herein (mindestens) halbiert.
- Die CPU wird zur DMA-Programmierung und bei der Buszuteilung benötigt. Zumindest die DMA-Programmierung wird mittels einer Unterbrechungsanforderung ausgelöst. Gelegentlich ist sogar das Anstoßen des DMA-Ablaufes nur über die CPU möglich.
- Das Ein-/Ausgabegerät (E/A-Gerät) kann vom Rechner her ausschließlich über ein Register angesprochen werden. Gerade in Kommunikationssubsystemen sind jedoch größere, meist wahlfrei adressierbare, Speicher vorhanden. Eine Möglichkeit zum direkten Ansprechen in einer dem Hauptspeicherzugriff entsprechenden Form ist daher nicht möglich. Folglich muß auch innerhalb des E/A-Geräts kopiert werden.

Eine Verbesserung ergibt sich schon allein durch eine direkte Signalisierverbindung zwischen E/A-Gerät und DMA-Controller. Bei entsprechender Abstimmung kann damit das Lesen in ein Hilfsregister komplett entfallen, also eine Durchsatzsteigerung von bis zu 100% erreicht werden. Die Steuerung des E/A-Gerätes wird dabei nicht über den Systembus vorgenommen, sondern ausschließlich vom DMA-Controller. Freigabe des Datenwertes sowie Adressierung und Steuerung des Hauptspeichers müssen vom DMA-Controller zeitlich koordiniert werden. Das E/A-Gerät muß schnell genug das nächste Datum bereitstellen. Am besten eignet sich dafür eine Anordnung mit zwei im Wechsel arbeitenden Registern.

Die nächste Verbesserung besteht darin, daß die Speicheradresse eines DMA-Kanals kontinuierlich weitergeschaltet wird, also schrittweise zusammenhängende Bereiche transferiert werden können, ohne jeweils intern umkopieren zu müssen. Eine noch größere Flexibilität wird erreicht, wenn statt einem Parametersatz eine Liste von Parametersätzen in den DMA-Controller geladen und intern auch entsprechend

schnell aktiviert werden können. Läßt man diese Verbesserungen auch auf der E/A-Seite zu, so wird aus dem einfachen DMA-Controller schon ein eigenes programmierbares Mikrorechnersystem mit eigenem Speicher. Durch die dramatischen Preisverfälle bei den integrierten Bausteinen - 8-Bit Mikrorechnerchips mit integriertem Speicher sind z.B. weit unter 10 DM zu bekommen - scheitert der zunächst etwas überzogen klingende Einsatz solcher E/A-Prozessoren auch keinesfalls an wirtschaftlichen Überlegungen!

Weitere Leistungssteigerungen kommen von den Rechnerbusarchitekturen. Zusätzlich vorgesehene Busmodi erlauben unter bestimmten Voraussetzungen eine Übertragung mit wesentlich kürzeren effektiven Buszyklen. Diese Burst-, Block-, Stream- oder auch Turbomode genannten Arbeitsweisen sind auf die Übertragung von Datenblöcken beschränkter Länge und strikt fortlaufender Adressierung ausgelegt. Die mit TURBOchannel bezeichnete Busarchitektur der Firma Digital Equipment [415] soll für diese Art des optimierten DMA-Transfers kurzzeitige Spitzendatenraten bis zu 800 Mbit/s erreichen können.

Interessant sind auch DMA-Varianten, die den Bus nicht ausschließlich belegen, sondern sich diesen effektiv mit anderen Komponenten am Bus teilen. Obwohl das zugrundeliegende Verfahren unter dem Namen "Cycle-Stealing" lange bekannt ist, waren bisher kaum ausreichend schnelle und intelligente Hardwarelösungen verfügbar, um tatsächlich einzelne unbenutzte Buszyklen effektiv nutzen zu können oder z.B. im festen Wechsel mit der CPU immer genau jeden 2. oder 3. Zyklus zu bekommen.
In größeren Rechenanlagen werden zunehmend auch hierarchische Bussysteme und Speicher mit logischer Mehrtoreigenschaft eingesetzt, so daß auch ein längerer Transfer zwischen einem E/A-Gerät und dem Speicher möglich ist, ohne die CPU dadurch (völlig) zu blockieren.

Obwohl nur einige Aspekte der Integration in ein Rechnersystem angesprochen werden konnten, wurde das Potential für Leistungsverbesserungen in dem meist vernachlässigten Bereich deutlich.

4.2.1.6 Weitere Einzelaspekte

In diesem Abschnitt sind Vorschläge für Teilaspekte und Vorgehensweisen enthalten, die Einfluß auf den Verlauf der Arbeit hatten, jedoch in den bisherigen Abschnitten nicht beschrieben wurden.

Der Entwurf eines Kommunikationssystems orientiert sich an den logischen Protokollfunktionalitäten. Dabei wird übersehen, daß der größte Teil der Protokolle aus Abhandlungen selten auftretender Ereignisse besteht. Die durchschnittliche Leistungsfähigkeit des gesamten Kommunikationssystems wird jedoch vom Verhalten im Normalfall bestimmt. Das Augenmerk sollte daher auf dessen optimierter Implementierung gelegt werden, wobei dies sogar explizit zu einer erhöhten Komplexität von Sonderfällen führen darf. Zeitgeber laufen z.B. - im Gegensatz zu Einsätzen außerhalb von Kommunikationssystemen - in der Regel nicht ab, und PDUs sind im Regelfall weder reihenfolgegestört noch fehlerbehaftet. Von diesen Annahmen sind Datenstrukturen und eingesetzte Mechanismen abhängig. Clark fordert in [104] etwa, die Struktur für erneute Übertragungen (retransmission queue) so zu implementieren, daß das Löschen von Einträgen optimal unterstützt wird, da dies der Normalfall ist.

Neben der Anpassung an den Normalfall schlagen Saltzer, Reed und Clark in [267] auch eine Anpassung an die hauptsächlich vorkommende Benutzung vor. Mehr als die Beeinflußung der Wahl des benutzten Protokollturmes erscheint jedoch nur in Einzelfällen sinnvoll, da selbst einfachste Kommunikationsformen eine große Varianz von Anforderungen beinhalten.

Bei verbindungsorientierten Diensten verursacht das Auffinden der zu einer Verbindung gehörenden Informationen einen nicht vernachlässigbaren Aufwand. Andererseits belegen veröffentlichte Messungen [108], daß bis zu 93% der empfangenen PDUs zur selben Verbindung gehören, wie die zuletzt empfangene! Das Halten der Verbindungsinformation der zuletzt empfangenen Dateneinheit(en) kann sich daher trotz einigen zusätzlich notwendigen Kommandos im Mittel positiv auf die Leistungsfähigkeit auswirken.

In Senderichtung kann eine analoge Annahme über die nächste zu sendende PDU dazu ausgenutzt wer-

den, den zugehörigen Steuerkopf vorzubereiten. Dieses Verfahren der Steuerkopfvorschau (header prediction) kann auf alle bestehenden Verbindungen ausgedehnt werden. Der teilweise ausgefüllte Steuerkopf der nächsten zu sendenden PDU einer Verbindung wird zusammen mit den weiteren verbindungsspezifischen Daten gespeichert.

Wolf untersucht in [322] verschiedene Formen einer Zuordnungsverwaltung. Die Zuordnung von externen auf interne Bezeichner ist eine unumgängliche und aufwendige Aufgabe in jedem Kommunikationssystem. Bedingt durch die Größe und die nicht vorgeschriebene und daher meist fehlende innere Strukturierung der Bezeichner scheitert ein direktes tabellarisches Ablegen am Speicher- und Suchaufwand. Stattdessen werden Komprimierverfahren (hashing) eingesetzt. Die in [188] gemachten Erfahrungen führen dazu, daß Wolf für Bezeichner unterschiedlicher Länge auch jeweils verschiedene Hashverfahren benutzt. Zusätzlich führt er analog zu den obigen Überlegungen jeweils eine Zwischenspeicherung der letzten durchgeführten Zuordnung ein.

Die Berechnung der Prüfsumme ist zwar formal in den jeweiligen Standards festgelegt, überraschenderweise ergeben sich aber Implementierungen mit extrem unterschiedlichen Leistungsfähigkeiten. Nicht nur die Wahl der Programmiersprache kann zu großen Unterschieden führen, sondern schon die Wahl von Variablentypen und kleinste Programmänderungen wirken sich sofort aus. Dies ist dadurch begründet, daß jedes Oktett eines Datenstroms manipuliert werden muß. Selbst aus einem zusätzlichen Befehl auf Maschinenebene, für dessen Ausführung z.B. nur 0.1 µs benötigt werden, resultiert in HS-Netzen mit Bandbreiten von mehr als 10 Millionen Oktetts pro Sekunde eine signifikante Belastung. Nakassis gibt in [236] einige diesbezügliche Hinweise für die arithmetische Prüfsumme nach Fletcher, wie sie u.a. auch in OSI TP4 verwendet wird (siehe auch Abschnitt 3.3.1.5).

Durch die steigende Bedeutung von Netzmanagement erreicht auch der dafür notwendige Bearbeitungsaufwand signifikante Werte. Die Gesamtleistungsfähigkeit eines Kommunikationssystems kann daher durch eine optimierte Unterstützung von Netzmanagement positiv beeinflußt werden. Neben einer an das System angepaßten Struktur des Managements [279] müssen schon frühzeitig Zugriffsmöglichkeiten und dezentrale Managementunterstützung vorgesehen werden.

4.2.2 Gesamtbetrachtung

Die im vorigen Abschnitt 4.2.1 durchgeführten Betrachtungen einzelner Umgebungsaspekte können - im Gegensatz zu protokollinternen Detailoptimierungen - unabhängig voneinander in realen Systemen berücksichtigt werden. Eine gewisse Verzahnung und gegenseitige Abhängigkeiten zwischen den einzelnen Punkten sind dennoch vorhanden und legen eine kombinierte Sichtweise nahe, in die natürlich Schlußfolgerungen der Einzelbetrachtung miteinfließen.

Analog zu den protokollorientierten Ansätzen werden zunächst Ansätze zur besseren Nutzung bestehender Umgebungsbedingungen beschrieben. Die Entwicklung von speziell für Kommunikationszwecke optimierten Laufzeitumgebungen, dedizierte Hardwareunterstützung und Leistungssteigerung durch Ausnutzen von Parallelität werden daran anschließend angesprochen.

4.2.2.1 Effiziente Nutzung bestehender Betriebsystemfunktionen

Die Leistungsfähigkeit wird wesentlich durch folgende Bereiche der Laufzeitumgebung beeinflußt:

- Formen von Prozessen und deren Verwaltung
- Pufferverwaltung
- Zeitgeberimplementierung
- Zuordnungsverwaltung

Im Gegensatz zu vielen Protokollmechanismen sind diese Bereiche nur in geringem Maße mit einstellba-

ren Parametern versehen. Trotzdem hat jeder Benutzer einer gegebenen Laufzeitumgebung die Wahlmöglichkeit zwischen mehreren Mechanismen und Verfahren um ein und dasselbe Ziel zu erreichen.

Dominierend ist dabei der Grad an Modularität, der, entsprechend dem in Abschnitt 3.3.3.1 Gesagten, umgekehrt proportional zur erzielbaren Effizienz ist [104, 116, 247, 299]. In den traditionellen UNIX-Betriebssystemen, die keine Mechanismen zur expliziten Unterstützung leichtgewichtiger Prozesse vorsehen, kann nur über die Anzahl der verwendeten Einzelmodule korrigierend eingegriffen werden. Entsprechende Änderungen setzen allerdings grundsätzlich geänderte Systemstrukturen voraus. Werden Prozeßformen mit unterschiedlichem Grad an interner Korrelation - und daraus resultierender Komplexität - unterstützt, ist auch die Wahl jeder einzelnen Funktionseinheit effizienzbestimmend.

Zur Reduzierung des Aufwands, der beim dynamischen Anlegen und Beendigen eines neuen Prozesses während des Betriebs entsteht, können von der Speicherverwaltung her bekannte Verfahren eingesetzt werden: vor dem eigentlichen Betrieb wird in einer Initialisierungsphase eine gewisse Anzahl von Prozeßumgebungen auf Vorrat angelegt (process pool). Die später erfolgende Zuordnung einer derart vorbereiteten Prozeßumgebung zu einem realen, mit einer entsprechenden Benutzeranforderung übergebenen Prozeß, nimmt deutlich weniger Zeit in Anspruch als eine komplette Prozeßgenerierung. Gleiches gilt für das Beenden und Austragen von Prozessen. Die notwendige Verwaltung der Prozeßumgebungen kann allerdings nicht auf einfache Weise in bestehende Betriebssysteme integriert werden und muß daher als Teil der Benutzersoftware realisiert werden.

Die Effizienz von Pufferverwaltung und Zeitgebermechanismen wird unter anderem auch durch protokollspezifische Eigenschaften beeinflußt. Ist die von einer Protokollinstanz benutzte Segmentgröße nicht auf die Größe der vom Betriebssystem verwalteten Elementarpuffer abgestimmt, resultieren schlechte Speichernutzungsgrade und höherer Verwaltungsaufwand. Die protokollabhängige Anzahl von Zeitgebern und das Setzen deren Werte wirkt sich direkt auf die Effizienz der Zeitgeber aus. Das Wissen über die Implementierungsform des Zeitgebermechanismus und den zugehörigen Primitiven kann u.U. zur Optimierung beitragen: z.B. kann es durchaus günstiger sein, einen Zeitgeber zunächst zu löschen und danach gleich neu zu starten, als den Wert und Aktivitätszustand des ursprünglichen Zeitgebers "direkt" mittels eines Restart-Primitivs zu manipulieren.

Eine bessere Nutzung der bestehenden Pufferverwaltung im laufenden Betrieb kann wiederum durch Anlegen und Reservieren eines Vorrats während einer Initialisierungsphase plus einer zusätzlichen Vorverarbeitung im Benutzerbereich erzielt werden. Speziell die, in den protokollspezifischen Aufgabenbereichen Segmentierung und Reihenfolgesicherung, implizit enthaltene Relation mit der physikalischen Speicherung kann hierzu ausgenutzt werden. Der originalen Pufferverwaltung kann beispielsweise statt Anforderungen für einzelne Dateneinheiten (PDUs) komplette Dienstdateneinheiten (SDUs) in Form von Pufferlisten von der Protokollinstanz übergeben werden; natürlich nur, falls das Betriebssystem eine solche geänderte Syntax zuläßt. Die Anzahl der Interaktionen zwischen Protokollinstanz und Pufferverwaltung reduziert sich durch dieses Verfahren entsprechend. Der zusätzliche Aufwand innerhalb der Protokollinstanz ist gering, da eine Verwaltung einzelner PDUs sowieso erforderlich ist.

4.2.2.2 Entwurf einer (Software-)Laufzeitumgebung

Die effizientere Nutzung bestehender Betriebssysteme scheitert häufig jedoch an vorgegebenen unveränderlichen Strukturen und unzugänglichen Internas. Die logische Konsequenz ist, wie bei den Protokollen, die Entwicklung neuer Umgebungen. Einige der bekannteren sind:

- x-Kernel [18, 187, 246]
- Mach und eine zugehörige Anpassung für die Sprache C [117, 301]
- Die portable allgemeine Laufzeitumgebung PCR [316]
- PRESTO [64]

- µSystem [81] sowie
- die Laufzeitumgebung des Heidelberger Transportsystems HeiTs [322, 176, 177]

Von den vier im vorigen Abschnitt genannten Aspekten - Prozesse und deren Verwaltung, Speicher- und Pufferverwaltung, Zeitgeber sowie Zuordnungsverfahren - werden die zwei erstgenannten jeweils in allen Vorschlägen berücksichtigt. Ein zentrales Anliegen ist dabei die Reduzierung des Prozeßwechselaufwandes. Dazu werden leichtgewichtige Prozesse und Unterkomponenten (threads, tasks) eingeführt und auch Verfahren der Vorratshaltung [322] vorgeschlagen.

Für die Verwaltung der physikalischen Speicher und logischen Puffer werden unterschiedlich aufwendige Mechanismen, welche alle auf verzeigerten Listen basieren, eingesetzt. Als wichtiges Ziel wird einheitlich die Vermeidung des physikalischen Bewegens der Daten genannt. Dazu sind jeweils Mechanismen zum Anfügen an und Auftrennen von kontinuierlichen Datenbereichen, zum Erzeugen von logischen Kopien, zur Längenermittlung und zur Speicherfreigabe vorgesehen. Zur Verwaltung der unbelegten Speicherplätze - und damit zur Ermöglichung einer hohen Speicherausnutzung bei gleichzeitig zugelassenen variablen Datenlängen - werden teilweise komplexe Datenstrukturen und Verfahren eingesetzt.

Zur Zeitgeberoptimierung in kommunikationsorientierten Systemen wird in [322] der Einsatz eines modifizierten Zeitrades vorgeschlagen. Außerdem werden Verfahren zur Optimierung der Zuordnungsproblematik verglichen und Lösungsvorschläge für den Bereich der Prozeßsynchronisation gemacht. Sowohl für Puffer als auch für PDU-bezogene leichtgewichtige Module wird eine Vorratshaltung eingesetzt, um den Aufwand für Anfordern und Freigeben im laufenden Betrieb zu reduzieren (Bild 4.26).

In vergleichbarer Weise, wie bei den neu entwickelten Protokollen, stellt sich allerdings auch bei speziell entwickelten kompletten Laufzeitumgebungen die Frage nach deren realer Verwendbarkeit. Im Gegensatz zu den Protokollen kann eine Laufzeitumgebung zwar, ohne viel Rücksicht auf andere Stationen im Netz nehmen zu müssen, lokal auf einer Rechenanlage implementiert werden, allerdings wird diese Rechenanlage nicht ausschließlich mit Kommunikation beschäftigt sein. Die nicht kommunikationsorientierten Aufgaben sind oft nur für die Zusammenarbeit mit einem bestimmten Betriebssystem ausgelegt. Die gleichzeitige Aktivität zweier unterschiedlicher Laufzeitumgebungen innerhalb einer Rechenanlage ist andererseits durch die zwangsläufige Benutzung gemeinsamer Ressourcen praktisch ausgeschlossen.

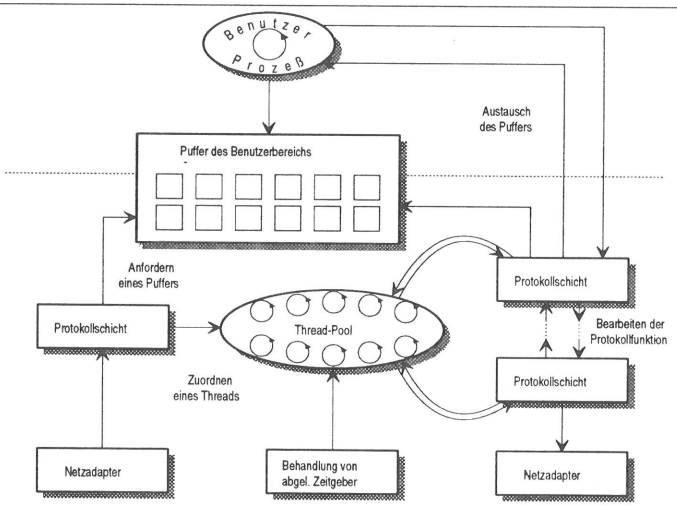


Bild 4.26: *Struktur einer kommunikationsorientierten Laufzeitumgebung*

Es bleibt also lediglich der Einsatz in dedizierten Kommunikations-(sub-)systemen, wie etwa Kommunikationscontrollern oder Adapterkarten. Diese Subsysteme zeichnen sich durch starke Spezialisierung und einen hohen Anteil von Hardware aus, so daß der Einsatz einer kompletten Laufzeitumgebung entweder nicht nötig ist oder an technischen Eigenschaften - wie falschen Prozessoren, fehlenden Verbindungsstrukturen oder anderen grundsätzlichen architekturellen Merkmalen - scheitert. Detaillösungen aus den verschiedenen Vorschlägen sind dagegen in realen Systemen durchaus anzutreffen.

4.2.2.3 Hardware-Unterstützung

Welche nicht protokollspezifischen Funktionen sind grundsätzlich für eine Umsetzung in Hardware geeignet oder lassen sich zumindest durch Hardware unterstützen? Im allgemeinen werden neben intelligenten DMA-Controllern (siehe 4.2.1.5) und besonders schnellen oder mit mehreren Zugängen ausgerüsteten Speicherchips lediglich Zeitgeber als mögliche hardwaretaugliche Komponenten eingeschätzt.

Keine der Funktionalitäten, die zur Realisierung von Zeitgebern benötigt werden, besitzt eine Komplexität, die eine Unterstützung durch Hardware unmöglich machen würde. Neben den fast ausschließlich als externe Hardwarebausteine erhältlichen Taktgebern ist die Bereitstellung von externem, d.h. nicht vom Betriebssystem kontrollierten, Speicher für die Zeitgeberinformationen die einfachste Form von Hardwareunterstützung. Um den notwendigen Aufwand für die Zeitgeber beim Anwendungsrechner jedoch effektiv zu reduzieren, sollte zusätzlich eine intelligente Verwaltung des Zeitgeberspeichers mit entsprechend einfach gestalteter integrierter Schnittstelle vorgesehen werden. Im Rahmen der Arbeit wurden verschiedene Varianten von unabhängigen Zeitgebermodulen entwicklet. Vorgestellt werden diese bei der Beschreibung der Hardware des Transportsystems in Abschnitt 5.3.5.

Nach der Durchführung von Entwurfsstudien und zugehörigen Aufwandsbewertungen ergeben sich weitere für eine Hardwarerealisierung geeignete Bereiche. Neben den schon genannten intelligenten DMA-Controllern und Zeitgebern sind dies u.a.:

- alle Arten von Such- und Sortiervorgänge,
- Vergleichsoperationen,
- die Verwaltung von Listen und anderen Datenstrukturen,
- die Kopplung von Prozessen sowie
- alle Formen von Steuerungsaufgaben mit beschränkter Komplexität.

Die Realisierung eines Mehrtorspeichers basierend auf Bausteinen, die physikalisch nur über einen Zugang verfügen, ist ein Beispiel für eine solche Steuerungsaufgabe mittlerer Komplexität. Da die Verwaltung von Pufferspeichern auf die Verwaltung von Listen plus eine zugehörige, nicht zu komplexe, Steuerlogik abgebildet werden kann, ist auch die Pufferverwaltung in Hardware umsetzbar. Im Verlauf der Arbeit wurde für jeden der genannten Bereiche Hardwarelösungen erarbeitet. Wesentliche Eigenschaften und Erkenntnisse werden bei der Beschreibung des Gesamtsystems in Kapitel 5 vorgestellt.

Neben den schon in Abschnitt 4.1.3 angesprochenen Arbeiten zu protokollorientierter Hardware, welche auch viele Aufgaben außerhalb der logischen Funktionalität erfüllen (hier ist besonders auf die fast ausschließlich nicht protokollorientierten Funktionen innerhalb des PE-Chipsatzes hinzuweisen!), sei an dieser Stelle lediglich ein explizit nicht protokollspezifischer Hardwareansatz erwähnt.

Die Franzosen Diot und Dang haben in ihrem MC3-Projekt [131, 132] völlig auf ein herkömmliches softwarebasiertes Betriebssystem verzichtet und die wesentlichen Aufgaben, d.h. Prozeßverwaltung, Interprozeßkommunikation und die Steuerung von Datenbewegungen, in festverdrahtete Hardware umgesetzt. Dazu war die Einführung optimierter Datenstrukturen eine wesentliche Voraussetzung. Für die logische Protokollfunktionalität haben sie einen RISC-Prozessorkern mit RAM- und ROM-Speicher vorgesehen. Der gesamte Entwurf ist von der Optimierung des Datenflusses geprägt:

- Datenbewegungen werden falls unumgänglich ohne Mitwirkung des zentralen Prozessorkerns vorgenommen. Dafür stehen vier spezielle Transfermaschinen zur Verfügung.
- Speicherverwaltung und Prozeßsynchronisation werden ebenfalls weitgehend von Hardwarekomponenten übernommen.
- Die Nutzung impliziter Parallelität wird durch getrennte Verbindungswege unterstützt.
- Einschränkungen der Formatvielfalt, insbesondere bei Steuerköpfen, werden empfohlen.

Die prinzipielle Entwurfsphilosophie des Konzepts weist viele Gemeinsamkeiten mit der von Greg Chessons Protokollmaschine PE auf. Allerdings bleibt der Bereich der externen Schnittstellenproblematik hier unklar. Die optimistische Abschätzung der Komplexität des Bausteins in Gattern und Fläche läßt sowohl eine Integration auf einem Chip, als auch eine Leistungsfähigkeit von bis zu 4.000 Nachrichten pro Sekunde oberhalb der ISO Schicht 4 zu. Berücksichtigt man die zu erwartenden Schwierigkeiten durch die Integration eines ROMs, mehrerer RAMs, PLAs, Prozessorkerne und Kontrollogik, sowie die enorm hohe Komplexität der vergleichbaren real existierenden vier Chips der Protokollmaschine, so sind jedoch Zweifel an der Einchiplösung legitim. Wäre mehr als ein Chip notwendig, müßten natürlich auch einige der Leistungsabschätzung zugrunde liegende Annahmen entsprechend angepaßt werden.

Unabhängig davon unterstützt das Konzept einige für die vorliegende Arbeit relevanten Ideen und Ansätze, die teilweise auch schon bei der Protokollmaschine wichtig waren. Diese sind nachfolgend aufgeführt:

- Der Schwerpunkt liegt eindeutig auf einer optimierten Implementierung der Umgebungsfunktionen. Protokolldetails spielen dagegen keine Rolle.
- Ein dediziertes, hardwareunterstütztes Teilsystem wird vorgeschlagen.
- Funktionalitäten bis einschließlich *zur* Schicht 5 des OSI-RMs sollen von der Baugruppe und nicht vom Anwendungsrechner erbracht werden.
- Es wird bewußt auf den Einsatz eines Standardbetriebssystems verzichtet.
- Die eigentliche Protokollbearbeitung wird durch die Realisierung mit programmierbarer Firmware flexibel gehalten.
- Der dominierende Faktor ist die Optimierung des Datenflusses.
- Die eingesetzten Datenstrukturen haben einen großen Einfluß auf das Leistungsvermögen.

4.2.2.4 Multiprozessorsysteme

Die Vorgehensweise zur Nutzung von Parallelität hängt nicht davon ab, ob die betrachteten Funktionen protokollspezifisch sind oder nicht. Auch bei den nicht protokollorientierten Funktionalitäten, die über 80% des Gesamtaufwandes eines Kommunikationssystems ausmachen können, muß die inherent vorhandene Parallelität zunächst lokalisiert und spezifiziert werden, wobei alle Formen der Parallelität gemäß Abschnitt 4.1.4 miteinbezogen werden können. Trotz einem großen theoretischen Potential an inherenter Parallelität - die Funktionen sind z.B. weitgehend unabhängig voneinander - werden keine umfassenden Vorschläge zu deren optimierter Implementierung durch Ausnutzen von Parallelität gemacht.

Dies ist durch die enge Verbindung mit den Anwendungen einerseits und der Gesamtarchitektur andererseits zu begründen. Ein eigener externer Prozessor, ausschließlich für eine optimale Interprozeßkommunikation, bringt keinerlei Vorteil, wenn sich die beteiligten Prozesse auf einem Prozessor befinden. Ähnlich wenig effizient wäre eine zwar optimierte aber externe Verwaltung von lokalen Prozeßwechseln oder die externe Kontrolle lokaler Betriebsmittel.

Inhalt dieses Abschnitts sind deshalb Vorschläge für Kommunikationssysteme, die aus mehreren Prozessoren aufgebaut sind und über die in Abschnitt 4.1.4 gemachte Betrachtung der rein logischen Funktion hinausgehen. Beim Umsetzen von theoretisch für die Nutzung von Parallelität in Protokollen entworfenen Multiprozessorsystemen werden verschiedene Möglichkeiten angewandt:

- Es wird von einem vorhandenen Betriebssystem ausgegangen, ohne dessen Leistungsfähigkeit detaillierter bzw. gar nicht zu berücksichtigen [198, 264, 303].
- Zur Wahrnehmung von Betriebssystemfunktionen wird pauschal ein zusätzliches Prozessorelement - eventuell jeweils für Sende- und Empfangsrichtung - vorgesehen [168, 196].
- Teilfunktionen werden durch Hardware oder eigenständige Prozessorelemente unterstützt, die eigentliche Betriebssystemfunktionalität bleibt auf jedem Prozessorelement vorhanden.

Zitterbart sieht z.B. in ihrer aus 8 Transputern aufgebauten Prototypimplementierung je einen Transputer für die Speicherverwaltung des gemeinsamen Sende- bzw. Empfangsspeichers vor [329, 331]. Die Verwaltung, Synchronisation und Kommunikation der Prozesse, die Behandlung von Unterbrechungen sowie alle weiteren nicht protokollorientierten Funktionen werden von den einzelnen Transputern durchgeführt.

Auch Diot nutzt die in Transputern vorhandenen Betriebssystemmechanismen für Prozeßzuteilung und -wechsel [133]. Die Synchronisation mit anderen Transputern kann direkt mit den transputertypischen Kommunikationskanälen ("Links") durchgeführt werden. Zur eigentlichen Prozeßsynchronisierung wird auf ein Rendezvous-Verfahren nach Hoares CSP-Modell [181, 16] zurückgegriffen. Kommunikation findet über einen für Transputer untypischen gemeinsamen Speicher statt, was die Einführung von Hardwaremechanismen zur Sicherstellung des Zugriffs mit wechselseitigem Ausschluß erforderlich macht.

Meleis und Tantawy vom IBM-Forschungslabor in Yorktown stellen in [224] eine Liste grundsätzlicher Funktionen einer modularen Kommunikationsmaschine MCM vor. Darin sind neben den protokollbezogenen Funktionsbereichen Datenbehandlung und Verbindungsmanagement explizit auch die Aufgabenbereiche Netzwerkmanagement, Datenbewegung und Speicherverwaltung sowie eine Laufzeitumgebung enthalten. Die Gesamtheit aller Funktionen floß in die Überlegungen zu optimierten Architekturen ein. Der resultierende generische Vorschlag ist in Bild 4.27 gezeigt.

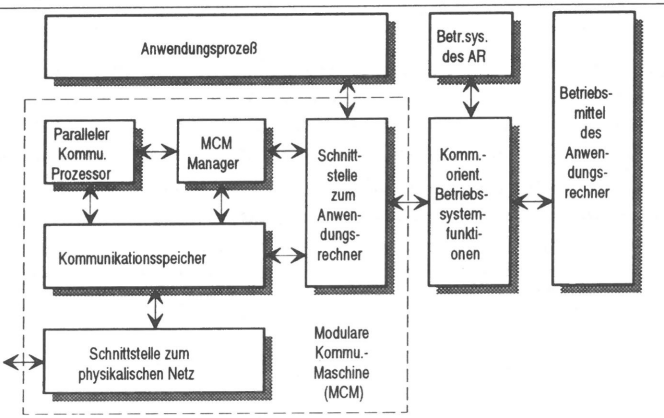


Bild 4.27: *Architektur der modularen Kommunikationsmaschine (aus [224])*

Wie bei Chessons Protokollmaschine PE wird jeweils ein eigenständiger Block für die Wahrnehmung der Schnittstellenfunktionen vorgesehen (PNI: Physical Network Interface und MHI: MCM Host Interface). Für die Steuerung des gesamten Kommunikationssubsystems ist der Block MCM Manager zuständig. Zur Ausführung der logischen Funktionalität dient der Block PCP (Parallel Communication Processor), der gegebenenfalls aus mehreren Prozessoren bestehen kann. Ein eigenständiges Speichersystem CMS (Communication Memory Subsystem) ist natürlich auch vorhanden. Interessanterweise ist im AR außer dem eigentlichen Betriebssystem noch ein Block mit für Kommunikation optimierten Betriebssystemfunktionen und eine direkte Verbindung zwischen MCM und Anwendungsprozeß enthalten.

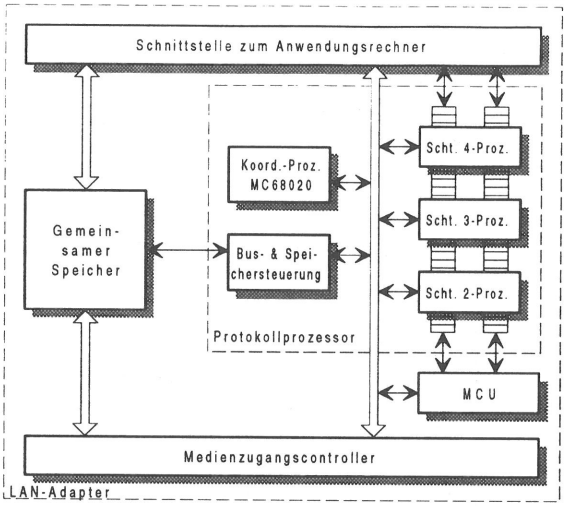


Bild 4.28: *Architektur eines Kommunikationsadapters nach [300]*

Auch die schon in Abschnitt 4.1.4.1 erwähnte Arbeit einer japanischen Gruppe [300] beschreibt ein Kommunikationssystem, das sowohl aus mehreren Prozessorelementen aufgebaut ist, als auch Ideen zur Integration von einigen Betriebssystemfunktionen enthält. Hauptmerkmal ist der Einsatz von schichtenspezifischen Spezialprozessoren, die ausschließlich für den normalen fehlerfreien Datentransfer (normal / best path only) optimiert sind.

Ein gemeinsamer Speicher für Nutzdaten und die Steuerköpfe aller relevanten Schichten wird eingesetzt, um das Bewegen von Daten zu vermeiden. Die Kommunikation der Schichtenprozessoren wird über paarweise vorhandene Befehlswarteschlangen entkoppelt (siehe Bild 4.28). Eine Verwaltung von Prozessen entfällt weitgehendst durch die fest vorgegebene Abfolge in den einzelnen Schichtenprozessoren.

Eine Hardwareeinheit ermöglicht Speicherzugriffe auf 1 bis 4 Oktetts unabhängig von deren Ausrichtung bezüglich Langwortgrenzen. Durch verbindungsspezifisches Reservieren von Puffern kann u.a. statische Steuerkopfinformation schon im Vorgriff eingetragen werden. Informationen zur generellen Strategie der Speicherverwaltung und eine Beschreibung der Interaktion mit dem AR werden nicht angegeben.

Wesentlicher Aspekt des japanischen Vorschlags ist die angedeutete Möglichkeit der Vereinfachung von Betriebssystemfunktionen durch

1. Trennen von Daten- und Kontrollfluß und
2. Serialisieren interner Abläufe

Trotz - oder gerade wegen - der Vielfalt der beschriebenen Lösungsansätze, von denen viele auch erst während der Arbeit veröffentlicht wurden, ist *der* dominierende Schwachpunkt von Kommunikationssystemen nicht gefunden. Stattdessen wurde in diesem Kapitel die Notwendigkeit für einen umfassenden *Gesamtsystementwurf* und der dominierende Einfluß der *protokollunabhängigen Implementierungsfaktoren* auf die Systemleistungsfähigkeit deutlich.

Kapitel **5**

Das Transportsystem

Die Beschreibung des entworfenen Systems erfolgt in 4 Teilen: bevor der Weg zur schlußendlich gewählten Implementierungsstruktur und die Herleitung wesentlicher Systementscheidungen geschildert wird, werden zunächst grundsätzliche Ziele und Randbedingungen der Arbeit zusammengestellt. Eine Übersicht über die Funktionen und Implementierungsaspekte der einzelnen Hardwarekomponenten des Systems schließen sich an. Im vierten Teil werden Merkmale der Systemsoftware angesprochen.

5.1 Zielsetzung und Vorgaben

Ausgangspunkt der Arbeit war die gravierende Differenz zwischen verfügbaren FDDI-Netzwerkadaptern mit Nutzbitraten über 90 Mbit/s oberhalb der Medienzugriffsschicht 2a und der geringen erzielbaren Leistungsfähigkeit von bestenfalls einem Mbit/s beim Nutzer der Transportschicht. Ursprüngliches Ziel der Arbeit war die Bestimmung der theoretischen Leistungsfähigkeit von ISO/OSI-Protokollen oberhalb der Schicht 2a. Parallelität in Protokollen und Unterstützung durch Hardware sollten dabei berücksichtigt werden (siehe auch Abschnitt 1.2).

Schon beim Erstellen von Ablaufdiagrammen geringer Granularität und anschließenden ersten Verfeinerungen wurde deutlich, daß die Begrenzung der Leistungsfähigkeit nur zum Teil von der logischen Funktionalität bestimmt wird. Bedingt durch die Vielzahl von beeinflussenden Parametern schied eine theoretische Betrachtung aus. Statt dessen sollte der Entwurf eines Implementierungsrahmens für die technologieunabhängigen Schichten eines Transportsystems für Hochgeschwindigkeitsnetze durchgeführt werden. Folgende Randbedingungen sollten beachtet werden:

1. Nach wie vor ist die erzielbare Leistungsfähigkeit von ISO/OSI-Protokollen von Interesse.
2. Der Entwurf soll so flexibel sein, daß eine Anpassung an unterschiedliche Protokollparameter und -versionen leicht machbar ist. Idealerweise sollte sogar der Einsatz von unterschiedlichen, bis hin zu nicht OSI-konformen, Protokolltürmen möglich sein.
3. Das System soll besonders für eine Optimierung der Datenübertragungsphase ausgelegt werden. Darin wiederum ist zunächst eine Konzentration auf den fehlerfreien Normalfall angebracht (best / normal path tuning).
4. Um die Ergebnisse verallgemeinern zu können, soll die Abhängigkeit der Ergebnisse von der Umgebung möglichst klein sein bzw. quantitativ angegeben werden können.
5. Der Entwurf ist hauptsächlich für eine LAN/MAN-Umgebung auszulegen.
6. Zunächst sind ausschließlich Endsysteme zu beachten, also keine auf dem Weg zwischen den eigentlichen Kommunikationspartnern liegenden Kopplungsstationen.
7. Die Bestimmung und Nutzung inherent vorhandener Parallelität ist zu beachten.
8. Ebenso ist der Einsatz jeglicher Art von Hardwareunterstützung bis hin zum Entwurf eigener anwendungsspezifischer integrierter Schaltkreise miteinzubeziehen.
9. Das Bewegen von Daten ist soweit als möglich zu vermeiden.

10. Das Gesamtsystem soll nicht rein akademischer Natur sein. Technische Machbarkeit und Wirtschaftlichkeitsüberlegungen sind daher unabdingbar.

Aus den Forderungen leiten sich direkt einige konkrete Vorgaben ab:

- Der bevorzugte Einsatzbereich (LANs und MANs - Punkt 5) in Verbindung mit der gewünschten OSI-Kompatibilität (Punkt 1) legt den Protokollturm fest: entsprechend den Ausführungen in Abschnitt 3.3.2 ist dies die Kombination von LLC Typ 1, dem verbindungslosen Vermittlungsdienst CLNS und der leistungsfähigsten Klasse TP4 auf Transportebene [304].
- Ein Nebeneffekt der Konzentration auf den LAN/MAN-Bereich ist das Vernachlässigen der Probleme, die durch ein großes Bandbreiten-Verzögerungs-Produkt enstehen können.
- Der Umfang von Arbeiten an Protokollmechanismen ist durch die Verwendung von ISO/OSI-Protokollen auf kleinere Abstimmungen reduziert. Die Entwicklung eines (weiteren) neuen Protokolls braucht als mögliches Mittel zur Leistungssteigerung damit ebenfalls nicht berücksichtigt werden.
- Die Beschränkung auf Endsysteme (Punkt 6) bewirkt, daß einerseits der komplexe Bereich der Wegewahl unberücksichtigt bleiben kann und andererseits eindeutige Richtungen der Informationsflüsse innerhalb des TpS festgelegt sind.
- Die Vermeidung von Datenbewegungen (Punkt 9) resultiert in einer Benutzung gemeinsamer Speicherkonzepte für die verschiedenen Schichten, wodurch die theoretisch im OSI-Referenzmodell enthaltene weitgehende Unabhängigkeit der Schichten zwangsläufig eingeschränkt wird.
- Um den Einfluß einer spezifischen Umgebung abschätzen zu können (Punkt 4) ist neben umfangreichem Detailwissen der Zugang zu und die aktive Einflußnahme auf Internas erforderlich. Dies ist im allgemeinen nicht oder nur sehr problematisch möglich. Deshalb liegt es nahe, von einem eigenständigen und damit unabhängigen Teilsystem auszugehen.

5.2 Systementwurf

Vier Ansatzpunkte zur Optimierung der Leistungsfähigkeit waren für den Entwurf von Bedeutung:

- Ausnutzung der Parallelität in OSI/TP4 durch Abbildung auf parallele Prozessoren
- Identifizierung zeitkritischer Protokollfunktionen und deren Unterstützung durch Hardware
- Systemüberlegungen zur optimierten Lasttrennung und Vermeidung von Engpässen
- Identifizierung zeitaufwendiger Systemfunktionen und deren Unterstützung durch Hardware

Eine Optimierung einzelner Ansätze kann einen erheblichen Implementierungsaufwand erfordern und dadurch zu Kollisionen mit der Forderung 10 zur Folge haben. So erfordert die Ausnutzung der Parallelität

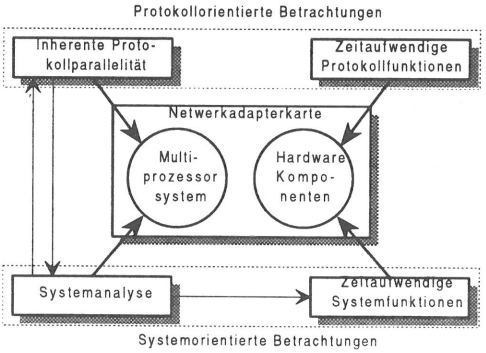


Bild 5.1: *Die Abhängigkeiten verschiedener Entwurfsfaktoren*

in einem Protokoll eine entsprechende Abbildung auf geeignete Systemstrukturen; deren Eignung ist häufig durch technologische oder wirtschaftliche Überlegungen begrenzt. Jede Parallelisierungsentscheidung macht daher eine Überprüfung der Folgen für die Implementierung nötig. Ergebnis dieser Überprüfung kann wiederum eine Revision der Ausnutzung der natürlichen Protokollparallelität sein. Bild 5.1 verdeutlicht diesen und ähnliche Zusammenhänge.

Einfach gezeichnete Pfeile sollen Auswirkungen eines Ansatzpunktes auf Überlegungen aufzeigen, die einem anderen Ansatzpunkt zugeordnet sind. Fett gezeichnete Pfeile stellen endgültige Entwurfsentscheidungen dar. Die Klassifikation in protokoll- und systemorientierte Ansätze verdeutlicht den Gegenstand der entsprechenden Betrachtungen. Das Ziel ist hier eine leistungsfähige und effiziente Implementierung der Protokolle und nicht etwa eine Veränderung der Protokollfunktionen.
Die getrennte Betrachtung der beiden grundsätzlichen Lösungsansätze wird auch hier beibehalten.

5.2.1 Protokollorientierte Optimierungen

Durch die Vorgabe der OSI-Kompatibilität konzentrieren sich protokollspezifische Verbesserungen auf das Ausnutzen von Parallelität, die Wahl der Sende- und Empfangsstrategie sowie die Festlegungen von benutzten bzw. erlaubten Protokolloptionen.

5.2.1.1 Parallelitäten der Protokolle

Höchste Funktionseinheit des betrachteten Protokollturms sind Verbindungen (Bild 5.2 a). Für jede Verbindung müssen zum einen die Protokollmechanismen realisiert sein und zum anderen ein Informationsaustausch zu den benachbarten Schichten möglich sein. Diese Unterscheidung führt zur Teilung einer Verbindung in drei Funktionseinheiten (Bild 5.2 b):

- Schnittstelle zum Anwendungsrechner AR (host),
- Protokollmechanismen und
- Schnittstelle zur Medienzugangseinheit.

Eine Fließbandverarbeitung bietet sich für die Parallelisierung dieser Funktionseinheiten an.

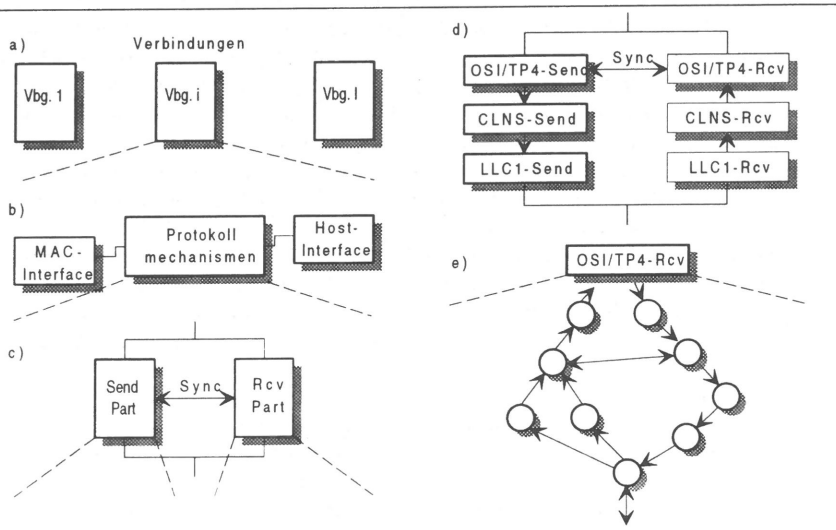


Bild 5.2: *Parallelitätsebenen innerhalb eines Protokollturms*

OSI-TP4 realisiert Vollduplexverbindungen. Die Protokollmechanismen für die Sende- und Empfangsrichtung hängen zwar im Sinne einer Synchronisation miteinander zusammen, sind aber vom Ablauf her verschieden. Letzteres gilt auch für die Protokolle der Schichten 3 und 2b. Eine logische Unterteilung in einen Sende- und Empfangsteil ist daher sinnvoll (Bild 5.2 c). Die Zuordnung dieser zu zwei isolierten Verarbeitungsbereichen parallelisiert diese Vorgänge.

Innerhalb des Sende- und Empfangsteils sind die Mechanismen der 3 Schichten realisiert. Die Verwendung einer 3-stufigen Fließbandverarbeitung macht deren Parallelisierung möglich (Bild 5.2 d).
Die Protokollmechanismen von OSI-TP4 werden sowohl für die Empfangs- als auch für die Senderichtung durch Unterfunktionen realisiert. In [329] wird eine Anordnung dieser Funktionen hergeleitet, die die Parallelisierbarkeit innerhalb einer maximalen Verarbeitungsstruktur widerspiegeln. Sie besteht sowohl aus Abschnitten mit Fließbandverarbeitung zur Nutzung zeitlicher Parallelität als auch aus Feldern von Verarbeitungseinheiten zur Nutzung räumlicher Parallelität. Kennzeichnend für diese Struktur ist eine komplizierte Verbindungstopologie. Die Struktur e) in Bild 5.2 (aus [329]) soll qualitativ die komplizierten Zusammenhänge zwischen den atomaren Unterfunktionen von OSI-TP4 verdeutlichen. Die Struktur des eingesetzten verbindungslosen Dienstes der Vermittlungsschicht und auch des LLC-1-Protokolls sind dagegen so einfach, daß eine weitere Unterteilung von vornherein nicht sinnvoll ist.

Eine erste Bewertung liefern folgende Systemüberlegungen:
Eine feste Zuordnung einzelner Verbindungen zu Prozessoren ist leicht zu implementieren. Auch die Migration von mehreren Prozessen eines Prozessors hin zu einem Multiprozessorsystem ist problemlos machbar. Durch die - von Sonderfällen abgesehene - inhaltliche Unabhängigkeit von Verbindungen gibt es keine Kollisionen zwischen den einzelnen Prozessen bzw. Prozessoren. Allerdings ist die erzielbare Leistungssteigerung durch die Anzahl gleichzeitig aktiver Verbindungen begrenzt. Die Last der einzelnen Prozessoren hängt zudem von der Charakteristik der jeweiligen Verbindung ab und wird daher stark unsymmetrisch sein. Eine in Multiprozessorsystemen grundsätzlich anzustrebende möglichst gleichmäßige Verteilung der Last ist nicht möglich. Der einzig mögliche Gewinn liegt in der Vermeidung der Verarbeitungszeit, die zur Herstellung des Verbindungskontextes nötig ist. Dieser Gewinn ist jedoch unerheblich.

