



# GMD Research Series

GMD –  
Forschungszentrum  
Informationstechnik  
GmbH

Ludger Fiege

## Dynamisch planbare Kommunikation in verteilten Echtzeitsystemen

© GMD 1999

GMD – Forschungszentrum Informationstechnik GmbH  
Schloß Birlinghoven  
D-53754 Sankt Augustin, Germany  
Telefon +49 -2241 -14 -0  
Telefax +49 -2241 -14 -2618  
<http://www.gmd.de>

In der Reihe GMD Research Series werden Forschungs- und  
Ergebnisse aus der GMD zum wissenschaftlichen, nicht-  
kommerziellen Gebrauch veröffentlicht. Jegliche Inhaltsänderung des  
Dokuments sowie die entgeltliche Weitergabe sind verboten.

The purpose of the GMD Research Series is the dissemination  
of research work for scientific non-commercial use.  
The commercial distribution of this document is prohibited,  
as is any modification of its content.

**Die vorliegende Veröffentlichung entstand im/  
The present publication was prepared within:**  
Institut für Autonome intelligente Systeme (AiS)  
Institute for Autonomous intelligent Systems  
<http://ais.gmd.de/>

**Anschrift des Verfassers/Address of the author:**

Ludger Fiege  
Linzer Straße 40b  
D-53489 Sinzig  
E-mail: [Ludger.Fiege@gmx.de](mailto:Ludger.Fiege@gmx.de)

**Die Deutsche Bibliothek - CIP-Einheitsaufnahme:**

**Fiege, Ludger:**

Dynamisch planbare Kommunikation in verteilten Echtzeitsystemen /  
Ludger Fiege. GMD – Forschungszentrum Informationstechnik GmbH. -  
Sankt Augustin : GMD – Forschungszentrum Informationstechnik, 1999  
(GMD Research Series ; 1999, No. 2)  
Zugl.: Bonn, Univ., Diplomarbeit, 1998  
ISBN 3-88457-351-9

ISSN 1435-2699

ISBN 3-88457-351-9

## Zusammenfassung

Die vorliegende Arbeit entwirft ein Kommunikationssystem für verteilte Echtzeitsysteme, das die für die vorhersagbare Kommunikation notwendigen Ressourcen zur Laufzeit reserviert und verschiedene Netzprotokolle gleichzeitig unterstützt. Es kann in verteilten Echtzeitsystemen eingesetzt werden, in denen der Ressourcenbedarf variabel und nicht vor der Laufzeit bestimmbar ist. Verschiedene Netzprotokolle können gemeinsam geplant und konfliktfrei nebeneinander genutzt werden. Ungenutzte Ressourcenbelegungen kann man damit deutlich verringern, weil der in den Teilanwendungen geforderte Quality-of-Service mit Hilfe geeigneter Protokolle erbracht werden kann. Der Entwurf konzentriert sich auf eingebettete Systeme wie sie zum Beispiel in der Robotik eingesetzt werden und er eignet sich auch für den Einsatz unter harten Echtzeitbedingungen. Die Implementierung zeigt die Praxistauglichkeit und die Vorteile der Kombination verschiedener Zugriffsprotokolle anhand einer verteilten Robotersteuerung auf.

Es wird ein zweistufiges Verfahren entwickelt, das Netzzugriffe und CPU-Anforderungen für die Kommunikationsverbindungen iterativ plant. Die Netzzugriffe plant im ersten Ansatz ein zentraler Rechner, um die Beeinflussung der anderen Knoten möglichst gering zu halten. Auf der Basis des weitverbreiteten Multiple-Access-Busses und eines modifizierten TDMA-Protokolls ermöglicht das Kommunikationssystem die Planung und den konfliktfreien Einsatz verschiedener Protokolle innerhalb verschiedener Zeitintervalle. In den kommunizierenden Knoten wird die für die Protokollverarbeitung notwendige CPU-Leistung jeweils lokal eingeplant.

**Schlagwörter:** Echtzeitsysteme, Kommunikation, Netzwerke, Dynamische Planung

## Abstract

In this thesis, a communication subsystem for distributed real-time systems is presented that schedules the necessary resources online and that integrates different network protocols at the same time. It is applicable in dynamic environments where the resource demand is variable and cannot be specified offline. The system enables the collision free use of different protocols which are jointly managed and scheduled in one infrastructure. Wasted resource reservations are notably reduced by concurrently using protocols in (sub-)applications which matches the demanded quality-of-service best. The design concentrates on embedded systems like robot control and is in particular suitable for hard real-time systems. The implementation of the concepts shows the practical deployment in a distributed robotic control environment and it emphasises the possibilities of combining different network protocols.

Network and CPU resources are scheduled in two stages. In a first approach a central unit schedules the network access, minimising the load of non-involved nodes. On the basis of a multiple access bus and a modified TDMA protocol the communication systems schedules and uses different protocols in different time intervals (conflict avoidance) and provides the appropriate network service transparently for the application. In the involved nodes locally schedule the CPU usage of the protocol stack.

**Keywords:** Real-Time Systems, Communication, Networks, Dynamic Scheduling



# Vorwort

Diese Diplomarbeit ist im Forschungsbereich Responsive Systeme des Instituts für Systementwurfstechnik der GMD–Forschungszentrum für Informationstechnik, St. Augustin, entstanden. Die Arbeit fügt sich in den Kontext des Projekts DIRECT ein, das für den Entwurf zukünftiger verteilter Echtzeitapplikationen Methoden und Werkzeuge entwickelt.

Ich möchte mich bei all denjenigen bedanken, die mich bei meinem Studium und vor allem bei dieser abschließende Arbeit in der GMD unterstützt haben.

Bonn, im Februar 1998

Ludger Fiege



# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>15</b>
1.1	Echtzeitsysteme . . . . .	15
1.1.1	Vorhersagbarkeit und Planung . . . . .	17
1.1.2	Klassifizierung . . . . .	18
1.1.2.1	Statische Planung . . . . .	18
1.1.2.2	Dynamische Planung . . . . .	19
1.1.3	Echtzeitbedingungen . . . . .	20
1.1.4	Fehlertoleranz . . . . .	22
1.1.5	Lokale Echtzeitsysteme . . . . .	23
1.1.6	Verteilte Echtzeitsysteme . . . . .	24
1.2	Problematik . . . . .	25
1.3	Ziel der Diplomarbeit . . . . .	29
1.3.1	Kommunikationsmodell . . . . .	29
1.3.2	Kommunikationsprotokoll . . . . .	30
1.3.3	Voraussetzungen . . . . .	32
1.3.4	Realisierung . . . . .	32
1.3.4.1	Kommunikationssystem . . . . .	32
1.3.4.2	Zugriffsprotokoll . . . . .	34
1.3.4.3	Planung . . . . .	34
1.3.4.4	Implementierung . . . . .	35
1.4	Aufbau der Arbeit . . . . .	36
<b>2</b>	<b>Verwandte Arbeiten</b>	<b>37</b>
2.1	Datenübertragung . . . . .	37
2.1.1	Multiple-Access-Busse . . . . .	37
2.1.1.1	Absprechende Zugriffsverfahren . . . . .	39
2.1.1.2	Kontrollierende Zugriffsverfahren . . . . .	41
2.1.1.3	Bewertung . . . . .	44
2.1.2	Token Ring . . . . .	46
2.1.3	ATM . . . . .	47
2.2	Netzwerkplanung . . . . .	48
2.3	Integration . . . . .	49

<b>3</b>	<b>Zielsystem</b>	<b>53</b>
3.1	Hardware-Architektur . . . . .	53
3.2	Software-Architektur . . . . .	54
3.2.1	Dynamische Planung . . . . .	54
3.2.2	Zeitsteuerung . . . . .	55
3.2.3	Laufzeitmodell . . . . .	56
3.3	Entwurfskriterien . . . . .	56
3.4	Kommunikation . . . . .	58
3.4.1	OSI-Referenzmodell . . . . .	59
3.4.2	Einbettung ins OSI-Modell . . . . .	61
<b>4</b>	<b>Kommunikationssystem</b>	<b>63</b>
4.1	Garantievertrag . . . . .	63
4.2	Struktur . . . . .	64
4.2.1	Globale und lokale Aspekte . . . . .	64
4.2.2	Kommunikationsmodell . . . . .	65
4.2.2.1	Kommunikationskanäle . . . . .	65
4.2.2.2	Aufgaben der Kanalverwaltung . . . . .	66
4.2.2.3	Datenübertragung in den Kanälen . . . . .	67
4.2.2.4	Kommunikationskanäle in der Applikation . . . . .	69
4.3	Globales Netzmanagement . . . . .	70
4.3.1	Verteilte Datenhaltung . . . . .	70
4.3.2	Verteilte dynamische Planung . . . . .	70
4.3.2.1	Beteiligte Komponenten . . . . .	71
4.3.2.2	Zentrale Reservierung . . . . .	71
4.3.2.3	Dezentrale Reservierung . . . . .	72
4.3.2.4	Folgerung . . . . .	73
4.3.3	Kommunikationsserver . . . . .	75
4.3.3.1	Kommunikationsverbindungen . . . . .	75
4.3.3.2	Aufbau des Servers . . . . .	75
4.3.4	Verbindungsaufbau . . . . .	77
4.3.4.1	Protokoll . . . . .	77
4.3.4.2	Ergänzungen . . . . .	80
4.3.4.3	Bewertung . . . . .	81
4.3.5	Kanalabbau . . . . .	82
4.4	Lokales Netzmanagement . . . . .	82
4.4.1	Aufgaben . . . . .	82
4.4.1.1	Senden . . . . .	83
4.4.1.2	Empfangen . . . . .	84
4.4.1.3	Dynamische Planung . . . . .	85
4.4.2	Aufbau des lokalen Kommunikationssystems . . . . .	85
4.4.3	Kommunikationsdaemon . . . . .	86
4.4.3.1	Planung . . . . .	86

4.4.3.2	NRT-Kommunikation . . . . .	88
4.4.4	Kommunikationsthread . . . . .	89
4.4.5	Kanalaufbau . . . . .	91
4.4.5.1	Interne Anfrage . . . . .	92
4.4.5.2	Externe Anfrage . . . . .	94
4.4.6	Kanalnutzung – das Kanalobjekt . . . . .	95
4.4.7	QoS-Umsetzung . . . . .	96
4.5	Gruppenkommunikation . . . . .	97
4.5.1	Broadcastkanal . . . . .	98
4.5.2	Aktivität . . . . .	98
4.6	Weitere Zugriffsprotokolle . . . . .	99
4.6.1	Kanalüberlagerung . . . . .	99
4.6.2	SRT-Kommunikation . . . . .	100
4.7	Fehlertoleranz . . . . .	100
4.7.1	Übertragungsfehler . . . . .	101
4.7.2	Zeitfehler . . . . .	101
4.7.2.1	Unterbrechung der Sendetask . . . . .	101
4.7.2.2	TaskPair im Kommunikationsthread . . . . .	102
4.7.3	Knotenausfall . . . . .	102
4.8	Verbindungslose Kommunikation . . . . .	103
4.9	Erweiterungsmöglichkeiten . . . . .	104
4.9.1	Planung komplexer Netze . . . . .	104
4.9.1.1	Verteilung der Reservierung . . . . .	104
4.9.1.2	Weitere Topologien . . . . .	105
4.9.2	Neue Dienste . . . . .	105
<b>5</b>	<b>Planung</b> . . . . .	<b>107</b>
5.1	Garantie . . . . .	107
5.2	Aktivitäten . . . . .	108
5.3	Betriebsmittel . . . . .	108
5.3.1	Vielfachheit . . . . .	108
5.3.2	Teilbarkeit . . . . .	108
5.3.3	Zugriff . . . . .	109
5.4	Zeitliche Spezifikation . . . . .	110
5.5	Netzleistungsspezifikation . . . . .	112
5.5.1	QoS-Parameter . . . . .	112
5.5.2	Harte Echtzeitkommunikation . . . . .	117
5.5.3	Andere QoS-Spezifikationen . . . . .	118
5.5.3.1	ATM . . . . .	118
5.5.3.2	Tenet . . . . .	119
5.5.3.3	RFC 1363 ‘Flow Spec’ . . . . .	120
5.5.4	Bewertung . . . . .	120
5.5.5	Unterstützung der QoS-Parameter . . . . .	121

5.6	Echtzeitplanung . . . . .	123
5.6.1	Planungsproblem . . . . .	123
5.6.2	Abhängigkeiten . . . . .	125
5.6.3	Dynamische Planung . . . . .	125
5.6.4	Betriebsmittelvergabe . . . . .	126
5.6.4.1	Zeitgesteuerte Vergabe . . . . .	126
5.6.4.2	Prioritätengesteuerte Vergabe . . . . .	126
5.7	Generisches Planungsmodell . . . . .	127
5.7.1	Problematik der TDMA-Planung . . . . .	127
5.7.2	Konzept . . . . .	129
5.7.2.1	Verwaltung der Zeitskala . . . . .	130
5.7.2.2	Reservierungseinheiten – Konflikttest . . . . .	133
5.7.2.3	Planungsalgorithmen . . . . .	134
5.7.3	Aufbau der Reservierung . . . . .	135
5.7.4	Reservierung . . . . .	136
5.7.4.1	Navigation . . . . .	137
5.7.4.2	Interna . . . . .	138
5.7.5	Slotverwaltung . . . . .	138
5.7.6	Reservierungseinheit . . . . .	138
5.7.7	Aktivitätsbeschreibung . . . . .	140
5.7.8	Reservierungsauftrag . . . . .	141
5.7.9	Zusammenfassung . . . . .	142
5.8	Echtzeitklassen . . . . .	142
<b>6</b>	<b>Implementierung</b>	<b>145</b>
6.1	Aufbau des Zielsystems . . . . .	145
6.1.1	Hardware . . . . .	145
6.1.2	Software . . . . .	146
6.2	Kommunikationssystem . . . . .	147
6.2.1	Datenübertragung . . . . .	147
6.2.2	Kanal . . . . .	148
6.2.3	Lokales Management . . . . .	148
6.2.4	Globales Management . . . . .	149
6.3	Planung . . . . .	150
6.4	Beispielapplikation – Robotersteuerung . . . . .	155
6.4.1	Schlangenroboter . . . . .	155
6.4.2	Steuerung . . . . .	156
<b>7</b>	<b>Zusammenfassung</b>	<b>161</b>
7.1	Was leistet das Konzept? . . . . .	161
7.1.1	Kanäle . . . . .	161
7.1.2	Globales Management . . . . .	162
7.1.3	Lokales Management . . . . .	162

<i>INHALTSVERZEICHNIS</i>	11
7.1.4 Datenübertragung . . . . .	162
7.1.5 Planung . . . . .	163
7.2 Was leistet die Implementierung? . . . . .	164
7.3 Ausblick . . . . .	164
<b>A Iteratoren</b>	<b>167</b>
A.1 Datenzugriff . . . . .	167
A.2 Navigation . . . . .	167
<b>Literaturverzeichnis</b>	<b>171</b>
<b>Index</b>	<b>179</b>



# Abbildungsverzeichnis

1.1	Aufbau eines lokalen Echtzeitsystems . . . . .	23
1.2	Unidirektionale Kommunikation durch einen Kanal, ohne Betrachtung der CPU-Aktivität des Kanals . . . . .	29
2.1	Charakterisierung von Multiple-Access-Protokollen . . . . .	38
3.1	Aufbau des Zielsystems . . . . .	54
3.2	Dienstmodell . . . . .	59
3.3	OSI-Schichtenmodell . . . . .	59
4.1	Aufgabenverteilung des Kommunikationssystems . . . . .	64
4.2	Petri-Netz des Verbindungsaufbaus . . . . .	78
4.3	Erweiterung zum vorzeitigen Abbruch des Protokolls . . . . .	81
4.4	Aufgabenverteilung des lokalen Kommunikationssystems . . . . .	86
4.5	Petri-Netz des Kanalaufbaus für Client $x$ . . . . .	93
4.6	Die Kanalnutzung über das Kanalobjekt. . . . .	96
4.7	Kanalüberlagerung und alternative Zugriffsverfahren . . . . .	100
5.1	Zeitmultiplexing als Realisierung einer anteiligen Nutzung . . . . .	110
5.2	Zeitliche Parameter einer Aktivität . . . . .	111
5.3	Kommunikationsformen . . . . .	113
5.4	Jitter einer periodischen Aktivität . . . . .	116
5.5	ATM-Dienstklassen . . . . .	119
5.6	WCET-Reduzierung durch Nebenläufigkeit . . . . .	130
5.7	Ring als Zeitskala . . . . .	133
5.8	Reservierungseinheiten beschreiben geplante Aktivitäten . . . . .	133
5.9	Verschiebung und Aufteilung als Konfliktlösung . . . . .	140
6.1	Aufbau der Testumgebung . . . . .	146
6.2	Aufbau von <code>CServer</code> . . . . .	150
6.3	Positionierung auf der Zeitachse . . . . .	151
6.4	Aufgabenverteilung zwischen den Steuerrechnern . . . . .	157
6.5	Zugriffsmuster der Sektionssteuerung . . . . .	158
6.6	Überlagerte Zugriffszeiten von Uhrenprotokoll/Calibrator . . . . .	159
6.7	Geplante Auslastung des Busses . . . . .	159

6.8	Umfang der Implementierung . . . . .	160
-----	--------------------------------------	-----

# Kapitel 1

## Einleitung

Die Kommunikation in verteilten, dynamischen Echtzeitsystemen ist Gegenstand dieser Diplomarbeit. Echtzeitsysteme haben im allgemeinen Kontakt zur realen Umwelt und müssen in ihrer Datenverarbeitung die Korrektheit eines Ergebnisses auch an zeitliche Bedingungen für den Ablauf der Berechnung knüpfen. Um Voraussagen über den ordnungsgemäßen zeitlichen Ablauf einer Applikation machen zu können, ist die Planung der Aktivitäten notwendig. In dynamischen Systemen erfolgt diese zur Laufzeit, um u.a. flexibel auf Veränderungen der Anwendungen reagieren zu können. Kooperierende Komponenten in verteilten Echtzeitsystemen müssen kommunizieren und diese *Echtzeitkommunikation* ist ebenso wie die Verarbeitung der Programme zeitlichen Beschränkungen unterworfen. Die Leistungsanforderungen der Kommunikation schwankt, wenn sie von variablen Umgebungsparametern abhängt. Die dynamische Planung kann hier das System besser auslasten als eine unveränderliche, statische Planung.

Im Abschnitt 1.1 werden die Grundlagen der Echtzeitverarbeitung vorgestellt. Dabei werden allgemeine Aspekte angesprochen, um eine ganzheitliche Sicht auf die Thematik zu erreichen. Durch diese umfassende Einleitung, die teilweise über die Belange dieser Arbeit hinausgeht, wird aber im späteren eine vereinfachte und gründliche Argumentation möglich. In 1.2 wird dann durch ein Beispiel die spezielle Problematik, in der sich diese Arbeit bewegt, beschrieben und die Notwendigkeit der dynamisch planbaren Echtzeitkommunikation gefolgert. In Abschnitt 1.3 gehe ich auf die Ziele und Hauptergebnisse der Diplomarbeit ein.

### 1.1 Echtzeitsysteme

Computer können mittlerweile in vielen verschiedenen Problembereichen eingesetzt werden. Nach dem anfänglichen Einsatz als reine *Rechenanlage*, die eine große Anzahl von Additionen und Multiplikationen pro Sekunde durchführen kann, wurde die Maschinensteuerung als Einsatzgebiet entdeckt. Computer und vor allen Dingen Programme, die die Steuerungsaufgaben übernahmen, traten

plötzlich in Interaktion mit der realen, uns umgebenden Welt. D.h. sich bewegendende Maschinen<sup>1</sup> treten aufgrund der physikalischen Gesetze, die auch für sie gelten, in einen Zusammenhang mit der sie umgebenden Welt und damit in zeitliche Abläufe, die von *außen* vorgegeben werden. Bei der Verarbeitung von Daten spielt damit der Zeitfaktor eine Rolle. Man spricht bei Computersystemen, die eine Verbindung zur realen Zeit haben, von *Echtzeitsystemen*. Einzelne Aufgaben, die als Teil davon eine solche Verbindung aufweisen und aus denen die Applikationen zusammengesetzt sind, nennt man *Echtzeitaufgaben*.

Herkömmliche, nicht-echtzeitfähige Systeme, z.B. Time-Sharing-Systeme wie UNIX, sind bei der korrekten Erfüllung von Aufgaben nur auf semantische Korrektheit angewiesen. Der Korrektheitsbegriff umfaßt das korrekte Ergebnis einer Berechnung, aber zeitliche Bedingungen für die Berechnung werden nicht berücksichtigt. Der Durchsatz des Systems wird optimiert, indem der Erwartungswert für die Verarbeitungszeit minimiert wird, ohne die zeitlichen Eigenschaften einzelner Aufgaben zu betrachten.

### Korrektheit in Echtzeitsystemen

Ein semantisch korrektes Ergebnis einer Berechnung, daß nach einem Zeitpunkt  $t$  keinen oder nur noch einen geringen Wert hat, impliziert für die Verarbeitung die Einhaltung zeitlicher Bedingungen. Der Begriff der Korrektheit der Datenverarbeitung wird also um zeitliche Spezifikationen erweitert. Die Korrektheit einer Verarbeitung wird nicht nur vom Ergebnis sondern auch von zeitlichen Bedingungen, die daran gebunden sind, bestimmt.

In einem Echtzeitsystem ändert sich der Status des Systems mit der fortschreitenden Zeit!<sup>2</sup>

### Beispiele für Echtzeitsysteme

Beispiele für den Einsatz von Echtzeitsysteme sind überall dort zu suchen, wo ein Bezug zur Umwelt besteht: *Roboter*, die selbständig navigieren und Hindernisse umfahren sollen, sind abhängig von der realen Zeit. Die eingehenden Sensorinformationen müssen in einer festen Zeitspanne ausgewertet werden, um gegebenenfalls die Motoren einer aktiven Bewegung zu stoppen oder eine Umgehung des Hindernisses zu veranlassen. Wenn die Informationsverarbeitung hier keinen zeitlichen Schranken unterworfen ist, droht der Roboter ständig „anzuecken“. Eine *Telefonvermittlung*, die pro Sekunde 100 Telefonate vermittelt, hat andere, aber in der Konsequenz ähnliche Anforderungen. In *Flugzeugen* und *Kraftwerken* kann die unkorrekte Erfüllung der gestellten Aufgabe schwerwiegende Folgen haben. Die korrekte Verarbeitung zu gewährleisten ist hier lebenswichtig.

---

<sup>1</sup>Wobei es sich hier streiten läßt, ob die Maschine *sich* bewegt oder ob sie bewegt wird (siehe [Wei78]).

<sup>2</sup>Der Status des Systems ist abhängig von der realen Zeit.

Die Darstellung von Video- und Audiodaten ist durch die menschliche Wahrnehmungsfähigkeit ebenfalls zeitlichen Bedingungen für die Konstanz der Verarbeitung der Informationen unterworfen. In letzter Zeit gewinnt diese Mensch-Maschine Schnittstelle (graphische Benutzeroberflächen (GUI), Multimedia) an Interesse.

Insgesamt werden zunehmend komplexere, verteilte Systeme verschiedene Applikationen verarbeiten und Rekonfigurationen sind keine Ausnahmen, so daß es wünschenswert ist, Änderungen dynamisch zur Laufzeit durchzuführen.

### 1.1.1 Vorhersagbarkeit und Planung

Ein wichtiger Aspekt von Echtzeitsystemen ist die Vorhersagbarkeit. Die Korrektheit des Systems, also die korrekte Verarbeitung aller Aufgaben, muß von vornherein garantiert werden können. Das System vergibt Garantien für die korrekte Verarbeitung angenommener Aufgaben. Diese Annahme bedarf einer *Planung* der Betriebsmittelzugriffe, um **Betriebsmittelkonflikte** zu vermeiden. Ein Konflikt entsteht, wenn für die nebenläufige Ausführung der Aufgaben nicht genügend Betriebsmittel (BM) bereitstehen, um die Einhaltung der zeitlichen Spezifikation zu gewährleisten.

Um die Verarbeitung vorhersagbar zu machen und um die erteilten Garantien einzuhalten, sind zwei sich ergänzende Voraussetzungen zu beachten:

1. Die Nutzungszeiten der erforderlichen Betriebsmittel für die Lösung eines Problems bzw. einer Aufgabe müssen spezifiziert werden. Neben einem Programm, das die Lösung berechnen soll, ist der Bedarf an Betriebsmitteln anzugeben.
2. Um von Seiten des Echtzeitsystems die Verfügbarkeit der spezifizierten Betriebsmittel zu sichern, müssen die Zugriffe darauf geplant werden. Die *Planung* regelt die BM-Zugriffe und erlaubt somit die Kontrolle und Vermeidung von Konflikten, die die Einhaltung der zeitlichen Spezifikationen stören.

In fast allen praxisrelevanten Fällen ist der Aufwand für die Suche nach einem Plan sehr komplex: NP-vollständig [GJ79].

Die Durchsetzung eines Planes und die Einhaltung der Spezifikationen sind zwei wesentliche Grundlagen für ein vorhersagbares Echtzeitsystem. Dies ist einer der wichtigsten zu beachtenden Aspekte in der Konstruktion solcher Systeme.

#### **Echtzeit vs. Schnelligkeit**

Es muß besonderer Wert darauf gelegt werden, Echtzeit nicht mit Schnelligkeit oder Effizienz gleichzusetzen [Sta88]. Die eigentliche Herausforderung ist hier, eine verlässliche Garantie für die Einhaltung der zeitlichen Spezifikation vergeben zu

können und somit die Verarbeitung vorhersagbar zu machen! Die um die zeitlichen Bedingungen erweiterte Korrektheit ist einzuhalten.

Wie in herkömmlichen, nicht-echtzeitfähigen Systemen ist die schnelle und effiziente Ausführung ein Ziel der Optimierung<sup>3</sup> bzw. Verbesserung eines *korrekten* Systems. Zu einer gegebenen Menge von Aufgaben soll die (geplante) Auslastung möglichst niedrig sein, um einen wirtschaftlichen Einsatz zu ermöglichen – ist die Auslastung zu hoch muß ein teureres, leistungsfähigeres System eingesetzt werden.

### 1.1.2 Klassifizierung

Über die Güte und den Umfang der Garantien kann man Echtzeitsysteme und Echtzeitaufgaben als Teile davon klassifizieren.  $\mathcal{A}$  bezeichne die Menge aller im System vorkommender Aufgaben.

1. 100%ige Garantie für alle Aufgaben

Alle Aufgaben  $A \in \mathcal{A}$  können sich auf die Verfügbarkeit der erforderlichen Betriebsmittel verlassen. Dazu müssen alle Spezifikationen für  $\mathcal{A}$  im Voraus bekannt und analysiert sein. Eine Planung vor dem Start des Systems (offline) sichert die Verfügbarkeit. Die Menge  $\mathcal{A}$  ist konstant und es können keine Aufgaben neu hinzukommen.

2. 100%ige Garantie für bestimmte Aufgaben

Nur für eine bestimmte Menge  $\mathcal{M} \subseteq \mathcal{A}$  von Aufgaben kann die konfliktfreie Ausführung garantiert werden. Eine vorangehende Planung entscheidet welche Aufgaben zu dieser Teilmenge gehören. Wenn nicht ausreichend BM für eine Echtzeitaufgabe verfügbar sind, wird die Aufnahme in  $\mathcal{M}$  verweigert.

Die Klassifizierung in den beiden Punkten kann auch auf einzelne Aufgaben bezogen werden. Dem ersten Punkt entsprechen Aufgaben, deren Ausführung *sofort* garantiert werden muß, wenn sie ankommt, d.h. dem System bekannt wird. Dem zweiten Punkt entsprechen Aufgaben, die nur ausgeführt werden, wenn sie garantiert worden sind.

Eine Modifikation der obigen Punkte erfolgt durch:

3  $p\%$ ige Garantie

Die konfliktfreie Nutzung der spezifizierten Betriebsmittel kann nur mit der Wahrscheinlichkeit  $\frac{p}{100}$  garantiert werden.

#### 1.1.2.1 Statische Planung

Die statische Planung findet vor dem Systemstart statt. Alle Aufgaben und Spezifikationen sind a priori bekannt (statische offline Analyse) und bleiben zur Laufzeit unverändert. Die Planung selbst ist nicht zeitkritisch und somit läßt sich

---

<sup>3</sup>In der Informatik versteht man unter ‘Optimierung’ die Verbesserung eines Ablaufs.

im allgemeinen trotz der Komplexität des Problems ein Plan finden, wenn einer existiert.

Die Vorhersagbarkeit von Echtzeitsystemen und Aufgaben, welche dem ersten Punkt der Klassifizierung entsprechen, muß mit der statischen Planung sichergestellt werden. Andernfalls kann die Planung später ankommender Aufgaben nicht gesichert werden.

Die notwendige vollständige Beschreibung erlaubt eine maximale Vorhersagbarkeit des statischen Systems, da von *allen* Komponenten das zeitliche Verhalten bekannt sein muß. Letzteres ist aber auch der eindeutige Nachteil der statischen Planung. Da der Bedarf an Betriebsmitteln vor dem Start festliegen muß und nicht mehr verändert werden kann, ist die Anpassungsfähigkeit einer solchen Planung sehr gering und es ist ein statisches Modell der umgebenden, realen Welt, mit der das System interagiert, notwendig. Für den Einsatz in dynamischen Umgebungen ist die statische Analyse schlecht geeignet, weil die BM-Anforderungen variieren.

Eine Möglichkeit die Anpassungsfähigkeit zu erhöhen sind Betriebsmodi mit verschiedenen Konfigurationen und Planungs*versionen*. Dafür muß das System während des Betriebs in die unterschiedlichen Modi ‘umschaltbar’ sein [SRLR89].

### 1.1.2.2 Dynamische Planung

Die dynamische Planung bearbeitet ankommende Aufgaben zur Laufzeit. Es wird während des Systembetriebs geprüft, ob die zeitlichen Spezifikationen eingehalten werden können. Diese inkrementelle Planung versucht einzelne Aufgaben, ohne die Kenntnis zukünftiger Laständerungen (Planungsanforderungen), in das bestehende System zu integrieren.<sup>4</sup>

Ändern sich die Umgebungsparameter und damit die BM-Anforderungen der Aufgaben, so kann das durch eine Neuplanung berücksichtigt werden (ausplanen, Spezifikation ändern, einplanen). In einer dynamischen Umgebung, in der die Ausführungszeit einer Aufgabe stark von einem Umgebungsparameter abhängt, führt die statische Planung zu einer schlechten Auslastung des Betriebsmittels. Die Offline-Analyse legt die Zeitbedingungen einmal fest und muß für eine sichere Erfüllung der Aufgabe von der maximalen BM-Anforderung ausgehen. Dadurch kann auf eine erheblich reduzierte Ausführungsdauer durch die aktuellen Umgebungsparameter nicht eingegangen werden. Die dynamische Planung erweitert das Modell der statischen Planung, indem solche Aufgaben zur Laufzeit mit einer verfeinerten Spezifikation geplant werden können.

Der Mehraufwand der dynamischen Planung steht der flexibleren Abstimmung der Spezifikation gegenüber: Der Anteil der von Echtzeitaufgaben tatsächlich genutzten an der verplanten BM-Auslastung, die *Planungsauslastung*, kann

---

<sup>4</sup>Diese Unkenntnis kann die Planbarkeit erheblich herabsetzen: [BKM<sup>+</sup>91] zeigt, daß die Summe der erfolgreich genutzten CPU-Zeit einer dynamischen Planung höchstens  $\frac{1}{4}$  der einer „allwissenden“ Planung sein kann.

erhöht werden. Der vorhersagbare Einsatz wird unter Umständen erst möglich, wenn die Planungsauslastung hinreichend hoch ist (vgl. Diskussion über ‘Schnelligkeit’ auf S. 17).

Die Garantievergabe zur Laufzeit kann abgelehnt werden, wenn der Planungsalgorithmus wegen einer Systemüberlastung (overload) oder einer unvollständigen Suche keine konfliktfreie Ausführung garantieren kann.<sup>5</sup> In diesem Fall müssen abgelehnte Aufgaben später, wenn laufende Applikationen beendet sind, erneut gestellt werden. Die Vorhersagbarkeit von Systemen und Aufgaben, welche dem zweiten Punkt der Klassifikation entsprechen, kann durch eine dynamische Planung sichergestellt werden; zur Laufzeit entscheidet die Planung über die Zugehörigkeit zur Menge  $\mathcal{M}$ .

Die Laufzeiten der benutzten Algorithmen sind kleiner zu halten als die der statischen Gegenstücke, um den Durchsatz des gesamten Systems nicht durch lange Planungszeiten zu bremsen und somit seine Planbarkeit zu sichern. Weil das Planungsproblem (meist) NP-vollständig ist, kann ein (dynamischer) Online-Algorithmus oft nur einen unvollständigen Planungsversuch durchführen, ohne daß eine erschöpfende Suche möglich ist. Daher kann nicht immer eine erfolgreiche Planung erfolgen, wenn dies auch theoretisch möglich wäre.

### 1.1.3 Echtzeitbedingungen

Zu der Aufgabenstellung an ein Echtzeitsystem gehört neben dem Programm als Rechenvorschrift auch eine Spezifikation der zeitlichen Bedingungen unter denen das Ergebnis erzielt werden soll. Die Verletzung dieser Bedingungen verursacht einen Zeitfehler. Die Bedingungen für eine zeitlich korrekte Erfüllung der Echtzeitaufgaben lassen sich in folgende Klassen unterscheiden (vgl. [ZA95] und [Net97]):

1. *Harte* oder strenge Zeitbedingungen (HRT<sup>6</sup>) müssen unter allen Umständen eingehalten werden, weil es sonst zu einem Systemversagen kommen kann. Mechanismen der Fehlertoleranz sind einzusetzen, wenn Zeitfehler nicht konstruktionsbedingt ausgeschlossen werden können. Ein Ergebnis außerhalb der Bedingungen ist wertlos bzw. hat einen negativen Wert (sogar  $-\infty$ ).

Aufgaben, die kritisch für die Erfüllung der Gesamtapplikation sind und deren Scheitern Kosten oder Schäden in hohem Maße verursachen, werden mit harten Echtzeitbedingungen versehen. Bestimmte Aufgaben in der Steuerung eines Kraftwerkes liegen z.B. in dieser Klasse.

---

<sup>5</sup>Aus der Unkenntnis der zukünftigen Laständerungen können Konflikte entstehen, die bei Vorabkenntnis der Spezifikationen vermieden werden könnten. Dies ist auch eine Form der Überlastung.

<sup>6</sup>engl.: *hard real-time*

2. *Weiche* oder lockere Zeitbedingungen (SRT<sup>7</sup>) müssen zum überwiegenden Teil erfüllt werden. Zeitfehler sind tolerierbar, ohne daß das System versagt. Der Wert eines Ergebnisses nimmt mit der Überschreitung der zeitlichen Bedingungen ab bzw. erreicht bei vorzeitigem Abbruch nicht das gewünschte Maximum.