Die restlichen Ansätze sind prinzipiell sinnvoll. Ihre Wirksamkeit wird durch eine Zunahme der Zahl der Verarbeitungseinheiten (VEs) unterstützt und durch entstehende Konkurrenz um gemeinsame Betriebsmittel gemindert. Die Frage, ab welcher Parallelitätsebene die Minderung dominiert, ist letzten Endes nur durch Simulation der Ansätze zu beantworten. Folgende Aussagen sind jedoch prinzipiell richtig:

- Die Schnittstellen wickeln Funktionen ab, die jedes Oktett einer PDU betreffen. Sie sind folglich zeitaufwendig, und ihre Isolierung und Parallelisierung ist daher sinnvoll.
- Eine Trennung in Sende- und Empfangsteil kann zu einer Reduzierung von Konkurrenz beim Speicherzugriff führen, falls dieser ebenfalls in einen Sende- und Empfangsteil getrennt ist. Der Synchronisationsaufwand zwischen den entsprechenden VEs ist in OSI-TP4 nicht als Engpaß anzusehen. Die beiden Teile sind für die Kommunikation mehrerer Verbindungen verantwortlich. In Umgebungen, in denen ein balancierter Gesamtverkehr in beiden Richtungen ausreichend häufig zu erwarten ist, ist damit die Vorsehung einer Sende- und einer Empfangs-VE sinnvoll.
- Eine Fließbandverarbeitung der einzelnen Protokollschichten, d.h. pro Schicht eine VE, ist allein schon durch die gravierenden Unterschiede in der Komplexität der beteiligten Schichten wenig sinnvoll. Darüber hinaus ist die geringe Komplexität der Schichten 2b und 3 gegenüber den Kosten für mehrere VEs und deren Synchronisierung nicht zu rechtfertigen.
- Eine begrenzte Aufteilung von OSI-TP4 kann nach Zitterbart [329, 331] Verbesserungen liefern. Voraussetzung für eine stark atomisierte Aufteilung auf VEs ist eine aufwendige Verbindungstopologie. Erhebliche Zugriffskonflikte bei gemeinsamen Verbindungswegen und Speichern sind zu erwarten. Das erste Problem läßt sich teilweise durch Verwendung von Transputern, die über 4

integrierte unabhängige Verbindungskanäle verfügen, auffangen [78, 133, 401, 330]. Das zweite Problem beeinflußt die erzielbare Leistungssteigerung erheblich, wie die Untersuchungen von Zitterbart zeigen. Die von ihr in [329] beschriebene Implementierung mit 8 Transputern und getrennten Sende- und Empfangsspeichern bleibt unter einer maximalen Durchsatzrate von 2000 TPDUs pro Sekunde

Für die entworfene Implementierung wurden daraus folgende Entscheidungen abgeleitet:

- Für die Sende- und Empfangsrichtung sind unabhängige Verarbeitungsbereiche mit jeweils einem oder mehreren Prozessoren (z.B. für TP4-Teilfunktionen), lokalen Speichern und Zugang zum globalen Kommunikationsspeicher vorzusehen.
- Kommunikationspfade zwischen den beiden Bereichen sind vorzusehen.
- Für die Schnittstellenfunktionen wird eine intelligente DMA-Unterstützung vorgesehen.

Implementiert wurde die Version mit je einem Prozessor für die Empfangs- und für die Senderichtung. Da die Leistungsfähigkeit über den anvisierten 100 Mbit/s lag, wurden andere Varianten nicht betrachtet.

5.2.1.2 Aufwendige Protokollfunktionen

Die Protokollfunktionen lassen sich in 3 Gruppen klassifizieren:

A) Funktionen, die nur bei Auf- oder Abbau einer Verbindung gebraucht werden,
B) Funktionen, die für die Verarbeitung der Steuerköpfe nötig sind und
C) Funktionen, die jedes Oktett einer PDU (oder deren Datenteil) manipulieren müssen.

Durch die Konzentration auf die Datenübertragungsphase (Punkt 3 in 5.1) ist eine intensivere Betrachtung und Optimierung der Funktionen vom Typ A zunächst nicht erforderlich. Mit einer typischen Übertragungszeit von 5 µs pro Kilometer ergeben sich im angestrebten LAN-/MAN-Umfeld Gesamtzeiten für Verbindungsaufbauten im Bereich von einigen wenigen ms. Bei zeitkritischen Anwendungen, für die selbst diese Zeiten zu hoch sind, wird die Einrichtung langlebiger Verbindungen vorgeschlagen.

Die zweite Gruppe erfordert eine Einzelbewertung der Teilfunktionen. Die Leistungsfähigkeit der implementierten Protokollsoftware (Abschnitt 5.4) zeigt, daß keine Funktion besondere Vorkehrungen erfordert! Bei den Funktionen der letzten Gruppe ist dagegen ein erheblicher Aufwand zu erwarten. Folgende Funktionen wurden hier identifiziert:

C1 - Ein-/Auslesen von Daten vom/zum AR
C2 - Ein-/Auslesen von Daten von/zur MAC
C3 - Kopieren von Daten zwischen Schichten des TpS
C4 - Prüfsummenerstellung in Senderichtung
C5 - Prüfsummenauswertung in Empfangsrichtung
C6 - Trennung von Steuerköpfen und Daten in Empfangsrichtung (Begründung folgt).

Die Lösung für C1 und C2 sind leistungsfähige DMA-Einheiten mit entsprechender Intelligenz. C3 wurde durch die Verwendung von gemeinsamen Speichern für alle Schichten und entsprechenden Adreßzeigern generell vermieden. Es ist anzumerken, daß die eigentlichen Nutzdaten, hier also Dienstdateneinheiten der Transportschicht (TSDUs), innerhalb des TpS nur für die Prüfsummengenerierung bzw. -auswertung interessant sind, ansonsten aber das TpS unbehandelt passieren. Werden für C4 und C5 gesonderte Mechanismen vorgesehen (siehe unten), beschränkt sich die Übergabe von Daten zwischen Schichten auf die Steuerköpfe. Eine gesonderte Behandlung von Nutzdaten und Steuerköpfen ist also möglich.

Für C4 wäre ein Baustein ideal, der die Erstellung der Prüfsumme gleichzeitig mit dem Einlesen der Daten vom AR ermöglichen würde. Dies erscheint durch die Einbeziehung aller Oktetts des Steuerkopfes für die Prüfsumme, durch die Position der Prüfoktetts im Steuerkopf und durch den positions- und längenabhängigen Algorithmus grundsätzlich ausgeschlossen zu sein. In Abschnitt 5.3.3.3 wird gezeigt, wie

es trotz den schwierigen Randbedingungen möglich ist.

Um auch in Empfangsrichtung eine Kopplung von Datentransfer und Auswertung der (Schicht 4-)Prüfsumme zu erreichen, ist die Kenntnis über den Beginn einer Transport-PDU innerhalb des von der Medienzugangseinheit empfangenen MAC-Rahmens erforderlich.
Eine - der Empfangs-VE vorgeschaltete - schnelle VE mit eigenem Speicher könnte diese Aufgabe erledigen. Allerdings wäre ein zusätzliches Kopieren der Daten notwendig. Eine Zwischenspeicherung ist nur zu vermeiden, falls die Bestimmung des Beginns einer TPDU in Realzeit, d.h. schritthaltend mit der Geschwindigkeit der angelieferten Daten (on-line oder on-the-fly), während der Übernahme des Rahmens von der Medienzugangseinheit erfolgen kann[3].

Durch unzählige Kombinationen von Formaten und Optionen zeichnen sich Steuerköpfe der OSI-Protokolle durch eine große Variabilität aus. Zusätzlich können sich prinzipiell auch mehrere TPDUs in einem einzigen MAC-Rahmen befinden. Ein Behandeln allgemeiner OSI-Steuerköpfe in Echtzeit erfordert daher eine hohe Rechenleistung. Mit einigen - größtenteils standardkompatiblen - Einschränkungen (Details siehe Abschnitt 5.2.1.3), die im wesentlichen die Vielfalt der Steuerkopfformate reduziert, kann diese Aufgabe mit Hardware handhabbarer Komplexität erfüllt werden. Zu diesem Zweck wurde ein OTF-Baustein (On-the-Fly) konzipiert. Ist der Beginn einer TPDU bekannt, kann auch für die Auswertung der Prüfsumme in Empfangsrichtung Hardware eingesetzt werden.
Alle Überlegungen erlauben eine gleichzeitige Ausführung der Funktionen vom Typ C.

5.2.1.3 Einschränkende Vereinbarungen

In einer Umgebung, die den Vorgaben und Randbedingungen aus Abschnitt 5.1 entspricht, sind einige Richtlinien zur Benutzung der ISO/OSI-Standards sowohl sinnvoll als auch möglich. Im Hinblick auf leistungssteigernde Maßnahmen stehen zwei grundsätzliche Ziele im Vordergrund:

1. Vermeidung von redundanter Funktionalität innerhalb des TpS und
2. Ermöglichung einer effizienten Implementierung durch Reduzierung der Variabilität.

Die ISO/OSI-Standards erlauben eine Segmentierung sowohl innerhalb der Transport- als auch innerhalb der Vermittlungsschicht. Durch eine geeignet vorgenommene Wahl der TPDU-Größe, kann dafür gesorgt werden, daß Segmentierung ausschließlich in der Transportschicht durchgeführt werden muß und ein zweites Segmentieren innerhalb der Vermittlungsschicht von vornherein überflüssig wird.

Für das vorliegende TpS wurde eine maximale TPDU-Länge von 1024 Oktetts vereinbart. PDUs dieser Länge passen ohne weitere Unterteilungen in die MAC-Rahmen aller gängigen Medienzugriffsprotokolle. Sollte das TpS an einer Kommunikationsbeziehung beteiligt sein, die auch Verbindungsabschnitte mit kleinerer maximal zulässiger Länge enthält (hier sind insbesondere die zukünftigen ATM-basierten Netze zu nennen), wird davon ausgegangen, daß beim Übergang auf entsprechende Verbindungsabschnitte ausreichend intelligente Zugangseinheiten vorhanden sind, die das dann notwendige Segmentieren und Zusammensetzen eigenständig - und damit transparent für das TpS - durchführen.

Eine weitere Einschränkung ist der Verzicht auf die Protokollmechanismen zur Verkettung mehrerer (N)-PDUs zu einer (N-1)-SDU und zur Blockbildung einer (N)-PDU aus mehreren (N)-SDUs. Die Schwierigkeiten bei letzterem Mechanismus und dessen daraus resultierende eher theoretische Bedeutung wurden schon in Abschnitt 3.1.6.1 behandelt. Die Verkettung von kurzen PDUs zu einer SDU hat die Erhöhung des Anteils von übertragener Nutzinformation durch die Einsparung von ansonsten notwendigen Steuerköpfen für jede einzelne PDU als Hintergrund. Gerade kurze PDUs gehören allerdings oft zu Anwendungen, die an einer kurzen Übertragungszeit interessiert sind, was jegliche Zwischenspeicherung aus-

[3] Eine sinnvolle Prüfsummenberechnung setzt das Vorhandensein *kompletter* TPDUs in einem MAC-Rahmen voraus. Eine Segmentierung auf der Vermittlungsschicht stünde offensichtlich im Widerspruch zu dieser Forderung. Geeignete Maßnahmen zur Auflösung des Widerspruchs sind im nächsten Kapitel beschrieben.

schließt. Außerdem ist in modernen Netzen die verfügbare Übertragungsbandbreite weder die leistungsbegrenzende noch die kostendominierende Ressource.

Schon in heutigen Implementierungen sind beide Mechanismen nicht zu beobachten und werden - wenn überhaupt - nur passiv unterstützt. Bei einem Verzicht kommt der Vorteil der wesentlich vereinfachten Trennung von Steuerköpfen und Daten hinzu, welcher erst eine optimierte Protokollverarbeitung und effiziente Speicherverwaltung ermöglicht. Die genannten Gründe wurden im vorliegenden Fall als gewichtiger eingestuft als der Verlust an 100%-iger Standardkompatibilität.

Explizit erwähnt werden sollen die entsprechenden Protokollmechanismen für Verbindungen. Da für die Vermittlungsschicht ein verbindungs*loser* Dienst gewählt wurde (sowie auf der Schicht 2b der ebenfalls verbindungslose Protokolltyp 1), entfällt sowohl das Multiplexen von mehreren (N)- Verbindungen auf eine (N-1)-Verbindung, als auch der umgekehrte Vorgang des Aufteilens (splitting) einer (N)-Verbindung auf mehrere (N-1)-Verbindungen. Wären die Mechanismen vorhanden, würden sie, was das Auswertung der Steuerköpfe und deren Zuordnung zu Verbindungen angeht, problemlos vom TpS unterstützt werden. Für die Problematik unterschiedlicher Dienstgüten bzw. deren Anforderungen würde das TpS allerdings auch keine Patentlösungen beinhalten.

Die Protokollfunktionen, die für das spezielle Behandeln von Duplikaten und Verlusten von Quittierungen vorgesehen sind, können erheblich vereinfacht werden. Diese Vereinfachungen sind gerechtfertigt, weil es nicht mehr erforderlich ist, mit allen Mitteln - u.U. sogar auf Kosten der erzielbaren Geschwindigkeit - Speicher einzusparen. Bleibt z.B. das Empfangsfenster immer offen, d.h. wird das in den Quittungen enthaltene Kreditfeld nie auf Null gesetzt, brauchen die Mechanismen, welche Wiederholungssendungen in diesem Sonderfall zulassen und gleichzeitig Verklemmungen durch fehlgeleitete Quittierungen ausschließen, nicht implementiert zu werden.

Ebenso kann der Einsatz weiterer Sondermechanismen und die Verwendung von zusätzlichen Subfolgenummern vermieden werden: es wird vereinbart, die obere Grenze des Sendefensters nicht zu reduzieren, während die betreffenden Nummern noch in Benutzung sein können [35, 280]. Benutzt die Gegenseite einen der Mechanismen, wird die zusätzlich vorhandene Information vom TpS nicht ausgewertet.

Der Standard sieht einen Parameter zur Bestätigung von Flußsteuerungsmaßnahmen (flow control confirmation parameter) vor. Dieser ist für den Einsatz nach einer Reduzierung der oberen Fenstergrenze, nach dem Schließen des Empfangsfensters und zur Kennzeichnung wiederholt ausgesandter Flußsteuerungsinformation vorgesehen. Da die beiden erstgenannten Punkte im TpS nicht vorkommen, wird auf den Einsatz des (ohnehin nur optionalen) Parameters ganz verzichtet.

Um den Anforderungen schneller Netze besser gerecht zu werden (siehe Abschnitt 3.3.1.1) und Wahlmöglichkeiten zu vermeiden, wird ausschließlich das erweiterte Format für die PDU-Numerierung (31 statt 7 Bit) und das Kreditfeld (7 Bit statt 4) festgelegt. Dies entspricht gleichzeitig der im Standard empfohlenen Default-Einstellung. Sollte die Gegenseite auf der Benutzung des reduzierten Formats bestehen, kann die Kommunikation durch die aktuelle Version des TpS nicht unterstützt werden. Die dazu notwendigen Änderungen wären leicht vorzunehmen, da sie nur die eigentliche Protokollsoftware betreffen.

Außer der möglichen Verwendung der bevorzugten Datenübertragung (expedited data transfer), die in Ausnahmesituationen wichtig sein kann, verspricht keine der im Standard aufgeführten Optionen einen Nutzen, der den Aufwand ihrer Implementierung rechtfertigen würde. Entsprechende Erweiterungen sind aber - wiederum durch Softwareänderungen - grundsätzlich möglich. Zum Ausrichten der Steuerköpfe auf einige wenige feste Längen ist die Verwendung des mit Null- oder auch NOP-Option bezeichneten Feldes innerhalb der Vermittlungsschicht vorgesehen. Im bestehenden TpS ist beispielsweise ein ganzzahliges Vielfaches von 4 Oktetts als Steuerkopflänge günstig.

Die wichtigsten Vereinfachungen sind nochmals in Tabelle 5.1 zusammengefaßt. Die Bezeichnung

konform kennzeichnet Änderungen, die im Einklang mit den Standards sind.

Tabelle 5.1: *Protokolloptimierungen*

Protokolloptimierung	konform	kompatibel
Keine Segmentierung auf Schicht 3		X
Kein Verbinden / Auftrennen von TPDUs		X
Kein Multiplexen/Demultiplexen und Splitting/Recombination	X	
Keine Sub-Sequenznummern in AK TPDUs (aktiv)	X	
Keine Sub-Sequenznummern in AK TPDUs (passiv)		X
Obere Fenstergrenze wird nie reduziert	X	
Empfangsfenster nie geschlossen (Kredite nie = 0)	X	
Kein Flußsteuerungs-Quittierungsparameter	X	
Erweitertes Format nur für DT TPDU Nummerierung		X

Das resultierende Transportsystem kann zwangsläufig nicht mehr 100%-ig konform zu den Standards sein. Untersuchungen bestehender ISO-konformer Implementierungen, die u.a. im Rahmen internationaler Kooperationsprojekte durchgeführt wurden, und praktische Erfahrungen von verschiedenen Netzbetreibern belegen andererseits, daß die beschriebenen und in Tabelle 5.1 mit kompatibel markierten Einschränkungen im praktischen Einsatz nicht zu Problemen führen.

Darüber hinaus ist im TpS ein Transparentmodus integriert, der nicht für das TpS geeignete MAC-Rahmen ohne jegliche Bearbeitung direkt an den AR weiterreicht. Damit ist dann neben der Bearbeitung anderer Protokolltürme auch eine Behandlung von, mit einer geringen Restwahrscheinlichkeit eventuell auftretenden, nicht TpS-geeigneten, aber OSI-kompatiblen, MAC-Rahmen möglich.

5.2.1.4 Sende- und Empfangsstrategie

Entsprechend den im Abschnitt 4.1.1.2 beschriebenen Untersuchungen wird für die Protokollsoftware die dort mit S6 bezeichnete Strategie gewählt. Deren Arbeitsweise ist durch die Verringerung des Sendefensters auf die Größe eines (erwarteten) Büschels und die zusätzliche Speicherung zusammenhängender Ketten von PDUs außerhalb des reduzierten Fensters charakterisiert.

Der Empfänger kann zwei Ketten von jeweils zusammenhängend empfangenen PDU-Blöcken verwalten. Die Größe des Sendefensters hängt im Normalfall (keine Empfangslücke) von der Größe des freien Empfangsspeichers ab. Liegt diese über einem einstellbaren Wert, wird das Fenster auf MaxCreditVal vergrößert. Liegt sie unter einem Minimalwert, wird das Fenster auf MinCreditVal verringert. Dazwischen entspricht die Größe des Sendefensters der Anzahl der im freien Empfangsspeicher ablegbaren PDUs.

Tritt eine Empfangslücke auf, wird das Sendefenster durch eine sofortige Quittung entweder auf einen als ReducedCreditVal bezeichneten Wert (entspricht der angenommenen Verlustbüschellänge) oder auf die Größe der Empfangslücke reduziert, falls diese größer als ReducedCreditVal ist. War aufgrund fehlenden Empfangsspeichers das Sendefenster vorher kleiner, so wird es nicht verändert. In keinem Fall unterschreitet die Größe des Sendefensters MinCreditVal. PDUs außerhalb des verkleinerten Sendefensters werden vom Empfänger gespeichert, sofern sie einen lückenlosen Block bilden. PDUs innerhalb des Fensters werden nur akzeptiert, falls sie in normaler Empfangs-Sequenz ankommen.
Nach Schließen der Empfangslücke wird das Fenster wieder entsprechend dem Normalfall berechnet. Die Werte ReducedCreditVal, MaxCreditVal und MinCreditVal sind für jede Verbindung bei Verbindungsaufbau festzulegen. MinCreditVal wird niemals zu Null gesetzt.

5.2.2 Systemorientierte Optimierungen

5.2.2.1 Allgemeine Festlegungen

Bedingt durch die Art der Operationen in den ausgewählten Protokollen wurden für die VEen RISC-Prozessoren ausgewählt. Kurze Instruktionszeiten, interne Zwischenspeicherung (caching) durch eine hohe Anzahl von Registern und die Verwendung von 32-bit-Wörtern sind Vorzüge, auf die ein guter Teil der Leistungsfähigkeit des TpS zurückzuführen ist. Die CPU-Taktrate wurde auf 33 MHz festgelegt.

Um den Aufwand durch Unterbrechungsanforderungen zu reduzieren, wurden grundsätzlich (Software-) Abfragemechanismen (polling) eingesetzt (siehe Abschnitt 5.4.2).

5.2.2.2 Minimierung der Zugriffskonflikte

Zugriffskonflikte wurden bei gemeinsam benutzten Speichern und Verbindungsstrukturen erwartet.

Speicherzugriffe

Neben der Einführung lokaler Speicher für die einzelnen Prozessoren und der grundsätzlichen Reduzierung gemeinsam genutzter Speicherbereiche, wurde dieses Problem durch eine mehrfache Teilung des Kommunikationsspeichers und die Vorsehung von Speichern mit (virtuell) gleichzeitiger Zugriffsmöglichkeit von zwei Seiten (dual-ported memories) gelöst:

A) Die Trennung in einen Daten - und einen Steuerkopfspeicher isoliert die steuerkopfbezogenen Zugriffe des Sende- bzw. Empfangsprozessors von den datenbezogenen Zugriffen des ARs und der Medienzugriffseinheit bzw. deren zugeordneten intelligenten DMA-Baugruppen. Diese Trennung trägt darüber hinaus folgenden Aspekten Rechnung:

- Steuerköpfe werden von den Protokollprozessoren maximal in einer Breite von 4 Oktetts angesprochen. Die Zugriffe müssen sehr schnell erfolgen. Bei den Daten sind theoretisch erheblich größere Zugriffsbreiten und auch langsamere Zugriffe möglich. Verschiedene Auslegungen des Speicherzugriffs sind deswegen sinnvoll. Schnelle 32-Bit-Zugriffe sind für den Steuerkopfspeicher geeignet, breitere Zugriffsbusse zu langsameren Speichern für die Datenteile. Die Größe des Steuerkopfspeichers (256-512 KB) liegt um mehr als eine Größenordnung unter der des Datenspeichers (4 - 8 MB). Die verringerten technologischen Anforderungen an den Datenspeicher reduzieren daher die Speicherkosten. Ist beim Steuerkopfspeicher der Einsatz von statischen CMOS-RAMs unbedingt erforderlich, kann zum Aufbau des Datenspeichers auf die weniger leistungsfähigen aber dafür erheblich preisgünstigeren dynamischen Speicherbausteine hoher Kapazität zurückgegriffen werden.
- Die Trennung ermöglicht eine getrennte Verwaltung der Speicher. Von der Medienzugangseinheit werden PDUs sequentiell abgelegt. Diese sind verschiedenen Verbindungen zugeordnet. Der AR übernimmt die Datenteile der PDUs je nach Verbindung unterschiedlich schnell. Die Folge ist eine Zersplitterung des Speichers. Das gleiche kann in Senderichtung durch unterschiedlich schnelle Quittierung innerhalb einzelner Verbindungen bewirkt werden. Eine zyklische Verwaltung des Speichers ist daher unwirtschaftlich. Die Verarbeitung der Steuerköpfe läßt dagegen eine zyklische Verwaltung zu. Die für die Senderichtung erforderliche Zwischenspeicherung der OSI/TP4-Steurkopfinformation (bis die PDUs quittiert sind) beschränkt sich auf wenige Oktetts pro PDU und kann in einem lokalen Speicher erfolgen.

In Empfangsrichtung erfordert die Trennung der Daten und Steuerköpfe im wesentlichen die gleiche Funktionalität wie sie für den OTF-Baustein zur Abwicklung der Prüfsummenverifikation bereits beschrieben wurde. Daher wurde die nötige Erkennung des Beginns des Datenteils innerhalb eines empfangenen MAC-Rahmens ebenfalls im OTF-Baustein realisiert. In Senderichtung übernimmt die Schnittstelle zur Medienzugangseinheit (DMAMacSend) die Funktion des Zusam-

menführens von Steuerkopf und Datenteil.

B) Trennung des Steuerkopfspeichers in einen Sende- und Empfangsteil (Abschnitt 5.2.1.1)

C) Trennung des Datenspeichers ebenfalls in einen Sende- und Empfangsteil

Eine physikalische Trennung entsprechend den beiden Kommunikationsrichtungen ist vordergründig betrachtet nur dann empfehlenswert, wenn die Medienzugangseinheit bzw. das zugehörige Medienzugriffsprotokoll gleichzeitig Daten abliefern und übernehmen kann. Aber selbst, wenn dies - wie z.B. bei FDDI - nicht der Fall ist, kann eine physikalische Separierung von Sende- und Empfangsbereich sinnvoll sein: Die einzelnen Bereiche sind für die Bearbeitung einer gewissen Zahl von PDUs pro Sekunde ausgelegt. Diese Zahl entspricht aber bei weitem nicht der theoretisch, durch ausschließliche Verwendung von PDUs minimaler Länge, möglichen PDU-Rate. Dies ist gleichbedeutend mit der Aussage, daß die Bearbeitungszeit einer (kurzen) PDU deutlich über der Belegungszeit der Medienzugangseinheit dieser PDU liegen kann. Liegt eine Anwendung vor, die einen hohen Anteil kurzer Meldungen impliziert, ist eine Trennung in Sende- und Empfangsteil sinnvoll.

Die wirtschaftliche physikalische Zusammenfassung - eine logische Trennung ist beizubehalten - der beiden Speicherbereiche (sowie der zugehörigen Verwaltungen und den Verbindungsstrukturen) ist jedoch leicht und ohne Auswirkungen auf andere Teile des TpS durchführbar.

Kommunikationspfade

Um die Kommunikation zwischen den drei (der dritte Bereich ist noch einzuführen) Verarbeitungsbereichen nicht zu einem Engpaß werden zu lassen, wurde kein gemeinsamer Kommunikationsbus vorgesehen. Vielmehr wurden die Bereiche jeweils paarweise durch einen Koppelbaustein (PCD, Process Coupling Device) verbunden. Die Funktionsweise des PCDs wird in Abschnitt 5.3.2.1 beschrieben.

5.2.2.3 Kommunikation zwischen den Verarbeitungsbereichen

Um eine gleichmäßige Auslastung der Verarbeitungsbereiche zu ermöglichen, wurde innerhalb des TpS eine lose Kopplung angestrebt. Die Kommunikation zwischen ihnen wird über Warteschlangen WSn mit FIFO-Charakteristik abgewickelt (first in - first out, d.h. die zeitliche Reihenfolge von Meldungen wird durch die WS nicht beeinflußt). Es werden keine Daten, sondern nur kurze Kommandos übergeben, die im Zielverarbeitungsbereich die entsprechenden Vorgänge anstoßen. Unterstützt wird die Verwaltung der FIFOs durch den oben erwähnten Koppelbaustein PCD. Ein Teil der Kommunikation zwischen Sende- und Empfangsbereich, der speziell die Aktualisierung von Verbindungsinformation betrifft, wird durch einen gemeinsamen Speicher verwirklicht.

5.2.2.4 Interaktion mit dem Anwendungsrechner

Die Gestaltung der Schnittstelle zum AR hat wesentlichen Einfluß auf die Leistungsfähigkeit. Sowohl die Art als auch der Umfang der Kommunikation zwischen TpS und AR können falsch gewählt werden. Mittlerweile ist auch eine Verschiebung des internationalen Forschungsinteresses von den protokollbezogenen Ansätzen hin zu diesem Problembereich festzustellen. Schon 1988 legten Cheriton und Kanakia [202, 203] bei der Entwicklung ihres Kommunikationsadapters einen Schwerpunkt auf die Schnittstellenrealisierung. Auch bei den weltweiten Überlegungen zu ATM wird die Problematik der Kopplung zwischen AR und Netzwerk immer deutlicher wahrgenommen. Davie und Brendan stellen in [124, 79] entsprechende Arbeiten vor. Verschiedene Arbeitstreffen und Konferenzen mit entsprechendem Themenschwerpunkt sind angekündigt, für 1993 sind gleich zwei Ausgaben des IEEE Journal on Selected Areas on Communications zu diesem Thema in Vorbereitung. Zwei Arten sind grundsätzlich unterscheidbar:

Bei einer engen Kopplung macht das TpS den AR durch eine Meldung auf bestimmte Ereignisse (Ankunft von SDUs, Bedarf an Speicher im Bereich des ARs, Ereignisse des Verbindungsmanagements

usw.) aufmerksam und wartet anschließend (untätig) darauf, bis der AR die Meldungen übernimmt. Erst danach fährt das TpS mit der Verarbeitung fort.
Besäße der AR Reaktionszeiten in der Größenordnung der Verarbeitungszeiten des TpS, wäre diese enge Kopplung vertretbar. Dies ist in heutigen Rechnersystemen häufig nicht der Fall. Zeiten bis in den Millisekundenbereich für einen Kontextwechsel sind auch in modernen Rechnern typisch. Selbst die Reaktion auf eine Unterbrechungsanforderung ohne eigentlichen Kontextwechsel benötigt nicht zu vernachlässigende Zeiten bis weit in den Mikrosekundenbereich hinein. Da der Verarbeitungsaufwand pro PDU im implementierten TpS in der gleichen Größenordnung liegt, könnte während dieser Zeit die Verarbeitung eines wesentlichen Teils einer TPDU abgewickelt werden. Dies erfolgt nicht, so daß die Durchsatzrate sinkt. Würde das TpS für die Dauer eines kompletten Kontextwechsels blockiert, wäre die Verschlechterung noch dramatischer. Die enge Kopplung ist daher in solchen Systemen ungünstig.

Die lose Kopplung sieht WSn zwischen AR und TpS vor. Bei einem Kontextwechsel im AR können damit mehrere Meldungen auf einmal verarbeitet bzw. anderen Prozessen zugeordnet werden. Deren Aufruf kann zwar immer noch eine Vielzahl von Kontextwechseln verursachen, das TpS ist jedoch davon entkoppelt und kann weiterarbeiten.

Der Umfang der Kommunikation bezieht sich auf die Zahl der ausgetauschten Meldungen zwischen AR und TpS. Die Meldungen sind selbst dann notwendig, wenn die eigentlichen Daten durch intelligente Transfereinheiten mittels erweiterter DMA-Fähigkeit schon zu ihrem Bestimmungsort transportiert worden sind. In heutigen Systemen ist es üblich, pro zu übertragender Dateneinheit mehrere Meldungen auszutauschen. Eine Reduzierung auf eine Meldung pro auszutauschender Einheit ist der offensichtliche erste Optimierungsansatz, eine weitere Reduzierung kann durch Sammeln und Zusammenfassen von Meldungen erreicht werden. Beide Ansätze werden nachfolgend für die Empfangsrichtung, also beim Übergang vom TpS auf den AR, diskutiert:
Eine Meldung pro Übertragungseinheit limitiert unter der Annahme eines mit 0.5 ms angenommenen AR-Kontextwechsel pro empfangener PDU die Durchsatzraten auf 2000 PDUs pro Sekunde. Da der AR nicht nur diesen Kontextwechsel durchführt, liegt die praktische Grenze noch niedriger. Der Ansatz einer Meldung pro empfangener Übertragungseinheit ist daher zu vermeiden.
In dialogorientierten (Request / Response) Anwendungen ist dies häufig nicht möglich, da hier jede einzelne Nachricht eine Reaktion auslösen kann, von der wiederum das nachfolgende Verhalten des Senders abhängt. Ein echter Dialog macht daher grundsätzlich häufige Kontextwechsel nötig. Die Optimierung dieser Art der Kommunikation erfordert deshalb zusätzlich eine Betriebssystemoptimierung zur Senkung der Zeiten für Kontextwechsel [246, 322].

Das Zusammenfassen aufeinanderfolgender Daten-PDUs in Empfangsrichtung ist möglich, wenn nicht besondere Anforderungen an die Verzögerung gestellt wurden. Deren gesammelte Übergabe an den AR erfordert weit weniger Meldungen. Voraussetzung in OSI/TP4 ist eine angepaßte Verwendung der TSDU-Endekennung EOT (End of Text) durch den Sender. Wird diese zu häufig verwendet, steigt die (durch das Protokoll vorgeschriebene) Anzahl der Meldungen an den AR entsprechend an.
Anmerkung: Bedingt u.a. durch die intern vorhandene Strukturierung der Speicherverwaltung, wird in heutigen realen Systemen eine große Datei in der Regel nicht als eine TSDU an das TpS übergeben, sondern in vielen einzelnen, voneinander unabhängigen, Blöcken mit der Länge einer Speicherseite. Die Sammlung von Empfangsmeldungen müßte daher über den engen Rahmen von Schicht 4 TSDUs einer Verbindung hinaus verallgemeinert werden. Entweder ist das Zusammenfassen mehrerer aufeinanderfolgender TSDUs einer Verbindung zuzulassen oder es wird ein Sammeln über alle Verbindungen hinweg erlaubt. Die erstgenannte Variante erfordert ein Wissen über die Anwendung; beiden gemeinsam ist die Notwendigkeit entsprechender Vereinbarungen über die Dienstgüte.

Aus diesen Betrachtungen wurden folgende Entscheidungen abgeleitet:

A) In Empfangsrichtung hat das TpS zu jeder Verbindung (DMA-)Zugriff auf einen Speicherbereich innerhalb des ARs. Anfang und Größe werden dem TpS vom AR über eine Meldung in einer TpS-WS mitgeteilt. Das TpS liefert standardkonform immer dann eine Meldung in eine weitere dem AR zugängliche WS, falls eine EOT-Kennung in einer PDU empfangen wurde oder der reservierte Empfangsspeicherbereich aufgebraucht ist.
Es wird nicht zwangsläufig durch jede Meldung ein Kontextwechsel ausgelöst. Ein Zusammenfassen von Meldungen gemäß beiden angeführten Varianten wird dadurch ermöglicht. Die Reaktionszeit hängt einzig vom Abfrageverhalten des ARs ab. Die problemlos machbare Einrichtung mehrerer WSn und einem zugehörigem variierenden Abfrageverhalten ermöglicht eine Unterstützung unterschiedlicher Dienstgüteforderungen.
Ist der zugeteilte Empfangsspeicher aufgebraucht, teilt das TpS dies dem AR über eine weitere WS mit und fordert weiteren Speicher an. Das TpS blockiert das Übertragen der Empfangsdaten der betreffenden Verbindung, bis eine Meldung über einen neuen Empfangsbereich eintrifft. Während dieser Blockierzeit eintreffende PDUs der Verbindung verbleiben zunächst im Empfangsdatenspeicher des TpS. Dies ist bei dessen Dimensionierung zu beachten.
Dieser Mechanismus stellt sicher, daß:
    - die EOT-Kennung von OSI/TP4 richtig behandelt wird
    - empfangene Daten einer Verbindung zusammenhängend übergeben werden
    - das TpS (begrenzt) PDUs empfangen kann, obwohl der AR keinen ausreichenden Empfangsbereich reserviert hat
    - Verbindungen sich nicht untereinander blockieren
    - der AR die Häufigkeit der Meldungen indirekt über die Größe der bereitgestellten Empfangsspeicherbereiche einstellen kann.

B) In Senderichtung ist die Vorgehensweise ähnlich: Der AR legt Kommandos in einer dem TpS zugänglichen WS ab. Dieses Kommando enthält neben der Verbindungskennung Anfangsadresse und Länge eines Datenbereiches, der gesendet werden soll. Ist der Datenbereich nicht zusammenhängend im Speicher abgelegt - was erfahrungsgemäß dem Normalfall entspricht - kann eine entsprechende Liste von Anfangsadressen und Längen übergeben werden. Werden ausschließlich Datenblöcke gleicher Länge benutzt, kann die Angabe der Länge entfallen. Allerdings muß dann für jeden Block - auch für direkt aufeinanderfolgende - ein Listeneintrag vorhanden sein.
Sind alle Daten in den Sendedatenspeicher des TpS kopiert worden, ergeht eine Meldung (über eine WS) an den AR zur Freigabe des Sendebereichs. Um das Senden innerhalb der Verbindung auch während dieser Austauschzeit zu ermöglichen, kann der AR dem TpS mehr als einen Sendebereich übergeben. Diese werden den Verbindungen zugeordnet und verwaltet. Beim (DMA-unterstützten) Einlesen der Daten muß beachtet werden, daß einzelne Verbindungen nicht bevorzugt werden. Hierzu dient Mechanismus, der das Einlesen für jede Verbindung auf eine bestimmte Zahl von PDUs beschränkt. Anschließend erhalten andere Verbindungen das Recht zum Einlesen.
Diese Lösung stellt sicher, daß:
    - die EOT-Kennung von OSI/TP4 richtig behandelt wird
    - sowohl AR als auch TpS kontinuierlich arbeiten können
    - eine gerechte Verteilung der Einlesebandbreite gegeben ist
    - die Sendebereichsgröße über die Zahl der Meldungen eingestellt werden kann.

Die Häufigkeit der Abfragen der WSn beeinflußt nicht nur deren Dimensionierung und die erzielbaren Dienstgüten, sondern - durch dazu im AR erforderliche Kontextwechsel und Bearbeitungszeit - auch dessen Effizienz. Eine enge Kopplung mit dem Betriebssystemmechanismus zur Aktivierung und Deaktivierung von Prozessen des ARs (scheduling) ist erforderlich. Abfragen innerhalb der Anwendungs-

programme sind unbedingt so auszulegen, daß eine begrenzte Zahl von erfolglosen Versuchen die Deaktivierung initiiert. Die erneute Aktivierung kann durch Erreichen eines vorgegebenen Füllstandes oder einer maximalen Zeit ausgelöst werden [200].

Alle oben beschriebenen Aufgaben beziehen sich auf die Datenteile von empfangenen bzw. zu sendenden PDUs. Die Verarbeitungseinheit VE, die die Aufgaben ausführt, muß daher Zugang zu den Datenspeichern haben. Insgesamt hat diese VE damit folgendes zu erledigen:

- Verwaltung der Empfangs- und Sendebereiche beim AR
- Steuerung des Kopierens aus den Sendebereichen des ARs in den Sendedatenspeicher des TpS und analog der umgekehrt gerichteten Übertragung. Steuerung ist hier gleichbedeutend mit Initialisieren der entsprechenden DMA-Einheiten
- Steuerung der Datenspeicherverwaltung
- Verwaltung der WSn zwischen AR und TpS.

Als zusätzliche Aufgabe kommt die Steuerung der DMAMacSend hinzu, die das Zusammenführen von Steuerkopf und Datenteil und deren Übergabe an die Medienzugangseinheit abwickelt. Der Umfang dieser Aufgaben legte die Einführung eines dritten Verarbeitungsbereichs mit eigener CPU nahe. Dieser wird als Verwaltungs- und Schnittstellenbereich (im folgenden mit IF abgekürzt, für Interface) bezeichnet. Die Implementierung der Protokollsoftware zeigte, daß die Reduzierung der Belastung von Sende- und Empfangsbereich erheblich und damit gerechtfertigt war.

5.2.2.5 Aufwendige Systemfunktionen

Die zyklische Verwaltung der Datenspeicher wurde bereits als unwirtschaftlich begründet. Eine Verwaltung von Speichersegmenten variabler Länge erfordert Strategien zum Auffinden und Nutzen von Speicherverschnitt (garbage collection) und nutzt den Speicher unzureichend aus. Dieser Nachteil wird durch eine Aufteilung des Speichers in Elementarpuffer fester Größe vermieden. Auch deren Verwaltung ist noch aufwendig genug, um eine Hardwareunterstützung zu rechtfertigen. Werden ausschließlich Daten-PDUs mit Längen von ganzzahligen Vielfachen der EP-Größe verwendet, so kann einerseits der Speicher einen höheren effektiven Füllgrad erreichen, andererseits vereinfachen sich die Mechanismen zur eindeutigen Feststellung des Datenendes sowohl im TpS als auch im AR.

Das OSI/TP4-Protokoll macht von einer Vielzahl von Zeitgebern Gebrauch; deren Realisierung und Verwaltung ist - wie z.B. in den Kapiteln 3.2.6 und 3.2.8 gezeigt - sehr aufwendig. Daher wurde für sie ebenfalls eine Hardwareunterstützung konzipiert.

Der bereits erwähnte Koppelbaustein (PCD) isoliert die Verwaltung der WSn und bietet Mechanismen, die eine Reduzierung des Synchronisationsaufwandes ermöglicht.

5.2.3 Gesamtarchitektur

Die angestellten Überlegungen führten zu der in Bild 5.3 abgebildeten TpS-Gesamtarchitektur.
Im folgenden werden die Ergebnisse der Entscheidungen kurz zusammengefaßt:

Drei Verarbeitungsbereiche mit jeweils eigenem RISC-Prozessor und lokalem Speicher für Programme und lokale Datenbereiche wurden vorgesehen:

A) Der Schnittstellen- und Verwaltungsbereich (IF-Bereich) wickelt die Kommunikation mit dem AR und der Medienzugangseinheit (MAC) ab. Zum Ein- und Auslesen von Daten von bzw. an den AR werden vom IF-Prozessor initialisierte intelligente Transfermaschinen mit erweiterter DMA-Fähigkeit (DMAHostSend/Rcv) verwendet. Entsprechend werden DMAMacSend und DMAMacRcv zum Übertragen von Daten an bzw. von der MAC eingesetzt. Die DMA-Einheiten greifen über eine hardwareunterstützte Speicherverwaltung (je eine für Sende- und Empfangsrichtung) auf

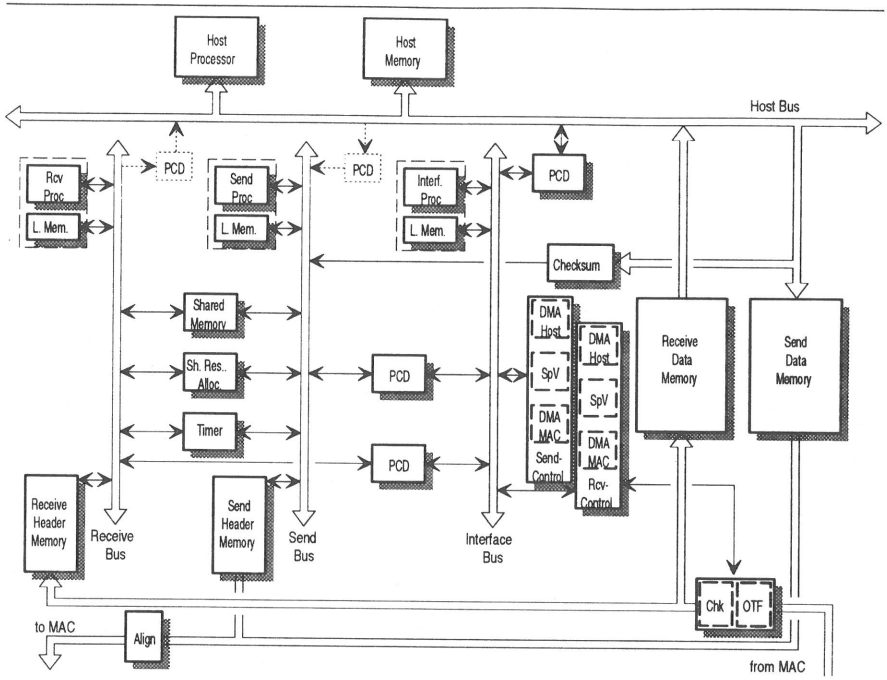


Bild 5.3: *Strukturdiagramm des Transportsystems (TpS)*

die Datenteile zu sendender bzw. empfangener Daten-PDUs (im Sende- bzw. Empfangsdatenspeicher) zu. Darüber hinaus haben die zwei DMA-Einheiten in Richtung AR Lese- und Schreibzugriff auf den Speicher des AR. Die DMAs zur MAC greifen ihrerseits zusätzlich auf die beiden Steuerkopfspeicher (RCV- bzw. Send-Header-Memory) zu, um zu sendende PDUs zusammenzustellen bzw. empfangene PDUs aufgeteilt nach Steuerkopf- und Datenteil getrennt abzuspeichern. Die Sende-DMAs und die Speicherverwaltung für die Senderichtung sind zur Sendesteuereinheit zusammengefaßt. Analog ist die Zusammenfassung zur Empfangssteuereinheit.

Zur Berechnung der Prüfsumme in Senderichtung wird ein Hardwarebaustein (checksummer), gesteuert von der DMAHostSend, verwendet. Das Ergebnis der Prüfung wird dem Sendeprozessor signalisiert. Der Kommandoaustausch zwischen dem IF-Prozessor und dem AR wird über Prozeßkoppelbausteine gesteuerte (FIFO-)WSn ausgeführt. Das gleiche gilt für den Kommandoaustausch mit den anderen 2 Verarbeitungsbereichen.

B) Der Sendebereich ist zuständig für die Zusammenstellung der Schicht 4, 3 und 2b Steuerköpfe zu sendender TPDUs und die Verwirklichung einzelner Protokollmechanismen, die mit dem Senden von TPDUs zu tun haben. Dazu hat der Sendeprozessor Zugriff auf den Sendesteuerkopfspeicher, in dem die fertig erstellten Steuerköpfe ablegt werden. Außerdem hat der Prozessor Zugang zur Verwaltung einer Zeitgebereinheit.

Der Kommandoaustausch mit den anderen 2 Bereichen erfolgt über - durch PCDs gesteuerte - WSn. Gemeinsam benötigte Daten - hauptsächlich Verbindungsinformation - verwalten Sende- und Empfangsbereich in einem gemeinsamen Speicher. Ein zweiter, wiederum durch einen PCD verwalteter, gemeinsamer Speicher ermöglicht dem AR eine direkte Kommunikation mit dem Sendebereich. Dieser Zugang ist zur Übergabe von Parametern während Verbindungsauf- bzw. abbau-

ten vorgesehen. Für die normale Datentransferphase ist er dagegen unerheblich.

C) Der Empfangsbereich wickelt die Empfangsmechanismen der Protokolle ab, löst das Senden von Kontroll-TPDUs aus und überwacht den Zustand der Verbindungen. Hierzu wertet der Empfangsprozessor die im Empfangsteil des Steuerkopfspeichers abgelegten Steuerköpfe sowie die zugehörige Statusinformation aus. Letztere wird von der DMAMacRcv mit Hilfe des OTF-Bausteins und des Bausteins zur Validierung der Empfangsprüfsumme zu jeder empfangenen PDU zusammengestellt und im Steuerkopfspeicher abgelegt.
Der Empfangsprozessor hat ebenfalls Zugang zur Verwaltung der Zeitgebereinheit.
Der Kommandoaustausch mit den anderen 2 Bereichen erfolgt wieder über PCDs gesteuerte WSn.
Analog zum Sendebereich gibt es auch im Empfangsbereich einen gemeinsamen Speicher zum AR, welcher zur Übergabe von Parametern während Verbindungsauf- und abbauten vorgesehen ist und während der normalen Datentransferphase nicht benötigt wird.

5.3 Die Hardwarekomponenten

Die bisherigen Überlegungen zu möglichen Optimierungen des TpS ließen bei folgenden Funktionen eine Hardwareunterstützung sinnvoll erscheinen:

1. Bewegung von Daten (Steuerung der Bus- und Speicherzugriffe)
2. Verwaltung der verschiedenen Speicher
3. Kopplung von Prozessen (Verwaltung und Ansteuerung von WSn)
4. Zeitgeber
5. Prüfsummenerstellung und -verifikation
6. Trennung von Steuerköpfen und Daten

Die Überlegungen zur Umsetzung der einzelnen Funktionalitäten in Hardware und deren jeweils wesentlichen Eigenschaften bilden den Kern dieses Kapitels.

5.3.1 Trennung von Steuerkopf und Daten

Eine wesentliche Eigenschaft des TpS ist die Trennung der Benutzerdaten, den Teilstücken von TSDUs, und aller Steuerköpfe der im TpS verarbeiteten Schichten 4, 3 und 2b.

In Senderichtung erfordert die getrennte Behandlung von Datenteilen und Steuerköpfen "lediglich" eine Synchronisation und Kommunikation zwischen IF- und Sendebereich. Zwischen TpS und MAC muß auch das lückenlose Zusammenführen der beiden Teile eines MAC-Rahmens sichergestellt werden.

In Empfangsrichtung ist die Trennung in Echtzeit, d.h. schritthaltend mit der Geschwindigkeit des ankommenden Oktettstroms, durchzuführen. Besäße jeder Steuerkopf der drei relevanten Schichten eine konstante Länge, würde sich die Funktionalität des Trennungsbausteins auf eine reine Zählfunktion reduzieren. Bedingt durch den allgemeinen Charakter der ISO/OSI Standards trifft diese idealisierte Annahme nur bei der im TpS vorhandenen Typklasse 1 des LLC-Protokolls zu, welche auf 3 Oktetts festgelegt ist. Die Standards der beiden verbleibenden Teilschichten CLNP und TP4 umfassen dagegen stark variierende Steuerkopflängen [366, 358]. Gründe hierfür sind:

- Unterschiedliche Arten von PDUs
- Mehrere zugelassene Formate, z.B. bei Krediten und Folgenumerierung
- Wahlfrei kombinier- und positionierbare Optionen

Als Folge der nicht deterministischen PDU-Länge enthalten die Steuerköpfe sowohl der Vermittlungs- als auch der Transportschicht Angaben über die Gesamtlänge des Steuerkopfes. Zur Feststellung des Datenbeginns brauchen die Oktetts nur sukzessive abgezählt werden. Die Werte der Zählungen sind aus dem Oktettstrom zu extrahieren und es ist eine Korrektur vorzunehmen, da die Längenangabe sich auf den

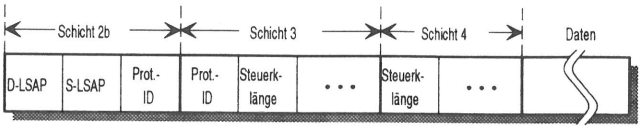


Bild 5.4: *Aufbau eines MAC-Rahmens*

gesamten Steuerkopf (inklusive Längenangabe sowie eventuell vorangegangener Oktetts) bezieht.

Wie Bild 5.4 zu entnehmen ist, stehen die Längen nicht an den gleichen Positionen innerhalb des Steuerkopfes - im Schicht 4 Steuerkopf steht die Angabe an erster Position, bei der Schicht 3 an zweiter. Die vorzunehmende Korrektur des Wertes muß daher kontextabhängig durchgeführt werden.
Es ergeben sich folgende funktionelle Blöcke für die Einheit zum Trennen von Steuerköpfen und Daten:

- Ladbare 8 Bit breite Zähler
- Vergleicher und Auswertelogik
- Korrekturlogik und
- Globaler Zustandsautomat

Zwei wichtige, über die beschriebene Basisfunktionalität hinausgehende Aufgaben, bewirken, daß die tatsächlich entworfene Einheit deutlich komplexer ausfiel, als diese Aufzählung zunächst vermittelt. Diese Aufgaben sind eine Konsistenzprüfung mit zugehöriger Umleitung aller empfangener Oktetts in den Empfangsdatenspeicher (transparenter TpS By-pass Modus) und die Umwandlung von 8 auf 32 Bit. Zur Ermöglichung der TP4-Prüfsummenberechnung muß außerdem neben dem Beginn der eigentlichen Daten auch der Beginn des Schicht 4 Steuerkopfes festgestellt und explizit signalisiert werden.