Aufgaben, bei denen sich die Kosten für ein Scheitern nicht wesentlich von ihrer Nützlichkeit unterscheiden, werden mit weichen Echtzeitbedingungen versehen. Die Darstellung einer Bildfolge aus einem Videodatenstrom kann solch eine weiche Echtzeitaufgabe sein (einzelne Bilder können ausgelassen werden).

3. Die Klasse der Aufgaben, deren korrekte Ausführung an keine zeitlichen Bedingungen geknüpft ist, bezeichnet man mit Nicht-Echtzeit (NRT<sup>8</sup>). Die „herkömmliche“ Datenverarbeitung gehört der Klasse der NRT-Aufgaben an.

Die drei Typen von Anforderungen existieren in der Realität nebeneinander und können somit auch zusammen in einem System vorkommen (HRT-, SRT, NRT-Aufgaben). Der Typ des Systems bestimmt sich dabei nach der strengsten Teilaufgabe; der unbeabsichtigte und unkontrollierte Einsatz verschiedener Echtzeitklassen gefährdet die Vorhersagbarkeit.

### **Harte Echtzeit**

Die strengsten Anforderungen liegen im Bereich der harten Echtzeit. Um die korrekte Ausführung einer HRT-Aufgabe zu sichern, müssen die zeitlichen Bedingungen unbedingt eingehalten werden. Daher ist der maximale Betriebsmittelbedarf mit einer Worst-Case Analyse der Aufgabe zu bestimmen. Andernfalls würde nicht immer die volle Funktionalität durch die Garantie abdeckt (Ausführungszeit). Der Entwickler ist für die ausreichende Spezifikation verantwortlich und muß gegebenenfalls eine Zeitfehlerbehandlung bereitstellen, wenn diese Fehler nicht ausgeschlossen sind.

Es existieren HRT-Aufgaben, die unbedingt ausgeführt werden müssen und bei denen eine Ablehnung der Garantie unannehmbar ist; z.B. auf externe Ereignisse reagierende Aufgaben. Die Zeitbedingungen sind mit der Ankunft im System hart und eine statische Planung ist für die vorhersagbare Ausführung notwendig.

Aufgaben, die erst nach erfolgreicher Planung harten Zeitbedingungen unterliegen, können dynamisch geplant werden. Zur Unterscheidung werden sie auch *essentiell* genannt [SRN91, Net97], aber im folgenden wird zur Vereinfachung auf diese Unterscheidung verzichtet, auch wenn die harten Bedingungen erst nach der Planung existieren.

---

<sup>7</sup>engl.: *soft real-time*

<sup>8</sup>engl.: *non-real-time*

### Weiche Echtzeit

Die Klasse der weichen Echtzeit ist klar abgegrenzt, weil Zeitfehler in der Ausführung nicht zu einem Systemversagen führen und somit toleriert werden können. Der BM-Bedarf kann geplant werden, um die Ausführung vorhersagbar zu machen (z.B. mittlerer, erwarteter oder minimaler Bedarf). Die Unterklasse der *Best-Effort* Verarbeitung enthält SRT-Aufgaben ohne geplanten BM-Bedarf.

Zeitfehler haben die unangenehme Eigenschaft, sich schleichend fortzupflanzen, d.h. verschleppte, nicht behandelte Zeitfehler verursachen durch die Verzögerung nachfolgender Aufgaben Fehler, die nichts mit der Ursache zu tun haben (Dominoeffekt, [Net97]). Diese Fehlerfortpflanzung muß verhindert werden. Man kann z.B. die Aufgaben vorzeitig abbrechen, um die Einhaltung der Zeitbedingungen zu erzwingen (vgl. TaskPair-Konstruktion, 4.7). Eine Möglichkeit die Fehler zu charakterisieren, ist die Angabe einer Wahrscheinlichkeit, mit der die Berechnung komplett durchgeführt werden kann.

Der Begriff „weiche Echtzeitkommunikation“ leitet sich direkt daraus ab.

Die Definition des Begriffes der weichen Echtzeit ist – auch in der Literatur – noch unbefriedigend. Im wesentlichen kann man ihn aber zur Differenzierung der Zeitbedingungen nutzen und damit eine wahrscheinlichkeitsbasierte Korrektheit definieren. Vor allem der Umgang mit verletzten Zeitbedingungen ist wichtig für einen sinnvollen Einsatz von HRT- und SRT-Aufgaben in einem System.

### Nicht-Echtzeit

Die NRT-Aufgaben sind deutlich von den Echtzeitaufgaben getrennt, da weder eine Spezifikation von zeitlichen Bedingungen noch ein Einfluß der Zeit auf die Korrektheit vorhanden ist.

#### 1.1.4 Fehlertoleranz

Die Bearbeitung von Echtzeitaufgaben erfordert die Untersuchung möglicher Fehlerquellen, um durch geeignete Maßnahmen die korrekte Verarbeitung zu sichern. Für HRT-Aufgaben muß auch im Fehlerfall die Spezifikation von Seiten der Aufgabe eingehalten werden, weil darüber hinausgehende BM-Anforderungen nicht geplant sind. Bei SRT-Aufgaben ist potentiell immer mit einem Zeitfehler zu rechnen. Dieser muß abgefangen werden.

Ein Fehlermodell muß die möglichen Fehler beschreiben und ist die Grundlage für alle Maßnahmen, die für eine (Zeit-)Fehlerbehandlung notwendig sind. Zum Beispiel können durch Redundanz in der Zeit (Prüfsummen, wiederholte Übertragung; jeweils eingeplant) oder im Raum (mehrere physikalische Kommunikationsverbindungen) temporäre oder dauerhafte Übertragungsfehler toleriert werden. Der Ausfall eines Senders erfordert ein fail-safe-Verhalten und Replikation der Funktionalität auf andere Rechner.

Auf die Problematik gehe ich nicht ausführlich ein, sondern zeige nur an einigen Stellen Ansatzpunkte auf; die entsprechenden Maßnahmen können nachträg-

lich eingesetzt werden.

### 1.1.5 Lokale Echtzeitsysteme

Lokale Echtzeitsysteme bilden die Bausteine, aus denen das verteilte System, das letztlich in dieser Arbeit betrachtet werden soll, aufgebaut ist. Ein lokales System ist ein Computer, der über *einen* physikalischen Hauptspeicher verfügt, in dem alle Programme ablaufen. Die Datenverarbeitung kann prinzipiell ohne Unterstützung anderer Rechner (Server für Daten, Rechenleistung, o.ä.) erfolgen: es handelt sich um ein autonomes System.

Der Aufbau eines lokalen Echtzeitsystems ist in Abb. 1.1 skizziert. Die zugrunde liegende Hardware wird ganz oder teilweise von einem Echtzeitbetriebssystem (RT-OS<sup>9</sup>) verdeckt, das den Zugriff darauf regelt, und dessen relevanten *Dienste* eine zeitbeschränkte Verarbeitung erlauben – Dienst ist hier ein Synonym für die Nutzung eines oder mehrerer Betriebsmittel. Die Regulierung der BM-Zugriffe muß letzten Endes die oben angesprochene Planung beinhalten, um die Vorhersagbarkeit sowohl des OS als auch der Applikationen zu gewähren.

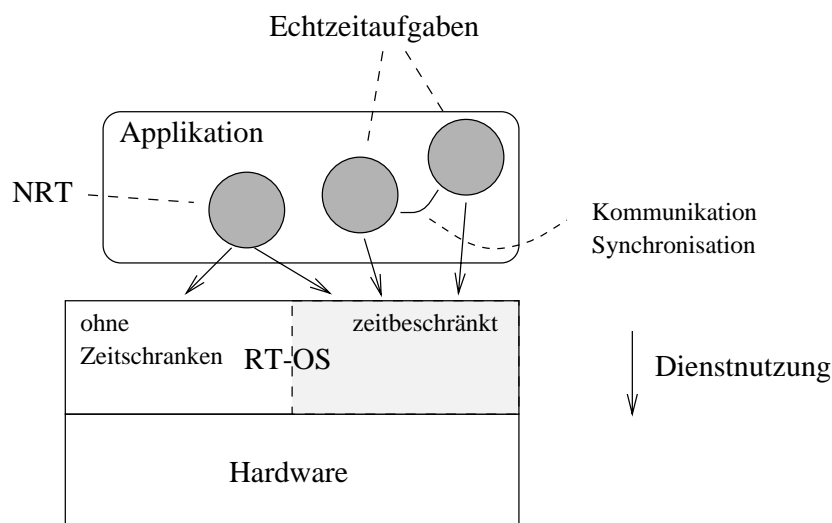


Abbildung 1.1: Aufbau eines lokalen Echtzeitsystems

Im lokalen Fall sind alle Informationen 'vor Ort' vorhanden und die kurzen Verzögerungen beim Zugriff auf den Speicher sind konstant oder zumindest aus der Planung bzw. Analyse abschätzbar. Unbekannte Größen einer (externen) Verarbeitung können durch diese Abgeschlossenheit unberücksichtigt bleiben.

<sup>9</sup>engl.: *real-time operating system*

### 1.1.6 Verteilte Echtzeitsysteme

Eine einheitliche Definition eines verteiltes System ist sehr schwierig und man findet in der Literatur mehrere, zumindest in den Details, unterschiedlichen Ansätze. Ein minimales verteiltes System ist ein Rechnernetz, also über ein Netzwerk verbundene autonome Rechner, das zusätzlich einen von mehreren Rechnern erbrachten Dienst anbietet. Weitere in der Literatur<sup>10</sup> aufgeführte Eigenschaften wie etwa der Begriff der Transparenz bleiben hier unerwähnt und sind einer späteren Erweiterung vorbehalten. Das wichtigste Unterscheidungsmerkmal zum lokalen System ist der Verlust des gemeinsamen physikalischen Speichers und der dadurch notwendigen Kommunikationswege über das Netzwerk.

#### Kommunikation

Jede Applikation, die Aufgaben auf verschiedene Rechner verteilt, braucht die Kommunikation als einen verteilten Dienst mit dem die Aufgaben verbunden werden. Das Netzwerk führt dabei zu signifikanten Verzögerungen in der notwendigen Kommunikation. In verteilten Echtzeitsystemen ist es entscheidend, daß die (maximale) Verzögerung bekannt ist, um die Vorhersagbarkeit einer Applikation zu gewähren. Man spricht in diesem Zusammenhang von **Echtzeitkommunikation**, die eine Grundlage für verteilte Echtzeitsysteme darstellt und als Betriebsmittel wie alle anderen Anforderungen geplant werden muß! Als solche wird sie in Unterkapitel 1.2 genauer untersucht und als Thema dieser Diplomarbeit herausgearbeitet.

#### Einsatz

Das Paradigma des verteilten Systems für den Entwurf von Computersystemen hat eine weite Verbreitung gefunden. Viele Probleme sind inhärent verteilt und unterstützen so dieses Design: die Gesamtapplikation läßt sich in Aufgaben zerlegen, die auf verschiedenen Rechnern getrennt bearbeitet werden und nur ihre Ergebnisse über das Netz austauschen.

Für eine Verteilung auf einzelne, selbständig arbeitende Computer können zwei Dinge sprechen. Erstens ist oft eine örtliche Verteilung vorhanden, so daß eine Verarbeitung von Daten am Ort ihrer Entstehung möglich ist. Und zweitens kann die Nutzung getrennter Rechner Betriebsmittelkonflikte zwischen Aufgaben vermindern, die konkurrierende BM-Anforderungen besitzen: das System skaliert. Das Kommunikationsaufkommen zwischen diesen Aufgaben muß natürlich vom Netz verkraftet werden können; es sind Garantien hierfür notwendig. Die physikalische Trennung der Verarbeitung ist ein wichtiger Aspekt der Fehlertoleranz und kann die Zuverlässigkeit der Verarbeitung erhöhen.<sup>11</sup>

---

<sup>10</sup>In [Tan95] ist definiert: „Ein verteiltes System ist eine Sammlung voneinander unabhängiger, vernetzter Computer, die dem Benutzer des Systems den Eindruck vermitteln, es handle sich um einen einzigen Computer“.

<sup>11</sup>Die Wahrscheinlichkeit, daß *ein* Rechner ausfällt, nimmt zwar zu, aber mit Hilfe von Mecha-

Die Skalierbarkeit des verteilten Echtzeitsystems hängt von der Netztechnologie und den verwendeten Protokollen ab. Die in der Regel gegenüber der lokalen Kommunikation um Größenordnungen langsamere<sup>12</sup> und mit dem Netzwerk als verteiltes Betriebsmittel schwierig zu planende Kommunikation beschränkt die Anzahl der verteilten Aufgaben.

## 1.2 Problematik

Zukünftige Echtzeitapplikationen werden komplizierter und in verteilten Umgebungen ablaufen, wo sie nur einen Teil des Gesamtsystems ausmachen. Infolgedessen sind Rekonfigurationen und Änderungen von Teilen des Systems keine Ausnahme und die Entscheidung, ob dies möglich ist, muß zur Laufzeit *neben* bereits aktiven Anwendungen getroffen werden. An einem Beispiel soll dies verdeutlicht werden.

Eine Anzahl von Roboter  $R_1, \dots, R_n$  arbeitet an der Erfüllung eines gemeinsamen Ziels in einer realen Umgebung. Die Steuerung findet jeweils lokal in den einzelnen Robotern statt und läuft unter harten Echtzeitbedingungen ab; die Kommunikation zwischen ihnen geschieht z.B. über Funk. Verschiedene Aktionen  $A_1, \dots, A_n$  sollen ausgeführt werden, für die ebenfalls harte Zeitbedingungen gelten (Übergabe eines Werkstücks, synchronisierte Arbeiten). Je nach Aktion nehmen verschieden viele Roboter mit unterschiedlichen lokalem Arbeitsaufwand daran teil, so daß  $A_i$  und  $A_j$  unter Umständen nebenläufig ausgeführt werden können. Der BM-Bedarf einer Aktion hängt wesentlich von dem aktuellen Status der Roboter ab (Positionen, Arbeitsgeschwindigkeiten usw.). Die Gesamtstrategie zur Erfüllung des Ziels und der Einfluß der Umwelt ermöglichen bzw. bedingen verschiedene Aktionsreihenfolgen, die nicht von vornherein bekannt sind.

Eine gemeinsame Aktion  $A_1$  soll von  $R_1$  und  $R_2$  ausgeführt werden und erfordert neben der jeweils lokalen Planung eine Kommunikation über den Fortschritt der Aktion (Statusmeldungen, Verarbeitungsergebnisse, Synchronisation, u.ä.). Ein Roboter kann an mehreren Aktionen gleichzeitig teilnehmen.

Statusinformationen werden an einer zentraler Stelle gesammelt, um das System zu visualisieren. Diese Kommunikation ist weniger strengen, weichen Zeitbedingungen unterworfen: Die Visualisierung sollte möglichst nah am aktuellen Zustand des Systems sein, aber es können durchaus Verzögerungen akzeptiert werden.

---

nismen der Fehlertoleranz kann das restliche System u.U. die Aufgaben übernehmen und somit ein komplettes Versagen verhindern; es handelt sich dann um einen partiellen Fehler (partial failure).

<sup>12</sup>Die Einführung von neuen Hochgeschwindigkeitsnetzen wie ATM kann hier eine Veränderung bringen [vEBBV95].

### Verteilte Echtzeitsysteme

Die Steuerung der Roboter macht den Einsatz von *Echtzeitsystemen* notwendig. Bewegliche Roboter müssen auf Hindernisse reagieren, um Zusammenstöße zu vermeiden, und die interne Kontrolle der Abläufe erfordert Zeitbedingungen für Sensor/Aktor-Kontrolle. Die *Verteilung* der Steuerung erleichtert die Konstruktion. Die komplexe Aufgabe der einzelnen Robotersteuerung wird jeweils lokal im Roboter durch einen Steuerrechner durchgeführt. Dies macht sie (teilweise) autonomen Systemen, die den internen Arbeitsaufwand vor dem Rest der Anlage verbergen.

### Echtzeitkommunikation

Während gemeinsamer Aktionen erfordert die notwendige Absprache zwischen beteiligten Robotern bzw. zwischen den jeweiligen Steuerrechnern eine zeitlich beschränkte Kommunikation gemäß spezifizierter Leistungsparameter (QoS – Quality of Service). Eine zeitgerechte Reaktion von  $R_2$  auf Ergebnisse von  $R_1$  machen einen Datenaustausch innerhalb fester Fristen notwendig.

Die *Echtzeitkommunikation* bildet die notwendige Grundlage einer reibungslosen Abstimmung zwischen den Rechnern und ist als solche wie die Nutzung anderer Betriebsmittel zu planen!

HRT-Aufgaben erfordern in der Regel eine HRT-Kommunikation, denn übertragene Daten, die als Eingabe dienen, können sonst durch eine Verzögerung die korrekte Ausführung der abhängigen Aufgaben verhindern: Zeitfehlerfortpflanzung durch einen Dominoeffekt.

An der Visualisierung und Kontrolle der Roboter wird die Notwendigkeit der Integration von harter und weicher Echtzeit deutlich. In komplexen Systemen werden in der Regel Aufgaben aus unterschiedlichen Echtzeitklassen verarbeitet. Die Steuerung ist harten Zeitbedingungen und das Sammeln und Auswerten der Visualisierungsinformationen ist weichen Zeitbedingungen unterworfen.

### Dynamik

Der Ablauf des Systems und die entsprechenden Betriebsmittel können nicht im voraus geplant werden und daher sind sie zur Laufzeit, vor dem Start einer Aktion, anzufordern. Die Aktionsreihenfolge steht nicht fest und vor allem ist der BM-Bedarf in erheblichem Maße von dem aktuellen Zustand des Systems bzw. der Umgebung abhängig. Eine statische Planung ist damit aufgrund der großen Zahl an Kombinationsmöglichkeiten (Modi der Planung) und des variierenden BM-Bedarfs der Aktionen nicht sinnvoll einzusetzen.

Die Rekonfiguration von *Teilen* des Systems verändert auch das Kommunikationsverhalten und die Netzwerkzugriffe müssen ebenfalls zur Laufzeit geplant werden.

### Systemmodell

Aus diesem Beispiel läßt sich nun das Modell des Systems, das in dieser Arbeit betrachtet wird, ableiten. Eine Menge autonomer, selbständig arbeitender Einprozessorrechner sind über ein lokales Netzwerk mit geringer Ausdehnung (LAN) verbunden. In diesem laufen verteilte Echtzeitapplikationen, die sowohl harte als auch weiche Bedingungen an die Verarbeitung stellen. Die Planung der für die Verarbeitung notwendigen lokalen Betriebsmittel (CPU) erfolgt jeweils in den Computern dynamisch zur Laufzeit. Die verteilten Anwendungen sind auf eine vorhersagbare Echtzeitkommunikation angewiesen, an die sie folgende Forderung stellen: Zur Laufzeit müssen verbindungsorientierte, zeitlich vorhersagbare Datenübertragungen mit spezifizierten Leistungsparametern einplanbar und zerstörbar sein.

Dieses Modell eines verteilten Systems ist vielfältig einsetzbar. Der Verzicht auf dedizierte Hardware<sup>13</sup> erschwert die Planung des Systems, ist aber für einen Einsatz in räumlich und wirtschaftlich engen Grenzen sehr gut geeignet. Dies erleichtert den Einsatz in eingebetteten Systemen<sup>14</sup> und Robotersteuerungen (siehe Beispielapplikation in 6.4.1).

### Hauptprobleme

Welche Probleme sind nun konkret bei „Dynamisch planbarer Kommunikation in verteilten Echtzeitsystemen“ zu lösen? Um ein vorhersagbares Kommunikationsverhalten zu erreichen, ist das Netzwerk als verteiltes Betriebsmittel dynamisch zu planen und dabei sind in erster Linie zwei Bedingungen zu erfüllen:

1. Das Netzwerk muß in der Lage sein, die Nachrichten gemäß ihrer zeitlichen Spezifikationen zu übertragen. Betriebsmittelkonflikte in der Nutzung des Netzes stören die Einhaltung der zeitlichen Spezifikationen. *Kollisionen* auf dem physikalischen Kommunikationsmedium verhindern eine fehlerfreie Übertragung. *Verstopfungen* in der BM-Nutzung erzeugen Zeitfehler wegen der begrenzten Kapazität des Betriebsmittels.<sup>15</sup>

Die Planung der Kommunikation hat das Ziel, die Kollisionen und Verstopfungen zu vermeiden, die die Zeitbedingungen verletzen würden.

2. Der Aufwand für die Protokollverarbeitung auf der Sender- und Empfängerseite muß in die Planung der CPU integriert werden. Für das zeitgerechte Übertragen/Senden der Daten und für die Abgrenzung gegen weitere laufende Applikationen ist der Aufwand entsprechend zu planen.

---

<sup>13</sup>Zusätzliche Hardware wie z.B. ein weiterer (Kommunikations-)Prozessor erleichtert die Realisierung einer vorhersagbaren Kommunikation durch die Aufgabentrennung zwischen Applikation und Systemleistung (Kommunikationsdienst läuft dediziert auf dem Kommunikationsprozessor).

<sup>14</sup>engl.: *Embedded Systems*

<sup>15</sup>Jeder BM-Konflikt bei der CPU-Nutzung ist eine Verstopfung. Kollisionen treten beim Netzwerk zusätzlich auf.

Die **Integration** der Datenübertragung in die verteilte Echtzeitverarbeitung des gesamten Systems ist eine fundamentale Voraussetzung für einen praxisnahen Entwurf. Wenn dies, wie es in der Literatur oft geschieht, nicht berücksichtigt wird, sind ungewollte Interaktionen mit laufenden Echtzeitanwendungen die Folge.

Die Echtzeitkommunikation besteht also aus der Datenübertragung zwischen einzelnen Rechnern, der Anbindung der RT-Aufgaben (Kommunikation zwischen Aufgaben) und der Integration in die Echtzeitverarbeitung des gesamten Systems.

Die Datenübertragung setzt sich aus der Signalübertragung einzelner Bits und Nachrichten, auf die in dieser Diplomarbeit nicht eingegangen wird, sowie aus einem Zugriffsprotokoll, das den Beginn der Signalübertragung festlegt, zusammen. Die Datenübertragung, d.h. in erster Linie das Zugriffsprotokoll, muß als Grundlage der Echtzeitkommunikation für einzelne Rechner eine begrenzte Verzögerung für den Zugriff auf das Netzwerk sichern; gleiches gilt für die Übertragungsdauer. Für eine Kommunikation mit harten Zeitbedingungen muß eine maximale Verzögerung existieren und bekannt sein. Andernfalls werden höchstens weiche Bedingungen unterstützt, weil eine feste Deadline nicht garantiert werden kann.

Der Zugriff auf das Netz und den Prozessor muß gemeinsam zur Laufzeit geplant und durch eine geeignete Laufzeitrepräsentation in die Planung des gesamten Systems integriert werden. Diese Repräsentation steht in der weiteren Planung zur Verfügung, um zum Beispiel abhängige Aufgaben vor bzw. nach der Datenübertragung für die Datenauf- bzw. -verarbeitung eingeplanen zu können [Mü97].

Die Dynamik und die Verteilung bringen weitere Probleme mit sich. Zur Laufzeit muß über das Netz, neben der Kommunikation der Applikationen, auch eine Absprache über neu zu planende Kanäle erfolgen und in den beteiligten Rechnern sind, neben der jeweils lokalen CPU-Planung, auch die Netzzugriffe zu planen. Diese Aufgaben bilden die verteilte Planung, die sich über mehr als einen Rechner erstreckt und in die Echtzeitverarbeitung des gesamten Systems eingreift, um die (zeitlich) korrekte Kommunikation zu garantieren.

Sowohl harte als auch weiche Zeitbedingungen für die Kommunikation kommen in der Verarbeitung vor und beide Echtzeitklassen sind zu unterstützen; eine Einschränkung auf reine HRT-Kommunikation ist für komplexe Systeme oft nicht möglich bzw. nicht sinnvoll. Abhängige Teilaufgaben einer Anwendung sind in den meisten Fällen nur dann verteilbar, wenn die zeitlichen Bedingungen an die Kommunikation zwischen ihnen mindestens von der gleichen Klasse sind wie die Echtzeitbedingungen der Prozessornutzung: HRT-Applikationen benutzen HRT-Kommunikation.<sup>16</sup> Wenn die Inhalte einer Kommunikation  $A \rightsquigarrow B$  als Eingabe für eine Aufgabe  $B$  dienen, dann kann  $B$  nur harten Bedingungen genügen, wenn

---

<sup>16</sup>Evtl. können SRT-Aufgaben mit einer potentiell unbeschränkten Verzögerung in der Kommunikation auskommen, wenn im Mittel ein akzeptabler Wert erwartet wird [BMR96, KGM96].

dies auch für die Kommunikation gilt. Die verschiedenen Echtzeitklassen müssen nebeneinander existieren können.

### Kommunikationssystem

Für die Planung und Kontrolle der Datenübertragungen ist ein Kommunikationssystem als ein verteilter Dienst notwendig, der auf allen kommunizierenden Rechnern eingesetzt wird. Die Übertragungen müssen geplant sein, um ungewollte BM-Konflikte zu vermeiden. Die Planung beginnt mit der Spezifikation der gewünschten Leistung (QoS), die daraufhin in einer Verhandlung (QoS-Negotiation) zwischen Sender, Netzwerk und Empfänger festgelegt und garantiert wird. Der Dienst muß in die CPU-Planung der einzelnen Knoten bzw. der Sende- und Empfangsaufgaben integriert werden.

## 1.3 Ziel der Diplomarbeit

### 1.3.1 Kommunikationsmodell

Die Kommunikation zwischen verteilten Echtzeitaufgaben erfolgt über verbindungsorientierte, unidirektionale *Kanäle* (s. Abb. 1.2). Der Zugriff auf den Kanal erfolgt aus der Applikation heraus über einen Puffer in den bzw. aus dem die Daten kopiert werden. Die Puffer auf der (eindeutigen) Sender- und Empfängerseite werden durch den Kanal verbunden. Die Protokollverarbeitung und Datenübertragung wird vom Kanal asynchron zur Applikation in einem eigenen Kontrollfluß ausgeführt.

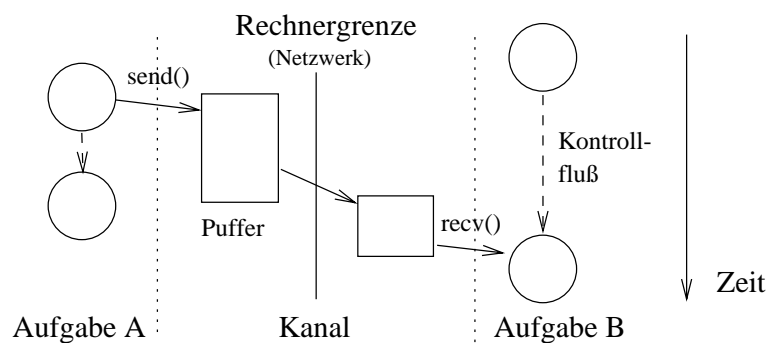


Abbildung 1.2: Unidirektionale Kommunikation durch einen Kanal, ohne Betrachtung der CPU-Aktivität des Kanals

Die Integration der Kommunikation in die einzelne Applikation wird über diese minimale Schnittstelle realisiert: Aufbauend auf der Kommunikation über die Puffer kann ein weiterführendes Kommunikationsmodell für die verteilte Applikation in zukünftigen Arbeiten behandelt werden.

Für eine Kommunikation  $A \rightsquigarrow B$  muß ein Kanal aufgebaut und die Betriebsmittel Netzwerk und CPU entsprechend der spezifizierten Leistung bzw. der zeitlichen Parameter (QoS) geplant werden. Das Kommunikationssystem vergibt für erfolgreich geplante Kanäle eine Garantie, daß spezifikationsgemäß bereitgestellte Daten innerhalb der zeitlichen Bedingungen durch den Kanal übertragen werden. Die sendebereiten Daten, welche die Aufgabe  $A$  in den Puffer kopiert hat (`send()` bzw.  $A \rightsquigarrow$ ), werden unter geplanter Nutzung der Betriebsmittel Netzwerk und CPU (in beiden Rechnern) in den empfängerseitigen Puffer übertragen ( $A \rightsquigarrow B$ ). Somit wird sichergestellt, daß nach den durch die Planung vereinbarten Zeiten  $B$  auf die Daten mit `recv()` zugreifen kann ( $\rightsquigarrow B$ ). Der Kanal bietet eine geplante „Ende-zu-Ende-Kommunikation“ zwischen Echtzeitaufgaben im verteilten System.

Auf dem Konzept der Kanäle können zusätzlich RPC<sup>17</sup> und bidirektionale Kommunikation nachgebildet werden und es stehen somit ausreichende Mechanismen für die zeitlich gebundene Kommunikation in verteilten Echtzeitsystemen zur Verfügung, um die oben angesprochene Problematik zu lösen.

Eine Applikation, die aus den zwei kommunizierenden Aufgaben  $A$  und  $B$  besteht, wird in zwei Schritten geplant. Zunächst ist in einer verteilten Planung des Kommunikationssystems ein Kanal mit den erforderlichen Parametern aufzubauen. Mit der Planung werden die zeitlichen Parameter und damit die Leistung der Kommunikation innerhalb der Spezifikation festgelegt. Die erforderliche Belastung der CPU wird geplant und ist als Laufzeitrepräsentation im System sichtbar. Diese Information wird im zweiten Schritt für die lokale CPU-Planung der von der Kommunikation abhängigen Aufgaben  $A$  und  $B$  genutzt.<sup>18</sup> Damit läßt sich die Kommunikation als Teil der Applikation in deren Planungsvorgang integrieren. Die Diplomarbeit konzentriert sich auf die darunterliegende Kommunikationsverbindung zwischen den beiden Puffern.

### 1.3.2 Kommunikationsprotokoll

Die Anforderungen, die die Kommunikation erfüllen wird, kann man in zwei Punkte unterteilen, die den Netzzugriff innerhalb eines Rechners bzw. die Kommunikation zwischen verschiedenen Rechnern umfassen.

- *Verteilte* Applikationen müssen ihre BM-Zugriffe auf Netz und CPU planen und abstimmen.

Dieser Punkt entspricht der oben angesprochenen Problematik der BM-Planung und der Integration der Kommunikation in die Echtzeitverarbeitung der lokalen Systeme.

---

<sup>17</sup>engl.: *remote procedure call*

<sup>18</sup>Alternativ könnte man auch die Planungsreihenfolge verändern, aber dies käme einer modifizierten Kommunikationsspezifikation gleich und soll im weiteren nicht betrachtet werden.

- *Zeitgesteuerte* Applikationen werden unterstützt. Sie gehen von einem unbedingten, verzögerungsfreien Zugriff auf die Betriebsmittel aus. Diese Applikationen laufen (scheinbar) isoliert von anderen Aktivitäten des Systems und können zeitlich sehr genau vorhergesagt werden.

Zweitrangig werden daneben auch Ereignis- oder prioritätengesteuerte Applikationen unterstützt. Sie gehen nicht von einem verzögerungsfreien BM-Zugriff aus, denn alle Aufgaben höherer Priorität erhalten vorrangig Zugriff. Ihre Zugriffe werden denen der zeitgesteuerten Applikationen nachgestellt. Die Planung ist oft einfacher als die der zeitgesteuerten Applikationen, aber sie sind in ihrem zeitlichen Verhalten weniger genau vorhersagbar [SSNB95, Kop93].

Der zweite Punkt ergänzt die oben angegebene prinzipielle Problematik der Echtzeitkommunikation: Es werden zusätzliche Forderungen an die Einsatzfähigkeit gestellt. Das resultierende Kommunikationssystem wird leistungsfähiger und flexibler sein als rein zeitgesteuerte Systeme. Viele Vorteile der Ereignissteuerung lassen sich in das System integrieren. Die folgenden, verfeinerten Ziele für den Entwurf des Protokolls werde ich in der Diplomarbeit schrittweise erreichen.

- Z1** Die Kommunikation muß harten Echtzeitbedingungen genügen können und dynamisch planbar sein.
- Z2** Die Planung und die von ihr vergebene Garantie erlauben eine feine zeitliche Auflösung, so daß auch (zeitgesteuerte) Anwendungen, die darauf angewiesen sind, einplanbar sind.
- Z3** Ein sinnvoller Umgang mit einer groben zeitlichen Auflösung muß daneben möglich sein, weil in einem komplexen System viele unterschiedliche Anforderungen an die Planung gestellt werden. HRT-Aufgaben, deren zeitliche Bedingungen zwar unbedingt eingehalten werden müssen, können aber z.B. auch einen gewissen Spielraum für die Ausführung offenlassen. Auch SRT-Aufgaben sind im Allgemeinen weniger engen zeitlichen Bedingungen unterworfen.
- Z4** Eine Verallgemeinerung des letzten Punktes fordert zusätzlich den Einsatz alternativer Zugriffsprotokolle. Der kombinierte Einsatz eines zeitgesteuerten Zugriffsverfahrens und alternativer Protokolle wird möglich sein, ohne die Punkte 1 und 2 zu verletzen.

Die schrittweise Bearbeitung dieser Punkte entspricht der Priorität die ihnen zugewiesen wird. Wenn die Anforderungen eines Punktes erfüllt sind, wird *diese* Lösung für den nächsten Punkt verfeinert und erweitert.

### 1.3.3 Voraussetzungen

Ein Rechnernetz aus Einprozessorrechnern mit jeweils lokalen Echtzeitbetriebssystemen ist die Grundlage für den Entwurf des Kommunikationssystems. Eine globale Zeitbasis und eine jeweils lokale dynamische Planung von zeitgesteuerten CPU-Aktivitäten in den Knoten werden vorausgesetzt. Das ermöglicht eine sehr gut vorhersagbare Ausführung verteilter HRT-Aufgaben in dynamischen Umgebungen. Die globale Zeit kann zur Synchronisation eingesetzt werden und die zeitgesteuerte Planung nutzt dies aus; auch fehlertolerante Echtzeitverarbeitung mit Hilfe von TaskPairs (s. 4.7 und [Str95, NSB<sup>+</sup>96, Net97]) wird unterstützt.

Ein Multiple-Access-Netzwerk wird vorausgesetzt, das keine Unterstützung zeitbeschränkter Kommunikation bietet; IEEE Ethernet oder drahtlose Funkübertragung können als Beispiel dienen. Dies ist die geringste anzunehmende Unterstützung der Netztechnik für die Echtzeitkommunikation und auf das MA-Zugriffsprotokoll können viele andere Protokolle aufsetzen (z.B. Token-Verfahren). Wenn das Netzwerk weitere Funktionalität anbietet, wie zum Beispiel der prioritätsgesteuerte Zugriff des CAN-Feldbusses, so kann diese neben dem hier entworfenen Zugriffsprotokoll eingesetzt werden (Ziel Z4, s.o.).

### 1.3.4 Realisierung

In dieser Diplomarbeit wird der Entwurf des Kommunikationssystems und eines Protokolls für den Netzzugriff kombiniert, um durch eine enge Verbindung die Leistungen möglichst genau abstimmen zu können. Die Arbeit wird sich auf die Aspekte der Echtzeitkommunikation in verteilten Steuerungssystemen konzentrieren. Neben der theoretischen Untersuchung ist die Realisierbarkeit der Ergebnisse ein zentraler Punkt. Es soll möglich sein, für Rechner, Netz und Echtzeitbetriebssystem Standardkomponenten zu benutzen.