Zur Prüfung der Verträglichkeit mit dem TpS werden die Protokollkennungen der Schichten 2b und 3 herangezogen, die sich entsprechend Bild 5.4 im 3. und 4. Oktett eines empfangenen MAC-Rahmens befinden. In der realisierten Version ist ein 16 Bit breiter Vergleich mit beliebig extern maskierbaren Sollwerten vorgesehen. Die Position der Werte macht es möglich, daß man ohne aufwendige Zwischenspeicherung auskommt und statt dessen das Ergebnis der Verträglichkeitsprüfung direkt benutzen kann, das erste Langwort (32 Bit, 4 Oktetts) in den gewünschten Speicher zu übertragen.

Zur Wandlung von 8 auf 32 Bit kommen zyklisch adressierte Register zum Einsatz. Das Ziel der Wandlung ist die Viertelung der Geschwindigkeitsanforderung. Mit nur einem Registersatz verletzten Steuerköpfe mit Längen, die kein ganzzahliges Vielfaches von 4 Oktetts sind, dieses Ziel. Deren Berücksichtigung erfordert die Dopplung der Register (Wechselpufferprinzip) und eine zugehörige Steuerung.

Da analoge Probleme bei der Zusammenführung von Steuerköpfen und Daten in Senderichtung auftreten, und der Einsatz des TpS nur in modernen Umgebungen mit internen Verarbeitungsbreiten von (mindestens) 32 Bit überhaupt sinnvoll ist, wurde von einer generellen Ausrichtung der Steuerkopflängen auf Vielfache von 4 Oktetts ausgegangen! Für die Auffüllung eventuell fehlender Oktetts wird gezielt die NOP (no operation) Option der Schicht 3 eingesetzt.

Es sei angemerkt, daß die entworfene Hardware keinerlei Einschränkungen bezüglich Optionen und Steuerkopfformaten vorschreibt. Diese sind durch die damit verbundene stark ansteigende Komplexität der eigentlichen Protokollverarbeitung begründet. Da die Baugruppe in ihrer derzeitigen Auslegung lediglich Mechanismen zur Feststellung des Beginns von Datenteilen enthält, resultiert allerdings die schon in Abschnitt 5.2.1.3 beschriebene Forderung nach maximal einem Steuerkopf jeder Schicht pro MAC-Rahmen und der resultierende Verzicht der verschiedenen Verkettungsmechanismen von PDUs und SDUs.

Wäre das Lokalisieren des Datenendes noch relativ einfach durch weitere Zähler und Korrekturlogik machbar, ergeben sich aus der unmittelbaren Folge der verschiedenen PDUs erhebliche zeitliche und or-

ganisatorische Schwierigkeiten. Aufeinanderfolgende MAC-Rahmen weisen dagegen durch unumgängliche Maßnahmen zur Rahmensynchronisation - und häufig vorhandene Vereinbarungen zum minimalen Abstand von MAC-Rahmen - immer einen Abstand von zumindest einigen Oktetts auf.

5.3.2 Prozeßkopplung

Zu Beginn der Entwicklung des Bausteins zur Interaktion von Prozessen auf einem oder mehreren Prozessoren standen noch verschiedene Formen der Verteilung paralleler Funktionen zur Debatte. Als Konsequenz daraus und mit der zusätzlichen Intention eines möglichst universell einsetzbaren Bausteins wurde ein generisches Konzept entwickelt. Obwohl im speziellen Umfeld des TpS vereinfachte Randbedingungen gelten, werden sowohl die grundsätzlichen Anforderungen an den Baustein als auch dessen - teilweise über die im TpS tatsächlich eingesetzten - Eigenschaften vorgestellt.

Wie bereits erwähnt, wurde ursprünglich eine Parallelisierung von OSI/TP4-Funktionen angestrebt. Dieser Ansatz bedeutet, daß die Verarbeitung einer PDU (im Sende- bzw. Empfangsbereich) in einzelne Teilfunktionen zerfällt. Die Teilfunktionen sind zur Parallelisierung als Prozesse einzelnen Prozessoren zuzuordnen. Diese kommunizieren untereinander über einen gemeinsamen Speicher (aus Kostengründen oder weil, wie beim Kommunikationsspeicher, nicht anders machbar). Die Beziehung zwischen den zu einem bestimmten Zeitpunkt ablaufenden Teilfunktionen ist je nach Implementierung wie folgt:

- Eine Fließbandverarbeitungsstruktur bedeutet, daß die zeitgleich (parallel) ablaufenden Teilfunktionen verschiedene PDUs betreffen. Die Kommunikation zwischen den Teilfunktionen findet über (FIFO-) WSn im gemeinsam zugänglichen physikalischen Speicher statt. Die Synchronisation ist implizit durch diese verwirklicht (siehe Bild 5.5).
- Eine geeignete Wahl voneinander weitgehend unabhängiger Teilfunktionen macht für die Weiterverarbeitung einer PDU nur die Kenntnis des Abschlusses aller Teilfunktionen und die Kenntnis über deren Erfolg oder Mißerfolg im Sinne der Anwendung erforderlich (dieser Ansatz wird z.B. von Zitterbart und Rupprecht [264, 330] beschrieben).

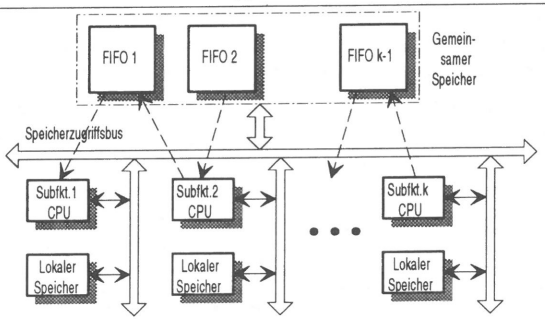


Bild 5.5: *Fließbandverarbeitung der Teilfunktionen*

In diesem Fall können die zeitgleich ablaufenden Teilfunktionen die gleiche PDU betreffen. Jede von ihnen stellt Informationen in einer gemeinsam benutzten Ausgangs-WS bereit. Zugleich entnehmen sie Informationen, die zur Durchführung der Teilfunktionen nötig sind, aus einer oder mehreren gemeinsam benutzten Eingangs-WS. Diese zeitlich horizontale Anordnung der Verarbeitung vermeidet die implizite Verzögerung einer Fließbandverarbeitung. Die Synchronisation muß im Gegenzug dafür sorgen, daß in der Ausgangs-WS der Abschluß aller Teilfunktionen und deren Ergebnis, die gegenwärtige PDU betreffend, angezeigt wird. Entsprechendes ist in der Eingangs-WS zur Freigabe des Speichers, der die Eingangsinformation enthält, nötig. Das Prinzip ist in Bild 5.6 abgebildet.

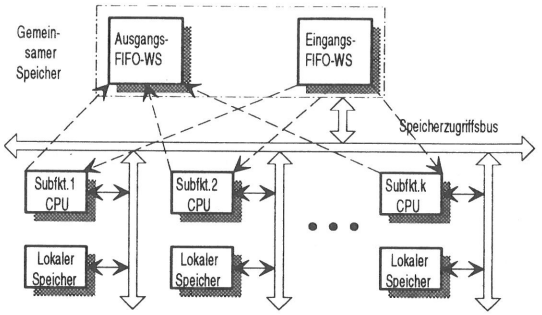


Bild 5.6: *Horizontale Verarbeitung der Teilfunktionen*

Beide Mechanismen sind in Software implementierbar. Allerdings erfordert diese Lösung ein wiederholtes Abfragen der einzelnen WSn. Da sich diese in einem gemeinsamen Speicher befinden, kann allein durch den zyklischen Abfragemechanismus eine erhebliche Belastung des Zugangsbussystems zum Speicher entstehen, insbesondere falls eine Vielzahl von angeschlossenen Prozessoren eine Vielzahl von WSn abfragen muß. Im Fall einer horizontalen Verarbeitungsstruktur kommt noch hinzu, daß das Abfragen des Weiterverarbeitungsprozessors den Abschluß aller Teilfunktionen überprüfen muß.

Unabhängig von der gewählten Form von Parallelität ergibt sich der Bedarf für eine zentralisierte hardwareunterstützte WS-Verwaltung mit einer gewissen Kommunikationsfähigkeit.

In den Bildern 5.5 und 5.6 ist jeweils nur *ein* Zugang zum gemeinsamen Speicher und den darin enthaltenen logischen WSn vorhanden. Eine erste Erweiterung ist die Auslegung auf zwei voneinander völlig unabhängige asynchrone Bussysteme. Setzt man zunächst einen unidirektional gerichteten Informationsfluß voraus, ergibt sich das funktionelle Blockbild 5.7 für die Steuerung einer (FIFO-)WS.

Die WS wird physikalisch in einem herkömmlichen Speicher mit wahlfreiem Zugriff (RAM, Random Access Memory) realisiert. SA steht für Schreib-, LA für Leseautomat. SZ und LZ sind die Zähler für die Schreib- und Leseadresse, V und L speichern den internen Zustand der WSn.

Der Zugriff auf den Speicher kann auf zwei Arten vorgenommen werden: entweder können Sender bzw.

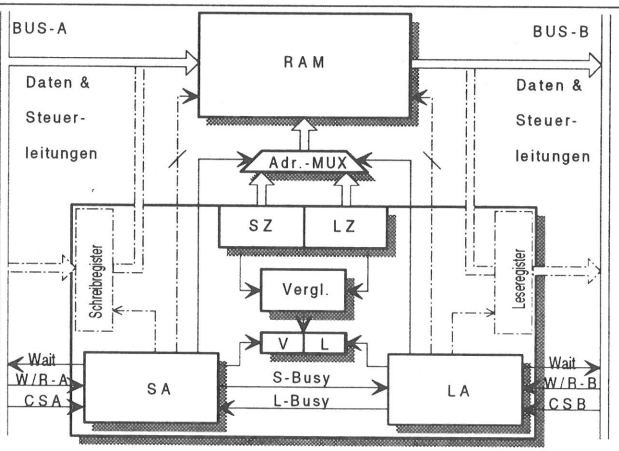


Bild 5.7: *Blockbild der Steuerung einer Warteschlange*

Empfänger direkt auf den Speicher zugreifen oder sie sehen jeweils nur ein Register (im Bild 5.7 strichliert dargestellt). Im ersten Fall übernimmt der Sender oder der Empfänger die Steuerung des Speichers und die Einhaltung des korrekten Zeitverhaltens. Der PCD-Baustein stellt, durch entsprechende Signale gesteuert, die synchronisierte Adressierung des richtigen Elements und - wahlweise - auch innerhalb eines Elements sicher. Sind in dem Speicher mehrere WSn zu verwalten, muß die Basisadressierung der Gewünschten vom Sender/Empfänger vorgenommen werden.
Im zweiten Fall besteht keinerlei direkte Verbindung zwischen Speicher und Anwendern. Der PCD übernimmt die komplette Steuerung und Adressierung des Speichers, die Daten werden über Register in bzw. aus dem PCD transferiert. Bei längeren Nachrichten führt die Entkopplung durch die Register durch die notwendigen Maßnahmen zur Synchronisierung zu erhöhtem Verwaltungs- und Zeitaufwand.

Da der Speicher physikalisch nur über einen Zugang verfügt, muß der PCD auch für die eindeutige Klärung eventuell auftretender Zugriffskonflikte ausgelegt sein. Eine Schreibanforderung ist dabei höher priorisiert als eine zum Lesen, laufende Aktionen können allerdings nicht unterbrochen werden. Ein Lese-(Schreib-)Zugriff wird zusätzlich nur dann zugelassen, wenn die WS nicht leer (voll) ist.

Die Benachrichtigung des Empfängers wird mit Unterbrechungsanforderung durchgeführt. Bei regelmässigem Informationsfluß und zyklischen Abläufen ist u.U. ein Abfragen vom Empfänger aus sinnvoll.

Die Bereitstellung eines solchen Bausteins mit Speicher pro benötigter WS wäre zwar aus funktioneller Sicht möglich, würde aber schnell zu einer unwirtschaftlich hohen Zahl von Bauelementen führen. Die Komplexität der obigen Schaltung - ausgedrückt in benötigten Gatteräquivalenten - ist zudem so gering, daß eine Umsetzung in ein ASIC nicht sinnvoll erscheint. Überlegungen zur Verallgemeinerung und Erweiterung der Funktionalität ergaben schließlich die folgenden Anforderungen an den Baustein:

- Effiziente Verwaltung mehrerer unabhängiger WSn
- Sicherstellen des geregelten bidirektionalen Zugriffs von zwei Seiten her (Semaphor- und Prioritätslogik)
- Mehrfaches Zugreifen auf einen Eintrag muß grundsätzlich möglich sein
- Anzahl der Elemente einer WS und Elementgröße sollen variabel sein
- Neben der reihenfolgebewahrenden Abarbeitungsstrategie (FIFO) sollte auch ein Stapelmodus (LIFO, Last In First Out) wählbar sein.

Es ergab sich die in Bild 5.8 gezeigte Struktur. Um verschiedenen VEs den Zugriff auf gemeinsam benutzte WSn zu ermöglichen, wurde das PCD als Buskoppelelement ausgelegt. Damit ist eine beliebige Verteilung der beteiligten Prozesse auf die verbundenen Verarbeitungsbereiche möglich.

Die Grundaufgabe des PCDs ist die Verwaltung von (FIFO-)WSn. Diese befinden sich physikalisch in einem regulären RAM, das vom PCDs kontrolliert wird. Die einzelnen WSn bestehen jeweils aus Elementen, deren Breite - d.h. Anzahl der Speicherwörter - definierbar ist. Gleiches gilt für die Längen der einzelnen WSn, also die Anzahl von Elementen einer WS. Der Zustand jeder WS ist zunächst durch zwei Zeiger, je einen für Lesen bzw. Schreiben, sowie Statusbits für Belegung und Füllstand gekennzeichnet. Die Statusbits aller WSn sind in einem Statusregister zusammengefaßt.

Die Zeiger legen für jede WS die gegenwärtig zugänglichen Elemente fest. Lese- und Schreibzugriffe auf das PCD beziehen sich auf diese Elemente. Die Auswahl eines Wortes innerhalb eines Elements wird dagegen direkt aus der angelegten Adresse abgeleitet.

Der Baustein zeigt jedes Weiterschalten eines Lese- oder Schreibzeigers mit der WS-Nummer und der Art der Veränderung über Signalleitungen der Umwelt an. Diese (repräsentiert durch auf Prozessoren ablaufende Prozesse) erhält die Signale in Form von direkten Unterbrechungsanforderungen oder gepuffert in einem lokalen Register, auf das durch lokale Abfragen zugegriffen wird. Jeder Prozeß weiß damit, daß ein neues Element zum Schreiben bzw. Lesen vorhanden ist, sofern die entsprechende WS nicht voll bzw.

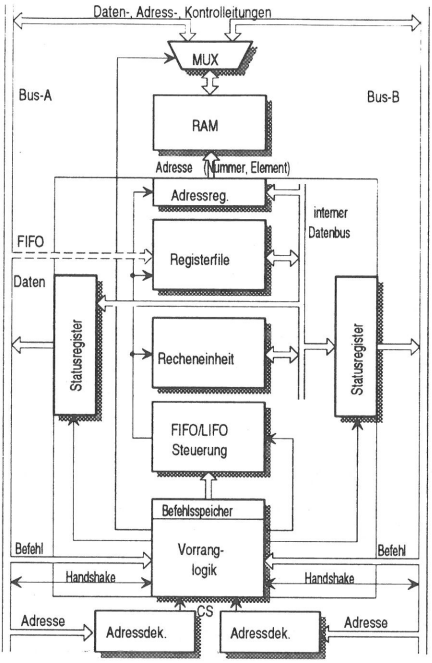


Bild 5.8: *Funktionsblöcke des Prozeßkoppelbausteins PCD*

nicht leer ist. Die (einmalige) Anzeige jedes Weiterschaltens ist es, die einen Teil des oben erwähnten wiederkehrenden Abfragemechanismus überflüssig macht. Der Rest wird durch die Abfrage des Status aller WSn in einem einzigen Zugriff auf das Statusregister vermieden.

Das Weiterschalten eines bestimmten WS-Zeigers wird durch die Softwareprozesse ausgelöst, die diesen benutzen. Jede WS ist für jede Zugriffsart (Lesen, Schreiben) einer festen Menge von Prozessen zugeordnet. Nach Abschluß des Zugriffs auf die WS teilt jeder Prozeß dem PCD mit, daß er mit der Verarbeitung des gegenwärtig zugänglichen WS-Elements fertig ist. Das PCD unterhält hierzu zu jeder WS und Zugriffsart einen Zählwert, der bei einer Fortschaltung des entsprechenden Zeigers zunächst auf 0 gesetzt ist. Jede Fertigmeldung eines Prozesses erhöht diesen Zählwert um einen bestimmten (auf dem Datenbus bereitgestellten) Wert. Erreicht die neue Summe eine bestimmte Grenze (festverdrahtet, 2-er Potenz), so wird der entsprechende Zeiger weitergeschaltet, das dazugehörige Statusbit u.U. geändert und der Zählwert auf 0 gesetzt. Dieser Vorgang wird als Stempeln bezeichnet.

Beim Stempeln teilt jeder Prozeß mit, ob seine Verarbeitung erfolgreich war. Was "erfolgreich" bedeutet, ist im Einzelfall durch die Anwendung vorgegeben. Hierzu wird beim Stempeln eine Datenleitung verwendet. Ist sie auf 1 (Erfolg), so wird das Ergebnisbit der zugehörigen WS und Zugriffsart nicht verändert. Ist sie auf 0, so wird das Ergebnisbit auf 0 gesetzt. Bei Fortschaltung eines Zeigers wird das Ergebnisbit in das erste Wort des neuen Elements eingetragen. Damit haben die beteiligten Prozesse die Möglichkeit, festzustellen, ob alle von ihnen erfolgreich waren. Nach erfolgtem Eintrag wird das Ergebnisbit auf 1 gesetzt. Die Ergebnisbits sind in einem Statusregister zusammengefaßt.

Durch Berücksichtigung fachlicher und organisatorischer Randbedingungen des gesamten Projekts wurden bei der implementierten Version folgende Festlegungen getroffen:.

- Die Anzahl der WSn ist auf maximal 16 beschränkt.
- Durch externe Beschaltung kann sowohl die Breite eines WS-Elements als auch die Länge einer WS einheitlich für alle WSn eines PCDs in 2-er Potenzen gewählt werden.
  (Die Beschränkung auf 2-er Potenzen ermöglicht eine einfache und direkte Zuordnung einzelner Bits der Gesamtspeicheradresse ohne aufwendige arithmetische Umrechnungen.)
- Als Konsequenz der einheitlichen Längen und Breiten sowie der Programmierbarkeit durch externe Beschaltung entfällt darüber hinaus die Notwendigkeit für das Laden der verschiedenen WS-Parameter; folglich verringert sich die Größe des PCD-intern benötigten Speichers.
  Sollte sich ein Bedarf für WSn stark unterschiedlicher Breite und Länge ergeben, kann durch das Einführen von 8 weiteren Chipanschlüssen (Pins) das Programmieren von einem der beiden Busse aus ermöglicht werden (in Bild 5.8 von Bus A aus).
- Da im TpS nur ein einzelner Prozessor pro Anschlußbereich angeschlossen ist, besteht kein Bedarf für das Weiterleiten von Meldungen an mehrere Empfänger mit dem beschriebenen Stempeln.
  Ein eventuell notwendiges Verteilen einer Meldung an mehrere Prozesse eines Prozessors wird durch entsprechende Verteilerprozeduren innerhalb des Prozessors übernommen. Der PCD-Baustein braucht damit nur mit einem Partner zu kommunizieren.
- Treten Mehrfachzugriffe nicht bzw. nur selten auf, kann auch auf die Ausgabe der aktuellen Lese- und Schreibzeiger verzichtet werden, mit denen die Nutzer ansonsten eigene Konsistenzprüfungen durchführen können. Interner und externer Ablauf vereinfachen sich durch diese Maßnahme erheblich und es können weitere Pins eingespart werden.

Bisher war ausschließlich vom Einsatz des PCD für die asynchrone Kopplung von Prozessen die Rede. Der Baustein kann jedoch u.a. auch zur Ressourcenverwaltung eingesetzt werden. Im TpS sind etwa die Zeitgeber oder die Elementarpuffer Systemressourcen, die häufig benötigt werden.

Dazu müssen zunächst die Elemente einer WS mit Verweisen auf die physikalischen Ressourcen initialisiert werden. Wird dann der Stapelmodus gewählt, können Anforderungen von Ressourcen durch einfaches Lesen des Stapels befriedigt werden. Das Zurückgeben von nicht mehr benötigten Ressourcen läßt sich direkt auf einen Schreibvorgang abbilden. Eine Abbildung auf die durch zyklische Ringspeicherstrukturen nachgebildete FIFO-WSn ist mit wenig Aufwand grundsätzlich ebenfalls möglich.

5.3.3 Speicherverwaltung

Die globalen Anforderungen an die Einheit zur Speicherverwaltung waren:

- Weitgehende Entlastung des externen Prozessors (im TpS: der IF-Prozessor)
- Direkter und wahlfreier Zugriff auf einzelne Speicherworte
- Hohe Speicherauslastung, d.h. wenig Speicherverschnitt
- Paralleles Lesen und Schreiben mit Datenraten von jeweils über 100 Mbit/s
- Nicht ausschließlich für eine Umgebung ausgelegt (⇒ programmierbar), trotzdem möglichst "echte" festverdrahtete Hardware und kein Einsatz von Prozessoren
- Vermeidung des (teuren) physikalischen Bewegens von Daten
- Unterstützung von Segmentierung und Reassemblierung sowie Mechanismen zur Behandlung von reihenfolgegestörten PDUs

Bild 5.9 zeigt die Anordnung von Verwaltungsrechner (VWR), Speicher und -verwaltung (SpV).

Offensichtliches Merkmal ist, daß der VWR keinerlei Verbindung mit dem Speicher hat. Die Steuerung des Speichers, d.h. physikalische Adressierung und Einhaltung des vorgegebenen Zeitverhaltens, wird vollständig von der SpV durchgeführt. Der Speicher verfügt aus logischer Sicht über zwei voneinander unabhängige Datenpfade (im Bild 5.9 willkürlich oben und unten angebracht). Zwischen VWR und SpV besteht eine Schnittstelle zur Steuerung und Signalisierung.

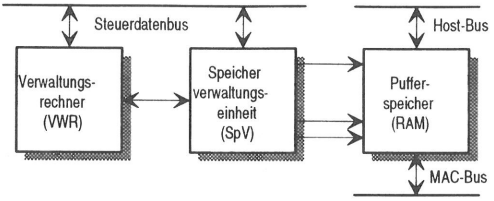


Bild 5.9: *Prinzipielle Anordnung der Speicherverwaltungseinheit*

Die letzten beiden Forderungen führen unmittelbar zum Prinzip der logischen Verkettung von Speichersegmenten (Elementarpuffer, EP), die sich an beliebiger Stelle im Speicher befinden können. Die Festlegung einer einheitlichen Länge für alle EPs reduziert die Komplexität der Verwaltung erheblich. Wird eine Zweierpotenz gewählt, kann die physikalische Adresse eines Speicherworts - wie im Bild 5.10 gezeigt - ohne Zwischenrechnungen direkt durch die Aneinanderreihung von umgesetzter EP-Nummer und Relativwert innerhalb des EPs (offset) gebildet werden.

Soll nicht auf ein bestimmtes Speicherwort zugegriffen werden, sondern (sequentiell) auf alle Worte des gesamten EPs, so kann der jeweils benötigte Relativwert von einem Binärzähler innerhalb der SpV bereitgestellt werden. Mit einem ladbaren Zähler kann auch der wahlfreie Zugriff ermöglicht werden.

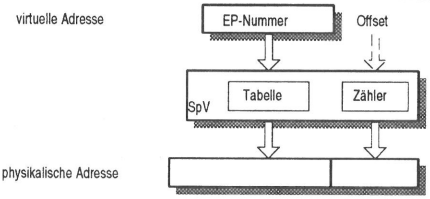


Bild 5.10: *Generierung der physikalischen Adresse eines Speicherwortes*

Die Umsetzung einer EP-Nummer in die höherwertigen h Bits der physikalischen Adresse kann auf einfache Weise realisiert werden: die EP-Nummer ist die Adresse eines Speichers innerhalb der SpV, die adressierte Speicherstelle enthält die gesuchten h Bits. Ein eventuell nachfolgender EP sollte idealerweise gänzlich ohne Mitwirkung des VWRs gefunden werden. Dazu kann in der gleichen Speicherstelle die Information über den Folge-EP in Form dessen Nummer abgelegt sein.

In Bild 5.11 soll auf die PDU, welche mit dem EP 0012 beginnt, zugegriffen werden. Sind alle Worte aus EP 0012 behandelt worden, wird für die weitere Adressierung auf den Inhalt des EPs 0432 zugegriffen usw., bis als Nachfolgenummer eine vordefinierte Endekennung (im Bild 5.11 willkürlich 0000) erreicht wird oder das Datenende auf andere Weise signalisiert wird.

Will man s EPs verwalten, wäre dazu ein Speicher der Größe s•(h + ld s) erforderlich. Bilden die s EPs einen zusammenhängenden Block, können die zugehörigen h Bits der physikalischen Adresse auch durch einfache arithmetische Operationen aus der Nummer des EPs abgeleitet werden. Bei optimaler Anordnung des zusammenhängenden Speicherblocks können diese sich sogar zur Identitätsabbildung reduzieren; die EP-Nummer stellt dann unmittelbar einen Teil der physikalischen Adresse dar.

Von dieser vereinfachenden Annahme konnte im vorliegenden Anwendungsfall ausgegangen werden, denn der Speicher ist ausschließlich über die SpV ansprechbar. Eine Rücksichtnahme auf geschützte oder belegte Adreßbereiche ist daher nicht notwendig, die EPs können also entsprechend ihrer Nummer angeordnet werden: z.B. beginnt der 7. von 16 vorhandenen EPs ab der Adresse 0111 00..0 (die Zahl der folgenden Nullen ist durch den Logarithmus dualis der EP-Größe bestimmt).

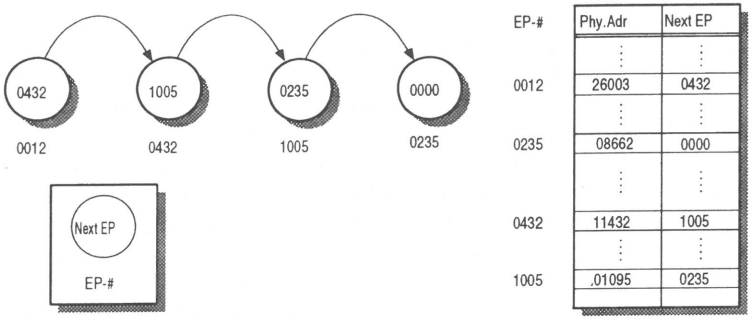


Bild 5.11: *Realisierungsprinzip der Verkettung*

Bewußt wurde keine Unterscheidung zwischen Schreib- und Lesezugriff vorgenommen; das Auffinden des nächsten unbelegten EPs beim Einschreiben von Daten verläuft tatsächlich völlig analog zum Ermitteln des Folge-EPs beim Lesen: alle unbelegten EPs sind nämlich zu einer Leerkette verbunden.
Es lassen sich folgende Funktionalitäten ableiten (zusätzlich ist eine Rücksetzmöglichkeit vorhanden):

1. Lesen ohne Austragen, ab einer EP-Nummer
2. Lesen ohne Austragen, ab einer EP-Nummer mit Offset
3. Lesen mit Austragen, ab einer EP-Nummer
4. Schreiben (Rückgabewert: Nummer des ersten EPs)
5. Nur Austragen, ab einer EP-Nummer
6. Initialisieren

Die Initialisierungsfunktion 6 muß vor allen anderen Funktionen mindestens einmal ausgeführt werden. Sie legt die verkettete Liste aller freien Segmente des Pufferspeichers an.
Die drei Lesevarianten ergeben sich durch die Wahl zwischen zerstörendem und nichtzerstörendem Zugriff, sowie durch die Forderung nach wahlfreiem Zugriff auf jedes einzelne Speicherwort. Zerstörendes Lesen bedeutet, daß die EPs sofort nach dem Auslesevorgang den freien EPs zugeordnet werden. Das Lesen ohne Austragen ist speziell beim sendenden Teil in Kommunikationssystemen von Vorteil: abgesandte Datenteile dürfen erst nach Erhalt einer Quittung überschrieben werden. Zwangsläufig muß für diesen Fall die Funktion 5 "Nur Austragen" eingeführt werden. Ohne direkten Verweis auf den letzten EP ist es hierzu unumgänglich, durch die ganze verzeigerte Liste zu gehen, bis die Endekennung detektiert wird. Mit vorhandenem Verweis auf den letzten EP brauchen "nur" einige Zeiger umgesetzt werden. Allerdings erhöht sich die Komplexität sowohl der Baugruppe als auch der Schnittstelle deutlich.

Sofort nach dem Erhalt eines Schreibkommandos muß die Nummer des ersten beschriebenen EPs bekannt gemacht werden, um spätere Lesevorgänge zu ermöglichen. Stellt die SpV bei einer Schreiboperation fest, daß kein freier Platz mehr im Pufferspeicher zur Verfügung steht und wird auch von extern nicht rechtzeitig das PDU-Ende signalisiert, werden ab der zugewiesenen Startsegmentnummer alle bis zu diesem Zeitpunkt eingeschriebenen Segmente wieder den Freien zugeordnet und damit gelöscht.

Um solche Verluste zu vermeiden, muß der VWR die Möglichkeit haben, sich über den augenblicklichen Füllstand der SpV zu informieren. Es ergeben sich zwei weitere Funktionen:

6. Lesen des Füllstandes und
7. Programmieren und Anzeige eines Schwellwert
   (im Englischen oft mit half full bezeichnet)

Da EPs nur als ganzes vergeben werden können, hat deren Größe einen direkten Einfluß auf den erzielbaren effektiven Speichernutzungsgrad. Andererseits steht im allgemeinen der Verwaltungsaufwand in einem umgekehrt proportionalen Verhältnis zur EP-Größe.

Im vorliegenden Entwurf kann das Umschalten auf den nachfolgenden EP durch zeitgleiche Vorschau während des Zugriffs auf das letze Speicherwort des aktuellen EPs - und damit *ohne* (!) jegliche sichtbare Zeitverzögerung - vorgenommen werden. Dadurch sind EP-Größen bis herunter zu einem Speicherwort möglich. Die Größe der EPs wirkt sich lediglich auf die Breite des Relativwertzählers und entweder auf die Anzahl von EPs oder die maximal verwaltbare Speichergröße aus.

Bei der Dimensionierung wurde die Anzahl der EPs auf 65.535 - entsprechend $2^{16} - 1$ (wegen der Endekennung) - festgesetzt. Für die Datenspeicher wurde eine EP-Größe von 128 Oktetts als sinnvoll erachtet, da in heutigen Netzen die Zahl kurzer Nachrichten klar dominiert [96, 108, 317]. Es ergibt sich eine maximale Speichergröße von $2^{16} \cdot 2^7 = 2^{23} = 2^3 \cdot 2^{20} = 8$ MB. Bei den Steuerkopfspeichern ist eine EP-Größe von 8 oder 16 Oktetts angebracht. Die Begrenzung von 0.5 bzw. 1 MB ist dabei nicht relevant.

Die Steuerlogik der SpV wurde durch hierarchisch strukturierte miteinander kommunizierende endliche Automaten realisiert. Das Operationswerk entspricht einem einfachen Rechnerkern mit Registern, schaltbaren Verbindungswegen, Komparatoren, Zählern und üblichen Zusatzgattern. Erwähnenswert ist der 16 Bit Zähler, der für die Initialisierung und die Füllstandsanzeige benötigt wird.

Vom Gatteraufwand minimale asynchrone Zähler (ripple carry counter) kamen auf Grund des zu großen Zeitbedarfs für das Weiterreichen des Übertrags durch alle 16 Stufen nicht in Frage. Synchrone Binärzähler mit vorausschauender Übertragsberechnung (carry look ahead) zeichnen sich durch eine mit der Zählerbreite weit überproportional steigende Gatteranzahl aus [29, 45]. Die Auf- Abwärtszählung erfordert bei diesem Zählertyp ebenfalls eine hohe Gatterzahl. Statt dessen kann hier ein linear rückgekoppeltes Schieberegister (Linear Feedback Shift Register, LFSR) eingesetzt werden.

Die theoretische Grundlage eines solchen Zählers bildet die aus der Kodierungstheorie bekannte Eigenschaft der Potenzreste im Galoisfeld (d.h. alles Modulo 2), bezüglich eines Polynoms M(u) [41]. Dabei bilden die Restpolynome u' mod M(u) dann einen Ring mit 2^n Elementen, wenn das Polynom M(u) vom Grad n ist. Die maximal auftretende Periode ist $2^n - 1$, da das Nullelement mit u multipliziert wieder Null ergibt. Polynome, die die maximale Periode erreichen, heißen primitive Polynome. Der Zyklus der Polynomreste kann mittels Modulo 2 Addition (entspricht EXOR-Gattern) rückgekoppelten Schieberegister nachgebildet werden, wobei die Polynomkoeffizienten als Rückkopplungszweige interpretiert werden. Ein Beispiel eines 3 Bit breiten LFSR ist in Bild 5.12 zu sehen.

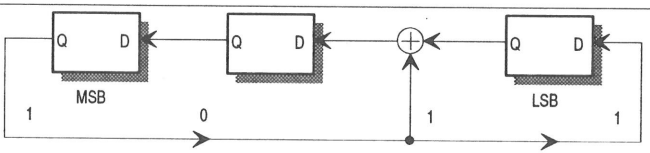


Bild 5.12: *Realisierung des Polynomrestezyklus durch ein LFSR*

Da es für (fast) alle Längen primitive Polynome mit einem oder wenigen Rückkopplungszweigen gibt [31], ist die benötigte Gatterzahl nur minimal höher als beim asynchronen Zähler. Zusätzlich zum Schalten der Speicherglieder muß nur eine einzige Gatterlaufzeit bis zum Anliegen des korrekten Zählerwertes beachtet werden; damit ergibt sich nahezu das ideale Zeitverhalten eines echten Synchronzählers. Nachteil dieses Zählertyps ist der Verlust der sequentiellen Binärreihenfolge und die gegenüber einem Binärzähler um 1 verringerte Periodenlänge (fehlendes Nullelement). Für die interne Arbeitsweise der SpV machen sich diese beiden Nachteile nicht bemerkbar, beim Programmieren des Schwellwertes und bei der Abfrage des aktuellen Füllstands muß allerdings die Kodierung beachtet werden.

Das Rückwärtszählen erfordert inverse Polynome. Statt jedoch mathematisch vorzugehen, hilft in diesem Fall auch die folgende anschauliche Überlegung zum Auffinden der passenden Rückkopplungspositionen:

Beim Rückwärtszählen muß der letzte Schiebetakt rückgängig gemacht werden. Das entspräche

(gedanklich) einem Schieben in die andere Richtung und einer Subtraktion an den Rückkopplungsstellen. Bei der Modulo 2 Rechnung ist eine Subtraktion gleichbedeutend mit einer Addition. Anschaulich kann daher die Anordnung der EXOR-Gatter und Flipflops beibehalten werden!

In der Realität muß allerdings beachtet werden, daß sowohl die Speicherglieder als auch die EXOR-Gatter ihre Richtung nicht einfach ändern können: physikalische Eingänge können nicht in Ausgänge umgewandelt werden und umgekehrt. Zur Realisierung des Dekrementierens müssen daher Wegeschalter und eine entsprechende Verdrahtung vorgesehen werden. Für das obige Beispiel wäre die in Bild 5.13 angegebene Verdrahtung für die Abwärtsrichtung zu schalten.

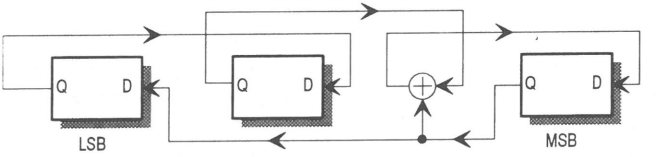


Bild 5.13: *Realisierung des inversen Polynoms durch geänderte Verdrahtung*

Im Verlauf der Arbeit wurden Hilfsmittel zum Auffinden primitiver Polynome und deren automatische Umsetzung in zugehörige Gatterentwürfe erstellt. Grundlage dafür war der Aufbau von geeigneten Standardzellen und die Abbildung der impliziten mathematischen Zusammenhänge auf eindeutige Verdrahtungsregeln. Mit diesen Werkzeugen können "auf Knopfdruck" parallel und seriell ladbare Ab-/Aufwärtszähler (momentan bis zu 64 Bit Breite) oder beliebigen funktionalen Untermengen davon erzeugt werden!

5.3.4 Realisierung des Zweitorverhaltens

Nachdem mit der beschriebenen SpV zeitlich nacheinander Schreib- und Lesezugriffe auf einen Speicher realisiert werden können, liegt es nahe, zwei davon zu kombinieren und intelligent zu koordinieren, um nach außen hin den Eindruck eines gleichzeitigen Zugriffs von zwei Seiten aus zu erwecken. Das entsprechende logische Blockdiagramm ist in Bild 5.14 zu sehen.

Die mit BCC (für Buffer Control Chip) bezeichneten Blöcke enthalten jeweils die Funktionalität der im vorigen Abschnitt beschriebenen SpV. Da beide BCCs jedoch auf denselben Speicher zugreifen, braucht die Anpassung an den Speicher sinnvollerweise auch nur einmal vorhanden zu sein - im Bild 5.14 ist dafür der Block 'RAM-Adaption' vorgesehen.

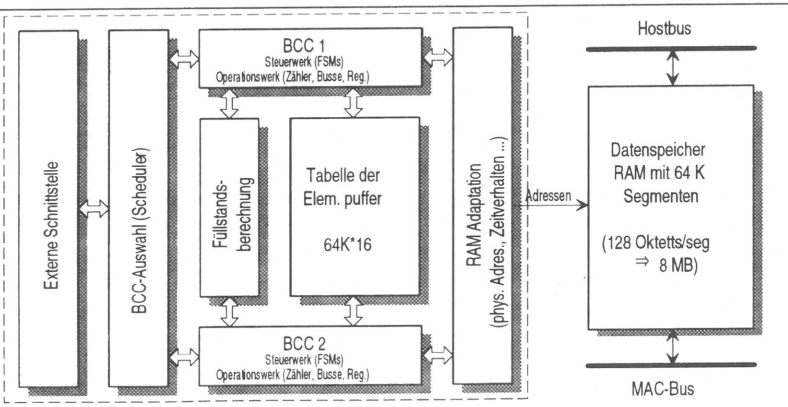


Bild 5.14: *Prinzip der Zweitorfähigkeit*

Steht bei der Auslagerung der Speicheranpassung die Vermeidung funktioneller Redundanz und damit die Einsparung von Gattern im Vordergrund, so ist ein gemeinsamer Füllstand funktionell unabdingbar. Deshalb ist in Bild 5.14 auch für die Füllstandsberechnung ein gesonderter Block vorhanden.

Den Kern der Zweitorrealisierung bildet ein übergeordnetes Steuerwerk ('Scheduler'), welches für die Koordinierung der beiden BCCs und die logische Abwicklung der externen Interaktionen zuständig ist. Die möglichen Zuteilungsstrategien reichen vom exklusiven Speicherzugriff einer der beiden BCCs mit der vollen verfügbaren Speicherbandbreite bis hin zum generellen bedarfsunabhängigen Umschalten des Zugriffrechts nach jedem einzelnen Speicherzyklus.

Die zuerst genannte Verfahrensweise erfordert Prioritäts- und Unterbrechungsmechanismen, kann dafür aber theoretisch in Umgebungen mit bis zu doppelten Bandbreitenbedarf - im Vergleich zur zweitgenannten Variante - eingesetzt werden und die für Kommunikationssysteme typische Büschelförmigkeit besser ausnutzen. Wurden die Annahmen über die Länge der kurzzeitigen Verkehrsspitzen zu optimistisch angesetzt, kann allerdings der Speicher schnell vollaufen.

Grundsätzlich können über jeden der beiden Zugänge beliebige Zugriffe ausgeführt werden. Durch die Speicheranordnung im TpS liegen dort allerdings eindeutige Richtungen vor: ein BCC wird ausschließlich Daten in den Speicher einschreiben, der zweite wird nur lesend zugreifen. Entsprechende Vereinfachungen innerhalb der BCCs wurden trotzdem nicht vorgenommen, um verschiedene ASIC-Varianten zu vermeiden. Als Folge davon konnten auch keine statischen Prioritäten vorgesehen werden - z.B. BCC1 hätte in jedem Fall Priorität gegenüber BCC2.

Die anfangs angestrebte FDDI-Bitrate von 100 Mbit/s bei Datenpfadbreiten von 32 Bit erfordert andererseits Taktraten von "nur" 3.125 MHz bzw. Speicherzykluszeiten von 320 ns. Da preiswerte Speicherbausteine mit deutlich unter 160 ns Zykluszeit verfügbar sind, wurde zunächst ausschließlich das generelle bedarfsunabhängige Umschalten des Zugriffsrechts implementiert.

5.3.5 Zeitgebereinheit

Angestrebtes Ziel war die Entlastung der Prozessoren des TpS von Zeitgeberfunktionen. Zur Interaktion mit der Zeitgebereinheit wurden eine Liste von abstrakten Primitiven, jeweils mit Parametern, festgelegt:

- Timer_Modus
- Timer_Einrichten (ID, Zeitwert, Aktivitätskennung, Ablaufaktion)
- Timer_Löschen (ID)
- Timer_Setzen (ID, Zeitwert)
- Timer_Rücksetzen (ID, [Zeitwert])
- Timer_Start (ID)
- Timer_Stop (ID)

Das Primitiv Timer_Modus gibt das Suchen nach abgelaufenen Zeitgebern und Auslösen einer entsprechende Aktion frei. Die Bedeutung der restlichen Primitive und ihrer Parameter ist selbsterklärend.

Anzahl und Inhalt der Primitive erlauben ein komfortables Arbeiten. Das Anhalten und Wiederanlaufen wird genauso mit eigenen Befehlen unterstützt, wie das Rücksetzen eines Zeitgebers. Bestehende Zeitgeberrealisierungen stellen häufig nur zwei Primitive zur Verfügung: Eintragen und Löschen. Die übrigen Funktionen müssen indirekt realisiert werden. Zum Rücksetzen eines Zeitgebers - eine für Kommunikationssysteme typische Funktionalität - muß beispielsweise der entsprechende Zeitgeber gelöscht werden, der Zeitwert wieder bereitgestellt werden und anschließend ein neuer Zeitgeber angelegt werden (inklusive der aufwendigen Beantragung eines neuen Identifizierers und dessen Eintrag in die entsprechenden Übersetzungstabellen). Das Einrichten eines Zeitgebers ist zwangsläufig mit dem Starten gekoppelt, eine vorausschauende Initialisierung ist daher nicht möglich.

Ausgehend von den in Abschnitt 3.3.3.4 gemachten Betrachtungen wurden zunächst mögliche Konzepte für Zeitgeberrealisierungen erstellt. Bevor auf die letztendlich gewählte Variante eingegangen wird, sind Grundideen und wesentliche Eigenschaften in den folgenden Abschnitten zusammengefaßt.

5.3.5.1 Zeitgeber mit zyklischer Abfrage

Eine direkte Umsetzung des Verfahrens mit ungeordneten Listen führt zu dem Prinzipbild 5.15 für die Kernfunktionalität der Zeitgebereinheit.

Während eines Zyklus wird *jeder* Datensatz des Speichers einmal adressiert. Jeder besteht aus:

- Aktivitätskennung
- Zeitwert (plus evtl. originaler Zeitwert oder Startzeitpunkt)
- Zeitgeberidentifikation und
- Kennzeichnung der gewünschten Ablaufreaktion

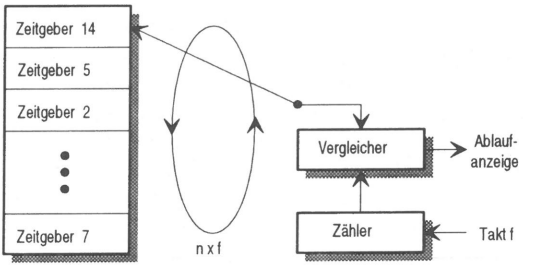


Bild 5.15: *Zeitgeber mit zyklischer Speicherabfrage*

Nach einer Prüfung auf einen aktiven Zeitgeber wird die Zahl der verbleibenden Ticks bis zum Ablauf des Datensatzes ausgelesen, dekrementiert und auf Null geprüft. Ist dies der Fall, d.h. bei Feststellung des Ablaufs des zugehörigen Zeitgebers, wird der Rest des Datensatzes ausgelesen und die gewünschte Ablaufreaktion ausgelöst. Ansonsten wird der geänderte Zeitwert zurückgeschrieben.

Einfügen, Starten, Anhalten und Löschen von Zeitgebern sind sehr einfach durchführbar. Die Kapazität des Speichers hängt nur von der maximalen Anzahl gleichzeitig vorhandener Zeitgeber ab und kann daher an die jeweilige Anwendung angepaßt werden. Theoretisch können beliebig viele Zeitgeber den gleichen Ablaufzeitpunkt besitzen. Der hohe Aufwand für das zyklische Abfragen und Testen aller Speicherstellen ist nur innerhalb der unabhängigen Zeitgebereinheit notwendig, stellt also kein prinzipielles Problem dar. Allerdings ist die Anzahl n der Zeitgeber und Auflösung A durch die Speicherzugriffszeit t_s beschränkt. Wenn für die Vergleich und Dekrementierung zusammen die gleiche Zeitdauer wie für einen Speicherzugriff (unabhängig ob Lesen oder Schreiben) angenommen wird, ergibt sich:

$$A = 3 \bullet n \bullet t_s \tag{5.1}$$

Dekrementieren und Zurückschreiben des Wertes kann vermieden werden, indem als Zeitwert nicht die Anzahl von Ticks bis zum Ablaufzeitpunkt, sondern dieser direkt gespeichert wird. Statt auf Null wird dann auf Gleichheit geprüft. Der Faktor 3 reduziert sich entsprechend. Bei der tatsächlichen Dimensionierung muß neben dem angegebenen Zusammenhang für den normalen Betriebsablauf auch der Zeitbedarf für "Ausnahmeaktionen" wie Anhalten und Starten, Eintragen und Austragen sowie Aktivieren und Deaktivieren eines Zeitgebers berücksichtigt werden. Auch eine geeignete Behandlung des Ablaufs von Zeitgebern in direkt aufeinanderfolgenden Datensätzen darf nicht unberücksichtigt bleiben.

5.3.5.2 Zeitgeber mit geordneter Liste

Auch das in Abschnitt 3.3.3.4 vorgestellte Verfahren zur Verringerung des Pro-Tick-Aufwandes mit sor-

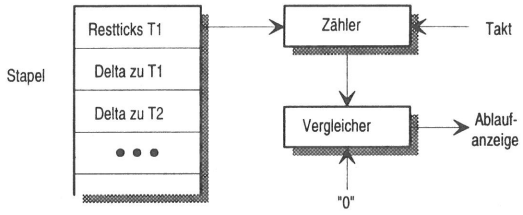


Bild 5.16: *Prinzip des Zeitgebers mit geordneten Listen*

tierten Zeitwerten kann leicht in ein Hardwarekonzept (siehe Bild 5.16) abgebildet werden.

Die Ablaufzeitpunkte werden dazu in aufsteigender Reihenfolge sortiert abgelegt (entweder als absolute Endzeitpunkte oder als Ticks relativ zum Vorgänger). Pro Tick braucht nur der vorderste Datensatz des Stapels manipuliert zu werden. Sehr aufwendig sind dagegen die anderen gewünschten Aktionen.

Werden die Zeitwerte als Anzahl von Ticks relativ zum Vorgänger angegeben, ist beim Einfügen eines neuen Zeitgebers zum Auffinden des richtigen Platzes die aufwendige Summation der Werte aller Vorgänger durchzuführen und eine Korrektur des Nachfolgerwertes erforderlich. Das Anhalten und Weiterlaufenlassen gestaltet sich noch schwieriger. Bei expliziter Angabe des gewünschten Ablaufzeitpunktes sind etwas weniger Berechnungen erforderlich, dafür muß der Anhaltezeitpunkt gespeichert werden und es sind Maßnahmen zur Berücksichtigung von Überlaufeffekten vorzusehen: ein absolut betrachtet kleiner Zeitwert, z.B. hexadezimal 02, braucht nicht zwingend eine kleinere Zeitspanne zu bedeuten als ein absolut großer Zeitwert, z.B. B3, entscheidend ist die relative Position zum aktuellen Stand des als Zeitnormal fungierenden Zählers; im Beispiel gilt: für alle Zählerstände zwischen 02 und B3 ist die Zeitspanne bis zum Ablauf des 02-Zeitgebers größer als die des Zeitgebers mit Ablaufwert B3, nur für Zählerstände größer als B3 wird der Endzeitpunkt 02 früher erreicht als B3.

Die vereinfachte PRO_TICK_ROUTINE rechtfertigt den Aufwand für die "sonstigen" Funktionen nicht.