#### 1.3.4.1 Kommunikationssystem

Das Kommunikationssystem wird als eine Erweiterung des Betriebssystems einen verteilten Dienst (verteilte BM-Verwaltung) bereitstellen und in jedem Rechner aktiv sein. Als Dienstleistung bietet es den dynamischen Auf- und Abbau von Kanälen an.<sup>19</sup> Die verteilte Planung eines Kanals umfaßt die globale Planung der Netzzugriffe und die jeweils lokale Planung der CPU für den sender- und empfängerseitigen Kommunikationsaufwand (Flußkontrolle und Protokollverarbeitung). Das Kommunikationssystem koordiniert die BM-Verwaltung von *Netzwerk* und *CPU*.

- Das *globale Netzmanagement* behandelt die globalen Aspekte der Kommunikationsplanung, die mehr als einen Rechner beeinflussen (Netzzugriffe).

---

<sup>19</sup>In 3.4.2 wird der Problembereich anhand des OSI-Modells beschrieben.

Alle Teilnehmer des Netzes müssen ihren Kommunikationsbedarf darüber planen, um die vergebenen Garantien nicht zu beeinflussen.

Die Planung der Netzzugriffe erfolgt an zentraler Stelle und ist für eine Garantie der angeforderten Kommunikationsleistung notwendig, weil das Netz alleine keine Echtzeitkommunikation unterstützt. Ein Reservierungsprotokoll verbindet die zentrale Netzplanung mit den jeweils lokalen CPU-Planungen der Kommunikationsteilnehmer (des zu planenden Kanals), die in den lokalen Netzmanagements stattfinden. Die Zentralisierung beschränkt den Planungsaufwand auf die Zentrale und die Knoten, die an dem Kanal teilnehmen.

Die Planung erfolgt kalenderbasiert und sie vergibt Zugriffszeiten für eine zeitgesteuerte Zugriffskontrolle (s.u.). Auf dem Broadcastmedium MA-Netz sind die geplanten Zugriffsrechte/-zeiten zunächst verbindungslos, werden aber im lokalen Netzmanagement zu verbindungsorientierten Kanälen erweitert.

Das globale Netzmanagement arbeitet in der zweiten Schicht des OSI-Modells (s. 3.4.1), die für die grundlegende, verbindungslose Übertragung einzelner Datenblöcke verantwortlich ist. Eine Lösung für die dritte Schicht (Netzwerkschicht), in der Routingprotokolle physikalisch getrennte Netzsegmente verbinden, wird nur kurz aufgezeigt und nicht weiter verfolgt.

- Das *lokale Netzmanagement* behandelt die lokalen Aspekte innerhalb eines Rechners und verwaltet die lokalen Zugriffsrechte und die CPU-Nutzung der Kommunikation. Die Dienstnutzer innerhalb des Knotens (Echtzeitaufgaben) bekommen die unter dem vorigen Punkt ermittelten Zugriffsmöglichkeiten der zugrundeliegenden zweiten OSI-Schicht zugewiesen und werden durch die explizite Einplanung des zeitgerechten Sendens und Empfangens in die Echtzeitverarbeitung des übrigen Systems integriert. Diese entscheidende *Integration* der CPU- und der Netzwerknutzung in die Echtzeitverarbeitung macht das System vorhersagbar.

Die eingeplante CPU-Aktivität ist eine Laufzeitrepräsentation des Kanals. Die Leistung des Kommunikationssystems ist darüber für weitere Planungen sichtbar; Abhängigkeiten in der Applikation zwischen Datenübertragung und -verarbeitung sind spezifizierbar.

Das lokale Netzmanagement, das der Transportschicht im OSI-Modell entspricht, stellt unzuverlässige Simplexverbindungen (Kanäle) bereit. Unzuverlässigkeit bedeutet hier, daß neben der Bereitstellung freier Zeitintervalle keine Integritätssicherung der übertragenen Daten (CRC<sup>20</sup>, erneutes Senden, o.ä.) stattfindet; ich gehe zunächst von einem fehlerfreien Netz aus.

---

<sup>20</sup>Prüfsumme, engl.: *cyclic redundancy check*

Darüber hinaus kann das Kommunikationssystem auch Kanäle ohne zeitliche QoS-Spezifikation für die NRT-Kommunikation zur Verfügung stellen. Nicht-Echtzeit Anwendungen, die auch im System vorhanden sind, müssen diese Kanäle benutzen, um sich ins Kommunikationssystem einzugliedern und die anderen Übertragungen nicht zu stören.

#### 1.3.4.2 Zugriffsprotokoll

Ausgehend von einem Multiple-Access-Netzwerk wird ein zeitgesteuertes Zugriffsprotokoll entworfen, das kollisionsbedingte Verzögerungen vermeidet und die dynamische Kanalplanung unterstützt (dynamisches TDMA<sup>21</sup>). Das *zeitliche Multiplexing* des Zugriffsrechts zwischen den Rechnern bzw. Echtzeitaufgaben sichert jeweils den alleinigen, isolierten Zugriff auf das Netz. Diese Eigenschaft kann für die HRT-Kommunikation genutzt werden. Ein weiterer Vorteil dieses Verfahrens gegenüber anderen ist z.B. die Möglichkeit, enge zeitliche Abstimmungen zwischen verteilten, zeitgesteuerten Aktivitäten zu planen (siehe 2.1.1.3 für eine Diskussion der Vor- und Nachteile). Die oben angegebenen Ziele Z1 und Z2 werden erreicht.

Den Vorteil des isolierten Zugriffs werde ich für die Möglichkeit nutzen, *innerhalb* bestimmter Zeitintervalle mehreren Nutzern das Zugriffsrecht zu erteilen (Kanalüberlagerung), ohne die anderen, davon isolierten Intervalle zu beeinflussen. Damit sind die Ziele Z3 und Z4 erreichbar<sup>22</sup>, denn die Vergabe des Zugriffsrechts innerhalb eines Intervalls kann auch mit einem anderen Protokoll erfolgen, solange die äußeren zeitlichen Grenzen eingehalten werden.

Bei dem Broadcastmedium MA-Netz sind die geplanten Zugriffsmuster, die reservierten TDMA-Zeiten, verbindungslos, aber sie werden im lokalen Netzmanagement durch die CPU-Planung zu verbindungsorientierten Kanälen erweitert.

Zunächst wird zur Vereinfachung von einem idealen Netz ausgegangen: kein Datenverlust, keine -veränderung oder -duplikation. Erweiterungen können diese Einschränkung teilweise aufheben (s. 4.7), aber unter Echtzeitbedingungen sind die Möglichkeiten zur Fehlerkorrektur begrenzt und müssen generell in die Planung einbezogen werden.

#### 1.3.4.3 Planung

Die kalenderbasierte Planung, die im Kommunikationssystem bzw. für das Zugriffsprotokoll eingesetzt wird, reserviert explizit die Zeitintervalle der BM-Zugriffe und garantiert deren Verfügbarkeit. Es findet keine Interpretation<sup>23</sup> der reservierten Ausführungszeiten statt, so daß das System keine WCET (Worst Case

<sup>21</sup>engl.: *time division multiple access*, s. 2.1.1.2

<sup>22</sup>Die Kanalüberlagerung innerhalb eines Rechners oder – je nach Unterstützung des Netzes – zwischen verschiedenen Rechnern hilft, auch ohne den Einsatz weiterer Zugriffsprotokolle das Ziel Z3 zu erreichen und die mögliche Planungsauslastung zu erhöhen.

<sup>23</sup>Abschnitt 5.8 geht kurz darauf ein.

Execution Time) oder EET (Expected Case Execution Time) kennt, sondern lediglich mit einer RT (Reservation Time) umgeht. Die BM-Nutzung wird am Intervallende explizit abgebrochen und es findet keine weitere Zeitfehlerkontrolle statt.

Die in den Zielen Z3 und Z4 geforderte Möglichkeit, neben der exklusiven Reservierung von Zeitintervallen auch weniger enge zeitliche Bedingungen und unterschiedliche Protokolle (nicht-exklusiv) einzusetzen, wird durch eine differenzierte Betrachtung der Intervalle erreicht. Der Test, der die Planbarkeit eines Intervalls im Kalender prüft, ist erweiterbar und die notwendigen Planungsspezifikationen *und* die -algorithmen werden an die Intervalle gebunden. Der einfachste Algorithmus prüft auf eine exklusive Nutzung des Intervalls: Unabhängig von den Planbarkeitstests in anderen Intervallen kann somit die exklusive BM-Nutzung sichergestellt und eine reine TDMA-Steuerung geplant werden. Darüber hinaus kann man in diesen Tests Algorithmen für die „gleichzeitige“ Vergabe von Nutzungsrechten (Überlagerung) und für weitere Zugriffsprotokolle implementieren, die jeweils die Planbarkeit innerhalb des spezifizierten Intervalls beurteilen.

Grobe und ungenaue Spezifikationen führen zu einer schlechten Planungsauslastung in der rein zeitgesteuerten Planung (Ziel Z3). Die Überlagerung kann man dazu benutzen, diese generellen Nachteile der TDMA-Planung (teilweise) zu beheben.

Die reservierten Zeiten werden weder für eine HRT- noch für eine SRT-Nutzung interpretiert, aber die Planung kann für beide Echtzeitklassen genutzt werden. Die exklusiv vergebenen Zeitintervalle sind isoliert für genau einen BM-Nutzer verfügbar und unterstützen dadurch die Einhaltung von HRT-Bedingungen. Weiche Echtzeitbedingungen kann man über eine erweiterte Zeitfehlerbehandlung auf die festen Reservierungszeiten aufsetzen oder die Kanalüberlagerung bzw. weitere Zugriffsprotokolle nutzen, um verschiedene Algorithmen und Leistungscharakteristika ins System zu integrieren.

Ein Hauptziel der Diplomarbeit ist hierbei der Entwurf generischer Datenstrukturen für die kalenderbasierte Planung, die die Reservierungsinformationen bzw. -zeiten enthalten. Ich stelle einfache Algorithmen für die Planungsverfahren vor, die die Einsatzfähigkeit des Konzepts demonstrieren. Die Arbeit kann als Basis für weitere Untersuchungen dienen: Diese Kombinationsmöglichkeit verschiedener Zugriffsprotokolle ist neuer Ansatz, der bisher noch nicht untersucht worden ist.

Die entwickelten Planungsverfahren und die generischen Datenstrukturen sind nicht an das Betriebsmittel *Netz* gebunden, sondern zum Beispiel auch für die CPU-Planung einsetzbar.

#### 1.3.4.4 Implementierung

Die Implementierung wird die verteilte Steuerung eines innovativen Roboters realisieren und eine Demonstration und Bewertung der erarbeiteten Konzepte erlau-

ben. Das verteilte System besteht aus einem Verbund autonom arbeitender Einprozessorrechner, die jeweils lokal mit einem Echtzeitbetriebssystem ausgestattet sind. Das gesamte System arbeitet zeitgesteuert mit einer globalen Zeitbasis. Als Kommunikationsmedium kommt der CAN-Feldbus zum Einsatz, auf den mittels TDMA zugegriffen wird. Dieser Mechanismus für die Zugriffskontrolle auf das CAN-Medium, anstatt der zur Verfügung stehen Prioritäten, erlaubt eine zeitlich genaue Vorhersagbarkeit der Kommunikation und ist auch für harte Zeitbedingungen sehr gut geeignet. Die zur Verfügung stehenden Prioritäten können für Erweiterungen genutzt werden. Der in Ziel Z4 geforderte Einsatz weiterer Zugriffsprotokolle wird in der Applikation nicht genutzt. Die Kollisionsfreiheit der CAN-Zugriffe setze ich in einer Kanalüberlagerung ein, um die Planungsauslastung trotz einer groben zeitlichen Auflösung hoch zu halten.

## 1.4 Aufbau der Arbeit

Das nächste Kapitel stellt einige verwandte Arbeiten auf diesem Gebiet vor. Arbeiten über Zugriffsprotokolle und Datenübertragung sind die Basis der Echtzeitkommunikation, die sich mit den netztechnischen Fragen auseinandersetzen. Die Arbeiten über Kommunikationssysteme bearbeiten Fragen der rechnerinternen Kommunikationssteuerung und zeigen im Vergleich mit dem hier gewählten Aufgabenbereich, daß die kombinierte Betrachtung von Netz- und CPU-Planung selten vollständig durchgeführt wird.

In Kapitel 3 fasse ich die Grundlagen für den Entwurf des Kommunikationssystems zusammen. Dazu gehören die Voraussetzungen der Zielumgebung und es werden Kriterien aufgestellt, an denen die weiteren Entscheidungen gemessen werden. Die Leistungen des Kommunikationssystems werden anhand des beschriebenen OSI-Modells charakterisiert.

Darauf aufbauend stellt Kapitel 4 den Entwurf des Kommunikationssystems vor. Ein Reservierungsprotokoll, das die Planung von Netz und CPU koordiniert, wird für den dynamischen Auf- und Abbau von Kanälen entworfen. Daneben wird für die vorhersagbare HRT-Kommunikation in den Kanälen ein Zugriffsprotokoll vorgestellt. Die gemeinsame Entwicklung erlaubt eine enge Abstimmung der Leistungen.

In Kapitel 5 wird die kalenderbasierte Planung des Zugriffsprotokolls behandelt, nachdem im vorigen Kapitel nur auf deren Abstimmung eingegangen wurde. Die Aufstellung der erforderlichen Parameter und der Entwurf einer erweiterbaren Reservierungsstruktur sind die zentralen Punkte.

Kapitel 6 stellt die Implementierung und die Beispielapplikation als Demonstrator vor.

Kapitel 7 schließt mit einer Zusammenfassung.

# Kapitel 2

## Verwandte Arbeiten

Anhand der zwei wichtigsten Punkte der Arbeit, die zeitliche korrekte Datenübertragung und die Integration der Kommunikation in das Echtzeitsystem, können die verwandten Arbeiten in den Abschnitten 2.1 und 2.3 unterschieden werden. In der Literatur sind die beiden Aspekte meist getrennt behandelt und es wird nur die Datenübertragung zwischen Rechnern betrachtet bzw. es wird von einem zeitbeschränkten Zugriff auf das Netz ausgegangen (z.B. über Token Ring, s.u.). Eine strikte Trennung ist aber nicht immer sinnvoll, da Möglichkeiten einer engen Abstimmung zwischen CPU und Netz ungenutzt bleiben, die für das System wichtig sein können.<sup>1</sup> Eine Entscheidung über das Protokoll wird aufgrund der hier verglichenen Arbeiten im Kapitel 4.2.2.3 getroffen.

### 2.1 Datenübertragung

Das Hauptaugenmerk liegt in dieser Arbeit auf Multiple-Access-Techniken. Token Ring und ATM werden auch kurz vorgestellt, weil sie einen weitverbreiteten Einsatz erlangt haben bzw. eine vielversprechende Unterstützung für die Echtzeitkommunikation bieten.

#### 2.1.1 Multiple-Access-Busse

Im Bereich der lokalen Netzwerke (LAN) ist der Multiple-Access-Bus weitverbreitet. Zwischen den Rechnern existiert *ein* physikalisches Medium, auf dem zur gleichen Zeit immer nur eine Nachricht transportiert werden kann; ein Netzsegment. Wenn zwei oder mehr Knoten gleichzeitig Daten übertragen, kommt es zur Kollision: keine der Nachrichten kann mehr gelesen werden. Es handelt sich hierbei um ein Broadcastnetz, d.h. gesendete Daten können potentiell von

---

<sup>1</sup>Arbeiten, die mit aufwendigeren Modellen die Kommunikation in den Planungsvorgang und damit in die Applikation integrieren wollen, können von der Datenübertragung abstrahieren. Das ist aber kein Ziel dieser Diplomarbeit; vgl. Integrationsbegriff auf Seite 27.

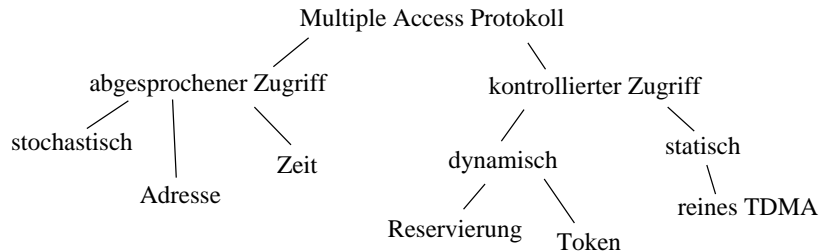


Abbildung 2.1: Charakterisierung von Multiple-Access-Protokollen

jedem angeschlossenen Rechner empfangen werden. Das einfachste Beispiel für solch ein Netzwerk stellt eine funkgestützte Übertragung dar. ALOHA gehört zu den klassischen Anfängen dieser Technik [Abr70]. Eine feste Funkfrequenz ist das gemeinsame Medium und ein gleichzeitiges, mehrfaches Senden invalidiert die Information der Übertragung. Ein unkontrolliertes Vorgehen in dieser Weise verringert den erreichbaren Durchsatz erheblich und resultiert in unvorhersagbaren Verzögerungen. Für einen Einsatz in Echtzeitsystemen muß ein vorhersagbarer Zugriffsmechanismus (medium access control, MAC) existieren.