5.3.5.3 Zeitgeber mit direkter Zeitwertadressierung

Eine Mischung aus den beiden vorherigen Vorschlägen zeigt Bild 5.17. Vom logischen Standpunkt her entspricht es der zweiten Variante mit geordneten Zeitwerten, vom Erscheinungsbild her eher der ersten Variante mit zyklischer Speicheradressierung.

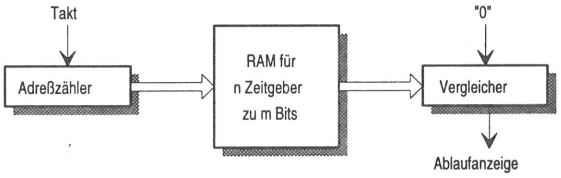


Bild 5.17: *Zeitgeber mit direkter Zeitwertadressierung*

Pro Endzeitpunkt ist ein Datensatz vorgesehen. Der Ausgang des Zählers wird direkt als Adresse darauf benutzt. Der Vorteil gegenüber der vorherigen Variante ist das Vorhandensein einer inneren Ordnung ohne Abhängigkeiten von Vorgänger- oder Nachfolgerelementen. Einfügen, Anhalten und auch alle anderen Aktionen können direkt und ohne Beeinträchtigung anderer Elemente durchgeführt werden.

Nachteil ist der hohe Speicherbedarf und die schlechte Speichernutzung. Jeder Datensatz umfaßt zumindest 2 Oktetts für Identifizierer, Aktivitätsmarkierung und Kennung der Ablaufreaktion, plus eventuell den Zeitwert. Schon bei einem lediglich 16 Bit breiten Zähler ergäbe sich unabhängig von der Zahl akti-

ver Zeitgeber ein (noch akzeptabler) Speicherbedarf von 65.536•4 Oktetts (256 KB). Bei 24 Bit wären schon etwas mehr als 16 Millionen•5 Oktetts (80 MB) erforderlich; wird gar ein Zähler mit 32 Bit Breite gewünscht, wären utopische 4 Milliarden•6 Oktetts (24 GB) Speicherplatz erforderlich!

Der Speicherbedarf läßt sich durch die Einführung einer Hierarchie erheblich reduzieren: wird z.B. ein weiteres Oktett pro Datensatz vorgesehen, welches die Anzahl der noch notwendigen Durchläufe bis zum Erreichen des Endzeitpunkts angibt, so kann mit lediglich 65.536•5 Oktetts (320 KB) der zeitliche Bereich eines 24 Bit breiten Zählers erzielt werden. Die komplexere Berechnung des Endwerts und ein generelles Dekrementieren des Durchlaufzählstandes sind der notwendige (geringfügige) Zusatzaufwand.

5.3.5.4 Zeitgeber mit Assoziativspeicher

Die in Bild 5.18 schematisch gezeigte Zeitgebervariante mit einem Assoziativspeicher - im Bild mit CAM (Content Addressable Memory, inhaltsadressierbarer Speicher) bezeichnet - scheint eng verwandt mit der gerade vorgestellten Variante zu sein.

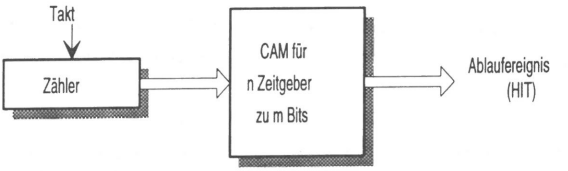


Bild 5.18: *Zeitgeber mit Assoziativspeicher*

Tatsächlich entspricht sie vom Grundprinzip aber der Variante mit zyklischer Speicherabfrage. Denn:

1. sind die Datensätze der Zeitgeber ohne inneres Ordnungskriterium im CAM abgelegt und
2. werden pro Tick alle Datensätze nach einem abgelaufenen Zeitgeber durchsucht.

> Der entscheidende Punkt ist jedoch, daß dieses Absuchen nicht sequentiell, sondern parallel über alle Datensätze gleichzeitig ausgeführt wird!

Das Adressieren jedes einzelnen Datensatzes ist hierbei überflüssig. Statt dessen kann sofort der Vergleich zwischen der aktuellen Zeit und den Endzeitpunkten der einzelnen Zeitgeber durchgeführt werden. Deshalb wird der Stand des Zählers wie in der vorigen Variante direkt an das Speicherelement angelegt. Allerdings hier nicht im Sinne einer klassischen Adresse, sondern als Suchmuster.

5.3.5.5 Implementierte Variante

Sowohl die hierarchisch strukturierte Variante mit direkter Zeitwertadressierung als auch der Vorschlag mit Assoziativspeicher wurden bis auf Gatterebene entworfen und verifiziert. Der Entwurf eines eigenen CAM-Bausteins ist mit der gewählten Semicustomtechnologie zwar prinzipiell möglich, die erzielbare Dichte von deutlich unter 1000 CAM-Zellen, was weniger als 20 einfachen Zählerdatensätzen entspricht, bei gleichzeitig mindestens 64 Anschlußpins läßt deren wirtschaftlichen Einsatz allerdings nicht zu.

Die wenigen am Markt verfügbaren CAM-Bausteine verfügen nur über (zu) kleine Speichertiefen und sind fast ausschließlich für die Verwaltung von schnellen Pufferspeichern (caches) konzipiert. Die Firma AMD (Advanced Micro Devices) bietet einen allgemeineren CAM Baustein an [417]. Dieser zeichnet sich durch komplexe Funktionen aus: u.a. automatisches Sortieren, Zugriff auch über eine bekanntgemachte Adresse, wahlfreie Vereinbarung von Wortbreiten und einfache Kaskadierbarkeit. Neben dem durch die - im TpS nicht benötigte - Komplexität und die geringe Stückzahl bedingten verhältnismäßig hohen Preis, sind der nur 8 Bit breit ausgelegte Zugriff und das uneinheitliche Zeitverhalten die Nachteile des mit CADM (Content Addressable Data Manager) bezeichneten Bausteins.

Getrieben von einem sich abzeichnenden steigenden Bedarf an schnellen hardwareunterstützten Tabellennachschauten, die z.B. bei der Adreßumsetzung in Kopplungseinheiten (bridges und routers) und insbesondere zur Umsetzung der VCI/VPIs (Virtual Connection/Path Identifiers) in ATM-Netzwerken dringend benötigt werden, gewinnen CAMs langsam an Bedeutung. Auch das Bellcore Forschungslaboratorium in Morristown, New Jersey (USA), hat einen entsprechenden Forschungschip entworfen [57]. Dessen Eigenschaften sind für die vorliegende Anwendung geradezu ideal:

- 64 Bit Wortbreite, die beliebig in Such- und Informationsfeld aufgeteilt werden können,
- flexible Maskierungsmöglichkeiten für alle Bits der Wortbreite,
- statische CAM-Zelle (⇒ geringer Leistungsverbrauch, einfache Handhabbarkeit),
- 64 Bit breite vollparallele Zugriffsmöglichkeit,
- schnelles und einheitliches Zeitverhalten: je 100 ns (!) für Schreiben, Lesen und Suchen, (bei der geplanten Auflösung von 1 ms könnte also zwischen zwei Ticks 10.000 mal für die verschiedenen Verwaltungsaufgaben auf den Baustein zugegriffen werden)
- einfache Feststellung und Signalisierung von mehrfach vorhandenen Suchmustern,
- erhöhte Fehlertoleranz durch eingebaute Selbsttest- und Umkonfigurationsroutinen,
- höchstmögliche Dichte durch optimal an typische Kommunikationsanwendungen angepaßte Funktionalität (256 Einträge à 64 Bit),
- Kaskadieren ist vorgesehen und durch spezielle Zusatzsignale leicht möglich.

Die Baugruppe (siehe Bild 5.19) wurde so ausgelegt, daß eine beliebige Aufteilung eines CAM-Wortes gewählt bzw. programmiert werden kann. Folgende Aufteilungen wurden mittels Simulation durchgespielt (jeweils mit 16 Bit für die Zeitgeberidentifikation, 7 Bit für die Kennzeichnung der gewünschten Ablauf reaktion und einem Aktivitätsbit):

1. 32 Bit Zeitwert (> 4 Milliarden Zeitschritte); ein Oktett bleibt ungenutzt oder kann als Zeiger auf ein Feld mit den ursprünglichen Zeitwerten benutzt werden,
2. 24 Bit Zeitwert (über 16 Millionen Schritte); es wird vereinbart, daß nur Zähler bis zu einem maximalen 16 Bit breiten Zeitwert unmittelbar zurückgesetzt werden können. Die 16 Bit des Zeitwertes sind in den zwei noch unbelegten Oktetts angeordnet.
3. 20 Bit Zeitwert (mehr als 1 Mio. Schritte bzw. mit 1 ms Auflösung etwa 17 Min.); Zähler des gesamten Bereichs können zurückgesetzt werden, da genau 20 Bits noch unbelegt sind.

Das Behandeln von Zeitgebern mit gleichem Ablaufzeitpunkt bereitet keine Probleme. Zur Verwaltung der Zeitgeberidentifikationen und dem Zugriff auf spezifische Zeitgeberdaten wird die Möglichkeit des beliebigen Maskierens ausgenutzt: Statt des aktuellen Zeitwerts wird einfach die Zeitgeberidentifikation als Suchmuster angelegt. Auf die gleiche Weise ist es bei Speicherplatzknappheit möglich, einen oder alle gerade inaktiven Zeitgeber zu lokalisieren und gegebenenfalls auszulagern oder zu löschen.

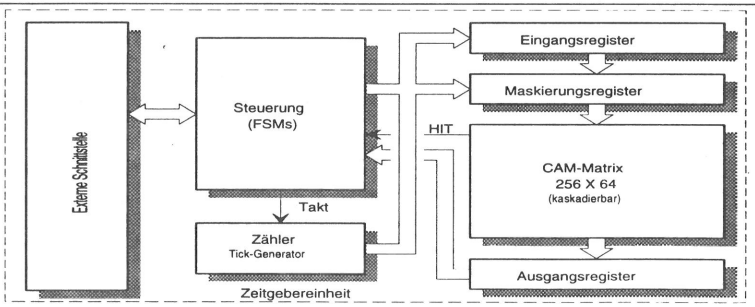


Bild 5.19: *Implementierte Zeitgeberbaugruppe mit Assoziativspeicher*

In der momentanen Version ist keine Umsetzung von externen in interne Zeitgeberidentifikationen vorgesehen. Auch die Verwaltung derselben wird zwar unterstützt aber nicht vollständig durchgeführt. Die Kaskadierfähigkeit wird ebenfalls nicht ausgeschöpft. Die dadurch resultierende Beschränkung auf maximal 256 gleichzeitig behandelbare Zeitgeber ist im Rahmen des Prototyps zu akzeptieren. Die Interaktion nach außen ist durch Register realisiert und dadurch zeitlich vollständig entkoppelt (siehe Abschnitt 5.4).

5.3.6 Prüfsummenbehandlung

Trotz geringer Bitfehlerrate der modernen Übertragungssysteme und geeigneten Mechanismen zu deren Entdeckung unterhalb der Transportschicht wird auf die Schicht-4-Prüfsumme nicht verzichtet. Denn durch den Einsatz des verbindungslosen Schicht-3-Dienstes besteht - zumindest theoretisch - keinerlei Kontrolle über die Qualität benutzter Verbindungsabschnitte [267, 312].

In Abschnitt 3.3.1.5 wurde der Algorithmus vorgestellt, der zur Fehlererkennung auf der Transportschicht bei TP4 standardisiert ist. Der arithmetische Algorithmus nach Fletcher läßt sich zwar in zwei Zeilen niederschreiben und ist dementsprechend nur von geringer logischer Komplexität, bei der Realisierung treten jedoch u.a. durch Variablenüberläufe und die geforderte Modulorechnung unerwartete Probleme auf. Der Zeitbedarf für die in einer Hochsprache definierte Berechnung der Prüfsumme liegt im Bereich von hundert Mikrosekunden pro Oktett (siehe Abschnitt 3.3.1.5). Mit optimierten Realisierungen auf Maschinenebene sind Werte unterhalb von 5 µs ro Oktett erreichbar. Bei angestrebten Gesamtbearbeitungszeiten deutlich unter 100 µs und realistischen mittleren TPDU-Längen von einigen 100 Oktetts sind allerdings selbst diese Zeiten unakzeptabel.

Wird die Prüfsummenoption zugelassen, so ist eine Unterstützung durch Hardware unbedingt erforderlich. Eine direkte Umsetzung würde jedoch durch die erforderlichen Multiplikationen und anderen vorzeichenbehafteten arithmetischen Operationen sowie die Modulorechnung relativ aufwendig.

Nachfolgend werden Schritte zur Reduzierung der Komplexität bei gleichzeitiger Ausrichtung auf maximal erzielbare Geschwindigkeiten beschrieben. Dabei werden Erstellung und Überprüfung der Prüfsummenoktetts getrennt behandelt, da bei der Prüfsummenerstellung zusätzliche Aspekte zu beachten sind.

5.3.6.1 Prüfsummenverifikation

Der Fletcheralgorithmus läßt es zu, daß die Werte der Prüfoktetts nicht für die Auswertung benötigt werden. Ist der Beginn einer zu prüfenden zusammenhängenden Folge von Oktetts bekannt (siehe Abschnitt 5.3.2), kann die Überprüfung daher sequentiell Oktett für Oktett bis zum (bekannten) Ende durchgeführt werden, ohne Position oder Wert der Prüfoktetts kennen zu müssen. Ein ansonsten erforderlicher Aufwand für das Lokalisieren, Extrahieren und Vergleichen der Prüfoktetts entfällt somit.
Verbleibende aufwendige Funktionen bei der Überprüfung eines Oktettstromes sind:

1. die pro Oktett notwendige Multiplikation und
2. die Moduloarithmetik

Ein erster Optimierungsschritt ist, die Multiplikation auf 8-Bit Operatoren zu beschränken.
(Zum einen sind entsprechende Multiplizierer in vielen Zellbibliotheken als Standardelemente verfügbar, zum anderen sind mit diesen Modulen Zeiten unter 50 Nanosekunden erzielbar. Auf Multiplizierer für größere Operandenbreiten trifft beides nicht zu). Schaut man sich die Fletcherforderungen

$$\sum_{i=1}^{L} a_i = 0 \quad \text{bzw.} \quad c0^n = c0^{n-1} + a^n \quad \text{für} \quad n = 1..L, \text{ und } c0^0 = 0 \quad \text{mod} 255 \qquad (5.2)$$

$$\sum_{i=1}^{L} i\,a_i = 0 \quad \text{bzw.} \quad c1^n = c1^{n-1} + na^n \quad \text{für} \quad n = 1..L, \text{ und } c1^0 = 0 \quad \text{mod} 255 \qquad (5.3)$$

an, sieht man, daß bei Oktettströmen bis zu einer Länge von 255 garantiert die 8-Bit Begrenzung eingehalten wird. Bei größeren Längen kann die Invarianzeigenschaft der Modulorechnung ausgenutzt werden.

Sei m der Modul, dann läßt sich i allgemein darstellen als

$$i = n \bullet m + k \quad bzw.\ i\,MOD\,m = k \tag{5.4}$$

Für den Summationsterm von c1 ergibt sich damit

$$(i \bullet a_i)\,MOD\,m = (n \bullet m + k)a_i\,MOD\,m = (k \bullet a_i)\,MOD\,m = ((i\,MOD\,m)a_i\,MOD\,m \tag{5.5}$$

Durch frühe Modulobildung des Positionsparameters kann garantiert werden, daß eine auf 8-Bit Operanden ausgelegte Multiplikation ausreicht. Wird ein Modul kleiner 256 gewählt, bei TP4 ist dies mit 255 bekanntlich der Fall, kann auch eine Addition eines weiteren 8-Bit Wertes zum Ergebnis der so durchgeführten Multiplikation nicht zu einem Überlauf über die 16 zur Verfügung stehenden Bits führen.

Die Modulokorrektur kann prinzipiell auf Operanden beliebiger Länge ausgeführt werden. Wenn TRUNC(r) eine Funktion bezeichnet, die angewandt auf eine beliebige reelle Zahl als Ergebnis den ganzzahligen Teil der Zahl liefert, also alle Nachkommastellen abschneidet, so läßt sich die Modulokorrektur zum Modul m einer Zahl i wie folgt angeben:

$$i\,MOD\,m = i - (TRUNC(\frac{i}{m}) \bullet m) \tag{5.6}$$

Allerdings stehen weder für die Division noch für die Funktion TRUNC( ) einfache und zugleich schnelle Schaltungsrealisierungen zur Verfügung, so daß der Modulowert einer Zahl in der Regel durch wiederholte Subtraktion des Moduls von der Zahl i ermittelt wird. Bei einer 16-Bit breiten Zahl und einem Modul von 255 können dazu bis zu 257 Subtraktionen (plus jeweils eine Prüfung des Abbruchkriteriums) notwendig sein. Bei größeren Bitbreiten erhöht sich der Aufwand proportional. Dies ist für den gegebenen Anwendungsfall der schnellen Kommunikation nicht tolerierbar.

Neben einer Subtraktion von *Vielfachen* des Moduls - günstig sind hier speziell 2-er Potenzen, die sich durch einfaches Linksschieben des Moduls ergeben - sollte nach Möglichkeiten gesucht werden, größere Bitbreiten von vorn herein zu vermeiden oder auf einfache Weise auf geringere Längen abzubilden.

Größere Bitbreiten können weitgehend dadurch vermieden werden, daß die Modulokorrektur sehr häufig, d.h. schon bei Zwischenschritten oder auf einzelne Operanden, ausgeführt wird. Der eingesparte Aufwand für die (einmalige oder seltene) Umwandlung großer Zahlen muß gegenüber dem Aufwand der zusätzlich notwendigen Modulokorrekturen abgewogen werden. Bei manchen Operationen - etwa der Multiplikation - lassen sich größere Operandenbreiten allerdings grundsätzlich nicht vermeiden.

Speziell im Falle des Moduls m=255 kann ein allgemein gültiger mathematischer Zusammenhang günstig zur Reduktion großer Zahlen ausgenutzt werden. Der Wert einer Zahl i bestehend aus den n Ziffern i_n, i_{n-1},... i_1 in einem Zahlensystem zur beliebigen Basis b ist definiert durch:

$$i = i_n b^{n-1} + i_{n-1} b^{n-2} + \ldots\ldots + i_2 b^1 + i_1 b^0 \tag{5.7}$$

Ersetzt man b durch (b-1)+1 verändert sich der Wert der Zahl i nicht. Obiger Ausdruck wird dadurch zu:

$$i = i_n((b-1)+1)^{n-1} + i_{n-1}((b-1)+1)^{n-2} + \ldots\ldots + i_1((b-1)+1)^0$$

$$i = i_n(b-1)^{n-1} + i_{n-1}(b-1)^{n-2} + \ldots\ldots + i_1(b-1)^0 + i_n + i_{n-1} + \ldots\ldots + i_1 \tag{5.8}$$

Der Wert dieser Zahl i zum Modul b-1 ergibt sich somit zu:

$$i\,MOD(b-1) = i_n + i_{n-1} + \ldots\ldots + i_1 \tag{5.9}$$

Im Dezimalsystem wird mit dieser einfachen Methode der *Quersummenbildung* die Teilbarkeit beliebig großer Zahlen durch 9, was der Zahlenbasis b=10 minus 1 entspricht, überprüft. Der Teilbarkeitsbegriff ist dabei einer Modulobildung äquivalent. Im Fall des Moduls m=255 kann analog vorgegangen werden:

> Der Wert einer beliebigen Zahl Modulo 255 entspricht dem Modulowert
> der zugehörigen Quersumme der Ziffern dieser Zahl zur Basis 256.

Die Ziffern zur Basis 256 sind die einzelnen Oktetts der Zahl. Ein Beispiel verdeutlicht den Sachverhalt. Gegeben sei die Zahl 50.537, gesucht deren Wert zum Modul m=255. Im binären Zahlensystem wird die Zahl durch 1100 0101 0110 1001 dargestellt. Die Ziffern im Zahlensystem zur Basis 256 haben die dezi-

malen Werte 197 bzw. 105. Addiert man beide Ziffern, d.h. die beiden Oktetts, ergibt sich 302 = 255+47, also der Modulwert 47. Geht man direkt von der Zahl aus, erhält man 50.537 = 198•255 + 47. Der Modulwert ist somit tatsächlich identisch.

Statt (maximal) 198 Subtraktionen muß mit diesem Verfahren nur eine Addition und eine nachfolgende Subtraktion durchgeführt werden - wenn die herkömmliche Vorgehensweise der wiederholten Subtraktion des Moduls für die Modulokorrektur eingesetzt wird. Jede beliebige ursprünglich 16-Bit breite Zahl wird auf eine gleichwertige Zahl mit einer maximalen Breite von 9 Bit abgebildet.

Stellt man im obigen Beispiel das Resultat der Addition (302) im Zahlensystem zur Basis 256 dar, (binär: 0000 0001 0010 1110 entsprechend 1 46 als dezimale Ziffernwerte), fällt auf, daß die niederwertige Ziffer sich um genau eins von dem gesuchten Modulwert unterscheidet und die höherwertige Ziffer gerade den Wert eins besitzt. Diese Feststellung ist völlig unabhängig von der gewählten Beispielszahl und muß sich auch durch die Wahl des Moduls zwangsläufig ergeben! Sieht man einmal von dem Wert 255 ab, der definitionsgemäß dem Wert 0 zum Modul 255 entspricht, so kann der resultierende Modulwert offensichtlich durch eine erneute Addition der zwei Ziffern konstruiert werden!

Hardwarebaugruppen zur Addition von zwei Oktetts liefern als Ergebnis jedoch keine zwei, sondern nur ein Oktett. Das höherwertige Oktett kann aber maximal den Wert eins annehmen (255 + 255 = 510, binär: 0000 0001 1111 1110), der Wert des höherwertigen Oktetts wird daher vollständig und korrekt von der meist ohnehin vorhandenen Übertragsmarkierung (carry flag) angezeigt!

Die Lösung zur Berechnung von c0 besteht somit (siehe Bild 5.20) aus einem einzigen 8-Bit Addierer, der in einem ersten Schritt das nächste Oktett zum bisherigen Wert addiert (im Zahlensystem zur Basis 256) und anschließend in einem zweiten Schritt sein eigenes Übertragsbit zum Ergebnis hinzuaddiert (Modulo 255 Korrektur). Am Ende des Oktettstroms muß nun nur noch ein eventuell vorhandenes Ergebnis bestehend aus lauter Einsen (dezimaler Wert: 255) in den Wert 0000 0000 umgewandelt werden. Führt man einen zweiten Addierer und einige Wegewahlschalter (Multiplexer) ein, ist eine Fließbandverarbeitung möglich; der minimal mögliche Abstand aufeinander folgender Oktetts ist dann durch die einfache Addierzeit bestimmt.

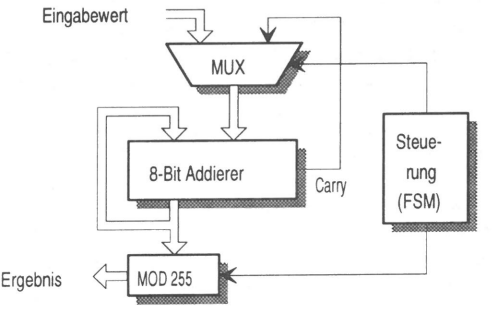


Bild 5.20: *Hardware zur Berechnung einer der beiden Prüfwerte (c0)*

Zur Berechnung von c1 wird ein MOD 255 Zähler, ein 8-Bit Multiplizierer, ein 8-Bit Addierer für die Addition der zwei Oktetts des Multiplikationsergebnisses und deren anschließender MOD 255 Korrektur sowie ein 2. funktionell identischer Addierer zur eigentlichen Aufsummation benötigt. Auch hier kann durch Fließbandverarbeitung die Dauer bis zur Bereitschaft für das nächste Oktett reduziert werden.
Die Multiplikation kann aber durch Rückführung auf eine rekursive Addition auch völlig entfallen!
Angestrebt ist (die Berechnung von c0 soll unverändert bleiben):

$$c0^n = c0^{n-1} + a^n \qquad \text{für } n=1..L, \text{ und } c0^0 = 0 \tag{5.10}$$

$$c1'^n = c1'^{n-1} + c0^n \qquad \text{für } n=1..L, \text{ und } c1'^0 = 0 \tag{5.11}$$

In Tabelle 5.2 sind die Schritte für die Oktetts a_1 bis a_4 sowie für das letzte Oktett a_L aufgeführt.

Tabelle 5.2: *Einzelne Schritte bei der Prüfsummenberechnung*

	c0	c1
a_1	a_1	a_1
a_2	$a_1 + a_2$	$a_1 + a_1 + a_2$
a_3	$a_1 + a_2 + a_3$	$a_1 + a_1 + a_1 + a_2 + a_2 + a_3$
...	...	...
...	...	...
a_L	$a_1 + a_2 + ... + a_L$	$L \cdot a_1 + (L-1) \cdot a_2 + ... + 2 \cdot a_{L-1} + a_L$

Der Nachweis der Korrektheit ist nachfolgend aufgeführt:

$$c1 = \sum_{i=1}^{L} i\,a_i = \sum_{i=1}^{L} (i + (L+1+i) - (L+1+i))\,a_i$$

$$= \sum_{i=1}^{L} (L+1-i)\,a_i + \sum_{i=1}^{L} (2i - L - 1)\,a_i = c1' + 2c1 - (L+1)c0$$

gefordert laut Standard: $c0=0$ und $c1=0 \Rightarrow c1'=0$ (5.12)

Mit dieser Berechnung von c1' entfällt beim Überprüfen eines Oktettstromes die Notwendigkeit für jegliche Multiplikation. Es ergibt sich das in Bild 5.21 gezeigte funktionelle Blockdiagramm.

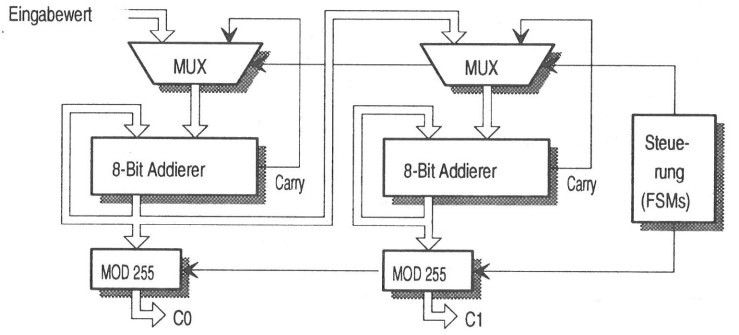


Bild 5.21: *Blockdiagramm des Prüfsummenverifizierers*

Ohne zusätzliche Addierer für die Modulokorrektur, d.h. die Summation der Übertragsmarkierung im 2. Schritt, ergeben sich bei einer konservativen 2 µm Semicustom Technologie Zeiten für eine Stufe der Fließbandverarbeitung, die mit 40 ns drei bis vier Größenordnungen unter den mit Software erzielbaren Werten liegen. 40 ns pro Oktett bedeutet einen Maximaldurchsatz von 8 Bit•25 MHz = 200 Mbit/s!

5.3.6.2 Prüfsummengenerierung

Liegt ein vollständiger zu prüfender Oktettstrom der Länge L vor, der an den zwei Positionen k und k+1 der späteren Prüfoktetts den Wert Null enthält, verläuft die Berechnung zunächst völlig identisch wie bei der Überprüfung eines empfangenen Datenstroms mit gesetzten Prüfoktetts. Für jedes Oktett werden die Werte c0 und c1', wie im vorigen Abschnitt beschrieben, durch sukzessive Addition erstellt. Nach dem L-ten Oktett ergeben sich zwei Gleichungen für die Prüfoktetts x und y. Aufgrund der geänderten Berech-

nungsweise für c1' ist nur die erste zur Gleichung (3.11) in Abschnitt 3.3.1.5 identisch

$$(c0+x+y)\,MOD255=0 \tag{5.13}$$

$$(c1'+(L-k+1)x+(L-(k+1)+1)y)\,MOD255=0 \tag{5.14}$$

Aus den zwei Gleichungen lassen sich x und y wie folgt berechnen:

$$x=((l-k)c0-c1')\,MOD255 \tag{5.15}$$

$$y=(c1'-(L-k+1)c0)\,MOD255 \tag{5.16a}$$ oder $$y=-(x+c0)\,MOD255 \tag{5.16b}$$

Die Berechnungen für die Prüfoktetts beinhalten Multiplikationen und Subtraktionen, beide wiederum in Modulo 255 Arithmetik. Mit dem oben Gesagten ist der Bedarf eines relativ großen Zeitaufwands einsichtig. Eine spezielle Hardware für die Berechnung von x und y wäre zwar machbar, würde aber in keinem Verhältnis zum erzielbaren Gewinn stehen. Im Gegensatz zu den Summationen für c0 und c1', die für jedes Oktett ausgeführt werden müssen, wird die Berechnung von x und y nur einmal pro TPDU benötigt. Eine zeitaufwendigere Berechnung in Software kann deshalb an dieser Stelle toleriert werden.
Zwei andere Aspekte der TP4-Prüfsumme stehen einer effizienten Realisierung grundsätzlich entgegen:

1. die Prüfsummenoktetts befinden sich im Steuerkopf einer PDU und
2. die Prüfsumme ist über die komplette PDU, also Steuerkopf plus Nutzdaten, auszuführen.

Die 1. Eigenschaft verhindert, daß Daten direkt vom Speicher des ARs an die MAC gegeben werden können, während gleichzeitig die Prüfsumme erstellt wird. Statt dessen muß jedes Oktett nach dem Durchlaufen des Prüfsummenbausteins solange im Sendedatenspeicher des TpS gespeichert werden, bis alle Oktetts bearbeitet wurden sowie x und y berechnet und an der richtigen Stelle eingetragen worden sind.

Die Berechnungsvorschrift für c1 und auch c1' enthält einen Bezug zur absoluten Position eines Oktetts. Eine - im Widerspruch zur 2. Eigenschaft stehende - voneinander entkoppelte Prüfsummenberechnung nur über die Daten, welche sinnvollerweise gleichzeitig mit dem Übertragen der Daten vom AR zum TpS erfolgen könnte, ist nicht möglich, da die Prüfsumme auch über den vom TpS anschließend erst zu erstellenden Steuerkopf gebildet werden muß. Ist der Steuerkopf erstellt und sind alle Daten in den Datenspeicher des TpS übertragen worden, verhindert wiederum die 1. Eigenschaft eine mit dem Auslesen in die MAC gekoppelte Prüfsummenerstellung. Statt dessen scheint ein zusätzliches Bewegen der kompletten TPDU - was eine erhebliche zusätzliche Verzögerung impliziert - unumgänglich zu sein.

Die als Folge der 1. Eigenschaft notwendige Zwischenspeicherung der Daten wird als nicht gravierend angesehen. Bei einer beschränkten TPDU-Größe von 1024 Oktetts, 32 Bit breiten Verbindungswegen und Speichern mit 100 ns Zykluszeit werden zwar mindestens 25.6 µs für die Übertragung vom AR zum TpS benötigt. Diese Zeit ist jedoch nicht völlig "verloren", da nur in wenigen Situationen (Niedriglast, verzögerungsfreies Medienzugriffsprotokoll etc.) von einem unmittelbaren Medienzugriff ausgegangen werden kann, sowie die Erstellung des Steuerkopfes und die Koordinierung innerhalb des TpS auch gewisse Zeiten benötigen. Außerdem werden AR und Medium bzw. MAC durch eine Zwischenspeicherung entkoppelt, was grundsätzlich positiv zu bewerten ist.
Das zusätzliche Bewegen einer TPDU für die Prüfsummenerstellung ist dagegen nicht zu akzeptieren.

Eine erste Möglichkeit dies zu vermeiden, ist die Erstellung des Steuerkopfes, verbunden mit dem gleichzeitigen Berechnen von c0 und c1' über diesen, bevor die Daten zum TpS übertragen werden. Die Übergabe von c0 und c1' an den Prüfsummenbaustein wäre dann gleichzeitig das Startsignal für die Datenübertragung (und die damit gekoppelte sukzessive Berechnung von c0 und c1'). Nachteilig ist der Zeitverzug und der erhöhte Kommunikationsbedarf sowohl zwischen AR und TpS als auch TpS-intern.

Der Zeitverzug läßt sich drastisch reduzieren, wenn man das Wissen über bestehende Verbindungen dazu ausnützt, den Steuerkopf der nächsten - für eine spezifische Verbindung - zu versendenden TPDU schon vorab bereitzustellen (header prediction). Die Steuerköpfe der Schichten 2b und 3 sind innerhalb einer Verbindung absolut konstant, im Schicht 4 Steuerkopf ändert sich meist nur die Folgenummer gegenüber

der vorangegangenen TPDU (bei Quittungen zusätzlich noch der Kreditwert). Ist der Steuerkopf für die nächste zu sendende TPDU schon bekannt und über den Verbindungsdatensatz auch auffind- und ansprechbar, so ist die Berechnung von $c0_S$ und $c1_S$ bzw. $c1'_S$ über die Oktetts des Steuerkopfs der nächsten TPDU ebenfalls leicht zu erstellen und zu speichern. Nach dem letzten Datenoktett stellen die im Prüfsummenbaustein enthaltenen Werte c0 und c1 (c1') direkt die für die Berechnung der beiden Prüfoktetts erforderlichen Größen dar. Der erhöhte Kommunikationsaufwand wird durch die Prüfsummenvorschau nicht beeinflußt.

Wird die originale Vorschrift für c1 benutzt ($c1^n = c1^{n-1} + i \cdot a_i$, Modulo 255 wird hier wie im folgenden nicht mehr explizit erwähnt!), so reicht zur Berechnung der Parameter $c0_D$ und $c1_D$ über alle Datenoktetts das Wissen über die Länge des Steuerkopfes aus. Die vollständigen Werte c0 und c1 ergeben sich durch einfache Addition der entsprechenden Parameter der Steuerköpfe ($c0_S$, $c1_S$) und Daten ($c0_D$, $c1_D$):

$$c0 = c0_S + c0_D \qquad bzw. \qquad c1 = c1_S + c1_D \tag{5.17}$$

Eine Verbesserung gegenüber der Variante mit Übergabe der vollständigen Parameter $c0_S$ und $c1_S$ ist nur für den Fall fester Steuerkopflängen erzielbar. In diesem Falle kann jederzeit eine Datenübertragung starten ohne vorher noch Daten aus dem Verbindungsdatensatz in den Prüfsummenbaustein übertragen zu müssen. Die Vereinheitlichung der Längen aller TPDUs überwiegt andererseits diesen Vorteil deutlich. Außerdem kann der im vorigen Abschnitt vorgestellte Baustein aus einfachen Addierern nicht eingesetzt werden, da die Überlegungen nur auf die originale multiplikative Berechnung von c1 anwendbar ist.

Auf der Suche nach einem Ausweg wurden schließlich die im folgenden beschriebenen mathematischen Umformungen vorgenommen. Zunächst werden die Grundgleichungen für $c0_S$ und $c1'_S$ sowie $c0_D$ und $c1'_D$ angegeben, S gibt die Länge des Steuerkopfes an, D entsprechend die der Daten (S + D = L). Das erste Datenoktett sei zunächst vereinfachend mit a_1 statt a_{S+1} bezeichnet. Zur Verdeutlichung wird als Laufindex j statt i eingesetzt.

$$c0_S = \sum_{i=1}^{S} a_i \qquad bzw. \qquad c0_D = \sum_{j=1}^{D} a_j \tag{5.18}$$

$$c1'_S = \sum_{i=1}^{S} (S - i + 1) a_i \qquad bzw. \qquad c1'_D = \sum_{j=1}^{D} (D - j + 1) a_j \tag{5.19}$$

Das Gesamt-c0 ergibt sich direkt aus der Summe von $c0_S$ und $c0_D$. Bei den c1'-Werten geht das nicht. Beide Ausdrücke können aber geeignet umgeformt und anschließend zusammengefaßt werden.

$$c1'_S = \sum_{i=1}^{S} ((S - i + 1) + D - D) a_i = \sum_{i=1}^{S} (S + D - i + 1) a_i - \sum_{i=1}^{S} D a_i = \sum_{i=1}^{S} (L - i + 1) a_i - D c0_S \tag{5.20}$$

Der Faktor (D - j + 1) in der Summe für $c1'_D$ ist durch die Differenz der oberen Grenze und der Laufvariablen bestimmt. Er ändert seinen Wert nicht, wenn sowohl zur Laufvariablen als auch zur oberen Grenze eine Konstante addiert wird. Daher gilt auch:

$$c1'_D = \sum_{j=1+S}^{D+S} (D + S - j + 1) a_j = \sum_{j=1+S}^{L} (L - j + 1) a_j \tag{5.21}$$

Der erste Faktor von $c1'_S$ - die verbleibende Summe - hat nun denselben Summationsausdruck wie die Summe für $c1'_D$, die Grenzen passen nahtlos aneinander, eine Zusammenfassung von $c0_S$ und $c0_D$ bietet sich folglich an. Der Laufindex j wird wieder in i umbenannt. Gleichzeitig wird das erste Datenoktett als a_{S+1} bezeichnet (was keinerlei Einfluß auf den Wert des Ausdrucks zur Folge hat, sondern lediglich eine Konvention für Bezeichner ist).

$$c1'_S + c1'_D = \sum_{i=1}^{S} (L - i + 1) a_i - D c0_S + \sum_{i=1+S}^{L} (L - j + 1) a_i = \sum_{i=1}^{L} (L - i + 1) a_i - D c0_S \tag{5.22}$$

Die Summe entspricht genau dem originalen c1'! Somit kann diese aus den Parametern der vorhandenen Teilberechnungen ermittelt werden (das Ergebnis für c0 ist zur Übersichtlichkeit aufgeführt):

$$c0 = (c0_S + c0_D) \, MOD 255 \tag{5.23}$$

$$c1' = (c1'_S + c1'_D + Dc0_S) \, MOD 255 \tag{5.24}$$

Die Berechnung für den Datenteil kann also völlig unabhängig von der des Steuerkopfs begonnen und durchgeführt werden!

Der Aufwand für die Verknüpfung der Teilparameter zu den benötigten Parametern c0 und c1' liegt unter dem der eigentlichen Berechnung der Prüfoktetts x und y und ist somit akzeptabel. Voraussetzung für die Verknüpfung ist das Wissen über die Länge der Daten. Ist die Datenlänge nicht eindeutig quantisiert, beispielsweise entweder 128 oder 1024 Oktetts, erfordert diese Angabe neben einem Zähler einen zugehörigen Kommunikationsvorgang. Entsprechende Zähler sind sowohl im Prüfsummenbaustein, in der Transfermaschine (DMA-Einheit) als auch innerhalb der Pufferverwaltung vorhanden.

Für die Erstellung der zwei Prüfoktetts kann somit der gleiche einfache Baustein zur Anwendung kommen, wie zur Überprüfung eines empfangenen Datenstroms. Es muß lediglich ein Auslesen der Ergebnisoktetts c0 und c1' (bzw. $c0_D$ und $c1'_D$) und gegebenenfalls ein ebenfalls lesbarer Oktettzähler hinzugefügt werden. Die Leistungsfähigkeit (für die innere Schleife) bleibt bei 200 Mbit/s (40 ns pro Oktett).

5.3.7 Aufwandsabschätzung

Die 6 ASICs haben jeweils eine Komplexität von einigen Tausend NAND2-Gatteräquivalenten. Mit den günstigen Bedingungen am Institut für Mikroelektronik Stuttgart (IMS), bei welchem statt den teuren Belichtungsmaskensätzen Strukturen direkt mittels Elektronenstrahl auf die Halbleiterscheiben (wafer) aufgebracht werden können, sind Kleinserien (10 - 25 funktionierende Exemplare) von ASICs dieser relativ geringen Komplexität zu Preisen deutlich unter 5000 DM zu erhalten [65, 66, 182].

Die Art der Schaltungen ist darüber hinaus für eine Implementierung mit den neuen Typen der programmierbaren Logikbausteine ((F)PLD für (Field) Programmable Logic Device oder FPGA für Field Programmable Gate Arrays, [115]) geeignet, die u.a. von den Firmen Altera, Xylinx und Actel angeboten werden. Mit Verwendung dieser Bauteile sinkt der Stückpreis unter 100 DM.

Da keine der Leiterplatten so komplex ist, daß aufwendige Mehrschichttechnologie erforderlich ist, kann pro Leiterplatte mit maximal 1.000 DM kalkuliert werden. Die Preise der Intel 80960 Prozessorfamilie liegen zwischen 200 und 1000 DM je nach Typ und Leistungsklasse.

Als letzter größerer Posten bleiben die Speicher anzuführen: für die Datenspeicher sind - wie schon mehrfach erwähnt - die derzeit gängigen dynamischen RAM-Bausteine mit Zykluszeiten unter 160 ns vorgesehen. Die Kosten für 1 MB (teilweise inklusive Paritätsbit) in Kleinplatinenform SIMM (Single Inline Memory Module) belaufen sich auf unter 100 DM (Stand Juni 1992). Die statischen CMOS-Speicher für die Speicherung der Steuerköpfe kosten pro MB zwar etwa vier bis fünfmal soviel, davon wird aber nur ein Bruchteil der Kapazität der Datenspeicher benötigt.

Selbst mit einer Maximalkonfiguration mit den leistungsfähigsten Prozessortypen i80960 CA 33 MHz, 2•8 MB Datenspeicher, jeweils 256 KB statischer Speicher für Steuerköpfe und jedes der Prozeßkoppelelemente bleiben die reinen Hardwarekosten des gesamten TpS deutlich unter 10.000 DM (werden IMS ASICs eingesetzt, gilt dieser Preis erst ab ca. 5 Systemen). Zum Vergleich seien hier die über 40.000 DM genannt, die für erste FDDI-Ankopplungen bis zur Schicht 2a angesetzt waren. Für die Ankopplungen an die LANs der nächsten Generation wird mindestens von solchen Summen ausgegangen.

Außerdem gehören Systeme, die einen hohen Kommunikationsbedarf als auch über genügend Rechenkapazität, Bus- und Speicherbandbreite verfügen und die extrem hohe Zahl von einigen Tausend Transaktionen pro Sekunde bearbeiten können, nicht in die Kategorie der einfachen Arbeitsplatzstationen, sondern mindestens in die der Abteilungsrechner, die im Preisbereich um 100.000 DM angesiedelt sind. Der Aufwand für das vorgestellte TpS bewegt sich daher in einem vernünftigen Rahmen.

5.4 Softwareaspekte

5.4.1 Abgrenzung und Verteilung der Programme

Programme stellen die Realisierung der gewünschten Funktionalität mit Hilfe der Betriebsmittel (Hardware, Firmware) dar. Ihre Abgrenzung und Verteilung spiegelt daher sowohl die prozeduralen Aspekte der Kommunikation, als auch die Art, Struktur und Verhalten der Betriebsmittel wieder.

Allgemein können Kommunikationssysteme logisch als ereignisgesteuerte Systeme aufgefaßt werden. Drei Arten von Protokollereignissen sind beim TpS unterscheidbar (nur Datentransferphase betrachtet):

1. Kommandos vom AR initiieren das Senden von PDUs
2. Empfangsereignisse (PDU-Ankünfte) signalisieren:
   - die Ankunft von Daten und u.U. die Notwendigkeit, den Erhalt der Daten durch Senden einer Quittung zu bestätigen
   - die Ankunft einer Quittung (inkl. Veränderung der Datenflußparameter)
3. lokale Zeitgeberereignisse lösen aus:
   - das Wiederholen von Daten-PDUs
   - das Senden einer Quittung
   - das Beenden einer Transportverbindung.

Eine ereignisorientierte Abgrenzung würde jede Funktion durch ein eigenständiges Programm verwirklichen. Bei jedem Ereignis wäre ein Kontextwechsel zur entsprechenden Routine durchzuführen.

Die Gründe für die Struktur des TpS wurden bereits genannt. Einmal festgelegt, erzwang sie die Zuordnung bestimmter Funktionen zu Verarbeitungsbereichen, in denen die nötigen Betriebsmittel (Speicher, Zeitgeber, Transfermaschinen mit DMA-Fähigkeit) vorhanden waren. Diese Zuordnung bewirkte die weitere Unterteilung der (ereignisorientiert abgegrenzten) Programme. Die Unterteilung war durch das TpS nur bis zu einem gewissen Grad vorgegeben. Wo dies der Fall war, wurden zur Strukturierung die gleichen Kriterien wie beim Entwurf des TpS beachtet:

- möglichst geringe Kommunikation zwischen den Programmen
- minimale Kopplung durch gemeinsam benutzte Betriebsmittel
- möglichst symmetrische, d.h. gleichmäßig verteilte, Auslastung der einzelnen Prozessoren und der zugeordneten Verbindungswege.

Die Implementierung aller Teilfunktionen auf Maschinenebene (i80960-Assembler [400]) konnte unmittelbar für erste Bewertungen herangezogen werden und lieferte zusammen mit den detaillierten Erkenntnissen aus den hardwarenahen funktionellen Simulationen die notwendigen Parameter für die Systemmodellierung und -simulation. Obwohl erst mit deren Ergebnissen genauere Aussagen über tatsächliche Lastverteilungen unter unterschiedlichsten Randbedingungen möglich sind, hat schon das Abzählen der Maschinenbefehle verschiedener möglicher Pfade und die daraus resultierende Anzahl von benötigten Maschinenzyklen in den einzelnen Bereichen (siehe 5.4.8) die vorgenommene Strukturierung bestätigt. Es ergab sich folgende Verteilung der Protokollfunktionen (in Klammern ist jeweils das Protokollereignis angegeben, das mittelbar oder unmittelbar die Ausführung des Programms bewirkt):

Programme im Empfangsbereich:

R1) Protokollverarbeitung
  R1A) Protokollverarbeitung empfangener Daten-PDUs (Ankunft von Daten-PDUs)
  R1B) Protokollverarbeitung empfangener Quittungen (Ankunft einer Quittung)

R2) Überwachung der Zeitgeber und Starten der Prozeduren bzw. Signalisieren von abgelaufenen Sendezeitgebern an den Sendebereich (Ablauf von Zeitgebern)

R3) Auswertung von Kommandos des IF-Bereichs, die das Freiwerden von Elementarpufferm (EPs)

anzeigen (Ankunft von Daten-PDUs)

Programme für die Empfangsrichtung des IF-Bereichs:

IFR1) Ausführung von Kommandos des Empfangsbereichs zum Verketten bzw. Freigeben von Datenteilen empfangener Daten-PDUs (Ankunft von Daten-PDUs)

IFR2) Auslesen von Daten an den AR (Ankunft von Daten-PDUs)

IFR3) Entgegennahme von für den Empfang von datenreservierten Speicherbereichen (Ankunft von Daten-PDUs)

Programme für die Senderichtung des IF-Bereichs:

IFS1) Entgegennahme der Kommandos vom AR zum Senden von Daten aus dessen Speicherbereichen (Sendekommando vom AR)

IFS2) Einlesen von Daten aus Sendebereichen des ARs (Sendekommando vom AR)

IFS3) Ausführung von Kommandos des Sendebereichs zur Freigabe von EPn und zum Aktualisieren des Kreditwertes in Senderichtung (Ankunft einer Quittung)

IFS4) Ausführung von Kommandos des Sendebereichs zum Starten der Übertragung einer PDU (Sendekommando vom AR)

Programme im Sendebereich:

S1) Zusammenstellung der Steuerköpfe für zu sendende Daten-PDUs

S1A) im Normalmodus (Sendekommando vom AR)

S1B) im Wiederholungsmodus (Ablauf des Sendezeitgebers)

S2) Verarbeitung von Empfangsbereich-Ereignissen

S2A) Verarbeitung von Zeitgeberabläufen (Ablauf des Sendezeitgebers)

S2B) Freigabe von Senderessourcen aufgrund angezeigter Quittung (Ankunft einer Quittung)

S3) Zusammenstellung einer Quittung (Daten-PDUs Ankunft bzw. Ablauf des Quittierungszeitgebers).

Der Ablauf der Programme im Zusammenhang wird im Abschnitt 5.4.3.2 erläutert.

5.4.2 Bestimmung der Programmreihenfolge

Wie schon mehrfach betont, gilt es, Unterbrechungsanforderungen und Kontextwechsel zu vermeiden. Wird ein eigenes Programm zur Bestimmung der Abfolge der einzelnen Verarbeitungsfunktionen eingesetzt (scheduler), können zwar bedarfsoptimierte Zuteilungsverfahren und Prioritätsmechanismen realisiert werden, es ist aber einiger Aufwand nötig, um das Programm selbst jeweils zu aktivieren.

Ähnliche Probleme würden bei einer direkten Aktivierung der einzelnen Funktionen durch Unterbrechungsanforderungen der jeweiligen Ereignisse auftreten. Abfolge und Priorität von schnell hintereinander oder gar gleichzeitig eintreffenden Ereignissen müßten eindeutig geklärt sein, Verschachtelungen von Ereignisverarbeitungen und eventuelle Abhängigkeiten wären zu berücksichtigen. Das Ziel der möglichst reduzierten Unterbrechungsanforderungen könnte mit diesem Verfahren auch nicht erreicht werden. Die Einführung von WSn für alle Kommunikationsformen zwischen den einzelnen Bereichen des Systems deutet auf die gewählte Lösung hin:

> Die Programme jedes Bereichs werden in einem festen Zyklus abgearbeitet!

Jedes der Programme besteht aus 2 Teilen (siehe Bild 5.22):

- Im Abfrageteil werden die Voraussetzungen für die Ausführung des 2. Teils geprüft. Abgefragt werden WSn- und DMA-Zustände oder der Status von Zeitgebern. Sind die Voraussetzungen nicht erfüllt, wird das nächste Programm im Zyklus gestartet.

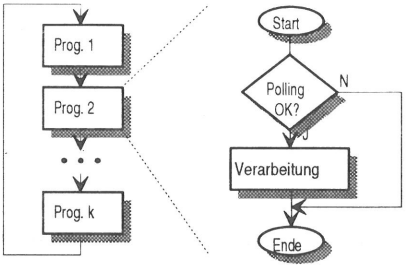


Bild 5.22: *Programmabfolge und prinzipielle Programmstruktur*

- Im 2. Teil findet die eigentliche Verarbeitung statt. Insbesondere werden hier die Kommandos erzeugt, die andere Programme desselben oder anderer Verarbeitungsbereiche anstoßen.

Im Hochlastfall wird durch das kontinuierliche Abarbeiten der vorgegebenen Programmabfolge ein optimaler Nutzungsgrad der VEs erreicht. Der Aufwand für jede Abfrage beschränkt sich auf wenige Maschinenbefehle und bleibt damit deutlich unter dem zeitlichen Aufwand der Reaktion auf eine Unterbrechungsanforderung. Außerdem sind viele Abfragen inhaltlich für die zu erbringende Funktion erforderlich, so daß sich der effektive Zusatzaufwand im Falle des Bedarfs einer Funktion weiter verringert.

Das TpS ist nur für die Wahrnehmung der Aufgaben, die mit dem Senden und Empfangen von Information zusammenhängen, ausgelegt. Sind keine Kommunikationsaufgaben durchzuführen (Niederlastfall), geht durch das stetige Abfragen daher keinerlei anderweitig nutzbare Rechnerkapazität verloren.

Das zyklische Aufrufen aller Programme und die Abfrage zu Beginn jedes Programms stellen sicher, daß:

- kein Aufwand durch Unterbrechungsanforderungen bzw. komplexe Programme zur Ablaufsteuerung entsteht,
- jedes Programm nach einer begrenzten Zeit aufgerufen wird,
- die reguläre Ablauffolge auch im Hochlastfall nicht gestört wird und daher keine entsprechenden Maßnahmen vorzusehen sind,
- der Verarbeitungsteil eines Moduls nur bei Bedarf ausgeführt wird.

Durch die Anordnung einzelner Programme innerhalb des Zyklus und gezielte Mehrfachaufrufe einzelner zeitkritischer Funktionen innerhalb eines Zyklus kann das Verhalten und die Leistungsfähigkeit des Gesamtsystems den gegebenen äußeren Umständen angepaßt werden bzw. Einfluß auf die Dimensionierung von Systemkomponenten genommen werden. Zwei Beispiele für einen sinnvollen Mehrfachaufruf seien hier genannt: zum einen die Funktion welche prüft, ob ehemals volle WSn wieder über Aufnahmekapazität verfügen und, falls ja, anderweitig zwischengespeicherte Aufträge in diese WS überträgt; zum anderen die Funktion, welche die logische Zuordnung zwischen den Anfangsadressen der beiden Teile von empfangenen MAC-Rahmen sicherzustellen hat und entsprechende Interaktionen mit den Speicherverwaltungen durchführen muß.

5.4.3 Kommunikation zwischen den Programmen

5.4.3.1 Kommunikationsmittel

Die Kommunikation wird über WSn abgewickelt: Programme werden durch Kommandos in Eingangs-WSn angestoßen und schreiben in Ausgangs-WSn Kommandos an andere Programme.

Die Kommandos beziehen sich meist auf Vorgänge, die innerhalb eines Verbindungskontexts abzuwikkeln sind. Die Kennung einer Verbindung (connection ID) ist daher fester Bestandteil dieser Kommandos. Die Zuordnung einer Transportverbindung zu einer Kennung wird bei Verbindungsaufbau festgelegt. Sie ist sowohl innerhalb des TpS als auch dem AR bekannt. Nach erfolgtem Verbindungsaufbau erfolgt

ein Kommandoaustausch zwischen AR und TpS grundsätzlich unter Angabe der Verbindungskennung. Die Ermittlung der Verbindungskennung aus dem zugehörigen Steuerkopffeld (destination reference, DST-REF) ankommender PDUs ist in Abschnitt 5.4.6 erläutert.

Zwischen dem Sende- und dem Empfangsbereich werden außerdem Daten in einem gemeinsamen Speicher gehalten. Weitere gemeinsame Speicher ermöglichen den Austausch von Informationen zwischen AR und jedem der drei Bereiche des TpS. Diese Möglichkeit wird während der normalen Datentransferphase nur zwischen AR und IF-Bereich genutzt. Es werden zwei Formen von WSn gebraucht:

1. Die bisher ständig erwähnten, durch PCDs realisierten FIFO-WS koppeln die Programme unterschiedlicher Verarbeitungsbereiche sowie die drei TpS-Bereiche mit dem AR.
2. Zu lokalen (d.h. im jeweiligen Bereichsspeicher unterhaltenen) Verbindungs-WS (VWS) werden Verbindungen verkettet, die auf die Abwicklung einer bestimmten Funktion warten. Das Ein- und Austragen von Verbindungen in bzw. aus einer VWS erfolgt durch verschiedene Programme unter Beibehaltung der zeitlichen Abfolge (FIFO-Charakteristik). Folgende VWS sind möglich:
   - Die Wiederholungs-WS (WWS) im Sendebereich enthält alle Verbindungen, für die aufgrund des Ablaufens des Sendezeitgebers Daten-PDUs wiederholt werden müssen.
   - Die Einlese- bzw. Auslese-WS (EWS, AWS) im IF-Bereich enthalten die Verbindungen, die auf das Freiwerden der DMA-Einheiten für die Übertragung vom bzw. zum AR warten.
   - Eine dritte Art von VWS kann entstehen, falls ein Programm nach Abschluß eines Verarbeitungsschritts ein Kommando in eine noch belegte Ausgangs-WS schreiben will. In diesem - bei geeigneter Dimensionierung selten auftretenden - Fall wird das Kommando im Speicherbereich der bearbeiteten Verbindung zwischengespeichert und die Verbindung wird in eine VWS gestellt, die der belegten PCD-kontrollierten WS zugeordnet ist (PCD-VWS). Für jede PCD-VWS existiert ein Auffüllprogramm, das zyklisch überprüft, ob die PCD-VWS nicht leer und die zugehörige Kommando-WS nicht (mehr) voll ist. Ist beides gegeben, so wird das zwischengespeicherte Kommando der ersten Verbindung der PCD-VWS in die eigentlich geplante WS übertragen. Die Verbindung wird außerdem aus der PCD-VWS entfernt und für die Weiterverarbeitung durch andere Programme vorbereitet.

Programme, die Kommandos generieren, die sich auf einzelne Daten-PDUs beziehen, überprüfen den Zustand der Ausgangs-WS, bevor sie mit der Verarbeitung beginnen. Ist die Ziel-WS voll, wird die Verarbeitung nicht ausgeführt und die Eingangs-WS wird nicht verändert. Auf diese Weise wird die Entstehung von potentiell großen WSn vermieden, die sich auf PDUs beziehen.

Die restlichen Programme können die Entstehung der beschriebenen PCD-VWS bewirken. Der Grund für diese Entscheidung liegt darin, daß die Eingangs-WS Kommandos enthalten kann, deren Ausführung nicht immer die Benutzung derselben Ziel-WS erfordert. Ein nicht entferntes Kommando in der Eingangs-WS (aufgrund blockierter Ziel-WS) könnte damit die Ausführung der folgenden Kommandos derselben Eingangs-WS verhindern, obwohl sie u.U. vollständig ausführbar wären (head-of-line blocking). Zudem ist die Verkettung von Verbindungen mit geringem Aufwand realisierbar.

5.4.3.2 Verzahnung der Programme

Bild 5.23 stellt die zeitliche Verzahnung durch vorgegebene Programmabfolgezyklen sowie die logische Verzahnung durch Kommunikations-WSn für alle Programme dar. Es enthält folgende Elemente:

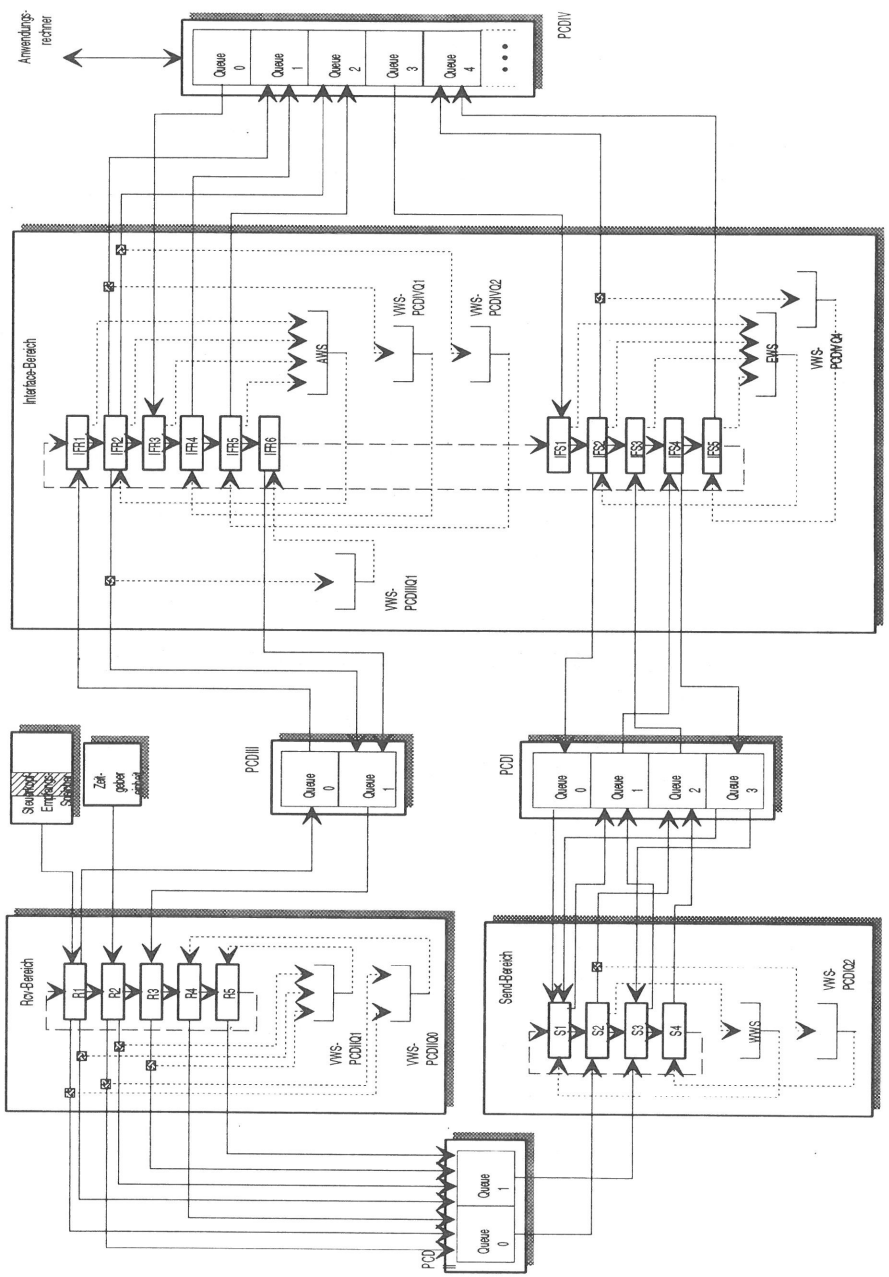


Bild 5.23: *Softwarestruktur des Transportsystems*

A) Programme innerhalb der Verarbeitungsbereiche (fett umrandet):
Zu den in Abschnitt 5.4.1 aufgeführten Funktionen R1-R3, IFR1-IFR3, IFS1-IFS4 und S1-S3 kommen noch die Auffüllprogramme für die verschiedenen WSn der einzelnen PCDs hinzu.

B) Durch PCDs realisierte WSn, die durch Pfeile mit zugehörigen schreibenden bzw. lesenden Programmen verbunden sind. Die markierten Abzweigungen symbolisieren den beschriebenen Überlaufmechanismus für den Fall voller Ziel-WSn.

C) Lokale VWS verbunden durch gestrichelte Pfeile mit den Programmen
- WWS, AWS, EWS sowie die PCD-VWS
(Bezeichnung: VWS-PCDxQy für WS y innerhalb des PCDs x)

D) Der Steuerkopfspeicher in Empfangsrichtung, welcher die Eingangs-WS für die Empfangsverarbeitung enthält

E) Die Zeitgebereinheit die als Quelle von Ereignissen logisch einer Eingangs-WS ähnelt.

5.4.3.3 Verbindungswarteschlangen

In die Wiederholungs-WS (WWS) im Sendebereich wird eine Verbindung gestellt, falls durch R2 ein Ablaufen des Sendezeitgebers festgestellt und über WS1 des PCDII dem Programm S2 angezeigt wurde. Die Verbindung wird durch S1 - bzw. durch den zugehörigen Wiederholungszweig - aus WWS entfernt, falls alle Daten-PDUs der Verbindung übertragen wurden oder alle Daten-PDUs, die zum Zeitpunkt des Ablaufens des Zeitgebers auf Quittierung warteten, quittiert wurden.

In der Einlese-WS (EWS) stehen alle Verbindungen, die vom AR Sendekommandos erhalten haben und über die Möglichkeit zum Senden verfügen (Sendekredit > 0). Zusätzlich darf keine Blockierung aufgrund einer vollen WS4 in PCDIV vorliegen, die ein Melden des Übertragungsendes eines Sendebereichs an den AR verhindern würde (und damit in VWS-PCDIVQ4 stehen würde).

In die EWS wird eine Verbindung daher durch das Programm eingetragen, das die letzte der 3 Bedingungen herstellt. Dies kann sein: IFS1 (Übernahme eines Sendekommandos), IFS3 (Aktualisieren des Sendekredits), IFS5 (Auffüllprogramm für PCDIVQ4). Außerdem sorgt das Einleseprogramm IFS2 dafür, daß eine Verbindung ans Ende der EWS gestellt wird, falls eine bestimmte Zahl von Datenteilen zu dieser Verbindung in Folge eingelesen wurden. Entfernt aus der EWS wird eine Verbindung durch IFS2, falls eine der 3 Bedingungen nicht länger zutrifft.

In der Auslese-WS (AWS) stehen alle Verbindungen, die (in Sequenz empfangene) Daten zum Auslesen bereithalten und über einen reservierten Empfangsbereich im Speicher des ARs verfügen. Konnte ein Kommando zur Anzeige einer Daten-TSDU in der entsprechenden WS2 des PCDIV nicht abgelegt werden, wird die Verbindung in die VWS (PCDIVQ2) aufgenommen.

In die AWS wird eine Verbindung eingetragen, wenn die letzte der Bedingungen erfüllt wird. Dies kann geschehen: durch IFR1 nach Kenntnisnahme empfangener Daten, durch IFR3 nach Entgegennahme eines Empfangsbereichs vom AR oder durch das Auffüllprogramm IFR5 nach Anzeigen einer ausgelesenen Daten-TSDU in der zuvor blockierten WS2 des PCDIV. Analog zum Einlesen sorgt das Ausleseprogramm IFR2 dafür, daß eine Verbindung nur für eine begrenzte Anzahl empfangener PDUs die zuständige DMA-Einheit beschäftigt, und daß sie ans Ende der AWS gestellt wird, falls diese erreicht ist. Entfernt wird eine Verbindung aus der AWS, falls eine der 3 Bedingungen verletzt wird.

Die Verwendung der PCD-VWS wurde schon in Abschnitt 5.4.3.1 erläutert.

5.4.4 Struktur der Verarbeitungsspeicher

Als Verarbeitungsspeicher (VSp) werden im folgenden die lokalen Speicher der drei Bereiche sowie der

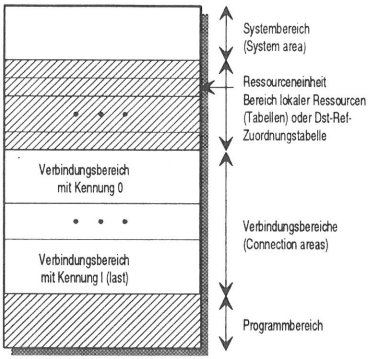


Bild 5.24: *Allgemeine Struktur der Verarbeitungsspeicher (VSp)*

gemeinsame Speicher zwischen Sende- und Empfangsbereich bezeichnet. Ein VSp ist in vier Bereiche strukturiert (siehe Bild 5.24):

1. ein Systembereich,
2. ein Bereich für lokale Ressourcen,
3. ein Verbindungsbereich und
4. der eigentliche Programmbereich

Die Inhalte der einzelnen Bereiche sind in den vier Speichern unterschiedlich. Einzelne Bereiche entfallen u.U. ganz. Die exakte Struktur aller vier Speicher ist in [127] beschrieben. Im folgenden werden typische Bestandteile der einzelnen Bereiche vorgestellt.

Für den Systembereich typische Informationen sind:

- Geräteadressen, u.a. der PCDs, der Zeitgebereinheit, der Sende- und Empfangssteuereinheiten oder der Prüfsummenbaugruppen
- Belegungsstatus der Geräte
- Information zur Verwaltung der VWS:
  - Anzahl der Verbindungen in der VWS
  - Zeiger auf die erste Verbindung bzw. auf deren Verbindungsbereich in der VWS
  - Zeiger auf letzte Verbindung in der VWS
- Information zur Verwaltung der lokalen Ressourcen (Tabellen)
  - Anzahl freier Tabellen
  - Zeiger auf erste freie Tabelle
- Zeiger auf Beginn der Verbindungsbereiche (PtConnAreas)
- Zeiger zur Manipulation eines Steuerkopfes (PtHeader).

Der Bereich der *lokalen Ressourcen* ist für den IF- bzw. den Sendebereich in Tabellenform vorgesehen. Im IF-Bereich nimmt eine Tabelle die Information über einen zu sendenden Datenbereich im Speicher des AR auf. Im Sendebereich werden die Steuerkopfinformationen einer gesendeten Daten-PDU in einer PDU-Tabelle aufgenommen. Im Empfangsbereich steht bei den lokalen Ressourcen die Tabelle der Zielreferenzen, die der Zuordnung ankommender PDUs zu Verbindungen dient (siehe Abschnitt 5.4.6).

Im Verbindungsbereich sind die Informationen zu jeder Verbindung enthalten, die zur Abwicklung des Transportprotokolls für die betreffende Verbindung nötig sind. Sie sind je nach Bereich sehr unterschiedlich [127]. Um einen möglichst schnellen Zugriff auf die Daten einer speziellen Verbindung zu ermöglichen, werden grundsätzlich Einträge einheitlicher Größe für alle 256 möglichen Verbindungen geführt.

Zur Verwirklichung der Verwaltung der VWS enthalten die Verbindungsbereiche zu jeder vorkommenden VWS einen Zeiger, der im Bedarfsfall auf den nächsten Verbindungsbereich zeigt, der in der VWS steht. Auf diese Weise kann eine Verbindung gleichzeitig in allen möglichen VWS eingegliedert sein. Das Verwalten der VWS beschränkt sich auf das Umtragen dieser Zeiger.

Im IF-Bereich enthält jeder Verbindungsbereich die nötige Information zur Verwaltung der eigenen (verbindungsbezogenen) Kette von Sendebereichen. Analog enthält jeder Verbindungsbereich im Sendebereich die Information zur Verwaltung der eigenen PDU-Tabellenkette.

Zum Schluß folgt der eigentliche Programmbereich für Programme und Daten.

5.4.5 Betriebsmittel und deren Verwaltung

5.4.5.1 Prinzipielle Arten von Betriebsmittel

Als Betriebsmittel (BM) werden die Ressourcen bezeichnet, die im Zusammenhang mit der externen Kommunikation belegt werden. Die zur internen Kopplung verwendeten WSn zählen nicht dazu.

Drei Klassen von BM werden unterschieden:

A) BM, die pro aktiver Verbindung einmal vorhanden sind. Diese werden bei Verbindungsaufbau gemeinsam belegt und bei Verbindungsabbau gemeinsam freigegeben.
Eine Gruppe von BM ist durch Angabe der Verbindungskennung eindeutig adressierbar (siehe Abschnitt 5.4.6). Die Zuordnung einer Verbindungskennung zu einer Transportverbindung stellt daher zugleich die Belegung und Zuordnung der entsprechenden BM dar. Zu diesen gehören (A1):

- die Zeitgeber einer Verbindung
- die Verbindungsbereiche in den Verarbeitungsspeichern

Zusätzlich wird bei Verbindungsaufbau eine Zahl von Empfangs-EPn der Verbindung exklusiv zugeordnet. Garantiert wird die Anzahl der EP, nicht aber eine feste Lokation innerhalb des Empfangsdatenspeichers. Die Anzahl hat sowohl die Größe des Empfangsfensters, als auch die mögliche Verzögerung der Datenübernahme durch den AR wiederzuspiegeln. Das Ableiten des Kreditwerts aus der Zahl exklusiv zugeordneter und freier EP (vergleiche Abschnitt 5.1.2.4) schließt PDU-Verluste aufgrund eines Überlaufs des Empfangsdatenspeichers zwar nicht aus, macht sie jedoch unwahrscheinlich (A2).

B) BM, die innerhalb jeder Verbindung pro Daten-PDU belegt werden:

- EP-Kette im Sendedatenspeicher für ein zu sendendes Datenteil (B1).
- EP-Kette im Empfangsdatenspeicher für eine empfangene PDU (B2).
- PDU-Tabelle im Sendebereich zur Zwischenspeicherung der Steuerkopfinformation einer gesendeten Daten-PDU (B3). Platz im Steuerkopfspeicher für zu sendende bzw. empfangene Steuerköpfe (B4).

C) BM, die durch den AR bereitgestellt werden.

- Datenbereiche im Speicher des ARs, die gesendet werden sollen.
- Die Verwaltungsinformation für jeden Sendebereich (Adresse, Größe) wird in Form eines (PCD-)Kommandos übergeben und von der angesprochenen Verbindung verwaltet (C1).
- Datenbereiche im Speicher des ARs, in die empfangene Daten geschrieben werden (Empfangsbereiche).
- Ein Empfangsbereich wird von einer Verbindung angefordert, falls kein reservierter Empfangsbereich mehr vorhanden ist und Daten zum Auslesen an den AR bereitstehen. Der AR übergibt in Form eines (PCD-) Kommandos die Information über einen reservierten Empfangsbereich, die zu deren Verwaltung nötig ist. Diese Information wird im entsprechenden Verbindungsbereich gespeichert (C2).

5.4.5.2 Verwaltung der Betriebsmittel

Verwaltung der Verbindungskennungen (A1)

Für das TpS wurde davon ausgegangen, daß *zu* keinem Zeitpunkt mehr als 256 Verbindungen unterschieden werden müssen. Für die Verbindungskennungen wurden daher 8-bit-Werte verwendet. Ihre Belegung bzw. Freigabe betrifft die Verbindungsauf- bzw. -abbauphase und kann sowohl vom AR als auch durch den Empfangsbereich des TpS initiiert werden. Der natürliche Ort für die Verwaltungsinformation der freien Verbindungskennungen ist daher der gemeinsame Speicher zwischen AR und Empfangsbereich. Darin sind die freien Kennungen als Kette abgelegt. Das Fehlen freier Kennungen wird anhand deren Anzahl festgestellt, die ebenfalls im gleichen Speicher abgelegt ist. Belegt wird eine Kennung durch die Entfernung der ersten Kennung aus der Kette und das Dekrementieren der Anzahl freier Kennungen. Die Freigabe einer Kennung erfolgt analog. Da die Operationen nicht atomar sind, ist ein Mechanismus zum gegenseitigen Ausschluß der Zugriffe von AR und Empfangsbereich vorzusehen.

Reservierung von Empfangselementarpuffern (A2)

Die Zahl der nicht zugeordneten EP wird im lokalen IF-Speicher gespeichert. Wird für eine Verbindung bei ihrem Aufbau eine Zahl von EPn gefordert, die nicht vorhanden ist, ist die Herabsetzung der Dienstgüte bzw. das Nichtzustandekommen der Verbindung dem Initiator des Aufbauvorganges (AR bzw. entfernter Netzknoten) zu signalisieren. Kommt eine Verbindung zustande, wird die Zahl der noch nicht zugeordneten EP entsprechend verringert. Bei Verbindungsabbau wird sie um denselben Wert erhöht.

Verwaltung der Sendeelementarpuffer (B1)

Die Anzahl freier Sende-EP wird im lokalen Speicher des IF-Bereichs festgehalten. Datenteile von TPDUs werden nur eingelesen, falls freie Sende-EP vorhanden sind. Ist dies der Fall, wird die Belegung der EP dynamisch durch die Kreditwerte der Verbindungen bzw. durch die Einlesestrategie gesteuert.

Zu jeder gesendeten Daten-PDU werden die Anzahl der belegten EP sowie die Adresse des ersten bzw. letzten EPs in einer PDU-Tabelle im Sendebereich festgehalten. Die EP-Ketten aller gesendeten TPDUs werden zu *einer* Kette verbunden, so daß bei Eintreffen einer Quittung diese nicht einzeln freigegeben werden müssen. Vielmehr werden die Adresse des ersten freizugebenden EPs (aus der Tabelle der ältesten quittierten TPDU) und die Adresse des letzten freizugebenden EPs (aus der Tabelle der jüngsten quittierten TPDU) ermittelt. Die hierdurch definierte EP-Kette wird danach gleichzeitig freigegeben.

Die implementierte Speicherverwaltung unterstützt das Verketten von EPs jeweils einer TPDU in Hardware. Das Verketten mehrerer TPDUs ist indirekt über die Übergabe von entsprechenden Listen möglich.

Verwaltung der Empfangselementarpuffer (B2)

Die Basisoperationen wurden schon bei der Beschreibung der Hardware (Abschnitt 5.3.3) vorgestellt. Die Belegung der EP für ankommende Daten-PDUs erfolgt durch die DMA-Einheit MACreceive. Der Empfangsbereich kann auf drei Arten auf den Empfang einer DT-TPDU reagieren (siehe Abschnitt 5.2.1.4):

- der Datenteil der TPDU ist mit der EP-Kette zu verbinden, in der die Datenteile zuvor in richtiger Sequenz empfangener TPDUs enthalten sind (1. Sequenz von Datenteilen),
- die TPDU ist zu verwerfen, wenn sie zu dem reduzierten Fenster gehört,
- der Datenteil der TPDU ist mit der EP-Kette zu verbinden, in der die Datenteile der TPDUs enthalten sind, die nach einer Empfangslücke in Sequenz empfangen wurden.

Die Verkettung wird mit der letzten TPDU einer TSDU beendet, die Information über die Abgrenzung einzelner TSDUs bleibt bewahrt. Das Auslesen an den AR wird jeweils nach einem EP mit logischem Nullnachfolger unterbrochen. Eine solche Unterbrechung signalisiert dem IF-Bereich die Notwendigkeit, eine Meldung über eine ausgelesene TSDU in FIFO 2 des PCD IV abzulegen.

Die Größe beider Sequenzen wird im IF-Bereich für jede Verbindung gespeichert, so daß zu jedem Zeitpunkt bekannt ist, ob Daten an den AR ausgelesen werden können. Der Zugriff auf beide Sequenzen wird durch die Speicherung der Adresse des jeweils ersten EPs in jeder Sequenz ermöglicht. Zum Auslesen von Daten aus EPn an den AR wird der Befehlstyp "Lesen mit Austragen" verwendet. Die entsprechenden EP werden unmittelbar in die Freiliste übertragen.

Verwaltung der PDU-Tabellen im AR (B3)

Zu jeder gesendeten PDU wird im Sendebereich eine Tabelle angelegt. Sie enthält die Information über belegte Sende-EP sowie die Information, die bei einer eventuellen Wiederholung (Steuerkopf, Zahl der Wiederholungen, Ablaufzeitwert) benötigt wird. Jede Verbindung unterhält eine lokale Kette von Tabellen. Sie entspricht den PDUs, die gesendet, aber noch nicht quittiert wurden.

Freie Tabellen werden in einer Kette freier Tabellen erfaßt, nach Bedarf entfernt und in eine verbindungsbezogene Tabellenkette gestellt. Das Quittieren von Daten-PDUs bewirkt den umgekehrten Vorgang.

Verwaltung der zu sendenden Speicherbereiche (C1)

Zu sendende Bereiche werden im IF-Bereich vom AR übernommen. Die zum Einlesen aus diesen Bereichen nötige Information - jeweils Paare aus Anfangsadresse und Bereichsgröße - wird in einer zugeordneten Tabelle abgelegt. Jede Verbindung unterhält eine lokale Kette von Tabelleneinträgen, die den noch zu sendenden Speicherbereichen entspricht.

Das Belegen bzw. Freigeben der Tabellen entspricht den Vorgängen in B3 mit dem Unterschied, daß der Abschluß des Einlesens aus einem Speicherbereich dem AR durch ein Kommando (Anfangsadresse/ Bereichsgröße) angezeigt wird.

Verwaltung der Speicherbereiche für den Empfang (C2)

Da pro Verbindung nur ein Empfangsbereich im Speicher des ARs vorgesehen ist, beschränkt sich die Verwaltung auf Kommandos, die das Anfordern bzw. die Übergabe eines Empfangsbereichs anzeigen.

5.4.6 Herstellung des Verbindungskontexts

Mit Ausnahme der ersten Verarbeitung in Empfangsrichtung wird den Programmen der Kontext entweder durch die in PCD-Kommandos und Statusworten der Zeitgebereinheit enthaltene Kennung oder direkt durch einen Zeiger BA (Basisadresse) auf den zugehörigen Verbindungsbereich angezeigt.

Die Umrechnung einer Verbindungskennung in eine physikalische BA wird durch einfaches Verschieben der Verbindungskennung und Addition dieses Werts zur Anfangsadresse des Verbindungsbereichs (PtConnectionAreas) ermittelt. Dies setzt voraus, daß für alle möglichen 256 Verbindungen Einträge vorhanden sind und daß deren Größe durch eine 2-er Potenz gegeben ist.

Ist eine direkte Verwendung der 8 Bit Kennung als eines der beiden Oktetts des Zielreferenzfeldes nicht möglich, ist eine Umsetzungstabelle vorzusehen. Um sich hierdurch keine Verzögerungen einzuhandeln (siehe Untersuchungen in Abschnitt 3.3.3.5), ist Hardwareunterstützung unabdingbar.

Die Zielreferenz in TP4 umfaßt, wie gesagt, 16 Bit. Damit gibt es 216, das sind 65.536 unterscheidbare Werte. Zu maximal 256 dieser Werte gibt es eine 8 Bit lange Verbindungskennung. Vom logischen Standpunkt her, ist diese Anwendung eine klassische Aufgabe für inhaltsadressierbare oder auch Assoziativspeicher genannte Bausteine. Allerdings sind diese CAMs, wie schon bei den Realisierungsvarianten der Zeitgeber in Abschnitt 5.3.3.1 angesprochen, schwieriger zu behandeln als herkömmliche RAMs und auch erheblich teurer als diese.

Trotz einer maximalen Nutzung von 256/65.536 gleich 0.39% ist daher eine Tabelle mit allen 65.536 möglichen Zielreferenzen, die mittels eines einzigen 64K X 8 RAM-Bausteins realisiert werden kann, vorzuziehen! Eine beliebige der 256 möglichen Verbindungskennungen muß allerdings ausschließlich für

die Kennzeichnung unbenutzter Zielreferenzen zur Verfügung gestellt werden. Werden zwei dieser Bausteine verwendet (⇒ 64K•16), kann zusammen mit der Kennung auch weitere verbindungsbezogene Information, wie z.B. der Status der Verbindung, durch nur eine einzige Speicherabfrage erhalten werden.

5.4.7 Programmsequenzen

5.4.7.1 Senden von Daten

Das Senden von Daten leitet der AR durch die Übergabe eines oder mehrerer zu versendender Speicherbereiche ein. Das Programm IFS1 legt die Information in einer Tabelle ab und stellt die Verbindung in die EWS, sofern sie nicht schon darin stand, und sofern die nötigen Bedingungen erfüllt sind.

IFS2 veranlaßt das Einlesen jeweils eines Datenteils in den Sendedatenspeicher des TpS. Die gleichzeitige Berechnung der Original-TP4-Prüfsumme ist, wie in Abschnitt 5.3.3.3 detailliert erläutert, nur dann möglich, wenn entweder ausnahmslos Steuerköpfe exakt gleicher (bekannter) Länge möglich sind oder dem Prüfsummenbaustein wahlweise die Länge des aktuellen Steuerkopfes oder die vorausschauend berechneten Prüfoktetts des Schicht-4-Steuerkopfes zur Verfügung stehen. Die beiden letztgenannten Varianten erfordern eine Interaktion mit dem Sendebereich bevor das Kopieren der Daten aus dem Speicher des ARs in den Sendedatenspeicher des TpS beginnen darf! Ein möglichst frühes Informieren des Sendeprozessors ist auch deshalb nötig, um die vorbereiteten Steuerkopfschablonen, die sich noch in einem Verarbeitungsspeicher befinden, frühzeitig in den Steuerkopfspeicher kopieren zu können. Mit der abgewandelten (standardkonformen) Berechnungsvorschrift und den Überlegungen aus Abschnitt 5.3.3.3 entfällt jegliche Interaktion vor dem Datentransfer! Der Transfer kann also beliebig starten.

Immer dann, wenn 1024 Oktetts aus dem Speicher des ARs in den Sendespeicher des TpS übertragen worden sind, übergibt IFS2 ein Kommando zur Fertigstellung des zugehörigen Steuerkopfes an den Sendebereich, welches neben der obligatorischen Verbindungskennung lediglich die Information zum Berechnen der endgültigen Prüfoktetts dieser TPDU enthält. Außerdem wird der Datenteil mit der Kette zuvor eingelesener Datenteile derselben Verbindung verkettet. Im Fall, daß ein Sendebereich, d.h. eine TSDU, vollständig eingelesen wurde, wird dies dem AR angezeigt.

Im Sendebereich stellt S1A den Steuerkopf der TPDU zusammen. Dazu müssen die zwei Prüfoktetts zunächst aus der übergebenen Information und der vorausbestimmten Zwischenprüfsumme des Steuerkopfes berechnet werden, und anschließend an ihre Positionen im Steuerkopf eingetragen werden. Nach Erledigung wird das Kommando zur Übertragung an den IF-Bereich generiert, die nächste Steuerkopfschablone der gleichen Verbindung vorbereitet, und die Verbindungsinformation entsprechend aktualisiert.

IFS4 im IF-Bereich startet die Übertragung der Daten-PDU und zeigt dem Sendebereich die Beendigung an, damit der Steuerkopfspeicher durch Weiterschalten des Schreibzeigers freigegeben werden kann.

5.4.7.2 Senden einer Quittung

Außer durch R1A (siehe oben) wird das Senden einer Quittung durch R2 nach Feststellen des Ablaufes des LAT-Zeitgebers durch ein Kommando an den Sendebereich ausgelöst.

Hierbei stellt S3 die Quittung im Steuerkopfspeicher zusammen und übergibt das Kommando zur Übertragung an den IF-Bereich. Das Starten der Übertragung durch IFS4 verläuft wie beim Senden von Daten.

5.4.7.3 Wiederholung von Dateneinheiten

Im Empfangsbereich stellt R2 das Ablaufen eines Zeitgebers fest und zeigt dies dem Sendebereich an. S2A stellt daraufhin die zugehörige Verbindung in die WWS. Solange die WWS nicht leer ist, werden nur Daten-PDUs im Wiederholungsmodus gesendet. S1B stellt für jede unquittierte Daten-PDU den Steuerkopf aus der entsprechenden PDU-Tabelle im Steuerkopfspeicher zusammen und legt das Kommando zur Übertragung an den IF-Bereich ab. Der Sendezeitgeber wird nach Senden der ältesten Daten-

PDU erneut gestartet. Wurden alle Daten-PDUs wiederholt oder zwischenzeitlich quittiert, wird die Verbindung aus der WWS entfernt. Im IF-Bereich wird die Übertragung wie im Normalmodus ausgeführt.

5.4.7.4 Empfang von Dateneinheiten

R1A überprüft, ob die Daten-PDU gültig ist (Prüfsumme, Länge, Format, Duplizierung). Bezüglich der Empfangssequenz sind vier Fälle zu unterscheiden:

- Es ist zuvor keine Empfangslücke aufgetreten und die Daten-PDU ist in Sequenz angekommen.
  An den IF-Bereich ergeht das Kommando, das Datenteil mit der Kette der restlichen Empfangs-EP zu verbinden, die Daten derselben Verbindung enthalten. Der Empfang wird normal quittiert, d.h. eine Quittung wird gesendet, falls eine bestimmte Zahl von Daten-PDUs empfangen wurde. Der hierbei verwendete Kreditwert wird aus der Zahl der nichtbelegten, der Verbindung zugeordneten EP berechnet. Der Empfang der ersten Daten-PDU nach Senden einer Quittung bewirkt das Starten des lokalen Quittierungszeitgebers LAT (Local Acknowledgement Timer).
- Die Verbindung war reihenfolgekorrekt , aber die aktuelle Daten-PDU ist nicht in Sequenz.
  Die Verbindung wird als nicht in Sequenz markiert. Das sofortige Senden einer Quittung wird durch ein Kommando an den Sendebereich veranlaßt. Die Reduzierung des Empfangsfensters wird vorgenommen (siehe Abschnitt 5.2.1.4 und 4.1.1.2). Ist die empfangene PDU innerhalb des reduzierten Fensters, wird es verworfen. An den IF-Bereich wird ein Kommando gerichtet.
- Die Verbindung ist nicht in Sequenz, die TPDU ist in Sequenz.
  Wurde dadurch die Empfangslücke geschlossen, wird die Verbindung als in Sequenz markiert und das Senden einer Quittung durch den Sendebereich veranlaßt. Der Kreditwert berechnet sich dabei wieder wie im ersten Fall. Dem IF-Bereich wird im Kommando neben der Information über das zu verkettende Datenteil das Schließen der Lücke mitgeteilt.
- Weder die Verbindung noch die TPDU ist in Sequenz.
  Paßt es zur 2. Empfangssequenz, wird dies der IF-Seite angezeigt. Ansonsten wird das Datenteil wie im 2. Fall freigegeben.

Im IF-Bereich führt IFR1 die Verkettung bzw. Freigabe der Empfangs-EP durch. Sind EP in erster Sequenz vorhanden, und ist ein Empfangsbereich beim AR reserviert, wird die Verbindung in die AWS gestellt. Ist kein Empfangsbereich reserviert, wird einer angefordert.

IFR2 liest die Daten an den AR aus und entfernt u.U. die Verbindung aus der AWS. Das Freiwerden der EP wird dem Empfangsbereich angezeigt, sobald ein Schwellwert überschritten ist.

Im Empfangsbereich wird diese Anzeige durch R3 verarbeitet. Eventuell wird das Senden einer Quittung mit vergrößertem Empfangsfenster veranlaßt.

5.4.7.5 Empfang von Quittungen

R1B stellt zunächst sicher, daß die empfangene Quittung gültig ist (Prüfsumme, Länge, Format, Folgenummer). Die Datenflußparameter werden dem Sendebereich im gemeinsamen Speicher bereitgestellt. Der Erhalt der Quittung wird dem Sendebereich durch ein Kommando angezeigt.

S2B verarbeitet das Kommando, stellt fest, welche Tabellenkette freizugeben ist und leitet daraus die Kette freizugebender EPs ab. Dies und die Veränderung des Sendekreditwertes werden dem IF-Bereich zum Aktualisieren mitgeteilt.

Im IF-Bereich gibt IFS3 die Sende-EP frei und verändert gegebenenfalls den Wert des Kreditfeldes. Infolgedessen wird u.U. die zugehörige Verbindung in die EWS gestellt (falls das Nichtvorhandensein von Krediten dies vorher verhinderte und nun neue Kredite gewährt wurden) oder aus der EWS entfernt (falls der neue Kreditwert weiteres Senden unmöglich macht).

5.4.8 Leistungsfähigkeit der Software

Für die Bewertung wurden folgende Annahmen gemacht:

- als CPU wird der Prozessor 80960CA von Intel eingesetzt [400]:
  - elementare Operationen erfordern einen CPU-Zyklus
  - Verzweigungen erfordern zwei Zyklen
  - Speicherzugriffe mit indizierter Adressierung erfordern vier Zyklen
  - das Laden der Befehle wird überlappt durchgeführt und ist aufgrund des CPU-internen schnellen Zwischenspeichers (cache) selten nötig
  - indizierte Zugriffe auf Geräteregister (Sende- und Empfangssteuerung, Zeitgebereinheit) erfordern in der Regel vier Zyklen (asynchrone Zugriffe)
  - indizierte Zugriffe auf PCDs erfordern sowohl zum Schreiben als auch zum Lesen sechs Zyklen; zum Stempeln werden acht Zyklen benötigt.

Unter diesen Annahmen wurde für jedes Programm die Dauer in Form von 3 Werten ermittelt:

- minimal: nur der Abfrageteil wird ausgeführt
- regulär: der wahrscheinlichste Pfad wird betrachtet
- maximal: der längste mögliche Pfad wird zugrundegelegt.

Die resultierenden Summenwerte für jeden Prozessor sind in Tabelle 3 zusammengefaßt.

Tabelle 5.3: *Notwendige Gesamtzyklenzahl der 3 TpS-Prozessoren (minimal, regulär, maximal)*

Empfangsbereich	79	767	1442
Sendebereich	57	1453	1562
IF-Bereich (senden)	38	367	789
IF-Bereich (empf.)	49	356	790
IF-Bereich (Summe)	87	723	1579

Für eine idealisierte Durchsatzschätzung wurden folgende Annahmen gemacht:

- Sende- und Empfangsverkehr sind im Gleichgewicht
- im Schnitt wird pro fünf gesendeten Daten-PDUs eine Quittung empfangen und umgekehrt
- wird nach 5 empfangenen Daten-PDUs jeweils eine Quittung abgesetzt (im Mittel)
- PDU-Wiederholungen sind nicht erforderlich
- Empfangslücken entstehen nicht
- PCD-VWS entstehen selten und sind vernachlässigbar
- 'Echte' Kommunikation mit dem AR ist selten (große Sende- bzw. Empfangsbereiche)
- Zyklusdauer: 0,03 µs bei 33 MHz Taktrate

Unter diesen Annahmen ist der Durchsatz von Daten-PDU-Paaren (je eines in Sende- und Empfangsrichtung) durch die Last von cá. 800 Zyklen (24 µs) im IF-Bereich begrenzt, was einem Durchsatz von über 40.000 PDUs pro Sekunde und Richtung entspricht!

Mit der generellen Annahme der maximal möglichen Pfadlängen der einzelnen Programme halbiert sich dieser Wert. Bei Daten-PDUs von 1 KB Länge entspricht dieser Wert einer Transportrate von ca. 170 Mbit/s pro Richtung! Bei TPDUs von einheitlich 8 KB Länge wäre das TpS somit - rein rechnerisch - in der Lage über 2.7 Gbit/s zu verarbeiten, allerdings ist dies ein eher unrealistisches Szenario, und Speicher, Busse sowie die Interaktion mit dem AR würden diese Raten nicht zulassen.

Bei Einsatz der billigeren 25 MHz CPU-Variante vermindern sich alle Werte linear um den Faktor 25/33.

Kapitel 6

Entwurfsablauf und Verifikation

Eine Besonderheit dieser Arbeit ist die Kombination einer großen Zahl von verschiedenen Einzelaspekten, von denen sich viele isoliert betrachtet auf einfache Weise - zumindest qualitativ - bewerten lassen. Aussagen zum Gesamtverhalten des TpS lassen sich allerdings nicht durch bloße Kombination der Einzelergebnisse ableiten, sondern erfordern komplexere Methoden. Vor einer Überprüfung des Zeitverhaltens einzelner Baugruppen steht dabei zunächst der Nachweis der (hardwarenahen) Funktionalität der Komponenten und deren Zusammenwirken im Vordergrund des Interesses.
Da formale Methoden zur Überprüfung der funktionellen Korrektheit nur auf einfache Systeme angewandt werden können, ist für die Validierung des gesamten TpS eine weitergehende Abstraktion erforderlich. Diese ist so ausgelegt, daß zusätzlich detailliertere Aussagen zu Dimensionierung und Leistungsfähigkeit des TpS ermöglicht werden. Beide Formen der Verifikation und einige Ergebnisse werden im 2. Teil dieses Kapitels beschrieben.

Dem Nachweis der Realisierbarkeit des TpS unter den Randbedingungen eines Hochschulinstituts kam eine zentrale Bedeutung innerhalb dieser Arbeit zu. Eine Übersicht der Entwurfsumgebung und des gewählten Entwurfablaufs rechtfertigt sich jedoch nicht allein aus dem erheblichen Zeitaufwand, der für die Implementierung der Soft- und Hardwarekomponenten benötigt wurde, sondern verdeutlicht auch die Problematik der Koordinierung eines Projekts mit mehreren Beteiligten. Darüber hinaus erwies sich die oft getroffene Annahme der einfachen und schnellen Umsetzung einer genauen Modellvorstellung in eine Implementierung im Sinne eines klassischen strukturierten Top-Down-Entwurfs (erwartungsgemäß) als Trugschluß. Dies wirkte sich auch auf den Entwurfsablauf aus.

6.1 Entwurfsablauf

Bild 6.2 zeigt den idealisierten Entwurfsablauf ohne jegliche Iterationen: wie gewöhnlich machte der erste Schritt von den informellen Ideen und Vorstellungen hin zu ersten groben Festlegungen der Systemspezifikation einen Großteil der Arbeit aus, was auch in den vorigen Kapiteln deutlich zum Ausdruck kamm. Neben der umfassenden Literaturstudie und allgemeinen Vorgaben (siehe 5.1) beeinflußten vorab ausgeführte "Implementierungen" der wesentlichen logischen (Protokol-)Funktionen sowohl in der Hochsprache C als auch auf Maschinebene (generischer RISC-Assembler) die grundsätzlichen Architekturentscheidungen. Im Bild 6.2 fallen folgende Punkte auf:

1. Keine frühzeitige Konzentration auf Modellierung und Verifikation
   (wie sie für einen echten Top-Down-Entwurf typisch wäre)
2. Weitgehende Trennung *von* Hard- und Softwareentwurf
3. Keine Unterteilung des eigentlichen physikalischen (Hardware-)Entwurfs.

Punkt 1 entsprang ursprünglich aus der Notwendigkeit heraus, möglichst frühzeitig unabhängige Teilprojekte zu definieren. Es zeigte sich jedoch schnell, daß sich aus der zunehmenden Kenntnis der Details mehrfach entscheidende Rückwirkungen auf Systemaspekte ergaben, die das Einführen der Bottom-Up-Vorgehensweise im nachhinein als unumgänglich einstufen.

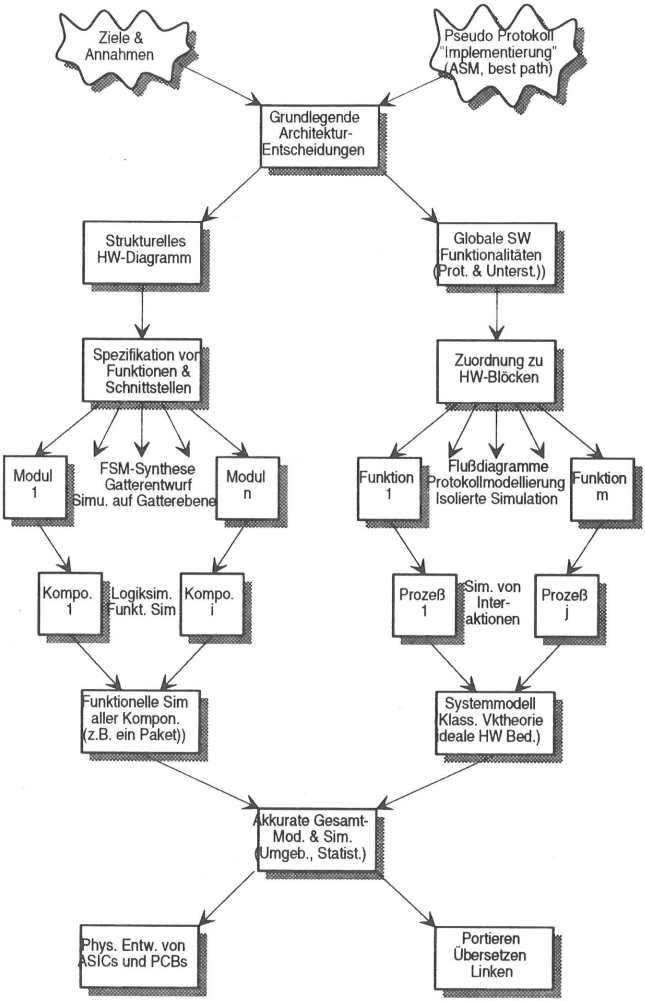


Bild 6.1: *Entwurfsablauf des TpS (Rückwirkungen vernachlässigt)*

Der 2. Punkt war nicht zu vermeiden und trägt gleichzeitig der traditionellen Sichtweise Rechnung, welche sich nach wie vor in streng getrennten Entwicklungswerkzeugen und -methoden ausdrückt. Der Verlauf des Projekts legt allerdings eine möglichst enge Verflechtung zwischen den zwei Bereichen während des gesamten Entwurfs nahe.