Die zeitliche Charakterisierung der MAC kann durch eine *Verzögerungsfunktion*  $VF(t)$  beschrieben werden. Dies ist eine Verteilungsfunktion einer die Verzögerung eines gestellten Sendeauftrages beschreibenden Zufallsvariable. Diese beschreibt die Wahrscheinlichkeit, daß der Beginn der erfolgreichen Übertragung innerhalb einer gegebenen Zeitspanne möglich ist. Der Erwartungswert

$$E(VF) = \int_0^{\infty} x \frac{d}{dx} VF(x) dx$$

beschreibt die mittlere Verzögerung. Wenn die Funktion ab einem Wert  $t_{max} < \infty$  gleich eins ist, läßt sich dies als maximale Verzögerung in der Echtzeitkommunikation nutzen. Wichtig ist hierbei, ob die Funktion für beliebige Zeitpunkte gilt oder ob zu bestimmten Zeitpunkten durch eine Absprache eine besondere Funktion erreicht wird. Oft ist die Bestimmung eines existierenden  $t_{max}$  abhängig von der aktuellen Auslastung des Netzes, so daß eine Beschränkung nur durch die Kenntnis dieser Last möglich ist. Harte Bedingungen für die Kommunikation sind nur im Falle einer bekannten Beschränkung von  $t_{max}$  zu erfüllen.

Verschiedene Vorschläge für eine Realisierung der MAC und ihren Nutzen für eine Echtzeitkommunikation stelle ich im folgenden vor (s. Abb. 2.1 und [KSY84]). Man kann zwischen *abgesprochenem Zugriff* und *kontrolliertem Zugriff*, der immer die Angabe einer maximalen Verzögerung  $t_{max}$  beinhaltet, unterscheiden. Beim abgesprochenen Zugriff ist eine genaue Kenntnis der Netzlast erforderlich, um  $t_{max}$  zu bestimmen. Bei beiden Verfahren ist das Ergebnis ein zeitliches Multiplexing der sendebereiten Stationen auf dem gemeinsamen Medium; im zweiten Fall als Mittel zum Ziel, im ersten Fall lediglich als Ergebnis der Absprache.

### 2.1.1.1 Absprechende Zugriffsverfahren

Beim abgesprochenen Zugriff werden auftretende Kollisionen durch eine (implizite) Absprache zwischen den kollidierten Rechnern aufgelöst (kollisionsauflösender Zugriff). Wie schnell diese Auflösung zum Ziel kommt und alle Nachrichten übertragen werden bestimmt das Aussehen der VF. Alle vorgestellten Kollisionsauflösungen versuchen natürlich den Erwartungswert  $E(VF)$  zu verkleinern, aber für die Bestimmung der maximalen Verzögerung  $t_{max}$  ist die Kenntnis der aktuellen Netzauslastung notwendig, da die Frage, wann und in welchem Umfang Kollisionen auftreten, dafür beantwortet werden muß.

Man setzt die CSMA Zugriffskontrolle in den absprechenden Verfahren voraus. Ohne eine solche einfache Voraussetzung bleiben die erreichbaren Durchsätze weit unterhalb der theoretischen Grenze, wie Untersuchungen an frühen MA-Architekturen wie ALOHA zeigen [Tan96].

#### CSMA

Eine CSMA<sup>2</sup> Zugriffskontrolle besagt, daß eine sendewillige Station das Medium auf eine laufende Übertragung abhört, um den eigenen Zugriff ggf. zu verzögern und ein korrektes Beenden der laufenden Übertragung zu ermöglichen. Wenn das Medium belegt ist, wird die Übertragung zu einem späteren Zeitpunkt angesetzt. Hierfür können verschiedene Strategien benutzt werden:

- **nicht-persistent CSMA:** Die Übertragung wird zu einem zufällig gewählten Zeitpunkt erneut angesetzt. Im Kollisionsfall wird genauso verfahren.
- **1-persistent CSMA:** Die Station wartet auf das Ende der laufenden Übertragung und beginnt dann sofort mit dem Senden. Auf diesem Zeitpunkt können nun verschiedene Rechner gewartet haben und es entsteht sofort eine Kollision, für deren Auflösung dann gesorgt werden muß.
- **p-persistent CSMA:** Die Station verhält sich mit Wahrscheinlichkeit  $p$  1-persistent, andernfalls wird eine bestimmte Zeit gewartet. Mit dieser Strategie kann man den höchsten Durchsatz erwarten.

1-persistentes CSMA hat sich aber durchgesetzt und ist Ausgangspunkt für die verschiedenen Arten der Kollisionsauflösung im abgesprochenen Zugriff. Das frühzeitige Erkennen von Kollisionen (collision detection, CD) hilft Zeit bei der Auflösung einzusparen. Die Kollisionszeit ist die maximale Zeitspanne zwischen dem Start einer Übertragung und dem Erkennen einer Kollision. Sie ist das Zweifache der maximalen Laufzeitverzögerung  $\tau^3$  der Signals auf dem physikalischen Medium.

---

<sup>2</sup>engl. *carrier sense multiple access*

<sup>3</sup>end-to-end propagation delay

### Stochastische Auflösung

Ethernet mit 1-persistentem CSMA Zugriff und Binary Exponential Backoff Kollisionsauflösung ist ein Beispiel für eine stochastische Auflösung, die auf Wahrscheinlichkeiten beruht [MB76, IEEa]. Nach einer Kollision warten alle Rechner einen zufälligen Zeitpunkt innerhalb einer Zeitspanne ab, bevor sie erneut senden. Bei jeder folgenden Kollision, die diese Nachricht erfährt, wird die Zeitspanne verdoppelt. Für eine gegebene Auslastung läßt sich eine Verzögerungsfunktion  $VF$  bestimmen, aber es gilt immer  $t_{max} = \infty$  !

Erweiterungen des Standards sind erarbeitet worden, um die Verzögerung durch die Kollisionen zu begrenzen, aber trotzdem in einem NRT-Modus bzw. bei kollisionsfreiem Zugriff die Vorteile des reinen CSMA/CD Verfahrens zu nutzen (Verfügbarkeit bei geringer Last, wenig Overhead) [LeL87, Pal92]. Neben dem Normalbetrieb (NRT) wird in einem anderen Modus Window-Protokolle eingesetzt bzw. ein Timed Token Protocol in Software emuliert; siehe unten, [VC95].

### Adreßbasierte Auflösung

Jeder Rechner hat eine eindeutige Adresse oder Priorität, anhand derer eine Kollision aufgelöst wird. Man kann sich diese als Blätter eines statischen, binären Baums vorstellen (Adaptive Tree Walk Protocol, [Cap79]). Eine Kollision bemerken alle angeschlossenen Rechner und warten deren Auflösung ab. Dazu wird der Baum halbiert und die erste Hälfte darf senden und die andere wartet zunächst. Dieser Vorgang rekuriert so lange, bis ein sendewilliger Knoten alleinigen Zugriff auf das Netz hatte. Nach einer erfolgreichen Übertragung bleibt das Netz für eine Kollisionszeit ( $2\tau$ ) ungenutzt und danach können *alle* Rechner erneut zugreifen. Diesen Ansatz muß man ändern, um eine maximale Verzögerung zu erreichen. Entweder muß zunächst die andere Hälfte des Baums ihre Übertragung beenden oder ein anderer Teilungsmechanismus muß das Verhungern von Nachrichten in aufeinanderfolgenden Kollisionen vermeiden. Im ersten Fall bilden die Adressen Prioritäten, die die Reihenfolge in der Kollisionsauflösung bestimmen. Für den Einsatz in der Echtzeitkommunikation ist zu beachten, daß die Prioritäten pro Knoten vergeben werden. Erweiterungen für die Vergabe der Prioritäten und die geschickte Unterteilung des Baumes wären hier noch zu untersuchen. Abgesehen von der Bestimmung von  $t_{max}$  ist allein für die Vergabe der Prioritäten eine globale Planung notwendig.

Der CAN-Feldbus<sup>4</sup> verwendet eine im Resultat ähnliche, in der Durchführung aber unterschiedliche Technik (binary countdown [Tan96]). Mit einem 1-persistenten Zugriff wird die Priorität der folgenden Nachricht übertragen. Da nie zwei gleiche Prioritäten gleichzeitig auf dem Netz sein dürfen, wird die niedrigere elektrisch überlagert ( $\tau \approx 0$ ) und der entsprechende CAN-Controller stellt die Übertragung ein. Es wird immer die Nachricht höherer Priorität versendet, ohne die Verzögerung einer baumbasierten Auflösung.

---

<sup>4</sup>Controller Area Network [Bos91, ISO92]

### Zeitbasierter Zugriff

Die sogenannten Window-Protokolle nutzen für die Kollisionsauflösung eine Ordnung der Nachrichten nach den Zeitpunkten der Erzeugung [Gal78, KS83]. Das Protokoll merkt sich den Zeitpunkt  $t_0$ , so daß alle vorher erzeugten Nachrichten erfolgreich auf dem Netz versandt wurden. Im einfachsten Fall ist die Zeitachse in Intervalle eingeteilt und  $t_0$  wird durch das Protokoll auf den Anfang des letzten freien Intervalls weitergesetzt. Dadurch wird ein Overhead in der Kommunikation erzeugt. Im Falle einer Kollision wird deren Auflösung abgewartet. Dazu wird ein Zeitfenster bekannter Größe gewählt, das die Zeiten der kollidierten Nachrichten umfaßt. Dann kann ein ähnlicher Teilungsmechanismus wie bei der adressbasierten Auflösung greifen. Das Fenster wird geteilt und die Nachrichten der älteren Hälfte erhalten das Zugriffsrecht. Nach einer erfolgreichen Übertragung erhalten zunächst die Rechner der anderen Fensterhälfte das Zugriffsrecht, so daß alle kollidierten Nachrichten in einer festen Reihenfolge übertragen werden.

Mit Hilfe einer virtuellen Zeit (virtual time) wird ein ähnliches Verfahren realisiert, das allerdings der Kollisionsvermeidung dient [Jef85, MK85, Kur84]. Neben der physikalischen Uhr ist in jedem Computer noch eine *virtuelle Uhr* vorhanden. Diese wird angehalten, wenn Daten auf dem Netz übertragen werden, und läuft ansonsten mit einer höheren Frequenz als die physikalische Uhr, wenn sie nachgeht. Hiermit wird im Prinzip das oben erwähnte  $t_0$  definiert. Eine Nachricht  $M$  wird immer nur dann übertragen, wenn ihre Bereit- oder Ankunftszeit<sup>5</sup>  $T_A(M) = t_0$  ist. Wenn Kollisionen auftreten, verhält sich das Protokoll wie nicht-persistentes CSMA. Zu einem Zeitpunkt  $t_0$  sind also alle Nachrichten, die vorher erzeugt worden sind, bereits versendet oder für einen späteren Zeitpunkt erneut angesetzt. In [ZR87] ist dieses Konzept auf andere Parameter der Planung (Spielraum, Frist, u.a.) erweitert worden.

Die maximale Verzögerung ist bei beiden Ansätzen aus der Zahl der beteiligten Rechner zu ermitteln und eine genauere Bestimmung der Verzögerung ist nur mit der Verzögerungsfunktion zu erreichen, die sich aus der Kenntnis der Netzlast ergibt.

#### 2.1.1.2 Kontrollierende Zugriffsverfahren

Beim kontrollierten bzw. kollisionsvermeidenden Zugriff werden die Zeitachse und die Rechte für den Netzzugriff in Intervalle (Slots) eingeteilt und an die Rechner vergeben. Zu einem Zeitpunkt  $t$  hat genau ein Rechner das Senderecht und kann so kollisionsfrei Daten übertragen.<sup>6</sup> Die Vergabe des Senderechts unterscheidet sich in den folgenden drei Verfahren.

---

<sup>5</sup>Ankunftszeit (arrival time) Sendeschnittstelle des Kommunikationssystems.

<sup>6</sup>Grundsätzlich muß darüber hinaus das so erhaltene Senderecht auf die im jeweiligen Rechner aktiven Applikationen verteilt werden.

### TDMA

Feste Zeitpunkte für den Beginn der Slots werden beim Zeitscheibenverfahren oder TDMA<sup>7</sup> explizit vergeben. Mit Hilfe einer globalen Zeitbasis kann ein eindeutiges Zugriffsrecht ermittelt werden. Im statischen Verfahren des *reinen* TDMA sind eine feste Anzahl von Slots (z.B. gleich Anzahl der Knoten) konstanter Größe in einem periodisch wiederholten Rahmen (frame) zusammengefaßt und vor der Laufzeit an die einzelnen Rechner vergeben, z.B. [Tan96, KSY84, KG94]. Zu diesen festgelegten Zeiten ist  $E(VF) = 0$ . Beim Zugriff zu beliebigen Zeitpunkten ergibt sich ein ähnliches Bild wie bei den Token-Verfahren, da auf den nächsten sendeberechtigten Zeitpunkt für den jeweiligen Rechner gewartet werden muß. Diese Wartezeiten sind unabhängig von der momentanen Netzlast.

Der größte Vorteil des TDMA-Verfahrens ist die gute Vorhersagbarkeit. Mit einer strikten Einhaltung der zeitlichen Spezifikation in den einzelnen Knoten ist immer ein konfliktfreier Zugriff gesichert. Als Nachteile des statischen Verfahrens sind die folgenden Punkte zu nennen:

- Die Anzahl der angeschlossenen Knoten ist zwingenderweise konstant, da alle Zugriffsrechte vor der Laufzeit vergeben werden.
- Variierender Kommunikationsbedarf in den einzelnen Knoten kann nicht berücksichtigt werden. Geplante Zugriffsrechte, die aufgrund einer Worst-Case-Planung ungenutzt bleiben, können nicht durch andere Knoten genutzt werden.
- Sendebereite Nachrichten werden (vorhersagbar) verzögert bis der Rechner zum entsprechenden Zeitpunkt das Zugriffsrecht erhält.
- Die Zugriffsmuster müssen sehr genau spezifiziert werden, um eine hohe Planungsauslastung zu erhalten. Der Planungsaufwand kann dadurch sehr stark ansteigen bzw. die minimale Granularität der Planung schränkt die Spezifikationsmöglichkeiten ein; siehe 5.7.1.

### Dynamische TDMA-Reservierung

Bei einer dynamischen Reservierung der Zugriffszeiten werden TDMA-Zeitintervalle zur Laufzeit an einzelne Rechner vergeben. Beim slot-switched TDMA werden die Slots eines Rahmens für einzelne Rechner reserviert; z.B. [FH76]. Typischerweise werden zu Beginn eines Rahmen die freien Slots in einer Planungsphase bis auf Widerruf vergeben. In [ML81] wird an zentraler Stelle die Reservierung nach einem festen Zeitraum an den aktuellen Bedarf angepaßt. Es sind immer alle Rechner an einer kompletten Neuplanung beteiligt. Die Vorteile des reinen TDMA bleiben erhalten. Weitgehend bleiben auch die Nachteile des reinen TDMA erhalten, aber mit Hilfe der dynamischen Reservierung berücksichtigt man

---

<sup>7</sup>engl.: *time division multiple access*

variable Leistungsanforderungen und verteilt die Bandbreite individuell auf die Rechner.

### Token-Verfahren

Das Token-Verfahren erteilt auf physikalischen oder logischen Ringen eine Sendeberechtigung in Form eines zirkulierenden Frei-Zeichens [KSY84, MZ95]; im Token Ring und Token Bus ist es standardisiert [IEEc, IEEb] und wird in vielen Arbeiten behandelt oder vorausgesetzt (s. 2.1.2). Genau ein Frei-Zeichen (Token, Sendeberechtigung) wird explizit im System weitergegeben und nur derjenige Rechner der es besitzt sendet auf dem Medium. Die Reihenfolge der Weitergabe ist durch den Aufbau des Rings oder die Adressen im Bus bestimmt; der Bus formt einen logischen Ring.

Durch eine Beschränkung der Zeit, die ein Rechner das Token behalten darf (token hold time, THT) und der festgelegten Reihenfolge, läßt sich eine obere Schranke für die Zeit zwischen aufeinanderfolgender Sendeberechtigungen (token rotation time, TRT) angeben. Bei hoher Auslastung und Ausnutzung der zur Verfügung stehenden THT konvergiert diese Verfahren gegen TDMA, ohne daß die tatsächlichen Sendezeiten bekannt sind. Für die  $VF$  eines zu beliebigen Zeitpunkt  $t$  gestellten Sendeauftrags der Länge Null – die Zeit für die Übertragung der Daten bleibt hier außer Betracht – gilt  $t_{max} = TRT - THT$  und  $VF(0) = \frac{THT}{TRT}$ . Der Verlauf dazwischen ist im Grenz-/TDMA-Fall linear und hängt sonst von der momentanen Last ab. Eine untere Grenze für die TRT ist die Verzögerung beim Weiterreichen des Tokens zwischen Rechnern, die nichts zu senden haben. Im Gegensatz zu TDMA gilt hier immer  $E(VF) \neq 0$ . Der Erwartungswert kann durch einen verteilten Planungsmechanismus, der zusätzlich zu der Netzwerktechnik aktiv sein muß, verringert werden.

Die durch TRT begrenzte Zugriffszeit bietet einen einfachen Mechanismus für die Kommunikation. Aber es sind auch einige Nachteile zu nennen:

- Alle Stationen tragen, auch wenn sie nicht senden, zur TRT und vor allem zur maximalen TRT bzw.  $t_{max}$  bei. Eine neu gestartete Applikation verändert durch ihre Kommunikation die mittlere TRT und damit auch das zeitliche Verhalten aller anderen kommunizierenden Aufgaben! Die Einhaltung der Spezifikation kann zwar über  $t_{max}$  vorhergesagt werden, aber eine zeitliche Vorhersage innerhalb der Spezifikation ist nur über den Erwartungswert  $E(VF)$  möglich, der mit Hilfe von Messungen zur Laufzeit verfeinert werden kann bzw. aktualisiert werden muß.
- Die Granularität einer verteilten Planung wird unnötig vergrößert, weil mittlere und maximale TRT weit auseinander liegen können (Varianz von  $E(VF)$ ). Die mit Token Ring eingesetzten Planungsalgorithmen wie Rate Monotonic haben Einschränkungen in der zeitlichen Spezifikationen, die oft

unerwünscht sind [LL73, SM89].<sup>8</sup>

- Die Unsicherheit  $t_{max}$  für den Zeitpunkt des Sendens/Empfangens einer Nachricht beeinträchtigt die Vorhersagbarkeit der Echtzeitverarbeitung. Die notwendigen DMA- und Interrupt-Aktivitäten können ungewollte Interaktionen mit gerade laufenden Applikationen hervorrufen und müssen eingeplant werden.