Nachdem einige Hardwarekomponenten - u.a. eine komplette FIFO-Speichereinheit, Multiplizierer, mehrere Steuerwerke (FSMs) sowie schnelle synchrone Zähler - als Vollkunden-ICs entworfen und gefertigt wurden, stellte sich heraus, daß der Vorteil der optimierten Leistungsfähigkeit und Chipfläche keinesfalls die Nachteile des hohen Entwicklungsaufwands und der höheren Kosten aufwog. Als Folge davon wurde im weiteren Verlauf auf das Semicustom-Entwurfsverfahren des IMS' (Institut für Mikroelektronik Stuttgart) zurückgegriffen [65, 66]. Für die dort vorhandenen, bis auf wenige Schichten vorgefertigten,

Master-Siliziumscheiben stehen sowohl vollständig verifizierte Gatterbibliotheken als auch leistungsfähige automatische Plazierungs- und Verdrahtungswerkzeuge zur Verfügung. Ist man weder auf jede Nanosekunde noch auf extreme Schaltungsintegrationsdichte angewiesen, kann der Datenaustausch zum IMS auf der Ebene von logischen Netzlisten erfolgen. Da analoges auch für die physikalische Fertigung von Platinen gilt, kann der tatsächlich relativ aufwendige Verlauf des physikalischen Entwurfs daher stark vereinfacht wiedergegeben werden (Punkt 3).

Von entscheidender Bedeutung für das Gelingen eines aus vielen Komponenten zusammengesetzten Systems sind frühzeitige Festlegungen der äußeren Schnittstellen jeder einzelnen Teilbaugruppe. Gerade dabei treten in der Realität Rückwirkungen auf: erst mit genauerer Kenntnis von Implementierungsdetails können die Schnittstellen exakt beschrieben werden. Insgesamt verlief der Entwurf sowohl bei Hard- als auch bei Software zeitweise mehr iterativ als streng dem idealen Entwurfsdiagramm folgend.

Zunächst wird der Entwurfsablauf von Hardwarekomponenten betrachtet. Um einheitliche, unzweideutige und damit nachvollziehbare Teilentwürfe sicherzustellen, wurden von Beginn an rechnerunterstützte Entwurfsverfahren eingesetzt. Stark erschwert wurde dieses ideale Vorgehen allerdings durch die allgegenwertigen Unzulänglichkeiten einzelner Werkzeuge (fehlende oder falsche Bibliothekselemente, keine oder falsche Prä- und Postprozesse für korrekte Datentransfers zwischen verschiedenen Werkzeugen etc.) sowie spärlich vorhandene Kompatibilität zwischen den verschiedenen Entwurfswerkzeugen und/oder deren Umgebungen.

Nach der graphischen Eingabe der Strukturdiagramme (PC-Cadstar von Racal-Redac [407]) wurde prinzipiell eine Auftrennung in Operations- und Steuerwerk vorgenommen. Die Operationswerke wurden schrittweise bis auf Gatterebene verfeinert, für Entwurf und Synthese der Steuerwerke konnte teilweise das Programmpaket LOG/IC - Gates (ISDATA, Karlsruhe [410]) eingesetzt werden. Dessen Qualitäten liegen allerdings bei der zweistufigen UND-ODER Synthese, wie sie z.B. in vielen programmierbaren Logikbausteinen (Programmable Logic Devices, PLDs) vorkommt. Die hier benötigte Umwandlung in mehrstufige ASIC-gerechte Gatterentwürfe, die Handhabung komplexerer Automaten und deren Partitionierung lieferte dagegen nur suboptimale Lösungen.
Auf die verschiedenen Formen der simulativen Verifikation der Hardwaremodule und Teilkomponenten wird im nächsten Abschnitt eingegangen.

Die Softwareentwicklung konnte sich zunächst an den vorhandenen (Pseudo-)Assemblerimplementierungen der reinen logischen Funktionalität orientieren. Durch den vollständigen Verzicht auf ein Betriebssystem und die Notwendigkeit mit eigenentwickelten Hardwarekomponenten zu interagieren erweiterte sich der Funktionsumfang der Software schnell.

Tabelle 6.1: *Ausschnitt aus einem auf Maschinenebene programmierten Modul*

```
put command 'SendPDU' in PCD I queue 1 to IF-processor
   MOV   #0..011, CmndWd
   SHRO  #2, PtHeader, HeaderAdr
;calculate time for retransmission, if unacknowledged
;load current time
MACRO:   Load TimeNow from timer
;load time-out value
   LD    (BA+i), SendTimeOut
   ADDO  TimeNow, SendTimeOut
   CLRBIT#16, TargetTimeOut, TargetTimeOut
;send timer active?
   LD    (BA+i), NoPakTab
   CMPO  NoPakTab, #0
   BG    LabelS3
;start SendTimer
MACRO: Start SendTimer
LabelS3: ...
```

Die Programmierung auf Maschinenebene (siehe Tabelle 6.1) induzierte - wieder Erwarten - keine alzu großen Probleme: durch die konsequente Nutzung von Makros, die funktionell den Elementen einer Gatterbibliothek beim Hardwareentwurf entsprechen, und die relativ geringe Komplexität jeder einzelnen Funktion, war der Mehraufwand im Vergleich zu einer Hochsprachenprogrammierung nicht signifikant. Auch die Änderungsfreundlichkeit unterscheidet sich durch die erwähnte konsequente Anwendung von Makros und die handhabbare Komplexität nicht wesentlich von der mit Hochsprachen erzielbaren.

Zur Spezifikation und Dokumentation der einzelnen Funktionen und ihres Zusammenwirkens wurden Ablauf- und Flußdiagramme (Bild 6.2) eingesetzt. Die Umsetzung in Code mußte allerdings in herkömmlicher fehleranfälliger Weise ohne Rechnerunterstützung durchgeführt werden.

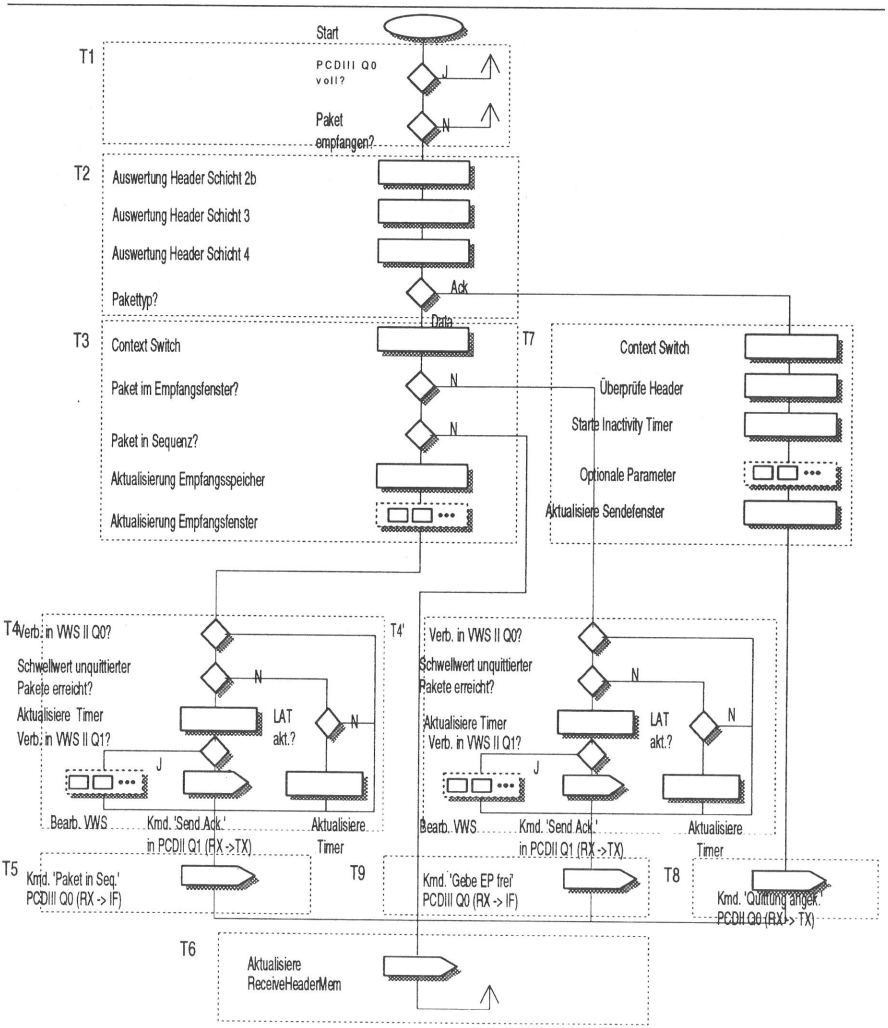


Bild 6.2: *Ausschnitt eines SDL-Diagramms des Empfangsprozessors*

6.2 Verifikation des Transportsystems

Schon zu Beginn eines Systementwurfs stellt die angestrebte bzw. erzielbare Leistungsfähigkeit einen wesentlichen Parameter dar. Um schon frühzeitig entsprechende Aussagen machen zu können, ist der Einsatz entsprechender Modellierungs- und Simulationswerkzeuge üblich. Bei vielen Systemen ist heutzutage zusätzlich ein Nachweis der logischen Korrektheit einzelner Komponenten und des Zusammenwirkens von Baugruppen schon in einem sehr frühen Entwurfsstadium unabdingbar. Die teilweise vorhandenen formalen Methoden zum Nachweis von z.B. interner Verklemmungsfreiheit sind nicht für Systeme mit der Komplexität des TpS anwendbar, und erfordern darüber hinaus in der Regel auch erhebliches Wissen über interne Details, welche zu diesem Zeitpunkt noch nicht verfügbar sind!

Statt eines idealen Top-Down Vorgehens wurde deshalb eine pragmatisch ausgerichtete Mischung des Zusammenfügens von fertig entworfenen und verifizierten (Hardware-)Komponenten mit dem schrittweisen Verfeinern des angestrebten Gesamtsystemmodells (Hard- und Software) vorgenommen. In analoger Weise wurde auch die Systemverifikation von zwei Seiten und mit zwei unterschiedlichen Zielsetzungen angegangen: einerseits standen interne Funktionalität der einzelnen Komponenten, deren funktionelles Zusammenwirken und das exakte Systemverhalten in spezifischen Situationen im Vordergrund des Interesses, andererseits ist das globale Systemverhalten, der Nachweis der prinzipiellen Korrektheit und das globale, nach außen hin sichtbare, Leistungsverhalten von Interesse. Die unterschiedlichen Methoden der Systemverifikation werden nachfolgend betrachtet.

6.2.1 Funktionelle Verifikation der Hardwarekomponenten

6.2.1.1 Granularität der Modellierung

Zur Verifikation von Hardwarekomponenten werden abstrahierende Modelle entwickelt, deren Verhalten dann mittels Rechnersimulationen und entsprechend gewählten extern vorgegebenen Szenarien nachgebildet werden kann. Mit steigendem Abstraktionsgrad, d.h. zunehmender Differenz zwischen Realität und Modell, verlieren die Resultate an Genauigkeit. Eine detaillierte Modellbildung resultiert andererseits direkt in komplexeren Simulationsprogrammen und höherem Rechenaufwand. Aus diesem Grund existiert eine große Spanne unterschiedlicher Simulationsebenen:

Die Prozeßsimulation untersucht die Auswirkung unterschiedlicher Dotierungsdichten auf die elektrischen Eigenschaften von einzelnen Schaltungselementen (Verbindung, Kondensator, Transistor etc.).

Bei der Analogsimulation sind Ein- und Ausgangsvektoren sowie innere Zustände durch einen (theoretisch) unendlichen Wertebereich charakterisiert. Die - überwiegend nicht linearen - elektrischen Grundelemente werden u.a. durch Ohm'sches Gesetz und Kirchhoffregeln beschrieben. Die Lösung der sich ergebenden Differentialgleichungssysteme mittels numerischer Verfahren stellt den Hauptteil der Analogsimulationsprogramme dar. Bedingt durch den mehr als quadratisch mit steigender Elementezahl zunehmenden Rechenaufwand ist diese Form der Simulation nur für kleinere Schaltungsteile, etwa beim Entwurf von Grundgattern oder für zeitkritische Pfade, geeignet. Bekanntester Vertreter hierfür ist SPICE.

Eine erhebliche Reduzierung der Komplexität kann durch die Modellierung eines Transistors als Schalter (switch) erreicht werden. Genau dies wird in Switch-Level-Simulationen gemacht. Nach wie vor ist jedes elektrische Grundelement im Simulationsmodell enthalten.

Der nächste Abstraktionsgrad ist das Zusammenfassen einzelner Transistoren zu logischen Gattern. Bei dieser als Logik- oder auch Gattersimulation bekannten Simulationsform werden Boolsche Operatoren zur Modellierung der logischen Funktionalität eingesetzt. Das reale Zeitverhalten wird durch den Einbau entsprechender Verzögerungsglieder nachgebildet. Die Logiksimulation erfreut sich insbesondere im Bereich der Digitaltechnik großer Beliebtheit, da die Grundelemente den verfügbaren Standardbauelementen entsprechen und damit der Denkweise der Entwickler entgegenkommen.

Komplexere Bausteine lassen sich allerdings nicht immer, und wenn, dann u.U. nur sehr aufwendig, auf Grundgatter basierte Schaltungen zurückführen. Die funktionelle Simulation erlaubt daher den Einsatz von hochsprachenähnlichen Sprachkonstrukten zur Beschreibung des internen Verhaltens eines Bauteils. Die äußere Struktur des Bauteils, also Anschlußbezeichner und elektrische Verbindung mit anderen Bauteilen, bleibt dagegen unverändert erhalten.

Mit zunehmender Abstraktion wird aus der funktionellen eine Verhaltenssimulation, die keinen Bezug zu realen Bauteilen oder realen Verbindungselementen mehr besitzt. Der häufig verwandte Begriff der Register-Transfer-Simulation verknüpft Elemente aus funktioneller und Verhaltenssimulation durch die Kombination von Funktionsblöcken mit unterschiedlichem Bezug zu realen Baugruppen. Durch die Verwendung einer einheitlichen Hardwarebeschreibungssprache (HDL, Hardware Description Language) ist der Übergang zwischen den einzelnen Formen in der Praxis fließend.

Bei der Entwicklung einzelner Baugruppen des TpS stand die Logiksimulation im Vordergrund. Obwohl die Kombination der so modellierten Einzelbaugruppen - mit entsprechend hohem Rechenaufwand - prinzipiell möglich gewesen wäre, wurde für die hardwareorientierte Simulation des Gesamtsystems eine rein funktionelle Vorgehensweise eingeschlagen.

6.2.1.2 Simulationsumgebung

Das vorhandene Simulationswerkzeug HILO [408] besteht aus den in Bild 6.3 dargestellten Teilen.

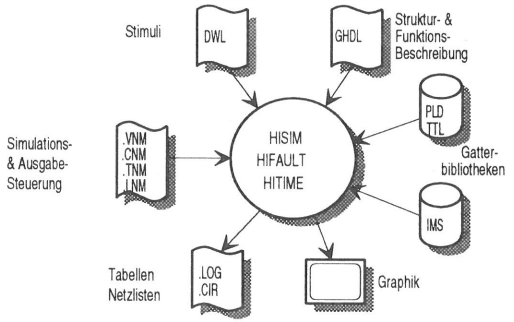


Bild 6.3: *Komponenten des Werkzeuges zur Hardwaresimulation*

Zur Beschreibung von Struktur und Funktion einer Schaltung steht eine alphanumerische Hardwarebeschreibungssprache (GHDL, GenRad Hardware Description Language) zur Verfügung, die Eigenschaften einer Hochsprache enthält. Die Beschreibung der Stimuli wird ebenfalls mittels einer alphanumerischen Sprache (DWL, Digital Waveform Language) durchgeführt. Die Ausgabe der Ergebnisse kann sowohl in Form von Impulsdiagrammen als auch tabellarisch vorgenommen werden.
Vorhandene Bibliotheken unterstützen Simulationen auf Gatterebene (eine entsprechende Unterstützung für die funktionelle Simulation ist nicht vorhanden).
Neben der Simulation mit festen Gatterparametern und jeweils *einem* speziellen Eingangsszenario mit dem Modul HiSim, können auch alle Formen der zeitlichen Varianz und daraus entstehendes Fehlverhalten durchgespielt werden (HiTime). Das Modul HiFault untersucht die äußere Einstell- und Beobachtbarkeit interner Knoten anhand vorgegebener Testmuster und ermöglicht somit deren Bewertung.

6.2.1.3 Simulationsmodell

Obwohl im TpS die Verwendung von RISC-Prozessoren des Typs INTEL i960 vorgesehen ist, wurde die Signalgebung und das Zeitverhalten der modellierten Prozessoren möglichst allgemein gehalten, so daß

problemlos auf andere 32 Bit Prozessoren ausgewichen werden kann.

Für die im Transportsystem verwendeten Speicher werden abstrakte Speicherzyklen eingeführt, die trotz einer verringerten Zahl von (generischen) Signalen das reale Zeitverhalten korrekt nachbilden. Der Zeit- und Steueraufwand für die Auffrischungszyklen bei den dynamischen Speichern wird berücksichtigt.

Neben dem Nachweis des Zusammenspiels der Einzelkomponenten war ein wesentliches Ziel der hardwarenahen TpS-Simulation, genaue Aussagen bezüglich Zugriffskonflikten machen zu können. Die exakte Nachbildung der Verbindungsstrukturen und Zugriffsverfahren nahm deshalb eine hohe Priorität ein. Ein Problem stellte dabei die Bidirektionalität von realen Verbindungswegen dar, welche sich nicht direkt modellieren läßt. Die übliche Methode des logischen Aufteilens in die zwei möglichen Richtungen löst zwar vordergründig das Problem, jedoch sind durch eine solche Parallelisierung offensichtlich keine exakten Aussagen zu Auslastung und realen Zugriffskonflikten mehr möglich. Erst der Entwurf und die Integration von hierarchisch übergeordneten virtuellen Steuereinheiten half weiter.

Die internen Funktionen der einzelnen Blöcke wurden - unter weitgehender Beibehaltung des äußeren Verhaltens - teilweise erheblich vereinfacht. Bild 6.4 verdeutlicht dieses Vorgehen am Beispiel des Bausteins zur Warteschlangenverwaltung PCD.

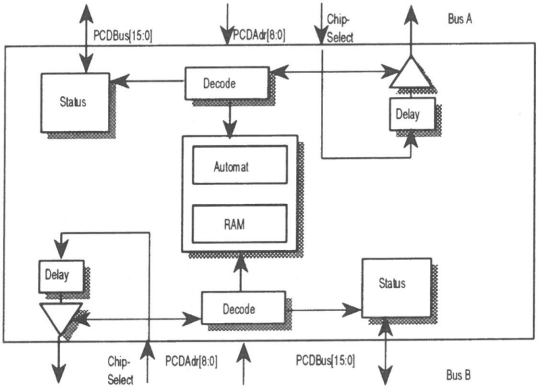


Bild 6.4: *Funktionelle Sichtweise des PCDs*

Die Programmfolgen innerhalb der Prozessoren wurden bei externen Aktionen genau nachgebildet, ansonsten aber zu Verzögerungen zusammengefaßt. Im Unterschied zu Systemmodellen für die Bewertung des statistischen Leistungsverhaltens werden in der hardwarenahen Modellierung allerdings auch "Details" wie z.B. die Mechanismen zur korrekten Ausrichtung von Daten oder Signalisierabläufe zwischen Systemkomponenten betrachtet.

6.2.1.4 Ergebnisse simulierter Szenarien

Der aufwendigste Teil der Hardwaresimulation bestand in der Modellierung jeder einzelnen Komponente. In dieser Phase wurden

- Optimierungsmöglichkeiten innerhalb von Baugruppen aufgezeigt,
- Schnittstellendefinitionen verfeinert und
- Wissen über zeitliches und logisches Verhalten einzelner Komponenten gewonnen.

Die detaillierten Informationen aus der Nachbildung einzelner Funktionalitäten machten mehrere Änderungen am Gesamtkonzept notwendig. Für Einzelheiten wird auf [229] verwiesen.

Das aus dem schrittweisen Zusammenfügen der verifizierten Komponenten entstandene Simulationsmodell des Gesamtsystems erlaubte die Überprüfung einfacher Meldungsszenarien. Für jede der zwei Kommunikationsrichtungen wurde dazu der Durchlauf von einzelnen Daten- und Quittierungs-PDUs durch das TpS betrachtet. Bild 6.5 zeigt Ausschnitte eines resultierenden Meldungsszenarios für den Empfang einer fehlerfreien Daten-PDU.

Neben der Anschaulichkeit des stockungsfreien Weiterreichens der Information enthält diese nachrichtenorientierte Darstellung implizit auch den Nachweis der zeitlich korrekten Steuerung der passiven Systemkomponenten.

Die Simulationsergebnisse wurden auch dazu benutzt, erste Aussagen zu Auslastungen von Verbindungsstrukturen und Prozessoren zu erhalten. Die erhaltenen Werte können naturgemäß nur die spezielle Einzelsituation widerspiegeln. Allgemeingültigere Aussagen bleiben der statistischen Simulation vorbehalten, die allerdings auf den hier gewonnenen genauen Kenntnissen aufbaut.

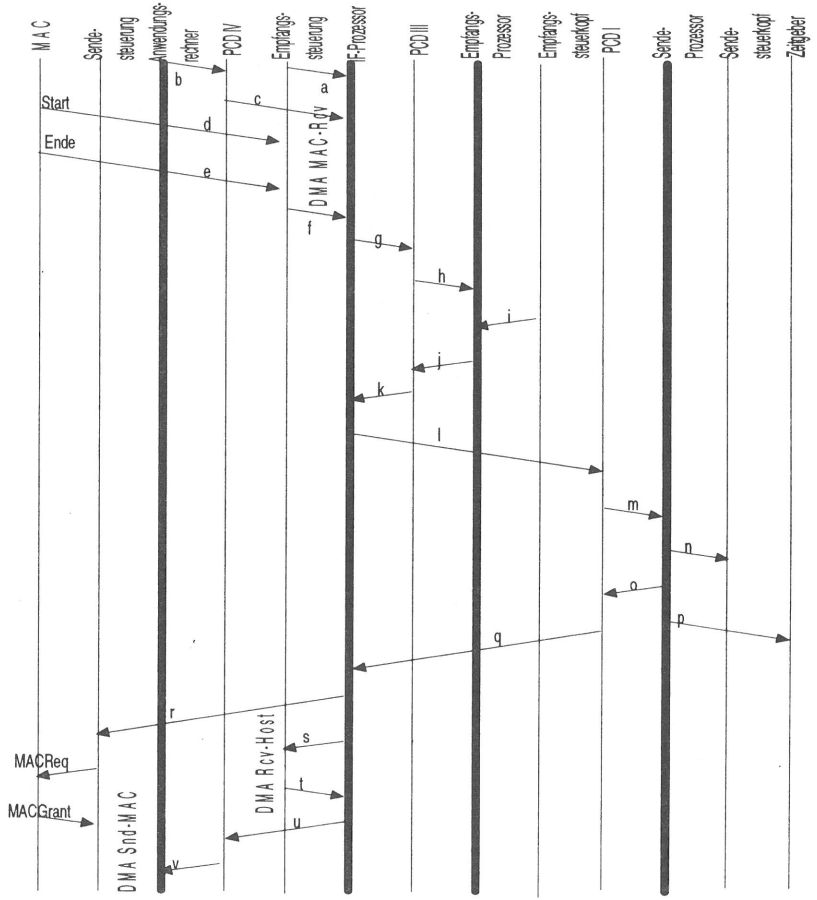


Bild 6.5: *Meldungsszenario für den Empfang einer Daten-PDU (vereinfacht)*

Die zusammengefaßten Ergebnisse der funktionellen hardwarenahen Simulation sind:

- Die grundsätzlichen Architekturentscheidungen und die vorgenommene Funktionsverteilung wurden bestätigt.
- Das korrekte Zusammenspiel aller TpS-Komponenten wurde nachgewiesen.
- Die Richtigkeit der in 5.4.8 gemachten Leistungsschätzungen konnte gezeigt werden.

6.2.2 Validierung und Leistungsuntersuchung des Gesamtsystems

Voraussetzung jeder Form von Leistungsbewertung sind abstrahierende Systemmodelle. Abhängig von Systemkomplexität und Abstraktionsgrad des Modells können verschiedene Methoden zur Leistungsbewertung eingesetzt werden. Bevor auf die spezielle Modellierung des TpS eingegangen wird, werden zunächst einige davon eingeführt.

6.2.2.1 Methoden zur Leistungsuntersuchung

Nur bei Modellen geringer Komplexität können exakte Analyseverfahren, basierend z.B. auf eingebetteten Markoff-Ketten, Mittelwertberechnungen oder der Produktlösungsform, angewandt werden. Durch den geforderten geringen Komplexitätsgrad der zugrundeliegenden Modelle können entweder nur einfache Systeme behandelt werden oder es ist eine so weitgehende Abstraktion notwendig, daß das reale Systemverhalten nur noch bedingt nachgebildet wird. Die exakte Analyse besitzt daher nur dort praktische Relevanz, wo Abschätzungen ausreichen, das Verhalten in Grenzfällen von Interesse ist oder Ergebnisse, die mit anderen Methoden gewonnen wurden, validiert werden sollen.

Verzichtet man auf die exakte Beschreibung und läßt bestimmte Näherungen zu, erweitert sich der sinnvolle Einsatzbereich der mathematischen Analyseverfahren. Durch die Anwendung von Näherungsverfahren lassen sich häufig einfach handhabbare Näherungsformeln gewinnen, deren Gestalt auch den Einfluß der Systemparameter auf die verschiedenen Systemgrößen deutlich machen kann.

Eine typische Vereinfachung ist die Annahme der Unabhängigkeit von Ereignissen. Die Definition eines äquivalenten Ersatzverkehrs und entsprechender Generatoren, Zerlegungsverfahren, Makrozustandsbetrachtungen oder die Approximation von Momenten höherer Ordnung sind weitere bekannte Beispiele. Sowohl exakte als auch approximative Verfahren können auf folgende Gleichungssysteme führen:

Systeme linearer Gleichungen;
Systeme linearer Differentialgleichungen;
Integralgleichungen;
Eigenwertprobleme;
Randwertprobleme;
Variationsaufgaben.

Zur Lösung werden Methoden der numerischen Mathematik eingesetzt. Speicherplatzbedarf, stark (z. B. exponentiell) ansteigende Rechenzeit und die numerische Stabilität setzen der Anwendung analytischer Methoden zur Leistungsuntersuchung enge Grenzen. Zur Bewertung des Normalverhaltens komplexerer Systeme sind somit auch die analytischen Nährungsverfahren nur bedingt sinnvoll einzusetzen. Zur Verifikation von Spezialfällen mit seltenen Ereignissen sind sie allerdings unverzichtbar.

Im Gegensatz zu analytischen Methoden ist die Leistungsuntersuchung durch Simulation eine sehr allgemein anwendbare Methode. Das Verhalten komplexer Modelle läßt sich häufig ausschließlich durch Simulation ermitteln. Ähnlich wie eine Analyse von Spezialfällen zur Validierung einer Simulation dienen kann, lassen sich auch analytisch ermittelte Ergebnisse durch Simulation validieren.

Die Simulation eines Systems ist eine experimentelle Methode, die entweder in einer realen oder einer nachgebildeten Umgebung vorgenommen werden kann. Am simulierten System werden Meßwerte in

Form von Stichproben vorgenommen und statistisch ausgewertet.

Bei der zeittreuen Simulation, welche hier angewandt wurde, wird der Verkehrsablauf zeitgetreu nachgebildet, indem die Ereignisse erfaßt werden, die mit einer Änderung des Systemzustands verknüpft sind.

6.2.2.2 Aspekte der Validierung

Zur simulativen Leistungsbestimmung wird ein geeignet vereinfachtes Modell bestimmt. In diesem Modell müssen sich nur die Mechanismen, die für den zeitlichen Ablauf wesentlich sind, wiederfinden.

Da die Simulation gleichzeitig auch zur Validierung des Gesamtkonzepts verwendet werden soll, müssen auch Mechanismen mit aufgenommen werden, deren zeitlicher Einfluß zu vernachlässigen wäre. Ebenso werden Elemente der Kommunikationssoftware nachgebildet, die für eine reale Implementierung von Bedeutung sind, nicht aber in einer Simulation zur Leistungsuntersuchung.

6.2.2.3 Modellierung des Transportsystems

Im TpS arbeiten drei Prozessoren parallel, die Verbindungsparameter und Systeminformation in ihren lokalen Speichern verwalten. Zur Validierung der Programmodule ist eine Konsistenzprüfung erforderlich. Im Simulationsprogramm kann die Validierung durch eine Plausibilitätsprüfung erfolgen.

Eine weitere Möglichkeit der Validierung ist die Einführung von Redundanz im Simulationsprogramm. Dazu werden Änderungen an relevanten Systemvariablen innerhalb (der später eingeführten) Datenstrukturebene durch übergeordnete Simulationsprozeduren detektiert und an zentraler Stelle protokolliert. Eine Konsistenzprüfung durch Vergleich ist so jederzeit möglich.
Ein einfaches Modell läßt sich durch eine Konzentration auf die Elemete erreichen, welche bei der regulären Abarbeitung einer Daten-PDU benötigt werden (siehe Bild 6.6). Obwohl dieses Modell im wesentlichen den logischen Datenfluß repräsentiert und nur wenige Implementierungsspezifika miteinbezieht, lassen sich damit schon aussagekräftige Leistungsuntersuchungen durchführen.

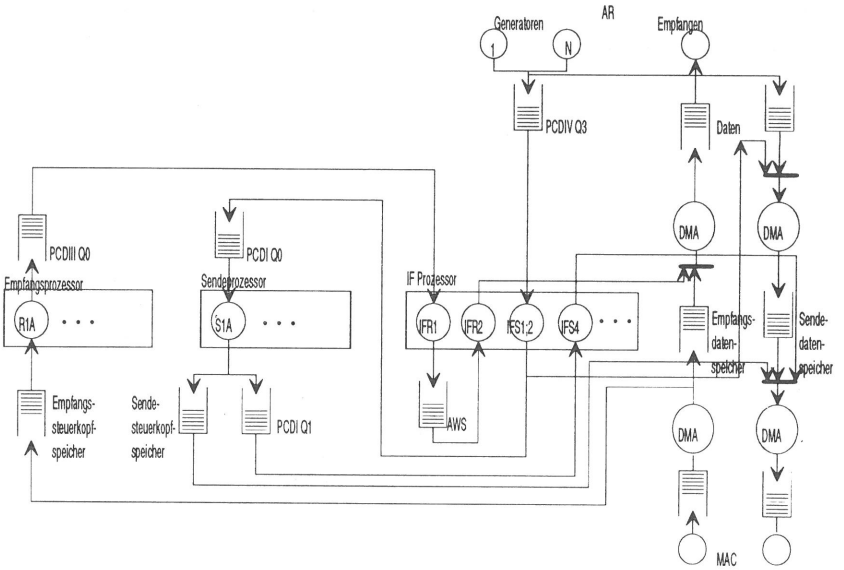


Bild 6.6: *Vereinfachtes Modell des TpS*

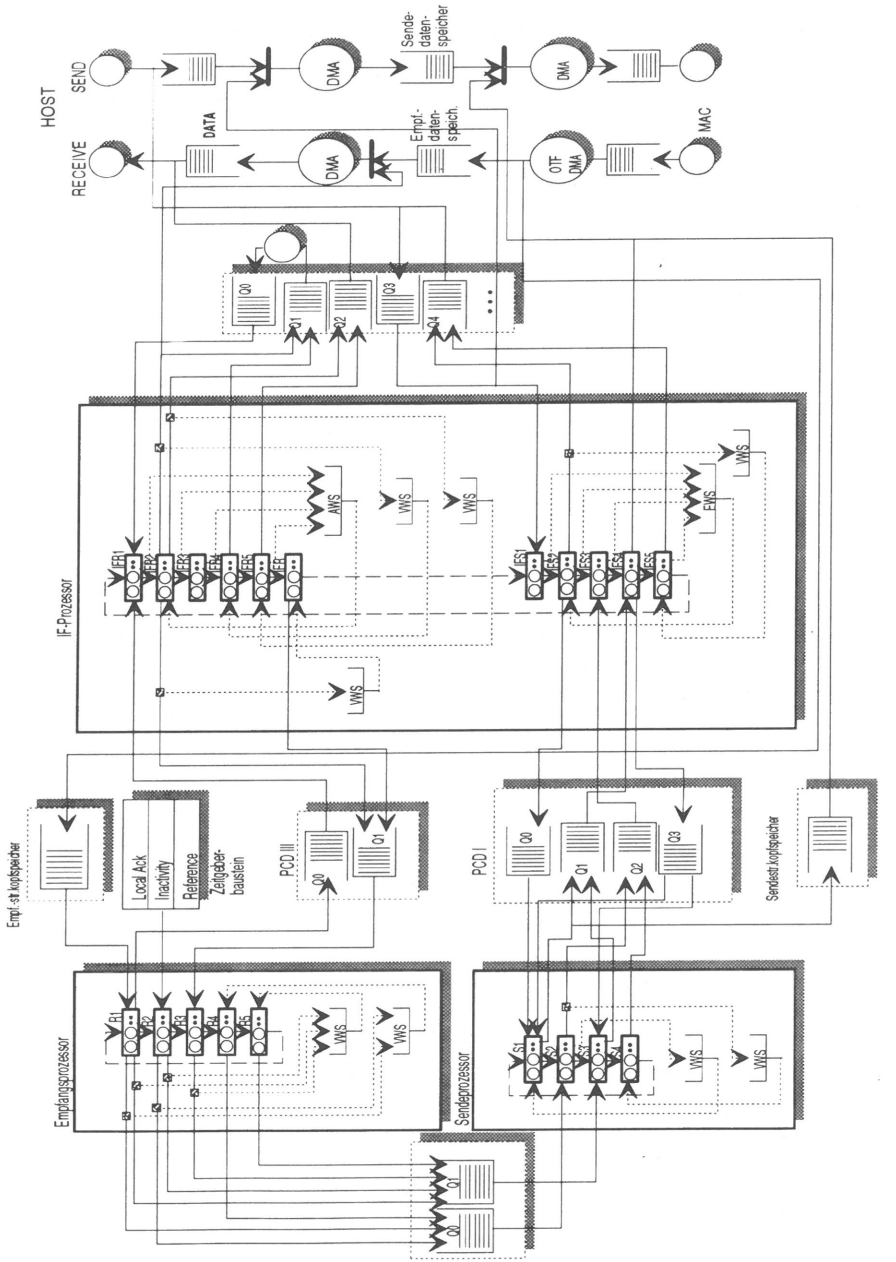


Bild 6.7: *Gesamtmodell des TpS*

Für eine präzisere Leistungsuntersuchung sowie für eine Simulation, die sowohl der Validierung des Systemkonzeptes dienen soll als auch eine Dimensionierung der Systemparameter ermöglicht, ist ein umfassenderes Modell erforderlich (siehe Bild 6.7).

Dieses Modell ist dadurch gekennzeichnet, daß es nicht nur die Elemente enthält, die zu einer hinreichend präzisen Leistungsbestimmung notwendig sind, sondern alle Elemente (Verbindungsstrukturen, Warteschlangen, interne Prozessorphasen etc.) berücksichtigt, die in der Datentransferphase eine Funktionalität besitzen, um eine Validierung des Systementwurfs zu ermöglichen.

Um die Komplexität des Gesamtmodells besser handhabbar zu machen, und um eine bessere Strukturierung der Aufgaben Leistungsmessung und Validierung zu ermöglichen, wurde eine Aufteilung des Modells in vier Ebenen entsprechend Bild 6.8 vorgenommen.

Auf der Statistikebene werden die Werte all der Objekte gespeichert, für die eine Messung und/oder eine statistische Auswertung gewünscht ist.

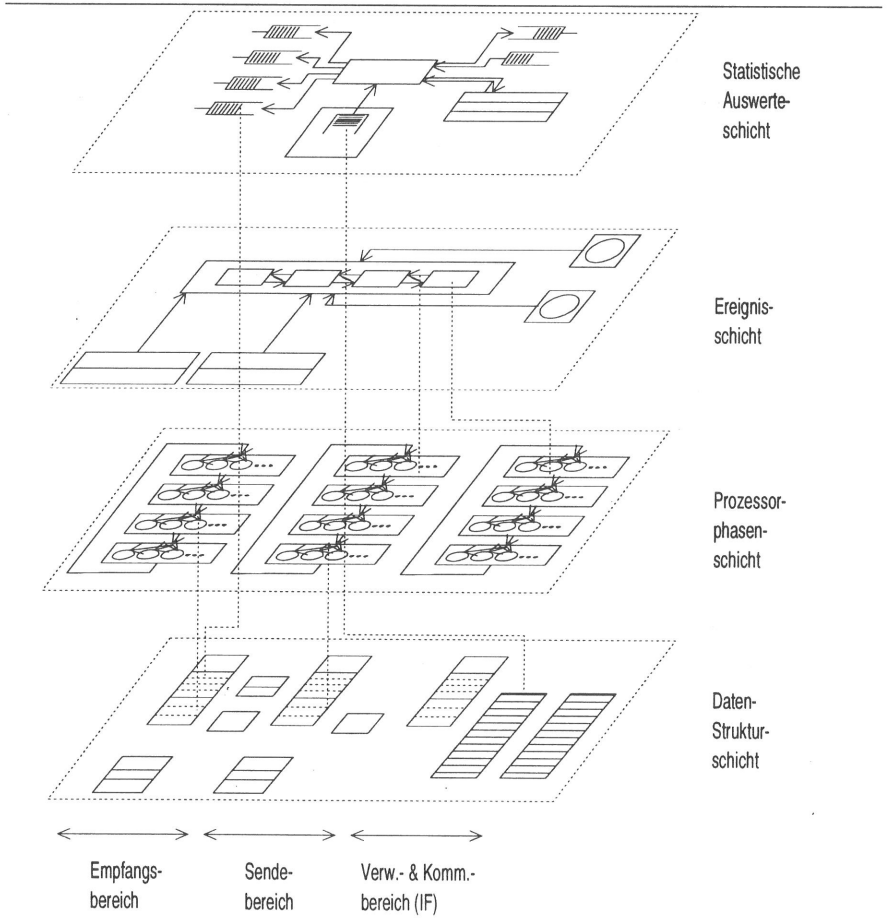


Bild 6.8: *Modell in vier Ebenen*

Datenstrukturebene

Innerhalb der untersten Ebene sind neben Eigenschaften der Hardwarekomponenten die verwendeten Datenstrukturen von zentraler Bedeutung für die Leistungsfähigkeit des Systems. Bild 6.9 zeigt einen Ausschnitt der Datenstrukturen der Pufferverwaltung. Zu erkennen sind logische Verkettungen innerhalb und zwischen Verwaltungsebenen. Überwiegend lassen sich diese Verkettungen auf entsprechende (FIFO-)Warteschlangen im Modell abbilden.

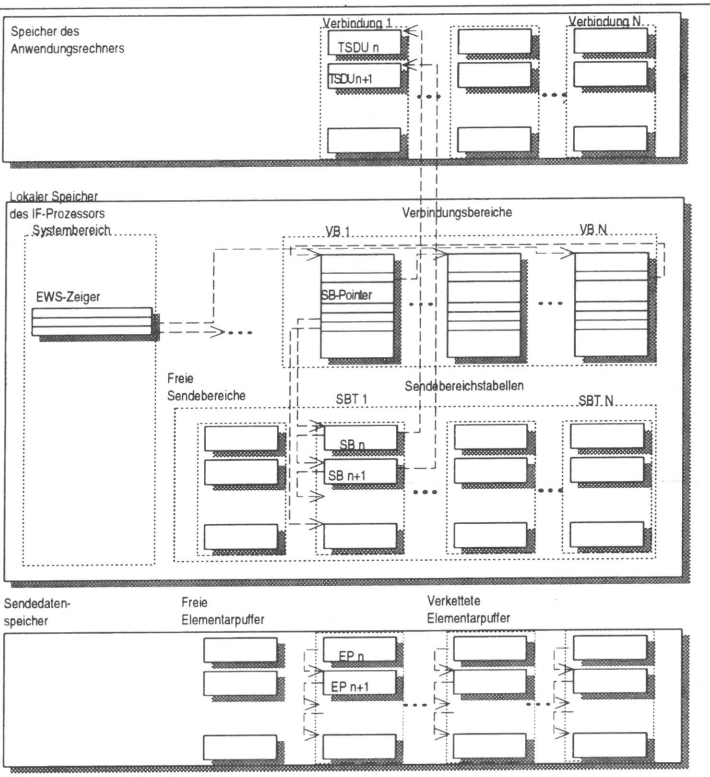


Bild 6.9: *Datenstrukturen der Pufferverwaltung in Senderichtung*

Im Simulationsprogramm wird jede Änderung an einer solchen Dateneinheit zusammen mit Art und Zeitpunkt der Änderung in einer zugeordneten Variablen erfaßt. Gemäß der Abstraktion des Modells in vier Ebenen bedeutet dies, daß zu einem bestimmten Simulationszeitpunkt vom Kalender (Ereignisebene) die Ausführung einer Prozessorphase (Prozessorphasenebene) veranlaßt wird. Wenn diese Prozessorphase an einer Datenstruktur (Datenstrukurebene) eine Veränderung vornimmt, wird diese Veränderung in der zugeordneten Variablen erfaßt (Statistikschicht) und steht dort zur statistischen Auswertung bereit.

Prozessorphasenebene

Um die abstrahierten Abläufe für die Prozessorphasenebene zu erhalten, wurde eine Analyse der Softwarefunktionalität in mehreren Schritten durchgeführt: nach der Unterteilung in Phasen bzw. Subprozesse (Tasks) folgte das Erkennen von Verzweigungen und der Abhängigkeiten von den Aktionen der benachbarten Prozessoren (Verzahnung) sowie die Erfaßung der Ausführungszeiten für die verschiedenen Pfade.

Die Kommunikationssoftware ist in i80960 Assemblersprache [400] geschrieben (vergleiche Tabelle 6.1, Seite 163). Es erscheint trivial, die Abarbeitungsdauer eines solchen Assemblerquelltextes durch einen Prozessor exakt zu ermitteln. Da die verwendeten RISC-Prozessoren aber über einen integrierten Befehlsspeicher (On-chip Cache), eine mehrstufige Fließbandverarbeitung und drei Einheiten zur Befehlsausführung verfügen, ist die genaue Abarbeitungsdauer nur schwer zu ermitteln. Die Berücksichtigung von Programmlokalität und Verzweigungshäufigkeit läßt jedoch eine (konservative) Annahme von im Mittel zwei parallel ausgeführten Befehlen zu.
Zur Erläuterung: die gesamte Software jedes TpS-Prozessors besteht aus jeweils nur wenigen hundert Assemblerbefehlen, so daß für reguläre Bedingungen grundsätzlich aus dem integrierten Befehlsspeicher gelesen werden kann. Ein Großteil der TpS-internen Abläufe ist sequentieller Natur, zusätzlich wurde auch auf eine möglichst verzweigungsfreie Programmierung geachtet.

Bild 6.2 auf Seite 164 zeigt das Flußdiagramm des Programmoduls, das die Bearbeitung eines empfangenen Steuerkopfes steuert. Gruppen von Funktionen werden zu Subprozessen (Tasks T1 bis T9) zusammengefaßt. Für die Simulation werden die Bearbeitungszeiten dieser Tasks für die verschiedenen Ausführungsbedingungen ermittelt. Dabei können für Fälle, in denen die Bearbeitungszeit im allgemeinen schon vorher bekannt ist (z.B. Funktion 'Aktualisiere Empfangsfenster') und selten vorkommende Pfade (z.B. Funktion 'Bearbeite VWS') Vereinfachungen vorgenommen werden, ohne die Qualität des Simulationsergebnisses wesentlich zu beeinflussen.

Verzweigungen lassen sich in zwei verschiedene Gruppen unterteilen: solche, die von Datenstrukturen abhängen, die vom gleichen Prozessor manipuliert werden, der auch die Verzweigung ausführt ('interne Verzweigungen'), und solche, die vom Zustand einer DMA-Einheit oder vom Inhalt eines PCDs (und damit indirekt von der Aktion eines anderen Prozessors) abhängen ('externe Verzweigung'). Letzteres kann auch mit dem Begriff 'Verzahnung' beschrieben werden.

6.2.2.4 Das Simulationsprogramm

Zur Simulation des TpS werden alle Mechanismen und Elemente der entwickelten Modelle in einem entsprechenden Simulationsprogramm nachgebildet. Dabei wird auf die am IND vorhandene VAX-PASCAL-basierte Simulationsbibliothek [409] und auf Elemente aus einem bestehenden Programm [208] zurückgegriffen. Das dynamische Ablaufgeschehen wird mit Hilfe einer Systemzeit und eines Kalenders nachgebildet. Im Kalender werden zukünftige Ereignisse in zeitlich geordneter Reihenfolge verwaltet. Ereignisse sind Ankünfte von Anforderungen (vom AR bzw. MAC), Kommandos zwischen den Prozessoren, das Ausführungsende eines laufenden Programmoduls und abgelaufene Zeitgeber.

Die Steuerung der Simulation erfolgt durch die in der Reihenfolge ihres Auftretens aus dem Kalender ausgetragenen Ereignisse. Bei jedem Austrag aus dem Kalender wird die Systemzeit aktualisiert, und es werden alle Aktionen durchgeführt, die zu diesem Zeitpunkt nötig sind. Beispiele für solche Aktionen sind Einträge in eine Warteschlange, Abarbeitung der Warteschlangen durch die TpS-Prozessoren oder die Bestimmung eines Folgeereignisses durch Erzeugung weiterer Anforderungen im AR. Der Eintreffzeitpunkt von Anforderungen wird mittels Zufallszahlen bestimmt, die durch Generatoren mit vorgegebener Verteilungsfunktion erzeugt werden.

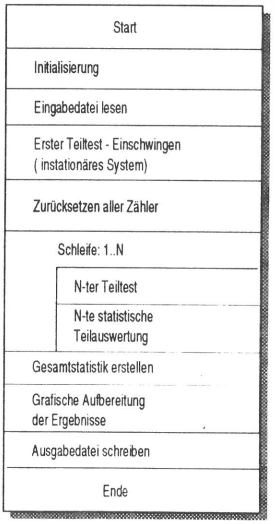


Bild 6.10: *Struktogramm des Simulationsprogramms*

Bild 6.10 zeigt den prinzipiellen Ablauf des Simulationsprogramms. Nach der Programminitialisierung wird die Eingabedatei gelesen, in der die Parameter für den jeweiligen Simulationslauf enthalten sind. Anschließend erfolgt ein Warmlauf des Systems, der typischerweise 1.000 TPDUs umfaßt. Dabei geht das System vom instationären Anfangszustand in den eingeschwungenen Zustand über. Anschließend werden alle statistischen Zähler neu initialisiert und TPDUs für die eigentliche Simulation übertragen. Eine Unterteilung der Simulation in Teiltests, die unabhängig voneinander ausgewertet werden können, erlaubt die Ermittlung von Vertrauensintervallen in einer abschließenden statistischen Auswertung. Die Ergebnisse dieser Auswertung werden in eine für die grafische Repräsentation geeignete Form gebracht und in eine Ausgabedatei geschrieben.

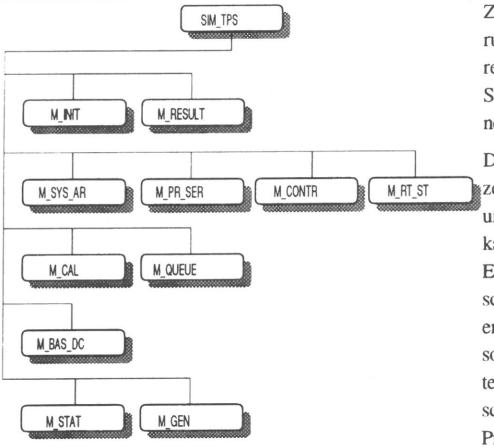


Bild 6.11: *Hauptmodule des Simulationsprogramms*

Zur Erhöhung der Übersichtlichkeit, Erweiterungsfähigkeit und Testbarkeit ist ein modulares Vorgehen unablässig. Die Hauptmodule des Simulationsprogramms sind Bild 6.11 zu entnehmen.

Das Programm ist so ausgelegt, daß der Benutzer über eine Eingabedatei Einfluß auf Inhalt und Darstellungsform der Ausgabe nehmen kann. Grundsätzlich können für jedes einzelne Element und auch für Elementgruppen statistische Aussagen unterschiedlicher Detaillierung erstellt bzw. auch ausgegeben werden. Von besonderem Interesse sind dabei Größen wie Warteschlangenlängen und deren Überlaufwahrscheinlichkeit, Bearbeitungszeiten einzelner Prozessorphasen, Gesamtdurchlaufzeit einer PDU oder gesamte Prozessorauslastung.

6.2.2.5 Simulierte Szenarien und Ergebnisse

Bei der Simulation wird von zwei Stationen ausgegangen, die über eine konstante Anzahl von Verbindungen miteinander kommunizieren. Der Medienzugriff erfolgt transparent, mit für die Verbindungen individuell einstellbaren Übertragungscharakteristika wie Verzögerungszeit und Fehler- bzw. Verlustwahrscheinlichkeit. Die Generatoren sind unabhängig voneinander einstellbar. Dadurch kann Simplex-, Halbduplex- und Vollduplexverkehr mit unterschiedlichen Verkehrscharakteristika nachgebildet werden. Für die simulative Leistungsuntersuchung sind damit nicht mehr als zwei Stationen nötig.

Nicht betrachtet werden Verbindungsauf- und -abbau sowie das Eintreffen von PDUs, die vom TpS nicht verarbeitet werden können, sondern transparent an den AR weitergeleitet werden müßten.

Die zugrundegelegten Systemparameter sind in Tabelle 6.2 zusammengefaßt. Die Werte sind das Resultat aus Betrachtungen zu Wirtschaftlichkeit und Leistungsfähigkeit.