Zwei unterschiedlich wichtige Probleme werden dabei aufgeworfen: Erstens ist die Kommunikation eine ereignisgesteuerte Aktivität, so daß die Ununterbrechbarkeit einer Task nicht mehr gewährleistet ist. Da die Task aber nicht von einer weiteren Task unterbrochen, sondern praktisch nur von den Interrupts und DMA-Zugriffen „verzögert“ wird, kann bzw. muß die Spezifikation in der Planung entsprechend angepaßt werden. Die daraus resultierende Verringerung der Planungsauslastung ist das zweite Problem. Jede Task des Knotens kann im schlechtesten Fall von allen Nachrichten unterbrochen und „verzögert“ werden. Die Genauigkeit der zeitlichen Vorhersage sinkt und nur dedizierte Hardware vermeidet mit Hilfe der Trennung von Applikations- und Kommunikationsprozessor diese Interaktionen.

- Um eine genauere Abstimmung zwischen den Applikationen über die Beschränkung der Varianz

$$\sigma^2 = \int_0^{\infty} (x - E(VF))^2 \cdot \frac{d}{dx} VF(x) dx$$

zu erreichen, ist eine weitergehende Absprache zwischen *allen* angeschlossenen Rechnern notwendig.

### 2.1.1.3 Bewertung

TDMA und das Timed Token Protokoll sind konstruktionsbedingt in erster Linie für eine periodische Datenübertragung geeignet, können aber auch für aperiodische Übertragungen eingesetzt werden; die Protokolle unterstützen allerdings keine Zugriffskontrolle auf Basis einzelner Nachrichten, so daß deren Einbindung schwieriger ist. Virtual Time, Window-Protokolle und die adressbasierte Auflösung bieten Absprachen für einzelne Zugriffe und unterstützen somit besonders die aperiodische (verbindungslose) Übertragung von Nachrichten, für die keine harten Zeitbedingungen gelten – die in der Einprozessorplanung in vielen Fällen optimale Strategie *minimum-laxity-first* kann mit einer Variante von Virtual Time angenähert werden [MZ95].

Generell gelten die Zugriffsrechte der Protokolle für einzelne Rechner und *innerhalb* eines Rechners ist der Zugriff auf die Applikationen zu verteilen. Diese

---

<sup>8</sup>Für die Anwendung der Rate Monotonic Analyse muß die Deadline gleich der Periode sein.

Problematik wird in 2.3 weiter untersucht. Typischerweise können dafür ähnliche Mechanismen wie für die Planung des eigentlichen Netzzugriffs verwendet werden.

Man kann die angeführten MAC-Protokolle und ihren Einsatz auf dem Gebiet der Echtzeitkommunikation in zwei Klassen teilen, deren Eignung für die Echtzeitkommunikation im folgenden noch einmal zusammengefaßt wird:

- Der *Versand nach Raten* legt keine Zugriffszeiten fest, sondern beschreibt konzeptionell einen prioritätsbasierten Zugriff: Jeder Rechner greift in Raten auf das Netz zu (Datenmenge, Periode<sup>9</sup> bzw. maximaler Frequenz), so daß die spezifizierten Bedingungen der Echtzeitkommunikation eingehalten werden. Die Spezifikationen, auf ihre gegenseitige Verträglichkeit getestet (Planung), bestimmen zusammen mit dem zugrundeliegenden MAC-Protokoll die Prioritäten und damit die Raten.
- Ein *Versand nach Planung* dagegen vereinbart feste Zugriffszeiten für die Teilnehmer der Netzkommunikation und ist damit ein zeitgesteuertes Verfahren. Die Überprüfung und Reservierung von Echtzeitbedingungen erfolgen über diese Zeiten.

Jede Klasse eignet sich jeweils für einen bestimmten Bereich von Applikationen, aber alle vier in der Einleitung vorgestellten Ziele werden von keinem einzelnen Verfahren erreicht. In den nächsten Kapiteln werde ich das Zugriffsprotokoll derart entwerfen, daß, ausgehend von einem zeitgesteuerten Zugriff, schrittweise eine Kombination beider Klassen ermöglicht wird.

### Ratenbasierter Versand

Der Einsatz eines ratenbasierten Versands in der Echtzeitkommunikation setzt die Kenntnis einer Verzögerungsfunktion  $VF$  voraus. Das MAC-Protokoll und eine gegebene Netzlast bestimmen die Verzögerungsfunktion, die verwendet wird, um die Kommunikation zu planen. Mit dem Erwartungswert  $E(VF)$  und der maximalen Verzögerung  $t_{max}$  können Durchsätze bzw. Bandbreiten für die Datenübertragung überprüft werden: Können gestellte Bedingungen eingehalten werden? Wie tragen sie zur Netzlast bei?

Die absprechenden Zugriffsverfahren (stochastisch, adreßbasiert und zeitbasiert) erlauben mit der Kenntnis der Netzlast, eine Verzögerungsfunktion anzugeben. Eine weitere globale Absprache über die zu erwartenden Netzlast ist also neben dem jeweils lokalen MAC-Protokoll notwendig. Die stochastische Absprache kann aber trotzdem keine beschränkte Zugriffszeit garantieren ( $t_{max} = \infty$ ) und ist daher unter harten Zeitbedingungen nicht einsetzbar.

Das Token-Verfahren ist ein typischer Vertreter der ratenbasierten Methode. Es bietet als kontrollierendes Zugriffsverfahren ohne Kenntnis der Netzlast eine

---

<sup>9</sup>Es werden zunächst nur periodische Sendeansforderungen berücksichtigt; aperiodische Aktivitäten müssen nachträglich darin eingepaßt werden [SSL89, SM89].

minimale Bandbreite pro Knoten an, und damit eine Verzögerungsfunktion und ein bekanntes  $t_{max}$  für den HRT-Einsatz. Ungenutzte reservierte Zugriffsrechte sind wegen der flexiblen Weitergabe des Tokens anderen Rechnern zugänglich, lassen sich aber nicht verplanen – variierende Anforderungen in den Rechnern können nicht für die Planbarkeit des Systems genutzt werden und die Planungsauslastung sinkt. Die resultierende Differenz zwischen der mittleren und maximalen Verzögerung ist nur durch eine zusätzliche Absprache über die zu erwartende Netzlast zu verringern. Alle Vor- und Nachteile des Token-Verfahrens kann man auf den ratenbasierten Versand übertragen.

Die Zugriffsrechte können innerhalb eines Rechners ebenfalls über ein ratenbasiertes Verfahren verteilt werden; z.B. eine Prioritätswarteschlange für ausgehende Datenpakete [ZK91].

### Planbasierter Versand

TDMA und *dynamisches TDMA* sind Beispiele für den planbasierten Versand. Zu den festgelegten Zeiten (nicht unbedingt statisch geplant) ist die kollisionsfreie Nutzung des Mediums garantiert und die beiden Verfahren eignen sich dadurch sehr gut für HRT-Kommunikation. Sie sind die einzigen MAC-Protokolle, die für einen Zeitpunkt  $t$  in der Zukunft die sendeberechtigte Station angeben können. Die Beschränkung auf die Zeitintervalle ist natürlich auch der größte Nachteil. Ein Rechner, der sein zugestandenes Intervall nicht ausnutzen kann, ist nicht in der Lage seine Sendeberechtigung explizit weiterzugeben. Nicht-Echtzeit und SRT-Kommunikation werden dadurch schlecht unterstützt, weil sie sich in der Regel nicht durch feste (periodische) Zeitintervalle beschreiben lassen. Eine genaue BM-Spezifikation ist in komplexen Applikationen sehr umfangreich und kann in der Praxis die dynamische Planung erheblich erschweren; die ratenbasierte Planung ist davon weniger betroffen, weil sie nicht mit expliziten Zeiten für die Ausführung umgehen muß und Planbarkeitstests einfacher ausfallen [LL73].

In allen vorgestellten TDMA-Arbeiten wird eine konstante Anzahl von Slots einer festen Periode angenommen, die die möglichen Zugriffsmuster erheblich einschränken. Das System ist nur zur Bearbeitung einer bekannten Aufgabenstellung einsetzbar, auf die die Periode vor dem Start abgestimmt wurde. Die enge Bindung der Kommunikation an die Applikation schränkt damit auch die Planbarkeit des gesamten Systems ein.

Die Zugriffsrechte können innerhalb eines Rechners sowohl über ein ratenbasiertes Verfahren, das die Vorteile des TDMA verbirgt, als auch über ein planbasiertes Verfahren verteilt werden. TDMA-Intervalle lassen sich auch direkt einzelnen Aufgaben zuweisen.

### 2.1.2 Token Ring

Im Token Ring Netzwerk wird das auf Seite 43 vorgestellte Token-Verfahren für den Netzzugriff benutzt [IEEc]. Die Implementierung in den Netzwerkcontrollern

(NIC)<sup>10</sup> unterstützt darüber hinaus im Fehlerfall Maßnahmen, um eine ordnungsgemäß rotierendes Frei-Zeichen sicherzustellen. Der physikalische Aufbau weicht von einem Multiple Access Bus ab. Die Rechner sind in einem physikalischen Ring angeordnet, so daß die Daten (und das Token) in einer vorgegebenen Reihenfolge *durch* die Rechner rotieren; jedes Bit wird in jeder MAC-Schnittstelle kurz zwischengespeichert, bevor es weitergeschickt wird. Der Einsatz von Prioritäten bei der Weitergabe des Tokens kann genutzt werden; es stehen aber nur acht zur Verfügung und das kann zu knapp für die Unterscheidung verschiedener Wichtigkeiten sein [SRL90].

Der Token Ring wird in der Echtzeitverarbeitung oft eingesetzt (siehe z.B. [SM89, Ple92, Tin94]). Der mittlere Durchsatz ist bekannt und die zu erwartenden Verzögerungen lassen sich abschätzen, wenn auch eine „Unschärfe“  $t_{max}$  in der Verzögerung die Planbarkeit der Echtzeitkommunikation einschränkt. Die einfache dynamische Vergabe des Frei-Zeichens an die Rechner ist ein weiteres Vorteil, der eine einfache Anpassung an NRT- und SRT-Kommunikation erlaubt.

FDDI<sup>11</sup> arbeitet mit einem ähnlichen Token-Mechanismus [Ros89, SS93], der auf dem Timed-Token-Protokoll [Gro82] basiert. Dieses Protokoll ist ebenfalls Gegenstand vieler Arbeiten, siehe z.B. [CAZ92]. FDDI-II erlaubt zusätzlich eine Reservierung von Bandbreite in synchronen Nachrichten einer festen Frequenz ( $125\mu s$ ); notwendig für den Transport von digitalisierter Sprache (ISDN).

### 2.1.3 ATM

ATM (Asynchronous Transfer Mode) bietet für die Echtzeitkommunikation einige Unterstützung an [LB92]. Die Technik und die Leistungsspezifikationen sind besonders für den Transport von Multimedia-Daten geeignet.<sup>12</sup> Die Planung verbindungsorientierter Kommunikation wird vom Netzwerk unterstützt und die Datenübertragung ist zeitsensitiv (SRT).

Die Integration der Kommunikation in das System ist ein offenes Problem, da DMA-Zugriffe und Interrupts von der Treibersoftware ein System erheblich stören können. Die meist hohen Datenraten von bis zu  $155 \text{ MBit/s}$ <sup>13</sup> erfordern Anstrengungen auf dem Gebiet der Hochgeschwindigkeitsnetze (high-speed networking) und verhindern (momentan) einen Einsatz in der hier betrachteten Umgebung (Robotik). Konzeptuell bietet ATM die Funktionalität des globalen Netzmanagements und ist ein vielversprechender Ansatz für (zukünftige) SRT-Systeme.

---

<sup>10</sup>engl.: *network interface controller*

<sup>11</sup>*Fiber Distributed Data Interface*

<sup>12</sup>ATM ist eine Grundlage für Breitband-ISDN (integrated services digital network) der internationalen Telekommunikationsgesellschaften.

<sup>13</sup>25-155 MBit/s ins Endgerät und sogar 622 MBit/s innerhalb des Netzes sind möglich.

## 2.2 Netzwerkplanung

Die bisher betrachtete Datenübertragung bezog sich immer auf ein einziges Segment physikalisch verbundener Rechner. Bei Netzen größerer Ausdehnung sind mehrere, teilweise heterogene, Segmente über *Brücken* (bzw. *Router*) verbunden [Tan96]. Der Datenfluß in diesen verbindenden Knoten muß natürlich den zeitlichen Anforderungen der Kommunikation genügen und die Planung betrifft alle beteiligten Segmente und Brücken. Die vorgestellten Arbeiten sind damit teilweise Erweiterungen der Diplomarbeit, aber auch für den ratenbasierten Versand (Token Ring) und die Kommunikation in den verbindenden Rechnern werden in den Arbeiten Mechanismen untersucht.

Auch in der Implementierung weit fortgeschritten ist die Tenet-Protokollsuite [BF<sup>+</sup>94]. Der Entwurf eines kompletten Protokolls für die Kommunikation in WANs<sup>14</sup> ermöglicht die Vorhersage des Kommunikationsverhaltens in heterogenen Netzwerkumgebungen. Als Basis dienen hier FDDI und andere zum Teil experimentelle Hochgeschwindigkeitsnetze. Eine zeitlich beschränkte Reservierung (Beginn und Ende der Nutzung zu Zeitpunkten in der Zukunft), die in [BF<sup>+</sup>94] nicht vorkommt, ist erst für die nächste Überarbeitung vorgesehen. Interessant ist hier die Planung der ratenbasierten Kommunikation in den verbindenden Rechnern. Die Integration in eine Echtzeitverarbeitung und die besonderen Erfordernisse in HRT-Systemen sind aber nicht Gegenstand der Untersuchungen.

Ähnlich sind die Arbeiten über Punkt-zu-Punkt-Verbindungen in verteilten Echtzeitsystemen [B<sup>+</sup>93, KSF94] und über Routing-Algorithmen [KS90]. Verschiedene Warteschlangenalgorithmen und -analysen werden benutzt, um die Pufferung der Pakete in den verbindenden Rechnern und einen ratenbasierten Zugriff auf ein Segment zu realisieren. In [KSF94] wird z.B. eine Kombination aus konstanten und dynamischen Prioritäten benutzt, um die Verzögerung einzelner Pakete in den verbindenden Rechnern unterhalb einer Schranke zu halten. Mit dem Aufbau einer Kommunikationsverbindung werden konstante Prioritäten benutzt, um die maximale Verzögerung im Netzzugriff zu errechnen. Diese Verzögerung wird zur Laufzeit in einer Variante des EDD-Algorithmus<sup>15</sup> für die Zuweisung von variablen Prioritäten verwendet.

Allgemein kann der Netzzugriff ratenbasiert modelliert werden, wenn von der Verzögerungsfunktion wenigstens die maximale Verzögerung  $t_{max}$  bekannt ist: spätestens alle  $x \mu s$  kann eine Nachricht verschickt werden [ZK91]. Im Token Ring können neben einer bekannten maximalen Verzögerung Prioritäten die Weitergabe des Frei-Zeichens kontrollieren und damit die Verzögerungsfunktion ändern. Damit ist etwa eine EDD-Strategie durchzusetzen [SM89, Ple92].

Diese Ansätze können auch in der knoteninternen Planung für Token Ring und Window-Protokolle genutzt werden, um die Pufferung und die CPU-Planung

---

<sup>14</sup>engl.: *wide area network*

<sup>15</sup>*earliest due date* ununterbrechbarer Aktivitäten [LL73]

durchzuführen (z.B. HARTS, s.u.). Im nächsten Unterkapitel werden Arbeiten vorgestellt, die sich mit der Integration in das Echtzeitsystem auseinandersetzen.

## 2.3 Integration

Das Ziel der in diesem Unterkapitel vorgestellten Arbeiten liegt in der Integration der Kommunikation in das Echtzeitsystem. Aufbauend auf ein echtzeitfähiges Netzwerk, das nicht Gegenstand der Arbeiten ist, wird der Einfluß der Kommunikation auf die Echtzeitverarbeitung bzw. die Kommunikation als Teil der Planung untersucht.

**HARTS** ist ein dynamisches, verteiltes Echtzeitsystem, daß an der Universität Michigan, USA, entwickelt wird [Shi91]. Einzelne Knoten des Systems enthalten einen Multiprozessor (Applikations- und Kommunikationsprozessor sind über gemeinsamen Speicher verbunden) und sind über Punkt-zu-Punkt-Verbindungen in einem hexagonalen Netz mit jeweils sechs Nachbarn verbunden.