Variable Parameter sind u.a. die Häufigkeit der Weitergabe befreiter EPs vom IF- zum Empfangsprozessor (für alle Verbindungen individuell einstellbar); die Sequenz der Programmodule; Vorgaben für Empfangsstrategie; Werte für Wiederholungs-, Fenster- und lokalen Quittierungszeitgeber; maximale und minimale Kredite; Quittierungsschrittweite; Ankunftsverhalten; Block- und TSDU-Länge (pro Verbindung); Programmsteuerinformation: Vertrauensintervall; Anzahl der PDUs zum Warmlaufen;,

Tabelle 6.2: *Feste Systemparameter bei den Simulationen*

Prozessortakt	25 MHz
Speicherbandbreite	200 Mbit/s; ⇒Lese-/Schreibzyklus je 160 ns
Größe des Speichers für Datenteil in Senderichtung	4 MB
Größe des Datenspeichers in Empfangsrichtung	4 MB
Größe der Steuerkopfspeicher	256 KB
Größe der Warteschlangen in den PCDs	256 KB
Größe der Tabelle für unquittierte TPDUs	4000
Maximalzahl gleichzeitig aktiver Verbindungen	512
Bearbeitungszeit (konstant) für Schichten 2b und 3	150 Assemblerbefehle (= 6.7 µs [258])

Teiltestzahl sowie die Zahl der PDUs pro Teiltest.

Um den Einfluß der Verzahnung zwischen den Bearbeitungsphasen der drei Prozessoren zusammen mit der zeitlichen Abfolge der Bearbeitungsschritte in einem Bild übersichtlich darzustellen, wurde ein neuartiges Diagramm entwickelt. Dazu wurden ein herkömmliches Gant-Diagramm, das die einzelnen

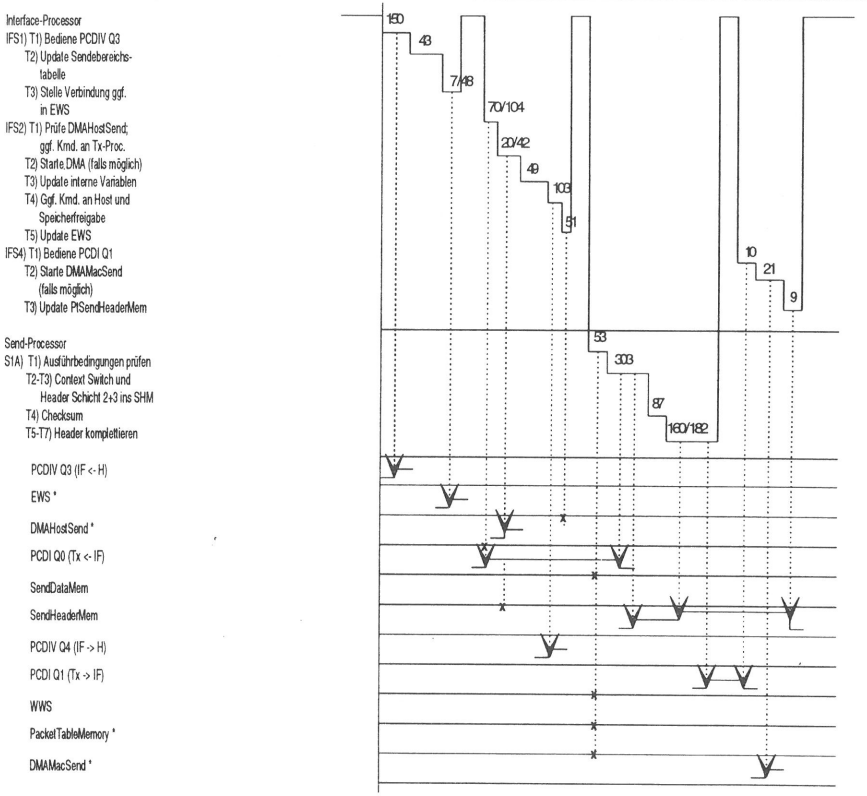


Bild 6.12: *Diagramm für Bearbeitung eines Datenpakets in Senderichtung*

Prozessorphasen enthält, um Verzahnungselemente erweitert.

Bild 6.12 zeigt ein solches Ergebnisdiagramm, das den Gesamtablauf der Bearbeitung einer regulären Daten-PDU in Senderichtung darstellt. Die Zahlen geben die Anzahl von Prozessorzyklen an, die zur Abarbeitung einer Teilaufgabe notwendig sind. Im Gegensatz dazu kann bei der Wiederholung einer PDU auf Information der ersten Zusammenstellung des Steuerkopfes zurückgegriffen werden.

Im Modell mit vier Abstraktionsebenen (Bild 6.8) beschreibt dieses Diagramm das Zusammenwirken von Datenstrukturebene und Prozessorphasenebene. Durch vertikale Pfeile wird die Änderung eines Datenelements angezeigt, das Verzweigungen beeinflußt. Externe Verzweigungen werden als Ereignis von der Ereignisebene ausgelöst, damit die gegenseitige Beeinflussung der parallel arbeitenden Prozessoren von der Simulation korrekt nachgebildet wird.

In der Simulation kann die Länge des Datenteils einer TSDU Vielfache von 128 und 1024 Oktetts annehmen. Wichtig ist eine Länge von 128 Oktetts für die Ermittlung der maximalen PDU-Rate und eine Länge von 1024 Oktetts für die Ermittlung des maximalen Durchsatzes.

Maximaler Durchsatz:
Für eine gute Nutzung der Übertragungskapazität des Netzes wird mit einer Länge des Datenteils der TPDUs von 1024 Oktetts gearbeitet. Um das TpS durch die Bedienung der Schnittstelle zum AR durch häufiges Anfordern neuer Sende- und Empfangsbereiche nicht unnötig zu verlangsamen, wird für die Bestimmung des maximalen Durchsatzes mit sehr großen Sende- und Empfangsbereichen und damit mit sehr großen TSDUs gearbeitet.

Maximale PDU-Rate:
Um die maximale PDU-Rate zu ermitteln, wird mit TPDUs der minimalen Länge des Datenteils von 128 Oktetts gearbeitet. Da die Systemressourcen limitiert sind, lassen sich zwei maximale PDU-Raten ermitteln. Im dynamischen Falle werden PDU-Büschel betrachtet, wobei diese Rate nur für kurze Büschel gilt. Eine praxisrelevante Büscheldauer liegt in der Größenordnung einer Millisekunde, was etwa 80 kurzen PDUs entspricht.

Verzögerungszeit:
Die mittlere Verzögerungszeit des TpS wird für wachsende Last überproportional anwachsen. Nach Möglichkeit sollte der Einfluß begrenzter Speicherbandbreite und begrenzter Prozessorzyklen auf die Verzögerung getrennt sichtbar werden.

Für die Bestimmung des maximalen Durchsatzes genügt es, ein AR-Verhalten vorzugeben, bei dem immer Anforderungen zur Abarbeitung durch das TpS bereitstehen. Im Gegensatz dazu muß für die Bestimmung von Verzögerungszeiten und für die Dimensionierung von Speichergrößen das Verhalten des ARs im Teillastbetrieb ausreichend genau nachgebildet werden.

Für einen Prozessortakt von 25 MHz ergibt sich - unabhängig von der Anzahl der aktiven Verbindungen - eine maximale Leistungsfähigkeit von 32.000 TPDUs pro Sekunde im Vollduplexbetrieb. Schon für mäßig große TPDUs mit einer Länge von 1 KB ergibt sich damit ein Durchsatz von 260 Mbit/s pro Kommunikationsrichtung!

In vielen Veröffentlichungen werden sehr große PDU-Längen verwendet, um eindrucksvolle Durchsatzraten angeben zu können (z.B. 32.000 Oktetts in [62]). Um die Leistung des TpS zu verdeutlichen, sei angemerkt, daß sich schon mit der maximalen LLC-konformen Länge von knapp 8 KB rechnerisch ein Durchsatz von 2 Gbit/s (wiederum Vollduplex) ergäbe! Von einigen Ausnahmen abgesehen (z. B. ausschließliche Übertragung sehr großer Dateien von und zu einem dedizierten Fileserver) werden allerdings in zukünftigen Anwendungen PDUs kurzer Länge klar dominieren [96, 180, 269, 317]!

Außerdem verlangen solche Datenraten Datenspeicher und Verbindungsstrukturen mit entsprechend

hoher Bandbreite, welche - wenn überhaupt - nur mit hohen Kosten zu realisieren sind. Wird von heute wirtschaftlich machbaren Systemparametern ausgegangen, also z.B. von Datenspeichern mit einer Bandbreite von 200 Mbit/s, so ist der Systemdurchsatz automatisch auf 100 Mbit/s begrenzt (jede beschriebene Speicherstelle muß zumindest einmal ausgelesen werden!). In Bild 6.13 ist zu sehen, wie dieser Maximaldurchsatz für PDUs mit einer Länge von 1024 Oktetts schon bei etwas mehr als 20.000 PDUs/s erreicht wird. Für kurze PDUs mit einer Länge von 128 Oktett ist der Durchsatz dagegen nicht durch die Bandbreite, sondern durch die Verarbeitungsleistung des TpS (maximale PDU-Rate) begrenzt.

Das System ist von seiner Leistungsfähigkeit so ausgelegt, daß im normalen Betrieb nie die volle Auslastung erreicht wird. Die Verzögerungszeiten steigen erst für Auslastungen von mehr als 70 % deutlich an. Für eine Auslastung von 50 % ergibt sich als Gesamtdurchlaufzeit einer PDU durch das TpS 63,6 µs in Senderichtung und 67,3 µs in Empfangsrichtung. Das sind extrem kurze Verzögerungen, die einen Einsatz des Transportsystems auch für isochronen Verkehr problemlos möglich erscheinen lassen. Bild 6.14 zeigt, wie die Verzögerungszeiten für steigende Belastung des Systems zuerst langsam und dann immer stärker zunehmen.

Die kurzen Verzögerungszeiten kommen einher mit sehr kurzen Programmzykluszeiten. Für den Empfangsprozessor ergibt sich eine Zeit von 14,6 µs, für den Sendeprozessor eine Zeit von 12,5 µs und für den IF-Prozessor eine Zeit von 14,5 µs. Diese Zeiten zeigen, daß die Bearbeitungslast beinahe optimal auf die drei Prozessoren verteilt ist und bestätigt somit die Überlegungen, die zur Einführung des IF-Bereichs führten.

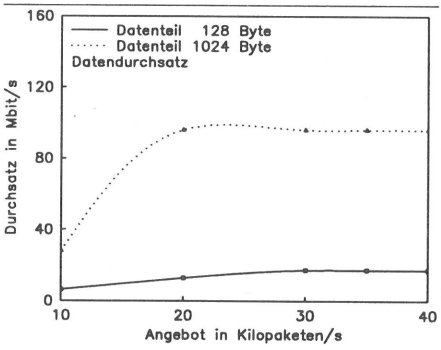


Bild 6.13: *Datendurchsatz bei Speicherbandbreite 200 Mbit/s*

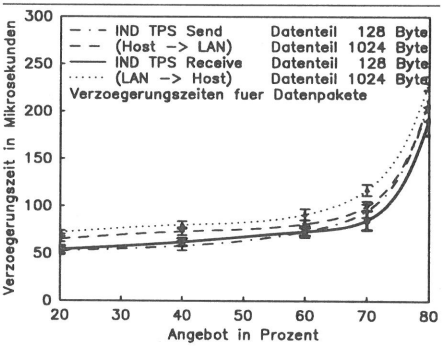


Bild 6.14: *Verzögerungszeit für Datenpakete*

Kapitel **7**

Zusammenfassung und Ausblick

Ausgangspunkt der Arbeit war es, Gründe für die gravierende Differenz zwischen der Leistungsfähigkeit auf den untersten Schichten des OSI-RMs und den oberhalb der Transportschicht erreichbaren Leistungswerten aufzuzeigen und entsprechende Lösungen zu erarbeiten. Nach einer Einführung allgemeiner Trends innerhalb der Kommunikationstechnik sowie grundsätzlicher Begriffsklärung und Modellvorstellungen beschrieb Kapitel 3 die durchgeführte Problemanalyse. Zusammengefaßt ergab sich folgendes:

- Die Leistungsfähigkeit von Protokollfunktionen hängt stark von deren Implementierung ab
- Die Protokolle wirken sich weniger aus als die notwendigen Unterstützungsfunktionen.
- Es existieren keine eindeutig leistungsdominierenden Teilbereiche.
- Zwischen Einzelfunktionen sind vielfältige Abhängigkeiten vorhanden.

Trotz dieser Erkenntnisse wurden in Kapitel 4 neben systemtechnischen Ansätzen bewußt auch Lösungsvorschläge für dedizierte Teilprobleme vorgestellt. Einerseits wird damit der Forschungstätigkeit auf diesem Gebiet Rechnung getragen, andererseits erhöht die zwangsläufig entstehende Strukturierung Lesbarkeit und Problemverständnis. Darüberhinaus wurde auch der tatsächliche Verlauf der Arbeit mit dem Übergang von Teilbetrachtungen zur Gesamtsystemsicht verdeutlicht.

Die abstrakte Modellierung eines aus den vorhandenen Teillösungen zusammengesetzen Gesamtsystems scheiterte an der zu großen Zahl möglicher Parameter. Stattdessen wurde die Implementierung eines kompletten Transportsystems unter den finanziellen und organisatorischen Randbedingungen eines Hochschulinstituts zum zentralen Gegenstand der Arbeit! Gleichzeitig verlor die ursprünglich geplante Suche theoretischer Optima für isolierte Teilprobleme sowie die zugehörigen wissenschaftlichen Nachweismethoden an Bedeutung.

Kapitel 5 schilderte den oft eher pragmatischen und ingenieurgemäßen als akademisch geprägten Lösungsweg, die wichtigsten Aspekte des entstandenen TpS, dessen Softwarestruktur und Hardwarekomponenten. Einige Hauptmerkmale werden hier wiederholt:

- Verzicht auf ein Betriebssystem sowie bewußt unstrukturierte Software
- Bezahl- und machbare Kompromisse statt teuren bzw. komplexen Lösungen
- Verwendung von 3 Standard RISC-Prozessoren
- Aufteilung der Speicher (Senden - Empfangen, Daten - Steuerköpfe)
- Entwurf von 6 ASICs bis auf (IMS-)Gatterebene
- Über 20.000 PDUs pro Sekunde und Richtung bei weniger als 100 µs Verzögerung.

Aus schaltungstechnischer Sicht besonders erwähnenswert sind der CAM-basierte Zeitgeberbaustein und die vollständig durch Hardware realisierte Speicherverwaltung. Die Prüfsummenbausteine werden trotz bzw. gerade wegen ihrer relativ wenig komplexen Schaltungstechnik ebenfalls genannt. Die überraschend einfache und trotzdem äußerst leistungsfähige Lösung ergab sich erst durch die (ungewöhnliche) Kombination aus hardwaregeprägten und mathematisch theoretischen Überlegungen (siehe Abschnitt 5.3.5.5).

Zur Verifikation und Validierung des TpS wurde neben hardwareorientierten Simulationswerkzeugen

auch eine eigenentwickelte Mischform der statistischen Modellierung eingesetzt. Diese zeichnet sich durch eine detaillierte Berücksichtigung von Details bis hin zu einzelnen Maschinenbefehlen aus, welche zur reinen Leistungsmodellierung nicht benötigt würden, zur Überprüfung des Systemkonzepts jedoch überaus hilfreich waren.

Die Kernaussagen der Arbeit lassen sich wie folgt zusammenfassen:

- Die OSI-Protokolle sind - mit einer entsprechenden Implementierung und einigen Festlegungen - durchaus auch für moderne Hochgeschwindigkeitsnetze einsetzbar.
- Modularität resultiert in einem Verlust an Effizienz. Dies gilt insbesondere für die 7 Schichten des OSI-RMs, welche nicht zwangsläufig in eine Implementierung umgesetzt werden müssen.
- Die eigentliche Protokollfunktionalität verursacht relativ wenig Aufwand, stattdessen sind u.a. Speicher- und Busstrukturen dominierend.
  (Logisch trivial erscheinende Funktionen sollten mehr beachtet werden!)
- Hochgeschwindigkeitskommunikation erfordert zwangsläufig Anwendungsrechner mit erheblich höheren Transaktionsraten, d.h. Aufgabenwechsel pro Sekunde. Zusätzlich bieten Verbindungswege und Speicherbausteine nach wie vor erhebliches Verbesserungspotential.
  (Anmerkung: Viele Rechner sind nicht oder kaum in der Lage, 10 Mbit/s zu verarbeiten!)
- Ein Kommunikationssystem kann nicht als Summe isolierter Einzelaspekte betrachtet werden.
- Der Entwurf eines komplexen Systems scheint nur mit einer Mischung aus "Top-Down" und "Bottom-up" Vorgehen möglich.

In den letzten Jahren nahm das weltweite Interesse an der grundsätzlichen Thematik der Arbeit erheblich zu. Inzwischen gibt es mehrere spezielle Konferenzen, Arbeitstreffen und komplette (Sonder-)Ausgaben bedeutenderer Zeitschriften zum Thema Hochgeschwindigkeitsprotokolle bzw. deren Realisierung. Allgemein ist eine Verschiebung hin zu den nicht protokollorientierten Aspekte festzustellen. Sehr stark wird dabei die Frage der Aufteilung zwischen Anwendungsrechner und Netzwerkkarte sowie - damit zusammenhängend - zwischen Soft- und Hardware diskutiert.

Obwohl das entwickelte Systemkonzept jeweils eine konkrete Position einnimmt, ist es flexibel genug, um an verschiedene Varianten angepaßt zu werden: die entworfenen Hardwarebaugruppen für Speicherverwaltung, Zeitgeber und asynchrone Prozeßkopplung sind grundsätzlich generischer Natur, die Protokolle sind als (änderbare) Software ausgeführt, und auch das Simulationsprogramm ist änderungsfreundlich aufgebaut. Somit kann die Arbeit - unabhängig von einem physikalischen Aufbau und Test - prinzipiell als Basis für weitergehende Untersuchungen zu Hochleistungsprotokollmaschinen eingesetzt werden.

Neben optimierten Architekturen des Anwendungsrechners sind dabei auch Konzepte zur Mehrfachnutzung eines solchen TpS - entsprechend der mehrfachen Nutzung eines Medienzugangs - möglich. Die Einbettung in das OSI-RM und die Bedeutung von MAC-Adressen eines solchen "Transporter"-Konzeptes sind dabei allerdings noch offene Fragen.

Literaturverzeichnis

L.1 Lehrbücher

[1] M.J. BACH; The Design of the UNIX Operating System, *Prentice-Hall International Edition,* Englewood Cliffs, NJ, USA, 1986.

[2] F. BAUSE; Funktionale Analyse zeitbehafteter Petri-Netze, *Deutscher Universitäts-Verlag (DUV),* Wiesbaden, BRD, 1992.

[3] P. BOCKER; ISDN: Das diensteintegrierende digitale Nachrichtennetz - Konzept, Verfahren, System, *Springer Verlag,* Berlin - Heidelberg, BRD, 1992.

[4] L. CIMINIERA, A. VALENZANO; Advanced Microprocessor Architectures, *Addison-Wesley Publishing Company,* Electronic Systems Engineering Series, 1990.

[5] D. COMER; Interworking with TCP/IP, *Prentice-Hall International Edition,* Englewood Cliffs, NJ, USA, 1988.

[6] A.A.S. DANTHINE, O SPANIOL (Eds.); High Speed Local Area Networks II, *Elsevier Science Publishers B.V. (North-Holland),* IFIP, 1988.

[7] D.W. DAVIES, D.L.A. BARBER, W.L. PRICE, C.M. SOLOMONIDES; Computer Communication Networks and their Protocols, *Wiley Interscience,* New York, NY, USA, 1983.

[8] R.J. DEASINGTON; X.25 Explained: Protocols for Packet Switching Networks, *Ellis Horwood Ltd.,* 1986.

[9] K.E. GANZHORN, K.M. SCHULZ, W. WALTER; Datenverarbeitungssysteme - Aufbau und Arbeitsweise, *Springer Verlag,* Berlin - Heidelberg, FRG, 1981.

[10] W.K. GILOI; Rechnerarchitekturen, *Springer Verlag,* Berlin - Heidelberg, BRD, 1981.

[11] W. GORA, R. SPEYERER; ASN.1, *Datacom,* Pulheim, FRG, 1987.

[12] F. HALSALL; Data Communications, Computer Networks and OSI (second Edition), *Addison-Wesley Publishing Company,* Wokingham, UK, 1988.

[13] R. HÄNDEL, M.N. HUBER; Integrated Broadband Networks - An Introduction to ATM-Based Networks, *Addison-Wesley Publishing Company,* Wokingham, UK, 1991.

[14] J.L. HAMMOND, P.J.P. O'REILLY; Performance Analysis of Local Computer Networks, *Addison-Wesley Publishing Company,* Reading, MA, USA, 1988.

[15] H.L. HARTMANN (Ed.); Circuit and Packet Switching for a Broadband Integrated Services Digital Network (B-ISDN), *B.G. Teubner,* Stuttgart, FRG, 1988.

[16] C.A. HOARE; Communicating Sequential Processes, *Prentice-Hall International Edition,* Englewood Cliffs, NJ, USA, 1985.

[17] F. HOFMANN; Betriebssysteme: Grundkonzepte und Modellvorstellungen, *Teubner Verlag,* Stuttgart, BRD, 1984.

[18] D. HUTCHISON; Local Area Network Architectures, *Addison-Wesley Publishing Company,* Wokingham, UK, 1988.

[19] P. JACKSON; Expertensysteme - Eine Einführung, *Addison-Wesley,* Bonn, FRG, 1987.

[20] M. JOHNSON (Ed.); Protocols for High-Speed Networks II; *Elsevier Science Publishers B.V. (North Holland),* IFIP, 1991.

[21] V.C. JONES; MAP/TOP Networking, *McGraw-Hill,* New York, N.Y., USA, 1988.

[22] F.J. KAUFFELS; Rechnernetzwerksystemarchitekturen und Datenkommunikation, *Bibliographisches Institut,* Reihe Informatik, Band 54, Zürich, Schweiz, 1987.

[23] F.J. KAUFFELS, K. SEITZ; Lösungen durch Netze, *Network Systems GmbH,* Frankfurt, BRD, 1991.

[24] D.E. KNUTH; The Art of Computer Programming, *Addison-Wesley Publishing Company,* Reading, MA, USA, 1973.

[25] B. LAMPSON, M. PAUL, H. SIEGERT; Distributed Systems - Architecture and Implementation, *Springer-Verlag,* New York, NY, USA, 1981.

[26] S.J. LEFFLER, M.K. MCKUSICK, M.J. KARELS, J.S. QUARTERMAN; Das 4.3-BSD-UNIX-Betriebssystem, Design und Implementierung, *Addison-Wesley Publishing Company.,* Bonn, FRG, 1990.

[27] M.M. MANO; Computer System Architecture (second Edition), *Prentice-Hall International Editions,* Englewood Cliffs, NJ, USA, 1990.

[28] J. MARTIN, K.K. CHAPMAN; Local Area Networks - Architectures and Implementations, *Prentice-Hall International Edition,* Englewood Cliffs, NJ, USA, 1989.

[29] C. MEAD, L. CONWAY; Introduction to VLSI Systems, *Addison-Wesley Publishing Company,* Reading, MA, USA. 1979.

[30] K. MEHLHORN; Data Structures and Algorithms 1: Sorting and Searching, *Springer-Verlag,* Berlin, FRG, 1984.

[31] W.W. PETERSON, E.J. WHELDON; Error-Correcting Codes, *MIT Press,* Boston, MA, USA, 1972.

[32] M. DEPRYCKER; Asynchronous Transfer Mode - Solution for B-ISDN, *Ellis Howard Ltd.,* 1991.

[33] G. PUJOLLE (Ed.); High-Capacity Local and Metropolitan Area Networks - Architecture and Performance Issues, *Springer-Verlag,* Berlin - Heidelberg, 1991.

[34] H. RUDIN, R. WILLIAMSON (Eds.); Protocols for High-Speed Networks, *Elsevier Science Publishers B.V. (North-Holland),* IFIP, 1989.

[35] M. SCHWARTZ; Telecommunication Networks: Protocols, Modeling and Analysis, *Addison-Wesley Publishing Company,* Reading, MA, USA, 1988.

[36] H.J. SIEGERT; Betriebssysteme: Eine Einführung (2. Auflage), *Handbuch der Informatik,* Band 4.1, Wien, Österreich, 1989.

[37] G.J. SIMMONS (Ed.); Contemporary Cryptology - The Science of Information Integrity, *IEEE Press,* New York, NY, USA, 1992.

[38] M. SLOMAN, J. KRAMER; Distributed Systems and Computer Networks, *Prentice-Hall International Series in Computer Science,* 1987.

[39] O. SPANIOL, A.A.S. DANTHINE (Eds.); High Speed Networking III, *Elsevier Publishers B.V. (North-Holland),* IFIP, 1991.

[40] W. STALLINGS; Local Networks (second edition), *Macmillan Publishing Company,* New York, NY, USA, 1984.

[41] J. SVOBODA; Codierung zur Fehlerkorrektur und Fehlererkennung, *R. Oldenbourg Verlag,* München, BRD, 1973.

[42] A.S. TANENBAUM; Computer Networks (second Edition), *Prentice-Hall International Edition,* Englewood Cliffs, NJ, USA, 1989.

[43] A.S. TANENBAUM; Structured Computer Organization (third Edition), *Prentice-Hall International Edition,* Englewood Cliffs, NJ, USA, 1990.

[44] C. TROOPER; Local Computer Network Technologies, *Academic Press,* New York, USA, 1981.

[45] N.H.E. WESTE, K. ESHRAGHIAN, Principles of CMOS VLSI Design, *Addison-Wesley Publishing Company,* Reading, MA, USA, 1985.

L.2 Veröffentlichungen

[46] B.W. ABEYSUNDARA, A.E. KAMAL; High-Speed Local Area Networks and their Performance: A Survey, *ACM Computing Surveys,* Vol. 23, No. 2, pp. 221 - 264, June 1991.

[47] H. ABU-AMARA, T. BALRAJ, T. BARZILAI, Y. YEMINI; PSi: A Silicon Compiler for Very Fast Protocol Processing, *in [34],* pp. 181 - 195.

[48] A. ALBANESE, M. GARRETT, A. IPPOLITI, H. IZADPANAH, M. KARR, M. MASZCZAK, D. SHIA; Bellcore METROCORE™ network - A test-bed for metropolitan area network research, *Proceedings of IEEE GLOBECOM'88,* Hollywood, FL, USA, November 1988.

[49] G. ANASTASI, M. CONTI, E. GREGORI; TPR: A Transport Protocol for Real Time Services in an FDDI Environment, *in [20],* pp. 313 - 331.

[50] M. AOKI, T. UCHIYAMA, S. TONAMI, A. HAYAKAWA, H. ICHIKAWA; Protocol processing for high-speed packet switching systems, *Proceedings of International Zurich Seminar,* pp. 125 - 130, Zurich, Switzerland, March 1986.

[51] J. APEL, F. FERRERO, L. GROSSI, J.E. HANSEN; Implementation Techniques for LION (Local Integrated Optical Network), *in [6],* pp. 207 - 221.

[52] R. ARONOFF, K. MILLS, M. WHEATLEY; Transport Layer Performance Tools and Measurements, *IEEE Network,* Vol. 1, No. 3, pp. 21 - 31, July 1987.

[53] H.R. VAN AS, W.W. LEMPPENAU, P. ZAFIROPULO, E.A. ZURFLUH; CRMA-II: A Gbit/s MAC Protocol for Ring and Bus Networks with Immediate Access Capability, *Proceedings of 9th European Fibre Optic Communications and Local Area Networks Conference (EFOC/LAN'91),* paper 4.2.3, pp. 56 - 71, London, UK, June 1991.

[54] M.S. ATKINS; Experiments in SR with different Upcall Program Structures, *ACM Transactions on Computer Systems,* Vol. 6, No. 4, pp. 365 - 392, November 1988.

[55] A New Ring Potocol for HS-PISN based on Asynchronous Transfer Mode, *Protocol ANSI working group X3T9.5/91-089 (FFOL-035),* ISO/IEC JTC1 / SC6/WG6 Private Integrated Services Networking / N 70 (submitted by Japanese National Body), September 1990.

[56] A.J. MCAULEY, C.J. COTTON ; Reliable Broadband Communication Using a Burst Erasure Correcting Code, *Proceedings of SIGCOMM'90,* ACM, pp. 297 - 306, Philadelphia, PA, USA, September 1990.

[57] A.J. MCAULEY, C.J. COTTON; A Self-Testing Reconfigurable CAM, *IEEE Journal of Solid-State Circuits,* Vol. 26, No. 3, pp. 257 - 261, March 1991.

[58] T. BALPH, J. VITTERA; Architectural considerations in the development of an IEEE 802.4 Token Bus chip set, *Proceedings of 4th Phoenix Conference on Computer Communication,* pp. 439 - 443, Phoenix, TX, USA, March 1985.

[59] T.S. BALRAJ, Y. YEMINI; PROMPT - A Destination Oriented Protocol for High Speed Networks, *in [20],* pp. 297 -261.

[60] R.E. BARKLEY, T.P. LEE; A Heap-based Callout Implementation to meet Realtime Needs, *Proceedings of USENIX Summer Conference,* pp. 213 - 222, San Francisco, CA, USA, June 1988.

[61] W. BAUERFELD, R. BRINKHUIJSEN, COSINE - Base for a European OSI-Infrastructure for R&D, *in: R. Speth (Ed.) "Research into Networks and Distributed Applications",* Elsevier Science Publishers B.V. (North Holland), pp. 537 -547, 1988.

[62] B. BEACH; UltraNet: An Architecture for Gigabit Networking, *UltraNet Product Overview,* Ultra Network Technologies, San Jose, CA, USA, September 1990.

[63] A.G. BELL, G. BORIELLO; A single chip NMOS Ethernet controller, *Proceedings IEEE International Solid-State Circuit Conference (ISSCC'83),* pp. 70 - 72, February 1983.

[64] B.N. BERSHAD, E. LAZOWSKA, H. LEVY; PRESTO: A System for Object-oriented Parallel Programming, *Software - Practice and Experience,* Vol. 18, No. 8, pp. 713 - 732, August 1988.

[65] M.A. BEUNDER, J.P. KERNHOF, B. HOEFFLINGER; The CMOS Gate Forest: An Efficient and Flexible High-Performance ASIC Design Environment, *IEEE Journal on Solid-State Circuits,* Vol. 23, No. 2, pp. 387 - 399, April 1988.

[66] M.A. BEUNDER, B. HOEFFLINGER, J.P. KERNHOF; New Directions in Semicustom Arrays, *IEEE Journal on Solid-State Circuits,* Vol. 23, No. 3, pp. 728 - 735, June 1988.

[67] E.W. BIERSACK, D.C. FELDMEIER; Transport Protocol Issues for ATM-based Networks, *Proceedings of 8th European Fibre Optic Communications and Local Area Networks Conference (EFOC/LAN'90),* pp. 104 - 113, Munich, FRG, June 1990.

[68] E.W. BIERSACK, C.J. COTTON, D.C. FELDMEIER, A.J. MCAULEY; An Overview of the TP++ Transport Protocol, distributed via public mailbox (erbi/dcf@bellcore.com), April 1991.

[69] E.W. BIERSACK; Connection Mgmt. using Synchronized Clocks, *in [39],* pp. 225 - 238.

[70] E.W. BIERSACK, C.J. COTTON, D.C. FELDMEIER, A.J. MCAULEY, W.D. SINCOSKIE; Gigabit Networking Research at Bellcore, *IEEE Network,* Vol. 6, No. 2, pp. 42 - 48, March 1992.

[71] A.D. BIRELL, B.J. NELSON; Implementing Remote Procedure Calls, *ACM Computer Systems,* Vol. 2, No. 1, pp. 39 - 59, February 1984.

[72] R.J. BOEHM; Standardized Fiber Optic Transmission Systems - A Synchronous Optical Network View, *IEEE Journal on Selected Areas in Communications,* Vol. SAC-4, No. 9, pp. 1424 - 1431, December 1986.

[73] T. BOLAND; Stable Implementation Agreements for OSI Protocols, *NIST Report SP/500/177,* U.S. Department of Commerce, Gaithersburg, MD, USA, December 1989.

[74] D.A. BORMAN; Implementing TCP/IP on a Cray Computer, *Computer Communications Review,* Vol. 19, No. 2, pp. 11 - 15, February 1989.

[75] M. BOSCH; Kopplung von Kommunikationsnetzen: Architekturen, Leistungsuntersuchungen und eine Beispielsimplementierung, *Dissertationsschrift,* Universität Stuttgart, 1992.

[76] J.Y. LE BOUDEC; The Asynchronous Transfer Mode: a tutorial, *Computer Networks and ISDN Systems,* Vol. 24, No. 2, pp. 279 - 309, February 1992.

[77] S.O. BRADNER; Testing Multiprotocol Routers - How Fast is Fast Enough?, *Data Communications International,* No. 2, pp. 71 - 86, February 1991.

[78] T. BRAUN, M. ZITTERBART; A Parallel Implementation of XTP on Transputers, *Proceedings of 16th IEEE Conference on Local Computer Networks (LCN'91),* pp. 321 - 331, Minneapolis, MN, USA, October 1991.

[79] C. BRENDAN, S. TRAW, J.M. SMITH; A High-Performance Host Interface for ATM Networks, *Proceedings of the SIGCOMM'91,* ACM, pp. 317 - 325, Zurich, Switzerland, September 1991.

[80] G.M. BROWN, M.G. GOUDA, R.E. MILLER; Block Acknowledgement: Redesigning the Window Protocol Networks, *Proceedings of SIGCOMM'89,* ACM, pp. 128 - 134, Austin, TX, USA, September 1989.

[81] P.A. BUHR, R.A. STROOBOSSCHER; The μSystem: Providing Light-weight Concurrency on Shared-memory Multiprocessor Computers running UNIX, *Software - Practice and Experience,* Vol. 20, No. 9, pp. 929 - 964, September 1990.

[82] F.M. BURG, PUT P.; X.25: It's Come a Long Way, *in: J. Raviv (Ed.) "Computer Communication Technologies for the 90's",* Elsevier Science Publishers B.V. (North Holland), pp. 280 -286, 1988.

[83] F.M. BURG, N. IORI; Networking of Networks: Internetworking According to OSI, *IEEE Journal of Selected Areas in Communications,* Vol. 7, No. 7, pp. 1131 - 1142, September 1989.

[84] J. T. BUTLER; Multimedia Systems, *Computer (special issue),* Vol. 24, No. 10, October 1991.

[85] B. BUTSCHER, L. HENCKEL, T. LUCKENBACH; Die Kommunikationsplattform BERKOM für multimediale Anwendungen, *Informatik Spektrum,* Springer-Verlag, Jahrgang 14, Heft 5, S. 261 - 269, Oktober 1991.

[86] W. BUX; Local-Area Subnetworks: A Performance Comparison, *IEEE Transactions on Communications,* Vol. COM-29, No. 10, pp. 1465 - 1473, October 1991.

[87] W. BUX, P. KERMANI, W. KLEINOEDER; Performance of an improved Data Link Protocol, *Proceedings of 9th International Conference on Computer Communication (ICCC'88),* pp. 251 - 258, Tel Aviv, Israel, November 1988.

[88] M. CAPURRO, L. VIVOLI; Telecommunication Services Evolution from LANs to MANs, *Proceedings of 9th European Fibre Optic Communications and Local Area Networks Conference (EFOC/LAN'91),* paper 5.5.1, pp. 332 - 340, London, UK, June 1991.

[89] G. CARLE; Modellierung, Simulation und Verifikation einer parallelen Kommunikationsrechner-Architektur für OSI-konforme Hochgeschwindigkeitsnetze, *Diplomarbeit,* IND D 1160, Juli 1992.

[90] J.B. CARTER, W. ZWAENEPOEL; Optimistic Implementation of Bulk Data Transfer Protocols, *Proceedings of Conference on Measurements and Modeling of Computer Systems (SIGMETRICS'89),* ACM, pp. 61 - 69, Berkeley, CA, USA, May 1989.

[91] V. CATANIA, S. CAVALIERI, M. IUDICA, L. VITA; A parallel VLSI Architecture for High-Speed Protocol Implementation, *Proceedings 10th International Conference on Computer Communication (ICCC'90),* pp. 276 - 283, New Delhi, India, November 1990.

[92] D.C.Y. CHAN, R.H.S. HARDY; Analysis of Transport Layer and LLC Sublayer Performance within a Local Area Network, *Proceedings of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing,* pp. 230 - 235, Singapore, June 1989.

[93] D.R. CHERITON, C.L. WILLIAMSON; VMTP as the Transport Layer for High-Performance Distributed Systems, *IEEE Communications Magazine,* Vol. 27, No. 6, pp. 37 - 44, June 1989.

[94] G. CHESSON; The Protocol Engine, *UNIX Review,* Vol. 5, No. 9, pp. 70 - 77, September 1987.

[95] G. CHESSON; XTP/PE Design Considerations, *in [34],* pp. 27 - 34.

[96] G. CHESSON; The Evolution of XTP, *in [39],* pp. 15 - 24.

[97] D.K. CHOI, B.G. KIM; The Expected (Not Worst-Case) Throughput of the Ethernet Protocol, *IEEE Transactions on Computers,* Vol. 40, No. 3, pp. 245 - 252, March 1991.

[98] S. CHRISTENSEN, H. SCHEUER, M. SKOV; A Flexible 1 Gbit/s Implementation of Physical Layer Protocols, *in [20],* pp. 173 - 199.

[99] I. CIDEON, Y. OFEK; Metaring - A Full-Duplex Ring with Fairness and Spatial Reuse, *Research Report RC 14961 (#66845),* IBM Research Div., Yorktown Heights, NY, USA, September 1989.

[100] I. CIDON, I. GOPAL, A. SEGALL; Fast Connection Establishment in High Speed Networks, *Proceedings of SIGCOMM'90,* ACM, pp. 287 - 295, Philadelphia, PA, USA, September 1990.

[101] CIO - Coordination, Implementation and Operation of Multimedia Services, *RACE'92, Technical description of project R2060 (CIO),* pp. A206 - 208, Brussels, Belgium; March 1992.

[102] D.D. CLARK; Window and Acknowledgement Strategy in TCP, *Network Information Center, RFC 813,* SRI International, Menlo Park, CA, USA, July 1982.

[103] D.D. CLARK; Names, Addresses, Ports and Routes in TCP/IP, *Network Information Center, RFC 814,* SRI International, Menlo Park, CA, USA, July 1982.

[104] D.D. CLARK; Modularity and Efficiency in Protocol Implementation, *Network Information Center, RFC 817,* SRI International, Menlo Park, CA, USA, July 1982.

[105] D.D. CLARK; The Structuring of Systems Using Upcalls, *Proceedings of 10th Symposium on Operating System Principles,* ACM, pp. 171 - 180, Orcas Island, WA, USA, December 1985.

[106] D.D. CLARK, M.L. LAMBERT, L. ZHANG; NETBLT: A Bulk Data Transfer Protocol, *Network Information Center, RFC 969,* SRI International, Menlo Park, CA, USA, March 1987.

[107] D.D. CLARK, M.L. LAMBERT, L. ZHANG; NETBLT: A High Throughput Transport Protocol, *Proceedings of SIGCOMM'87,* ACM, pp. 353 - 359, Stowe, VT, USA, August 1987.

[108] D.D. CLARK, V. JACOBSON, J. ROMKEY, H. SALWEN; An Analysis of TCP Processing Overhead, *IEEE Communications Magazine,* Vol. 27, No. 6, pp. 23 - 29, June 1989.

[109] D.D. CLARK, D.L. TENNENHOUSE; Architectural Considerations for a New Generation of Protocols, *Proceedings of SIGCOMM'90,* ACM, pp. 200 - 208, Philadelphia, PA, USA, August 1990.

[110] L. CLYNE; Lower Layer OSI Addressing (Tutorial), *Computer Networks and ISDN Systems,* North-Holland, Vol. 17, pp. 279 - 281, 1989.

[111] R.G. COCHRAN, G.R. SAMSEN; A VLSI chip set for token ring LANs, *Proceedings of 4th Phoenix Conference on Computer Communication,* pp. 428 - 431, Phoenix, TX, USA, March 1985.

[112] E.I. COHEN, G.M. KING, J.T. BRADY; Storage hierarchies, *IBM Systems Journal,* Vol. 28, No. 1, January 1989.

[113] R. COLELLA, R. ARONOFF, K. MILLS; Performance Improvements for ISO Transport, *Proceedings of 9th IEEE Data Communications Symposium,* pp. 9 - 16, Whistler Mountain, B.C., Canada, September 1985.

[114] I. CONNOR, S. MARGERY; 1M-bit video RAMs offer speed for high-resolution graphics displays, *EDN,* pp. 79 - 82, March 1988.

[115] D. CONNOR; High-Density PLDs, *EDN,* pp. 76 - 88, January 1992.

[116] G.H. COOPER; An Argument for Softlayering of Protocols, *Technical Report MIT-TR-300,* Laboratory for Computer Science, Boston, MA, USA, May 1983.

[117] E.C. COOPER, R.P. DRAVES; C Threads, *Technical Report,* Carnegie Mellon University, Pittsburg, PA, USA, March 1987.

[118] E.C. COOPER, P.A. STEENKISTE, R.D. SANSOM, B.D. ZILL; Protocol Implementation on the Nectar Communication Processor, *Proceedings of the SIGCOMM'90,* ACM, pp. 135 - 144, Philadelphia, PA, USA, September 1990.

[119] G.V. CORMACK; Short Communication: A Microkernel for Concurrency in C, *Software - Practice and Experience,* Vol. 18, No. 5, pp. 485 - 491, May 1988.

[120] A.A.S. DANTHINE, P. HENQUET, B. HAUZEUR, C. CONSTANTINIDIS, D. FAGNOULE, V. CORNETTE; Access Rate Measurement in the BWN Environment, *in [6],* pp. 89 - 115.

[121] A.A.S. DANTHINE; Commmunication Support for Distributed Systems - OSI versus Special Protocols (invited paper), *in: G.X. Ritter (Ed.) "Information Processing 89"*, Elsevier Publishers B.V. (North-Holland), pp. 181 - 190, IFIP, 1989; also published in: *Proc. of 11th World Computer Congress on Information Processing*, pp. 181 - 190, San Francisco, CA, USA, September 1989.

[122] A.A.S. DANTHINE; A New Transport Protocol for the Broadband Environment, *Proceedings of IFIP Workshop on Broadband Communications*, Estoril, Portugal, January 1992.

[123] P.B. DANZIG, S. MELVIN; High Resolution Timing with Low Resolution Clocks and A Microsecond Resolution Timer for Sun Workstations, *Operating Systems Review*, ACM Press, Vol. 24, No. 1, pp. 23 - 26, January 1990.

[124] B.S. DAVIE; A Host-Network Interface Architecture for ATM, *Proceedings of the SIGCOMM'91*, ACM, pp. 307 - 315, Zurich, Switzerland, September 1991.

[125] M. DEASON (Ed.); Broadband Technology, *Electrical Communication*, ALCATEL report, Vol. 64, No. 2/3 (whole issue), June 1990.

[126] D. DELODDERE, G. GASTAUD, R. PESCHI; DQDB MAN - a step towards B-ISDN, *Proceedings of 8th European Fibre Optic Communications and Local Area Networks Conference (EFOC/LAN'90)*, pp. 87 - 90, Munich, FRG, June 1990.

[127] G. DERMLER; Entwurf eines Transportsystems für Hochgeschwindigkeitsnetze, *Diplomarbeit*, IND D 1111, Dezember 1991.

[128] C.R. DHAS, V.K. KONAGI; X.25: An Interface to Public Packet Networks, *IEEE Communications Magazine*, Vol. 24, No. 9, pp. 18 - 25, September 1986.

[129] H. DIETSCH, R. ULRICH; Ein OSI-Kommunikationswerk auf Transputer-Basis für den Einsatz in der dezentralen Prozeßrechentechnik, *Tagungsband Prozeßrechensysteme'91*, Informatik Fachbericht 269, Springer Verlag, S. 271 - 280, 1991.

[130] J. DING, K. ROTHERMEL, K. URBSCHAT; ECSE: An Efficient Environment for Layered Communication Protocols, *Proceedings of 2nd IEEE Workshop on Future Trends of Distributed Computing Systems*, pp. 400 - 414, Cairo, Egypt, September / October 1990.

[131] C. DIOT, M.N.X. DANG; Performances of a Communication Circuit Intended for the Implementation of High Level Layers of the OSI Model (or equivalent), *Proceedings of IEEE Conference on Systems Engineering*, pp. 533 - 536, Dayton, OH, USA, August 1989.

[132] C. DIOT, M.N.X. DANG; A High Performance Implementation of OSI Transport Protocol Class 4: Evaluation and Perspectives, *Proceedings of 15th IEEE Conference on Local Computer Networks (LCN'90)*, pp. 223 - 230, Minneapolis, MN, USA, October 1990.

[133] C. DIOT, V. ROCA; XTP/KRM Implementation on a Transputer Network, *Proceedings of 16th IEEE Conference on Local Computer Networks (LCN'91)*, pp. 310 - 320, Minneapolis, MN, USA, October 1991.

[134] R. A. DIRVIN, A.R. MILLER; The MC68824 token bus controller - VLSI for the factory LAN, *Proceedings of IEEE Micro*, pp. 15 - 25, June 1986.

[135] R.W. DOBINSON, M.D. SZPAK; Interfacing to Ethernet using VLSI protocol chips, *Interfaces in Computing*, Vol. 3, No. 3/4, pp. 173 - 187, September - December 1985.

[136] W. DOERINGER, D. DYKEMAN, M. KAISERSWERTH, B. MEISTER, H. RUDIN, R. WILLIAMSON; A Survey of Light-Weight Transport Protocols for High-Speed Networks, *IEEE Transactions on Communications*, Vol. 38, No. 11, pp. 2025 - 2039, November 1990; also published as *Research Report RZ 1980 (#70152)*, IBM Research Division, Rüschlikon, Switzerland, May 1990.

[137] P. DOMSCHITZ, M. SIEGEL; Dual Ring Usage in FDDI: Implications on Systems and Management Architecture, *Computer Communications*, Vol. 15, No. 5, pp. 447 - 457, September 1992.

[138] O. EBINGER; Entwurf einer hardwareunterstützten Pufferverwaltung, *Studienarbeit*, IND S1 1113, April 1992.

[139] S.W. EDGE; An Adaptive Timeout Algorithm for Retransmission Across a Packet Switching Network, *Proceedings of SIGCOMM'83*, ACM, USA, August, 1983.

[140] W. EFFELSBERG, A. FLEISCHMANN; Das ISO-Referenzmodell für offenen Systeme und seine sieben Schichten, *Informatik Spektrum*, Heft 9, S. 280 - 299, 1986.

[141] M. EGERTER; Untersuchung von Zeitgeberimplementierungen und Entwurf einer ausgewählten Variante, *Studienarbeit*, IND S2 110?, April 1992.

[142] U. ELZUR; Getting the most out of Ethernet's data link and transport layers, *Data Communications,* pp. 53 - 59, March 1989.

[143] I. ERICKSON; Protocol controller chip manages X.25 interface, *Computer Design,* Vol. 24. No. 8, pp. 78 - 81, September 1985.

[144] J. MCFARLANE; Evolution of the Public Network for High Speed Services, *Participants Proceedings of Workshop on Broadband Communications,* pp. 2 - 10, Estoril, Portugal, January 1992.

[145] D.C. FELDMEIER; Multiplexing Issues in Communication System Design, *Proceedings of SIGCOMM'90,* ACM, pp. 209 - 219, Philadelphia, PA, USA, September 1990.

[146] D.C. FELDMEIER, E.W. BIERSACK; Comparison of Error Control Protocols for High Bandwidth-Delay Product Networks, *in [20],* pp. 271 - 295.

[147] D.C. FELDMEIER, A.J. MCAULEY; Ordering Issues in Communication System Design, *to appear in: B. Pehrson, P. Gunningberg (Eds.) "Protocols for High-Speed Networks III", Elsevier Science Publishers B.V. (North-Holland),* IFIP, Stockholm, Sweden, May 1992.

[148] Suomalainen OSI-profiili (Finnish Government OSI Profile), Version 1.0, *Tiedonsiirron yhteistyöelin,* Finnland, February 1990.

[149] M. FINK; Systemlösung mit FDDI-Chipsatz, *Design & Elektronik,* Heft 8, S. 30 - 33, April 1991.

[150] W. FISCHER, E.H. GÖLDNER, N. HUANG; The Evolution from LAN/MAN to Broadband ISDN, *Proceedings of International Conference on Communication (ICC'91),* pp. 128 - 134, Denver, CO, USA, June 1991.

[151] R. FITZGERALD, R.F. RASHID; The Integration of Virtual Memory Management and Interprocess Communication in Accent, *ACM Transcations on Computer Systems,* Vol. 4, No. 2, pp. 147 - 177, May 1986.

[152] A. FLEISCHMANN, S.T. CHIN, W. EFFELSBERG; Specification and Implementation of an ISO Session Layer, *IBM Systems Journal,* Vol. 26, No. 3, pp. 255 - 274, March 1987.

[153] J.G. FLETCHER; An Arithmetic Checksum for Serial Transmission, *IEEE Transactions on Communications,* Vol. COM-30, No. 1, pp. 92 - 106, January 1982.

[154] W.R. FRANTA; Measurement and Analysis of HYPERchannel Networks, *IEEE Transactions on Computers,* Vol. C-33, No. 3, pp. 249 - 260, March 1984.

[155] A.G. FRASER; The Universal Receiver Protocol, *in [34],* pp. 19 - 25.

[156] A.G. FRASER, W.T. MARSHALL; Data Transport in a Byte Stream Network, *IEEE Journal on Selected Areas in Communication,* Vol. 7, No. 7, pp. 1020 - 1033, September 1989.

[157] G. GEIGER, L. LERACH; ISDN-oriented modular VLSI chip set for central-office and PABX applications, *IEEE Journal on Selected Areas in Communications,* Vol. SAC-4, No. 8, pp. 1268 - 1274, November 1986.

[158] M. GERLA, H.W. CHAN; Window Selection in Flow Controlled Networks, *Proceedings of SIGCOMM'85,* ACM, pp. 84 - 92, August 1985.

[159] D. GIARRIZO, M. KAISERSWERTH, T. WICKI, R.C. WILLIAMSON; High-Speed Parallel Protocol Implementation, *Participants Proceedings of IFIP Workshops "Protocols for High Speed Networks",* IFIP WG 6.1/6.4, Rüschlikon, Switzerland, May 1989.

[160] Gigabit Network Testbeds, *IEEE Computer (Special Report),* Vol. 23, No. 9, pp. 77 - 80, September 1990.

[161] I. GOPAL, R. ROM; Improving ARQ Protocol Performance by Multiple FIFO Buffers, *Proceedings of IEEE INFOCOM'90,* pp. 426 - 432, San Francisco, CA, USA, June 1990.

[162] W. GORA, R. SPEYERER; ASN.1 - Das aktuelle Schlagwort, *Informatik Spektrum,* Heft 11, S. 207 - 211, 1988.

[163] L. GORYS; FDDI-Board für ATs, *Design & Elektronik,* Heft 8, S. 40 - 46, April 1991.

[164] A. GOSCINSKI, W. ZHU; The Development and Performance Study of the Rhodos Reliable Datagram Protocol (RRDP), *Proceedings of International Conference on Computer Communication (ICCC'90),* pp. 388 - 396, New Delhi, India, November 1990.

[165] D.J. GREAVES, I.D. WILSON; Cambridge HSLAN Protocol Review, *in [34],* pp. 257 - 268.

[166] P. GUNNINGBERG; Innovative communication processors: A survey, *Technical Report SICS-R-87003,* Swedish Institute of Computer Science, Spanga, Sweden, May 1987.

[167] P. GUNNINGBERG, M. BJÖRKMAN, E. NORDMARK, S. PINK, P. SJÖDIN, J.E. STRÖMQUIST; Application Protocols and Performance Benchmarks, *IEEE Communications Magazine,* Vol. 27, No. 6, pp. 30 - 36, June 1989.

[168] Z. HAAS; A Communication Architecture for High-speed Networking, *Proceedings of IEEE INFOCOM'90,* pp. 433 - 441, San Francisco, CA, USA, June 1990.

[169] Z. HAAS; A Protocol Structure for High-Speed Communication over Broadband ISDN, *IEEE Network Magazine,* Vol. 5, No. 1, pp. 64 - 70, January 1991.

[170] Y. HAN; An Optimal Linked List Prefix Algorithm on a Local Memory Computer, *IEEE Transactions on Computers,* Vol. 40, No. 10, pp. 1149 - 1152, October 1991.

[171] R. HÄNDEL, M.N. HUBER; B-ISDN Network Capabilities and Evolution Aspects, *Proceedings of Workshop on Broadband Communications,* pp. 12 - 22, Estoril, Portugal, January 1992.

[172] S. HEATLEY, D. STOKESBERRY; Measurements of a Transport Implementation Running over an IEEE 802.3 Local Area Network, *Proceedings of IEEE Computer Networking Symposium,* pp. 34 - 43, Washington, D.C., USA, April 1988.

[173] S. HEATLEY, D. STOKESBERRY; Analysis of Transport Measurements over a Local Area Network, *IEEE Communications Magazine,* Vol. 27, No. 6, pp. 16 - 22, June 1989.

[174] D.B. HEHMANN, M.G. SALMONY, H.J. STÜTTGEN; High Speed Transport Systems for Multi-Media Applications, *in [34],* pp. 303 - 321, also published as *Technical Report No. 43.8903,* IBM European Networking Center, Heidelberg, FRG, April 1989.

[175] D.B. HEHMANN, M.G. SALMONY, H.J. STÜTTGEN; Transport services for multimedia applications on broadband networks, *Computer Communications,* Vol. 13, No. 4, pp. 197 - 203, May 1990.

[176] D.B. HEHMANN, R.G. HERRTWICH, R. STEINMETZ; Creating HeiTS: Objectives of the Heidelberg High-Speed Transport System, *Technical Report No. 43.9102,* IBM European Networking Center, Heidelberg, FRG, 1991.

[177] D. HEHMANN, B. KÖHLER, N. LUTTENBERGER, L. MACKERT, W. SCHULZ, H. STÜTTGEN; Implementation Experience with a Communication Subsystem for B-ISDN, *in [39],* pp. 305 - 320.

[178] M. HEIN; Modulare Architektur beseitigt Engpässe, *VMEbus,* Heft 3, pp. 31 - 34, April 1990.

[179] B. HEINRICHS, M. RUPPRECHT; High Speed Communication with Standard Protocols, *Proceedings of IEEE GLOBECOM'91,* Phoenix, AZ, USA, December 1991.

[180] R.G. HERRTWICH, R. STEINMETZ; Towards Integrated Multimedia Systems: Why and How, *Technical Report No. 43.9101,* IBM European Networking Center, Heidelberg, FRG, 1991.

[181] C.A.R. HOARE; Communicating sequential Processes, *Communications of the ACM,* Vol. 21, No. 8, pp. 666 - 677, August 1978.

[182] B. HÖFFLINGER; ASICs zum Spartarif, *Elektronik,* Heft 26, S. 36 - 43, Dezember 1990.

[183] A. HOOPER, R.M. NEEDHAM; The Cambridge Fast Ring Networking System, *IEEE Transactions on Computers,* Vol. 37, No. 10, pp. 1214 - 1223, October 1988.

[184] A. HOOPER; Design and Use of High-Speed Networks in Multimedia Applications, *in [39],* pp. 25 - 38.

[185] E. HORSTMANN; 75 Jahre Fernsprechen in Deutschland, *Jubiläumsschrift,* Bundesdruckerei Bonn, Deutschland, 1952.

[186] C. HUITEMA, A. DOGHRI; A High Speed Approach for the OSI Presentation Protocol, *in [34],* pp. 277 - 287.

[187] N.C. HUTCHINSON, L.L. PETERSON; The x-Kernel: An Architecture for Implementing Network Protocols, *IEEE Transactions on Software Engineering,* Vol. 17, No. 1, pp. 64 - 76, January 1991.

[188] M.P. HUTH; A VLSI chip set for the IEEE 802.4 Token Bus, *Proceedings of 4th Intl. Phoenix Conference on Computer Communication,* pp. 434 - 438, Phoenix, TX, USA, March 1985.

[189] K. IMAI, T. HONDA, H. KASAHARA, T. ITO; ATMR: Ring Architecture for Broadband Networks, *NTT Technical Report,* NTT Communication Laboratories, Tokyo, Japan, December 1990.

[190] Integrated Services Digital Network, *IEEE Journal on Selected Areas in Communications (special issue),* Vol. 4, No. 4, May 1986.

[191] ISO Development Environment at NRTC, *Technical Report,* Northrop Research and Technology Center, Palos Verdes Peninsula, CA, USA, November 1988.

[192] V. JACOBSON; Congestion Avoidance and Control, *Proceedings of SIGCOMM'88,* ACM, pp. 314 - 329, Stanford, CA, USA, August 1988.

[193] R. JAIN; Divergence of Timeout Algorithms for Packet Retransmissions, *Proceedings of 5th International Phoenix Conference on Computers and Communications (IPCCC'86),* pp. 53 - 65, Scottsdale, AZ, USA, March 1986.

[194] R. JAIN; A Timeout-Based Congestion Control Scheme for Window Flow-Controlled Networks, *IEEE Journal on Selected Areas in Communications,* Vol. SAC-4, No. 7, pp. 1162 - 1167, October 1986.

[195] R. JAIN; Congestion Control in Computer Networks: Issues and Trends, *IEEE Network Magazine,* Vol. 4, No. 3, pp. 24 - 30, May 1990.

[196] N. JAIN, M. SCHWARTZ, T.B. BASHKOW; Transport Protocol Processing at Gbps Rates, *Proceedings of SIGCOMM'90,* ACM, pp. 188 - 199, Philadelphia, PA, USA, August 1990.

[197] R. JAIN; Myths About Congestion Management in High-Speed Networks, *Research Report,* Digital Equipment DEC-TR-726, Littleton, MA, January 1991.

[198] M.N. JENSEN, M. SKOV; VLSI-Architectures Implementing Lower Layer Protocols in very High Data Rate LANs, *in [6],* pp. 187 - 205.

[199] M.N. JENSEN, M. SKOV; Multi-Processor Based High Speed Communication Systems, *Proceedings of the European Fibre Optic Communications and Local Area Network Conference (EFOC/LAN'89),* Amsterdam, Netherlands, July 1989.

[200] M. KAISERSWERTH; The Parallel Protocol Engine, *IEEE Journal on Selected Areas in Communications,* to appear 1993.

[201] H. KANAKIA, F. TOBAGI; Performance Measurements of a Data Link Protocol, *Proceedings of International Conference on Communication (ICC'86),* pp. 116 - 120, Toronto, Canada, June 1986.

[202] H. KANAKIA, D.R. CHERITON; The VMP Network Adapter Board (NAB): High Performance Network Communication for Multiprocessors, *Proceedings of SIGCOMM'88,* ACM, pp. 175 - 187, Stanford, CA, USA, August 1988.

[203] H. KANAKIA, D.R. CHERITON; Universal Network Device Interface Protocol (UNDIP), *Proceedings of 13th IEEE Conference on Local Computer Networks (LCN'88),* pp. 301 - 309, Minneapolis, MN, USA, October 1988.

[204] P. KARN, C. PARTRIDGE; Improving Round-Trip Time Estimates in Reliable Transport Protocols, *ACM Computer Communication Review,* Vol. 17, No. 5, pp. 2 - 7, 1988.

[205] M.J. KAROL, R.D. GITLIN; High-Performance Optical Local and Metropolitan Area Networks: Enhancements of FDDI and IEEE 802.6 DQDB (invited paper), *IEEE Journal on Selected Areas in Communications,* Vol. 8, No. 8, pp. 1439 - 1448, October 1990.

[206] M. KAWARASAKI, B. JABBARI; B-ISDN Architecture and Protocol, *IEEE Journal on Selected Areas in Communications,* Vol. 9, No. 9, pp. 1405 - 1415, December 1991.

[207] T.J. KING, I. GALLAGHER; ORWELL - A Multiservice Network Protocol, *Proceedings of 8th European Fibre Optic Communications and Local Area Networks Conference (EFOC/LAN'90),* pp. 91 - 96, Munich, FRG, June 1990.

[208] H. KIRAM; Detaillierte Simulation eines ISO-Protokollstacks, *Diplomarbeit,* IND D 1139, Mai 1992.

[209] J. KREIDL, G. RÜCKER; Zusatzschub für den VME-Bus, *Elektronik Journal,* Heft 24, S. 140 - 144, Dezember 1989.

[210] A.S. KRISHNAKUMAR, B. KRISHNAMURTHY, K. SABNANI; Translation of Formal Protocol Specifications to VLSI Designs, *in: H. Rudin, C.H. West (Eds.): "Specification, Verification and Testing of Protocols VII",* pp. 375 - 390, Elsevier Publishers B.V. (North-Holland), IFIP, 1988.

[211] A.S. KRISHNAKUMAR, K. SABNANI; VLSI Implementations of Communication Protocols - A Survey, *IEEE Journal on Selected Areas in Communications,* Vol. 7, No. 7, pp. 1082 - 1090, September 1989.

[212] P. KRÖGER; OSI zum Anfassen, *Elektronik,* Heft 4, 1991.

[213] P.J. KÜHN; Sortierverfahren und Beispiele zum Heapsort-Vorgehen, *Vorlesungsunterlagen "Datenverarbeitung I",* Kapitel 3.3, Universität Stuttgart, Dezember 1990.

[214] R. KUN; A VLSI approach to support LAPD in an ISDN exchange termination, *Proceedings of Intl. Conference on Communication (ICC'86),* pp. 760 - 765, Boston, MA, USA, June 1986.

[215] M. LAHTI, J. MALKA, O. MARTIKAINEN, E. OJANPERÄ; OSI-Protocol Performance Measurements on FDDI, *Proceedings of the 9th European Fibre Optic Communications and Local Area Network Conference (EFOC/LAN'91),* paper 4.8.4, pp. 215 - 220, London, UK, June 1991.

[216] V. LASKER, M. LIEN, E. BENHAMOU; An Architecture for High Speed Protocol Implementations, *Proceedings of IEEE INFOCOM'84,* pp. 156 - 164, San Francisco, CA, USA, April 1984.

[217] P. MARTINI, M. RUPPRECHT; Designing High Speed Controllers for High Speed Local Area Networks, *Proceedings of IEEE GLOBECOM'89,* pp. 170 - 174, Dallas, TX, USA, November 1989.

[218] W.A. MASON; VMTP: A High Performance Transport Protocol, *ConneXions - The Interoperability Report,* Interop Inc., Vol. 4, No. 6, pp. 2 - 10, June 1990.

[219] J.L. MASSEY; An Introduction to Contemporary Cryptology (invited paper), *Proceedings of the IEEE,* Vol. 76, No. 5, pp. 533 - 549, May 1988.

[220] N. F. MAXEMCHUK, M. EL ZARKI; Routing and Flow Control in High-Speed Wide-Area Networks, *Proceedings of the IEEE,* Vol. 78, No. 1, pp. 204 - 221, January 1990.

[221] T.Y. MAZRAANI, G.M. PARULKAR; Specification of a Multipoint Congram-oriented High Performance Internet Protocol, *Proceedings of IEEE INFOCOM'90,* pp. 450 - 457, San Francisco, CA, USA, June 1990.

[222] B.W. MEISTER, P.A. JANSON, L. SVOBODOVA; Connection-Oriented Versus Connectionless Protocols: A Performance Study, *IEEE Transactions on Computers,* Vol. C-34, No. 12, pp. 1164 - 1173, December 1985.

[223] B.W. MEISTER; A Performance Study of the ISO Transport Protocol, *IEEE Transactions on Computers,* Vol. 40, No. 3, pp. 253 - 262, March 1991.

[224] H.E. MELEIS, A.N. TANTAWY; The Modular Communication Machine (MCM): An Architecture for High Performance Networks, *Research Report RC 14541 (#65062),* IBM Research Division, Yorktown Heights, NY, USA, April 1989.

[225] I. MILOUCHEVA, K. REBENSBURG; XTP Service Classes and Routing Strategy for an Integrated Services Broadband Environment, *Technical Report,* University of Berlin (TUB), January 1992.

[226] T. MINAMI; A 200 Mb/s Synchronous TDM Loop Optical LAN Suitable for Multiservice Integration, *IEEE Journal on Selected Areas in Communications,* Vol. SAC-3, No. 6, pp. 849 - 858, November 1985.

[227] P. MINET; Performance evaluation of GAM-T-103 Real-time transfer protocols, *Proceedings of IEEE INFOCOM'89,* pp. 766 - 772, Ottawa, Canada, April 1989.

[228] B. MCMINN; Chip accelerates sorting and searching, *EDN News,* pp. 14 - 16, March 1987.

[229] P. MISSEL; Funktionelle Verifikation eines Multiprozessor-Transportsystems auf der Ebene von Hardwarekomponenten, *Diplomarbeit,* IND D 1159, Juni 1992.

[230] D. MITRA, J.B. SEERY; Dynamic Adaptive Windows for High Speed Data Networks: Theory and Simulations, *Proceedings of SIGCOMM'90,* pp. 30 - 40, Philadelphia, PA, USA, September 1990.

[231] G. MITYKO; The FDDI Follow-On LAN, *Proceedings of 9th European Fibre Optic Communications and Local Area Networks Conference (EFOC/LAN'91),* paper 4.8.2, pp. 201 - 205, London, UK, June 1991.

[232] J.C. MOGUL, R.F. RASHID; The Packet Filter: An Efficient Mechanism for User-level Network Code, *Operating Systems Review (special issue),* ACM SIGOPS, Vol. 21, No. 5, pp. 39 - 51, November 1987.

[233] J.C. MOGUL, A. BØRG; The Effect of Context Switches on Cache Performance, *Operating Systems Review (special issue),* ACM SIGOPS, Vol. 25, No. 2, pp. 75 - 84, April 1991.

[234] E. MUMPRECHT, D. GANTENBEIN, R. HAUSER; Timers in OSI Protocols - Specification versus Implementation, *Proceedings of International Zurich Seminar (IZS'88),* paper C1.1, pp. 93 - 98, Zurich, Switzerland, March, 1988.

[235] S.L., MURPHY A.U. SHANKAR; Service Specification and Protocol Construction for the Transport Layer, *Proceedings of SIGCOMM'88,* ACM, pp. 88 - 97, Stanford, CA, USA, August 1988.

[236] A. NAKASSIS; Fletcher's Error Detection Algorithm: How to implement it efficiently and how to avoid the most common pitfalls, *Computer Communication Review,* ACM Press, Vol. 18, No. 5, pp. 63 - 88, October 1988.

[237] L. NEDERGÅRD, S.M. PEDERSEN, S. NIELSEN, M. SKOV; On the Design of Generic High-Speed Physical and Media Access Protocol Processors, *in [34],* pp. 197 - 216.

[238] A.N. NETRAVALI, W.D. ROOME, K. SABNANI; Design and Implementation of a High-Speed Transport Protocol, *IEEE Transactions on Communications,* Vol. 38, No. 11, pp. 2010 - 2024, November 1990.

[239] A. NEUMANN; Untersuchung der ISO TP4 PrüfsummenBerechnung und anderer zeitkritischer Transportsystemfunktionen sowie deren Umsetzung in ASIC-Entwürfe, *Diplomarbeit,* IND D 1110, November 1991.

[240] R. NITZAN, P. GROSS; The Role of the U.S. GOSIP, *Computer Networks and ISDN Systems,* North-Holland, Vol. 19, pp. 270 - 274, 1990.

[241] E. NORDMARK, D.R. CHERITON; Experiences from VMTP: How to achieve low response time, *in [34],* pp. 43 - 54.

[242] B. NOWICKI; Transport Issues in the Network File System, *Computer Communications Review,* Vol. 19, No. 2, pp. 16 - 20, March 1989.

[243] G.M. PARULKAR, J.S. TURNER; Towards a Framework for High-Speed Communication in a Heterogeneous Networking Environment, *IEEE Network Magazine,* Vol. 4, No. 2, pp. 19 - 27, 1990.

[244] G.M. PARULKAR; The next Generation of Internetworking, *Computer Communication Review,* ACM Press, Vol. 20, No. 1, pp. 18 - 43, January 1990.

[245] A. PATEL, V. RYAN; Introduction to Names, Addresses and Routes in an OSI Environment, *ACM Transactions on Communications,* Vol. 13, No. 1, pp. 27 - 35, January/February 1990.

[246] L.L. PETERSON, N.C. HUTCHINSON, S.W. O'MALLEY, H. RAO; The x-kernel: A Platform for Accessing Internet Resources, *Computer,* Vol. 23, No. 5, pp. 23 - 33, May 1990.

[247] G.S. POO, B.P. CHAI; Modularity versus Efficiency in OSI Systems Implementations, *Proceedings of IEEE INFOCOM'91,* paper 8D.2.1., pp. 950 - 959, Bal Harbour, FL, USA, April 1991.

[248] G.S. POO; Transport and link layer performance comparison of CSMA/CD, token bus and token ring networks, *Computer Communications,* Vol. 14, No. 4, pp. 235 - 243, May 1991.

[249] R. POPESCU-ZELETIN; From Broadband ISDN to Multimedia Computer Networks, *Computer Networks and ISDN Systems,* North-Holland, No. 18, pp. 47 - 54, December 1989 / January 1990.

[250] T.F. LAPORTA, M. SCHWARTZ; Architectures, Features, and Implementation of High-Speed Transport Protocols, *IEEE Network Magazine,* Vol. 5, No. 3, pp. 14 - 22, May 1991.

[251] J.B. POSTEL C.A. SUNSHINE, D. COHEN; The ARPA Internet Protocol, *Computer Networks,* Vol. 5, No. 4, pp. 261 - 271, July 1981.

[252] J.B. POSTEL; Internet Protocol, *Network Information Center, RFC 791,* SRI International, Menlo Park, California, September 1981.

[253] J.B. POSTEL; Transmission Control Protocol, *Network Information Center, RFC 793,* SRI International, Menlo Park, California, September 1981.

[254] R.L. PROBERT, K. SALEH; Synthesis of Communication Protocols: Survey and Assessment, *IEEE Transactions on Computers,* Vol. 40, No. 4, pp. 468 - 475, April 1991.

[255] R. A. QUINNELL; Standardized feature sets add versatility and speed, *EDN,* March 1992.

[256] S.M. RADACK; U.S. Government moves toward implementing OSI standards, *Computer,* Vol. 21, No. 5, pp. 82 - 83, June 1988.

[257] M. RAHNEMA; Frame Relaying and the Fast Packet Switching Concepts and Issues, *IEEE Network Magazine,* Vol. 5, No. 4, pp. 18 - 23, July 1991.

[258] M. REPNOW; Untersuchungen der Vermittlungsschicht in lokalen Netzwerken, *Studienarbeit,* IND S2 1105, August 1991.

[259] C.M. REEVES; Complexity Analysis of Event Set Algorithms, *Computer Journal,* Vol. 27, No. 1, pp. 36 - 47, February 1984.

[260] M.T. ROSE; Transition and Coexistence Strategies for TCP/IP to OSI, *IEEE Journal of Selected Areas in Communications,* Vol. 8 , No. 1, pp. 57 - 66, January 1990.

[261] F.E. ROSS; An Overview of FDDI: the Fiber Distributed Data Interface, *IEEE Journal on Selected Areas in Communications,* Vol. 7, No. 7, pp. 1043 - 1051, September 1989.

[262] F.E. ROSS, R.L. FINK; Overview of FFOL - FDDI Follow-On LAN, *Computer Communications,* Vol. 15, No. 1, pp. 5 - 10, January / February 1992.

[263] I. RUBIN, J.E. BAKER; Media Access Control for High-Speed LAN and MAN Communications Networks, *Proceedings of the IEEE,* Vol. 78, No. 1, pp. 169 - 203, January 1990.

[264] M. RUPPRECHT; Petrinetze als Organisationsprinzip für einen Kommunikations-Controller, *in: P.J. Kühn (Ed.) "Kommunikation in verteilten Systemen"*, Springer Verlag, Februar 1989.

[265] T. RUSSELL; UltraNet - High Performance TCP/IP, *UltraNet TCP Project Summary*, Ultra Network Technologies, San Jose, CA, USA, October 1990.

[266] J. SALTER, P. EVANS; The IEEE 802.6 Standard for MANs, its Scope & Purpose, *Proceedings of 8th European Fibre Optic Communications and Local Area Networks Conference (EFOC/LAN'90)*, pp. 151 - 163, Munich, FRG, June 1990.

[267] J.H. SALTZER, D.P. REED, D.D. CLARK; End-to-End Arguments in System Design, *ACM Transactions on Computer Systems*, Vol. 2, No. 4, pp. 277 - 288, November 1984.

[268] R.M. SANDERS, A.C. WEAVER; The Xpress Transfer Protocol (XTP) - A Tutorial, *ACM Computer Communication Review*, Vol. 20, No. 5, pp. 67 - 88, October 1990.

[269] J. SANDVOSS, B. KÖHLER, H. STÜTTGEN; Ein Anfrage-/Antworttransportprotokoll für verbindungsorientierte Vermittlungsdienste, *Praxis der Kommunikationsverarbeitung und Kommunikation (PIK)*, Jahrgang 14, Nr. 4, S. 202 - 210, Oktober - Dezember 1991.

[270] J. SANDVOSS; Entwurf und Implementierung eines transaktionsorientierten Transportprotokolls, *Diplomarbeit*, IND D 1119, Dezember 1991.

[271] D.V. SARWATE; Computation of Cyclic Redundancy Checks Via Table Look-Up, *Communications of the ACM*, Vol. 31, No. 8, pp. 37 - 43, August 1988.

[272] K. SAUER, A. WEBER; A flexible Customer Premises Network for Introducing ATM, *Proceedings of 14th International Switching Symposium (ISS'92) Yokohama*, Japan, October 1992.

[273] J.M. SCHRÖDER; Verteiltes Monitoring- und Diagnosesystem für Ethernet, *in: G. Hommel (Hrsg.) "Prozeßrechensysteme"*, Springer Verlag, Berlin, BRD, Februar 1991.

[274] W.D. SCHWADERER; XTP in VLSI - Protocol Decomposition for ASIC Implementation, *Proceedings of the 15th IEEE Conference on Local Computer Networks (LCN'90)*, pp. 249 - 252, Minneapolis, MN, USA, October 1990.

[275] Synchrone Digitale Hierarchie (gesamtes Heft), *Philips Innovation*, Technische Mitteilungen Telekommunikation, SDH Special, Februar 1991.

[276] A.U. SHANKAR; A Verified Sliding Window Protocol with Variable Flow Control, *Proceedings of SIGCOMM'86*, ACM, pp. 84 - 91, Stowe, VT, USA, August 1986.

[277] A.U. SHANKAR; Modular Design Principles for Protocols with an Application to the Transport Layer (invited), *Proceedings of the IEEE*, Vol. 79, No. 12, pp. 1687 - 1707, December 1991.

[278] M. SIEGEL, K. SAUER, W. SCHÖDL, M. TANGEMANN; Design of an Enhanced FDDI-II System, *Proceedings of International Zurich Seminar (IZS'90)*, Paper E9, pp. 418 - 432, Zurich, Switzerland, March 1990.

[279] M. SIEGEL, K. SAUER; Efficient Network Management for Highly Distributed Systems, *Proceedings of second International Conference on Local Communication Systems: LAN and PBX*, pp. 421 - 431, Palma de Mallorca, Spain, June 1991.

[280] M. SIEGEL, M. WILLIAMS, G. RÖßLER; Overcoming Bottlenecks in High-Speed Transport Systems, *Proceedings of 16th IEEE Conference on Local omputer Networks (LCN'91)*, pp. 395 - 407, Minneapolis, MN, USA, October 1991.

[281] M. SIEGEL; Die Zukunft der Hochleistungsnetze, *Proceedings of FDDI-Congress '91*, pp. 509 - 551, Düsseldorf, FRG, September 1991.

[282] R. SIMUNCIC, A.C. WEAVER, M.A. COLVIN; Experience with the Xpress Transfer Protocol, *Proceedings of 15th IEEE Conference on Local Computer Networks (LCN'90)*, pp. 123 - 131, Minneapolis, MN, USA, October 1990.

[283] A. DESIMONE; Generating Burstiness in Networks: A Simulation Study of Correlation Effects in Networks of Queues, *Computer Communications Review*, ACM Press, Vol. 21, No. 1, pp. 24 - 31, January 1991.

[284] K. SKLOWER; Improving the Efficiency of the OSI Checksum Calculation, *Computer Communication Review*, ACM Press, Vol. 19, No. 5, pp. 32 - 43, October 1989.

[285] M. SKOV; Implementation of High-Speed Physical and Media Access Protocols, *IEEE Communications Magazine*, Vol. 27, No. 6, pp. 45 - 53, June 1989.

[286] K. SMITH, R. MORELL, P. HOFHINE, A. SORIA, D. BROADHEAD, M. WHITAKER, M. HUTH; A local-area network VLSI chip set, *Proceedings of IEEE Custom Integrated Circuits Conference (CICC'86),* pp. 426 - 430, 1986.

[287] M. STARK, A. KORNHAUSER, D. VAN MIETROP; A high functionality VLSI LAN controller for CSMA/CD, *Proceedings of IEEE Conference on Computers,* pp. 510 - 517, March 1983.

[288] J. STEHR, J. SUPPAN-BOROWKA; Messungen von TCP/IP-Implementierungen, *Datacom,* Nr. 9, S. 166 - 171, September 1987.

[289] R. STEINMETZ, R.G. HERRTWICH; Integrierte verteilte Multimedia-Systeme, *Informatik Spektrum,* Springer-Verlag, Jahrgang 14, Heft 5, S. 249 - 260, Oktober 1991.

[290] J.P.G. STERBENZ, G.M. PARULKAR; AXON: A High Speed Communication Architecture for Distributed Applications, *Proceedings of IEEE INFOCOM'90,* pp. 415 - 425, San Francisco, CA, USA, June 1990.

[291] J.P.G. STERBENZ, G.M. PARULKAR; AXON: Application-Oriented Lightweight Transport Protocol Design, *Proceedings of 10th International Conference on Computer Communications (ICCC'90),* pp. 379 - 387, New Delhi, India, November 1990.

[292] STRAUSS P.; OSI throughput performance: Breakthrough or bottleneck, *Data Communications,* pp. 53 - 56, May 1987.

[293] L. SVOBODOVA; Implementing OSI Systems, *IEEE Journal of Selected Areas in Communications,* Vol. 7, No. 7, pp. 1115 - 1130 September 1989; also published as *Research Report RZ 1715,* IBM Research Division, Rüschlikon, Switzerland, June 1988.

[294] L. SVOBODOVA; Measured Performance of Transport Service in LANs, *Computer Networks and ISDN Systems,* Vol. 18, pp. 31 - 45, December 1989 / January 1990; also published as *Research Report RZ 1799 (#64.663),* IBM Research Division, Rüschlikon, Switzerland, March 1989.

[295] L. SVOBODOVA, P.A. JANSON, E. MUMPRECHT; Heterogeneity and OSI, *IEEE Journal of Selected Areas in Communications,* Vol. 8 , No. 1, pp. 67 - 79, January 1990.

[296] A.S. TANENBAUM VAN, R. RENESSE; Distributed Operating Systems, *ACM Computing Surveys,* Vol. 17, No. 4, pp. 419 - 470, December 1985.

[297] M. TANGEMANN; Timer Threshold Dimensioning and Overload Control in FDDI Networks, *Proceedings of IEEE INFOCOM'92,* pp. 363 - 371, Firenze, Italy, May 1992.

[298] A. TANTAWY, T. SCHÜTT, H. MELEIS, R. LAMAIRE, R. AUERBACH; High Performance Protocol Implementations: LLC Case Study, *in [20],* pp. 123 - 140, also published as *Research Report RC 15.850,* IBM Research Division, Yorktown Heights, NY, USA, June 1990,.

[299] D.L. TENNENHOUSE; Layered Multiplexing Considered Harmful, *in [34],* pp. 143 - 148.

[300] M. TERADA, T. YOKOYAMA, T. HIRATA, S. MATSUI; A High Speed Protocol Processor to execute OSI, *Proceedings of IEEE INFOCOM'91,* pp. 0944 - 0949, Bal Harbour, Fl, USA, April 1991.

[301] A. TEVANIAN, R.F. RASHID, D.B. GOLUB, D.L. BLACK, E. COOPER, M.W. YOUNG; Mach Threads and the UNIX Kernel: The Battle for Control, *Proceedings of the USENIX Summer Conference,* pp. 185 - 198, Phoenix, AZ, USA, June 1987.

[302] U.K. Government OSI Profile, Version 3.0, *Central Computer and Telecommunications Agency (CCTA),* March 1990.

[303] R. ULRICH, H. DIETSCH; A Transputer-Based Communication Controller for FDDI Stations, *Proceedings of 8th European Fibre Optic Communications and Local Area Networks Conference (EFOC/LAN'90),* paper 5.5.3, pp. 287 - 292, Munich, FRG, June 1990.

[304] U.S. Government OSI Profile, Version 2 (Draft), *U.S. Department of Commerce,* NIST, Gaithersburg, MD, USA, October 1989.

[305] G. VARGHESE, T. LAUCK; Hashed and Hierarchical Timing Wheels: Data Structures for the Efficient Implementation of a Timer Facility, *Proceedings of 11th Symposium on Operating Systems Principles (SIGOPS'87),* ACM, pp. 25 - 38, Austin, TX, USA, November 1987.

[306] I. VERBAUWHEDE, F. HOORNAERT, J. VANDERWALLE, H. DEMAN; VLSI Design Methods for an ASIC Cryptographic Processor based on DES, *in: F. Pichler (Ed.) "Secure Designs and Test of Cryptochips",* Elsevier Science Publishers B.V. (North-Holland), pp. 31 - 33, October 1991.

[307] C.A. VISSERS, L. LOGRIPPO; The Importance of the Service Concept in the Design of Data Communication Protocols, *Proceedings of SIGCOMM'85,* ACM, USA, September 1985.

[308] G. VOTSIS; Ethernet VLSI Controllers Comparison, *Proceedings of International Symposium on Local Communication Systems: LAN and PBX,* pp. 73 - 82, Toulouse, France, November 1986.

[309] TH. WAHL; Leistungsoptimierung in Hochgeschwindigkeitsnetzen: Transportschicht, *Diplomarbeit,* Institut für Betriebs- und Dialogsysteme, Universität Erlangen, April 1991.

[310] B. WALKE; Digitale Nebenstellenanlagen und LANs, *Informatik Spektrum,* H. 11, S. 9 - 28, 1988.

[311] R.W. WATSON; Timer-Based Mechanisms in Reliable Transport Protocol Connection Management, *Computer Networks,* North-Holland, Vol. 5, pp. 47 - 56, 1981.

[312] R.W. WATSON, S.A. MAMRAK; IPC interface and end-to-end (transport) protocol design issues, *in [25],* pp. 140 - 174 (chapter 7).

[313] R.W. WATSON, S.A. MAMRAK; Gaining Efficiency in Transport Services by Appropriate Design and Implementation Choices, *ACM Transactions on Computer Systems,* Vol. 5, No. 2, pp. 97 - 120, May 1987.

[314] R.W. WATSON; The Delta-t Transport Protocol: Features and Experience, *in [34],* pp. 3 - 18, also published in: *Proceedings of 14th IEEE Conference on Local Computer Networks (LCN'89),* pp. 399 - 407, Minneapolis, MN, USA, October 1989.

[315] A. WEBER; Entwurf und Implementierung eines generischen Prozeßkoppelbausteins, *Diplomarbeit,* IND D 1109, Oktober 1991.

[316] M. WEISER, A. DEMERS, C. HAUSER; The Portable Common Runtime approach to interoperability, *Operating Systems Review (special issue),* Vol. 23, No. 5, pp. 114 - 122, December 1989.

[317] M. WEIXLER; Distributed Measurement System for Protocols and Applications in ISO 8802/3 LANs, *Proceedings of 4th International Conference on Data Communications Systems and their Performance,* pp. 448 - 455, Barcelona, Spain, June 1990.

[318] M.J. WEIXLER; Ein Konzept zur Leistungsmessung in verteilten Rechenanlagen mit dezentralen Zeitgebern, *Dissertationsschrift,* IND, Universität Stuttgart, Juni 1992.

[319] R.K. WENDT; Echtzeitsimulationen, *Design & Elektronik,* Heft 8, S. 65 - 66, April 1991.

[320] A.D. WHALEY; The Xpress Transfer Protocol, *Proceedings of the 14th IEEE Conference on Local Computer Networks,* pp. 408 - 414, Minneapolis, MN, USA, October 1989.

[321] C.L. WILLIAMSON, D.R. CHERITON; An Overview of the VMTP Transport Protocol, *Proceedings of 14th IEEE Conference on Local Computer Networks (LCN'89),* pp. 415 - 420, Minneapolis, MN, USA, October 1989.

[322] L.C. WOLF; Entwurf und Implementierung einer Laufzeitumgebung für Hochgeschwindigkeitsprotokolle, *Diplomarbeit,* Institut für Mathematische Maschinen und Datenverarbeitung, Universität Erlangen, Juli 1991.

[323] C.M. WOODSIDE, J.R. MONTEALEGRE; The Effect of Buffering Strategies on Protocol Execution Performance, *IEEE Transactions on Communications,* Vol. 37, No. 6, pp. 545 - 554, June 1989.

[324] W.W. WU, A. LIVNE, J. GRIFFITHS (Eds.), Integrated Services Digital Networks, *Proceedings of the IEEE (special issue),* Vol. 79, No. 2, February 1991.

[325] P. ZAFIROPULO; On LANs and MANs: An Evolution from Mbit/s to Gbit/s, *Proceedings of 8th European Fibre Optic Communications and Local Area Networks Conference (EFOC/LAN'90),* pp. 15 - 22, Munich, FRG, June 1990.

[326] L. ZHANG; Why TCP Timers Don't Work Well, *Proceedings of SIGCOMM'86,* ACM, pp. 397 - 405, Stowe, VT, USA, August 1986.

[327] X. ZHANG, A.P. SENEVIRATNE; An efficient Implementation of a High-Speed Protocol without Data Copying, *Proceedings of 15th IEEE Conference on Local Computer Networks (LCN'90),* pp. 443 - 449,, Minneapolis, MN, USA, October 1990.

[328] H. ZIMMERMANN; The ISO Model of Architecture for Open Systems Interconnection, *IEEE Transactions on Communication,* Vol. COM-28, No. 4, pp. 212 - 219, April 1980.

[329] M. ZITTERBART, Funktionsbezogene Parallelität in transportorientierten Kommunikationsprotokollen, *Dissertationsschrift,* Universität Karlsruhe, Oktober 1990; erschienen im VDI-Verlag als Fortschrittsbericht der Reihe 10: Informatik/Komm.-technik, Nr. 183, Düsseldorf, BRD, 1991.

[330] M. ZITTERBART; A Multiprocessor Architecture for High Speed Network Interconnections, *Proceedings of IEEE INFOCOM'89,* pp. 212 - 218, Ottawa, Canada, April 1989.

[331] M. ZITTERBART; High Speed Transport Components, *IEEE Network Magazine,* Vol. 5, No. 1, pp. 54 - 63, January 1991.

[332] E.A. ZURFLUH, P. DILL, R. HELLER, W.W. LEMPPENAU, P. MÜLLER, H.R. SCHINDLER, P. ZAFIROPULO; The IBM Zurich Research Laboratory's 1.13 Gb/s LAN Prototype, *Research Report RZ 2278 (#76808)*, IBM Research Division, Rüschlikon, Switzerland, February 1992.

L.3 Standards (thematisch sortiert)

[333] ISO CD 7498-1; *Information processing systems - Open Systems Interconnection - Basic Reference Model* (⇒CCITT X.200), March 1992 (see also *Computer Networks*, Vol. 5, pp. 81 - 118, 1981).

[334] ISO 7498-1 / Add. 1; *Connectionless Data Transmission*, July 1987.

[335] ISO 7498-3; *Information processing systems - Open Systems Interconnection - Basic Reference Model - Part 3: Naming and Addressing*, March 1989.

[336] ISO TR 8509; *OSI Service Conventions* (⇒CCITT X.210), September 1987.

[337] ISO 8649; *Service Definition for the Association Control Service Element - ACSE* (CCITT X.217), December 1988.

[338] ISO 8650; *Protocol Specification for the Association Control Service Element - ACSE* (CCITT X.227), December 1988.

[339] ISO 9072-1; *Remote Operations P1: Model, Notation, and Service Definition*, November 1989.

[340] ISO 9072-2; *Remote Operations, P2: Protocol Specification*, November 1989.

[341] ISO 8571-1; *File Transfer, Access, and Management (FTAM), General Introduction*, Oct. 1988.

[342] ISO 9040; *Virtual Terminal Service - Basic Class*, March 1990.

[343] ISO 8831; *Job Transfer and Manipulation (JTM) Concepts and Services*, July 1989.

[344] ISO DIS 9594-1; *The Directory, Part 1: Overview of Concepts, Models, and Services* (⇒CCITT X.500), April 1990.

[345] ISO DIS 10021-1; *Message Oriented Text Interchange System (MOTIS), Part 1: System and Service Overview* (⇒CCITT X.400), May 1990.

[346] ISO DIS 10021-2; *Message Oriented Text Interchange System (MOTIS), Part 2: Overall Architecture* (⇒CCITT X.402), May 1990.

[347] ISO DIS 9506-1, *Industrial Automation Systems - Systems Integration and Communications - Manufacturing Message Specification (MMS), Part 1: Service Definition*, August 1988.

[348] ISO DIS 9506-1; *Industrial Automation Systems - Systems Integration and Communications - Manufacturing Message Specification (MMS), Part 2: Protocol Specification*, August 1988.

[349] ISO 8822; *Connection Oriented Presentation Service Definition* (⇒CCITT X.216), August 1988.

[350] ISO 8823; *Connection Oriented Presentation Protocol Spec.* (⇒CCITT X.226), August 1988.

[351] ISO 8824; *Spec. of Abstract Syntax Notation 1 (ASN.1)* (⇒CCITT X.208), December 1990.

[352] ISO 8825; *Specification of Basic Encoding Rules for Abstract Syntax Notation 1 - ASN.1* (⇒CCITT X.209), December 1990.

[353] ISO 8326; *Basic Connection Oriented Session Service Def.* (⇒CCITT X.215), August 1987.

[354] ISO 8327; *Basic Connection Oriented Session Protocol Spec.* (⇒CCITT X.225), August 1987.

[355] ISO 8072; *Transport Service Definition* (⇒CCITT X.214), June 1986.

[356] ISO 8073; *Connection Oriented Transport Protocol Spec.* (⇒CCITT X.224), December 1988.

[357] ISO 8073-Add. 2; *Connection Oriented Transport Protocol Specification, Addendum 2: Class 4 Operation over Connectionless Network Service*, September 1989.

[358] ISO 8073-DAM4; *Connection Oriented Transport Protocol Specification, Addendum 4: Transport Protocol Enhancements*, July 1990.

[359] ISO 8348; *Network Service Definition* (⇒CCITT X.213), April 1987.

[360] ISO 8348/Add. 1; *Connectionless-mode Transmission*, April 1987.

[361] ISO 8348/Add. 2; *Network Layer Addressing*, June 1988.

[362] ISO TR 10172; *Network / Transport Protocol Interworking Specification*, September 1991.

[363] ISO 8473; *Protocol for Providing the Connectionless-mode Network Service*, December 1988.

[364] ISO 8473/Add. 3; *Provision of the Underlying Service Assumed by ISO 8473 over Subnetworks which Provide the OSI Data Link Service,* September 1989.

[365] ISO 8208; *X.25 Packet Level Protocol for Data Terminal Equipment,* March 1990.

[366] ISO 8878; *Use of X.25 to provide the CONS* (⇒CCITT X.223), September 1987.

[367] ISO DIS 8886; *Data Link Service Definition* (⇒CCITT X.212), September 1988.

[368] ISO 3309; *High-level Data Link Control (HDLC) - Frame Structure,* June 1991.

[369] ISO 8802-1; *Local Area Networks Part1: Introduction, Overview and Architecture* (⇒ANSI/IEEE 802.1), September 1990.

[370] ISO 8802-2; *LAN Part 2: Logical Link Control* (⇒ANSI/IEEE 802.2), July 1990.

[371] ISO 8802-3; *LAN Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD), Access Method and Physical Layer Specifications* (⇒ANSI/IEEE 802.3), September 1990.

[372] ISO 8802-4; *LAN Part 4: Token-Passing Bus Access Method and Physical Layer Specifications* (⇒ANSI/IEEE 802.4), August 1990.

[373] ISO 8802-5; *LAN Part 5: Token Ring Access Method and Physical Layer Specifications* (⇒ANSI/IEEE 802.5), September 1990.

[374] ISO CD 8802-6; *LAN Part 6: Distributed Queue Dual Bus (DQDB) Access Method and Physical Layer Specifications* (⇒ANSI/IEEE 802.6 12/90), December 1991.

[375] ISO 9314-1; *FDDI, Part 1: Physical Layer Protocol - PHY* (⇒ANSI X3.148), April 1989

[376] ISO 9314-2; *FDDI, Part 2: Medium Access Control - MAC* (⇒ANSI X3.139), May 1989.

[377] ISO DIS 9314-3; *FDDI, Part 3: Physical Layer Medium Dependent - PMD-* (⇒ANSI X3.166), August 1990.

[378] ISO DP 9314-4; *FDDI, Part 4: Single-Mode Fiber / Physical Layer Medium Dependent - PMD* (ANSI X3.184), 1991.

[379] ISO 9314-5; *FDDI, Part 5: Hybrid Ring Control - FDDI-II* (⇒ANSI X3.186), May 1991.

[380] ISO DP 9314-6; *FDDI, Part 6: Station Management - SMT,* 1992.

[381] ISO DIS 7498-4; *Information processing systems - Open Systems Interconnection - Basic Reference Model - Part 4: Management Framework,* November 1989.

[382] ISO 9595; *Common Management Information Service (CMIS) Def.* (⇒CCITT X.710), July 1990.

[383] ISO 9596; *Common Management. Information Prot. (CMIP) Def.* (⇒CCITT X.711), June 1991

[384] ISO DIS 10040; *Systems Management Overview* (⇒CCITT X.701), April 1990.

[385] ISO DIS 10165 *Structure of Management Information,* March 1990.

[386] ISO DIS 10165-1; *Management Information Model* (⇒CCITT X.720), March 1991.

[387] ISO DIS 10165-2; *Definition of Management Information* (⇒CITT X.721), March 1991.

[388] ISO DIS 10165-4; *Guidelines for the Definition of MOs* (⇒CCITT X.722), March 1991.

[389] ISO/IEC JTC1/SC21 N 2686; *Working Draft of the OSI Management Specification, Part 6: Configuration Management,* Project JTC1.21.28, May 1988.

[390] ISO/IEC JTC1/SC21 N 3312; *Systems Management - Fault Management,* Working Document Project JTC1.21.28, January 1989.

[391] ISO/IEC JTC1/SC21/N 3313; *Performance Management,* Working Document, Project JTC1.21.28, January 1989.

[392] CCITT Recommandation I.121; *Broadband Aspects of ISDN,* Blue Book, Vol. III - Fascicle III.7, section 2, pp. 34 - 61, Geneva 1989.

[393] IEEE P996; *Personal Computer Bus Standard,* Draft D2.00, January 1990.

[394] A Transport Protocol for Highspeed Networking, *ANSI X3S3.3/91-265,* working group protocols, November 1991.

[395] VMTP: Versatile Message Transaction Protocol, *Network Information Center, RFC 1045,* Protocol Specification, Preliminary Version 0.7; Stanford University, February 1988.

[396] XTP Protocol Definition (Revision 3.6), *Technical Report PEI 92-10,* Protocol Engines Incorporated, Mountain View, CA 94043, USA, January 1992.

L.4 Handbücher

[397] Intel i386 Micoprocessor, Programmer's Reference Manual, Intel Corporation, Santa Clara, CA, USA, 1988.

[398] Intel i486 Micoprocessor, Programmer's Reference Manual, Intel Corporation, Santa Clara, CA, USA, 1990.

[399] Intel i960 Micoprocessor, Programmer's Reference Manual, Intel Corporation, Santa Clara, CA, USA, 1988.

[400] ASM-960 Operating Instructions for DOS Systems, Intel Corporation, Santa Clara, CA, USA, 1988.

[401] The Transputer Databook, INMOS Ltd., Lewins Mead Bristol, UK, 1989.

[402] Enhanced 32-Bit Microprocessor User's Manual (3rd Edition), *Prentice Hall,* Englewood Cliffs, NJ, USA, 1990, 1991.

[403] OS/2 Programmer's Reference, *Microsoft Press,* Redmond, WA, USA, September 1989.

[404] iRMX II.3 Operating System Documentation, Intel Corporation, Santa Clara, CA, USA, 1988.

[405] Networking Programming, Revision A, SUN Microsystems, Mountain View, CA, USA, 1988.

[406] VMS General User's Manual, Digital Equipment Corporation, Maynard, MA, USA, April 1988.

[407] Cadstar Reference Manual, Release 6, Racal Design Software Series, Tewksbury, UK, 1991.

[408] Systems HILO, System Reference Manual, GenRad Ltd., Fareham, UK, 1998.

[409] J. ZEPF, G. WILLMANN; Strukturierte Simulationstechnik mit der IND-Modulbibliothek unter VAX-Pascal, *Unterlagen zum Simulationskurs*, IND, Oktober 1991.

[410] Das Programmsystem LOG/iC, Syntax Handbuch, Release 3, ISDATA, Karlsruhe, BRD, 1989.

[411] The SUPERNET Family, Databook, Advanced Micro Devices, Sunnyvale, CA, USA, 1989.

[412] FDDI-Controller mit 29000-RISC-CPU, *Design & Elektronik,* Heft 8, S. 104 - 106, April 1991.

[413] Motorola mit FDDI, *Markt & Technik,* Heft 22, S. 3, Mai 1991.

[414] DECnet Digital Network Architecture (Phase IV) - General Description, *DEC Publication,* No. AA-149A-TC, Digital Equipment Corporation, Bedford, MA, USA, 1982.

[415] TURBOchannel Developper's Kit (Version 2), Digital Equipment Corporation, Maynard, MA, USA, September 1990.

[416] IBM Personal System/2, Model 80, *Technical Reference*, International Bussiness Machines Corporation, Armonk, NY, USA, April 1987.

[417] Am95C85 (CADM) Content Adressable Data Manager, Technical Manual, Advanced Micro Devices, Sunnyvale, CA, USA, 1986.

[418] SuperCrypt, *Preliminary Data Sheet Version 1.01,* Computer Electronic Infosys, 99C003, Bodenheim, FRG, October 1991.