[MIS96b, MIS96a] erläutern das Kommunikationssystem und gehen von einer beschränkten Übertragungszeit aus – damit sind auch Netze wie Token Ring einsetzbar. Die notwendige Nutzung der CPU und Speicher für die Verarbeitung der Kommunikation fließen in die Planung mit ein. Die Planung der Verzögerung wurde bereits im letzten Abschnitt erwähnt [KSF94]: die maximale Verzögerung  $t_{max}$  wird über konstante Prioritäten bei der Planung einer Verbindung ermittelt. Dabei wird die Datenübertragung durch eine Variable  $\mathcal{L}_{xmit}(s)$  in der Analyse der Wartezeiten berücksichtigt, die die Übertragungsdauer einer Nachricht der Länge  $s$  beschreibt. Das ist ein Schwachpunkt der Arbeit, denn für Punkt-zu-Punkt-Verbindungen sind geringe Werte  $\mathcal{L}_{xmit}(s)$  zu erwarten, aber beim Token Ring zum Beispiel ist ohne eine weitere Kontrolle der gesamten Netzlast nur ein unverhältnismäßig großer Wert garantiert. Damit sinkt die Planungsauslastung unter Umständen erheblich!

Die EDD-Planung von Prozessen, die jeweils einer Kommunikationsverbindung zugewiesen sind, sichert die Einhaltung von  $t_{max}$ . Die Leistungsanforderungen an die Datenübertragung werden durch die CPU-Planung dieser Prozesse garantiert. Überlast wird durch die Kontrolle der beanspruchten Leistung in den einzelnen Prozessen vermieden; der Versand erfolgt nach logischer Zeit, die das Nachrichtenaufkommen gemäß der Spezifikation „glättet“.

Nachrichten beliebiger Länge werden in Pakete fester Länge unterteilt. Nachrichten sind in der Übertragung unterbrechbar, aber einzelne Pakete müssen ununterbrechbar übertragen werden; damit werden die Verzögerungen durch die EDD-Planung verkleinert.

Für den Datenempfang und -versand ist der Aufwand (DMA, Protokollverarbeitung) nur als Rate, als ständiges „Hintergrundrauschen“ spezifizierbar, das die CPU nach Bedarf belastet. Die in der Arbeit angegebene Analyse geht auf diverse zeitliche Kosten für die Protokollverarbeitung ein (Kosten für Kontext-Switch,

Verwaltungsaufwand für Paketwarteschlangen), aber die CPU-Last setzt sich ausschließlich aus dem Kommunikationsaufwand zusammen. Folgerichtig muß von einem eigenen Kommunikationsprozessor ausgegangen werden, und das ist eine große Einschränkung für den Entwurf, die den Einsatz in Embedded Systems erschwert.

**Spring** ist ein verteiltes Echtzeitsystem, das HRT-Aufgaben zeitgesteuert ausführt und durch eine dynamische Planung zur Laufzeit eine Garantie für die Einhaltung der zeitlichen Spezifikation vergibt [SR91]. SpringNet ist der Entwurf einer Kommunikationsarchitektur für das Spring Projekt [SNR91, DNS94]. Jeder Knoten ist ein Multiprozessor, der über VME-Bus an einen verteilten gemeinsamen Speicher (DSM<sup>16</sup>) angeschlossen ist. Jede Änderung in diesen physikalisch getrennten Speichern wird über einen optischen Ring automatisch in alle angeschlossenen Rechner weitergeleitet und somit ein logischer gemeinsamer Speicher erzeugt. Die Änderungen (Nachrichten) werden lokal in einer FIFO-Liste gespeichert und sukzessive übertragen. Das Protokoll auf dem Ring sichert jedem Knoten eine konstante Bandbreite und maximale Verzögerung zu.

An der synchronen Kommunikation sind immer zwei Tasks, ununterbrechbare Programmteile (s. 3.2.3), beteiligt: Die erste ändert das DSM und stoppt und die zweite beginnt mit der Rückmeldung über die Kommunikation. Aus der offline-Analyse einer Applikation ergeben sich die Zeitbedingungen für die Tasks und damit für einzelne Nachrichten (Kommunikationsmuster), weil Änderungen am DSM durch eine Task sofort als Nachrichten in die FIFO-Liste aufgenommen werden – enge Kopplung von Netz und CPU. Der Planungsalgorithmus entscheidet über eine Garantie der zeitlichen Bedingungen einzelner Nachrichten nur mit Hilfe der Kenntnis der lokalen FIFO-Liste. Die aktuelle Netzlast wird nicht in den Algorithmus einbezogen und die Bandbreite, mit der sich die Liste abbaut, wird als konstant angenommen. Dies kann zu niedrigen Planungsauslastungen führen.

Eine explizite Verbindung zwischen den Rechnern wird nicht aufgebaut. Das Kommunikationsmuster der Applikationen wird bei der Planung zur Laufzeit synchronisiert und bestimmt die weitere lokale CPU-Planung. Die Kommunikation über das DSM ist konzeptionell verbindungslos und die Applikationen werden durch die zeitgesteuerte Ausführung verbunden. Die Arbeiten gehen nicht auf eine periodische Datenübertragung ein; Spring kennt periodische Tasks und eine dahingehende Erweiterung der Kommunikation sollte möglich sein. Die gestellten Anforderungen an die Hardware (Netztechnik, DSM, Multiprozessor) sind hoch und nicht ohne weiteres in der Praxis einsetzbar.

**Maruti** ist ebenfalls ein dynamisches, zeitgesteuertes System, das für HRT-Aufgaben explizit Reservierungen in einer kalenderbasierten Datenstruktur vornimmt [SdA94]. Zu jedem Betriebsmittel existiert ein solcher Kalender, der die Verwaltung der Zeitskala übernimmt. Die kleinsten planbaren Einheiten sind

---

<sup>16</sup> *distributed shared memory* ist physikalisch verteilt, aber logisch gemeinsam

die *elemental units* (EU). Alle Ein- bzw. Ausgaben erfolgen (logisch) vor bzw. nach der Ausführung einer EU. Diese konzeptuelle Entscheidung erschwert allerdings die praktische Planung. Die Modellierung eines komplexen, zeitlich fein aufgelösten Kommunikationsverhaltens ist zwar möglich, aber die Ausführungszeiten der EUs werden verkleinert und damit die Anzahl „künstlich“ erhöht, wenn eine hohe Planungsauslastung erreicht werden soll. Die Laufzeiten und Mißerfolgchancen der Echtzeitplanung vergrößern sich aufgrund der Komplexität der Planungsproblems (siehe 5.6.1 und 5.7.1).

Die Kommunikation ist unidirektional mit festen Nachrichtentypen. Der Versand ist immer asynchron, der Empfang kann sowohl synchron (blockierend) als auch asynchron durchgeführt werden. Es ist keine bestimmte Hardware für die Realisierung vorausgesetzt und die Datenübertragung nicht Teil des Konzepts. Für ein vorgeschlagenes Multiple-Access-Netz wird die kalenderbasierte Reservierung abgelehnt, weil es die Zugriffe auf CPU und Netz zu eng koppelt. Statt dessen wird ein zeitliches Multiplexing unabhängig von der CPU in weiterführenden Arbeiten bevorzugt [MSST93, SSMT94]; ähnlich dem Token-Verfahren werden dabei für das Multiplexing keine expliziten Zeiten festgelegt, sondern ratenbasiert zugegriffen. Die Vorteile einer zeitlich genauen Spezifikation gehen verloren. Eng gekoppelte, verteilte Applikationen sind nicht planbar.

Bei **MARS** handelt es sich um ein statisch geplantes, zeitgesteuertes Echtzeitsystem, das für die Prozeßkontrolle und Automatisierungstechnik entwickelt wurde. Alle Echtzeitaufgaben sind periodisch und ihre BM-Spezifikationen müssen vor dem Systemstart bekannt und maximal sein. Statisches TDMA wird auf einem Ethernet für die Kommunikation eingesetzt. Der Systemdesigner hat über die Größe und Anzahl von Slots und über die Vergabe an die einzelnen Rechner zu entscheiden. Das Zugriffsrecht für einen Slot erhält eine einzelne Echtzeitaufgabe. Die Planung der Echtzeitaufgaben geschieht offline und versucht die Reaktionszeit durch eine zeitlich nahe Ausführung an dem Slot klein zu halten. Die CPU-Belastung der Netzkommunikation berücksichtigt die Planung, indem die Ausführungszeit der Aufgaben entsprechend angepaßt wird (Sender und Empfänger der Nachrichten sind zu jeder Zeit bekannt).

Das Entwurfskriterium, die maximale Systemlast vorhersagbar zu verarbeiten, ist in MARS erfüllt, aber die statische Planung ist nicht immer möglich oder sinnvoll (siehe 1.1.2.2). Die Vorteile des Systems, die gute Vorhersagbarkeit und die planbar schnelle Reaktivität, können aber in einen neuen Entwurf eingehen.

In komplexen dynamischen Systemen kann die Planung der Kommunikation als Teil des Gesamtsystems in einem allgemeineren Ansatz betrachtet werden. Die Planung wird zu einer Leistungsverhandlung (QoS-Negotiation) über QoS-Beschreibungen. Mit den steigenden Hardwareleistungen wachsen auch die Anforderungen und die komplexer werdende Absprache über Nachfrage und Angebot an Leistung wird in der Literatur meist mit dem Begriff der Leistungsverhandlung belegt. Obwohl die verwendeten BM auch weiterhin geplant werden müssen, liegt

das Augenmerk auf einer ausführlichen Beschreibung der gewünschten Qualität des Service und auf umfangreichen (Neu-)Verhandlungen über die zu erbringende Leistung.

In letzten Jahren erscheinen zunehmend Arbeiten auf diesem Gebiet. In [NS95, CCH94] werden zum Beispiel Rahmen für solche Absprachen geschaffen, ohne auf die Funktionalität der einzelnen Systemkomponenten einzugehen. Diese Ansätze können für Multimedia-Systeme und -Anwendungen sinnvoll eingesetzt werden, sind aber für die hier betrachtete Problematik zu umfangreich, und die in der Diplomarbeit erarbeiteten Lösungen können später als Teil einer solchen komplexen Verhandlung eingesetzt werden.<sup>17</sup> Der Übergang zwischen den Begriffen der Planung und Leistungsverhandlung ist fließend und ich werde in 3.4 genauer darauf eingehen.

---

<sup>17</sup>Mit anderen Worten ist eine solche QoS-Negotiation ein Problem aller OSI-Schichten, während diese Diplomarbeit in den Schichten zwei bis vier angesiedelt ist, siehe Kapitel 3.4.

# Kapitel 3

## Zielsystem

Dieses Kapitel beschreibt die Grundlagen für den Entwurf des Kommunikationssystems. Dazu gehören die Voraussetzungen über den Aufbau des verteilten Systems, sowie Kriterien für die Bewertung des Systems, an denen sich die späteren Entwurfsentscheidungen messen müssen. Anhand eines Vergleiches mit dem OSI-Modell werden die Leistungen des entworfenen Kommunikationssystems charakterisiert.

### 3.1 Hardware-Architektur

Betrachtet werden soll ein verteiltes Echtzeitsystem, bestehend aus folgenden Komponenten:

- Der Hauptbestandteil des Systems sind **autonome Rechner**, die unter eigenständiger Kontrolle darin arbeiten. Ein anderer Ausdruck hierfür ist Knoten oder Rechnerknoten. Es handelt sich dabei um Einprozessorsysteme, die als Basis eine minimale Unterstützung bieten. Die Rechenzeiten für die Protokollverarbeitung der Kommunikation und für die Applikation belegen denselben Prozessor und müssen beide geplant werden, um eine ungewollte Interaktion zu vermeiden.
- Ein **Kommunikationsnetzwerk** verbindet die verschiedenen Knoten des Systems. Ein Multiple Access Bus wird zugrundegelegt; diese einfache Technik ist weit verbreitet und man kann über verschiedene Varianten des Netzzugriffs unterschiedliche Leistungscharakteristika erzeugen. Einige wurden in Abschnitt 2.1 vorgestellt und deren Einsatzfähigkeit für die Echtzeitverarbeitung bewertet. Der Bus bietet keine Unterstützung für Echtzeitkommunikation.
- **Dezentrale I/O** bildet einen weiteren Teil des Systems. Es handelt sich hier um Ein-/Ausgabegeräte, die über das Netzwerk zu erreichen sind, aber

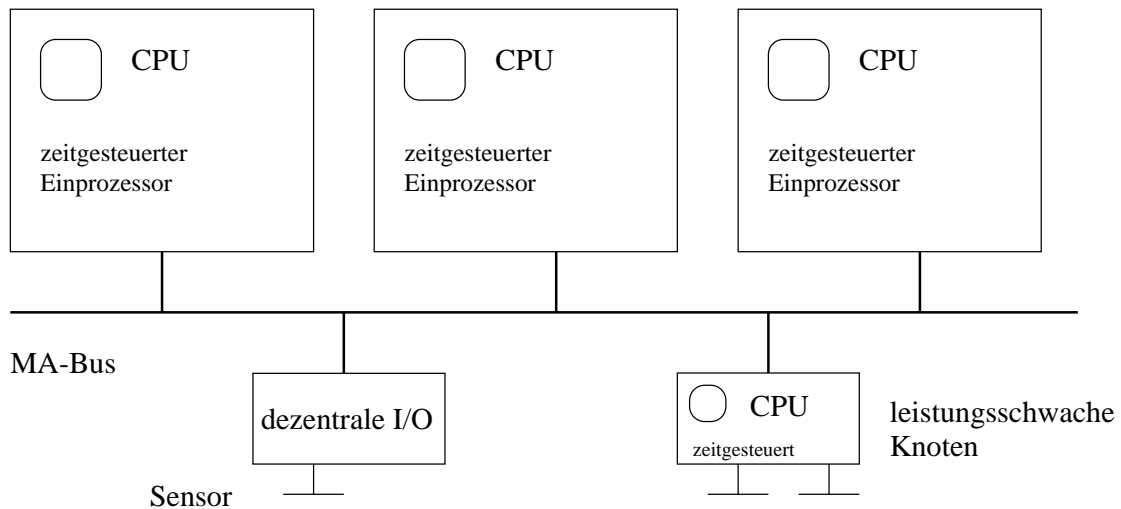


Abbildung 3.1: Aufbau des Zielsystems

keine oder nur begrenzte eigene ‘Intelligenz’ besitzen. Als Netzwerkteilnehmer sind sie nicht selber in der Lage ihre Kommunikation zu planen.

Zum Beispiel ist die periodisch übertragene Sensorinformation eines dezentralen Sensors von einem anderen Netzteilnehmer anzumelden und somit für die Verwaltung und Planung des Systems sichtbar zu machen; oder im einfachsten Fall reagiert der Sensor nach einer expliziten Anfrage mit der Rücksendung der Antwort innerhalb einer festen Zeitspanne.

- Leistungsschwache Knoten sind vorhanden und bilden einen Engpaß. Sie sind nur für eine dedizierte Aufgabe gebaut und können an der dynamischen (Um-)Planung des Systems gar nicht (statische Planung des Teilsystems) oder nur ihre eigenen Aufgaben betreffend teilnehmen.

## 3.2 Software-Architektur

### 3.2.1 Dynamische Planung

In dem betrachteten System werden Zugriffsrechte auf Betriebsmittel (CPU) jeweils lokal in den Knoten zur Laufzeit dynamisch geplant und in einer Reservierung<sup>1</sup> festgehalten. Mit einer erfolgreichen Planung werden Garantien für den Zugriff vergeben; auf Zeitfehlerbehandlung und Interpretation der Reservierungszeiten (HRT- oder SRT-Planung) gehe ich aber nicht ein (vgl. Seite 34). Durch die hohe Anpassungsfähigkeit zur Laufzeit wird das Design von Applikationen und der Einsatz in der Praxis erleichtert. In zukünftigen komplexen, verteilten Sy-

<sup>1</sup>Die Reservierung ist die Datenbasis der Planung.

stemmen, wo Änderungen und Rekonfigurationen keine Ausnahmen sind, ist diese Flexibilität unabdingbar.

### 3.2.2 Zeitsteuerung

Ein zeitgesteuertes, dynamisches Echtzeitsystem mit einer globalen Zeitbasis (max. Abweichung  $\Delta GT$  vom Normal) wird betrachtet. In *zeitgesteuerten Systemen* werden Programmteile nach einer festen zeitlichen Abfolge, unabhängig von auftretenden Ereignissen, aufgerufen. In verteilten Systemen können die Aufgaben über eine globale Zeitbasis koordiniert und synchronisiert werden. Dadurch wird die Planung komplexer Abhängigkeiten unterstützt und insbesondere eine hohe Vorhersagbarkeit der Planung erreicht.

Die Reaktionszeit auf zufällig erzeugte Ereignisse ist durch die geplanten Zeiten bestimmt und bei geringer Belastung schlechter als in einem ereignisgesteuerten System, in dem die mittlere Reaktionszeit oft erheblich niedriger ist. Aber gerade in Ausnahme- und Hochlastsituationen sind die Verzögerungen einfacher vorhersagbar (s. [Kop91]). Die Nachteile der Zeitsteuerung werden im Kapitel 5.7 analysiert und Verbesserungen erarbeitet.

**Globale Zeit** Im verteilten System ist die Kenntnis des globalen Zustandes, oder zumindest Teilen davon, eine Voraussetzung für kooperatives Arbeiten. Eine Zeitbasis kann zur Abstimmung verteilter Aktivitäten genutzt werden. Mit Hilfe von Zeitstempeln ist zum Beispiel eine zeitliche Ordnung von Nachrichten gegeben. Wenn es nur auf die Reihenfolge und nicht auf die genauen Zeiten ankommt, können *logische Uhren* benutzt werden [Lam78]. In diesem Zusammenhang werden auch zeitlose Protokolle für die Echtzeitkommunikation eingesetzt, die ohne eine globale Zeitbasis auskommen (Token Ring).

Für Echtzeitaufgaben, die auf genaue Zeitpunkte oder -differenzen wie z.B. global vergleichbare Zeitstempel angewiesen sind, muß eine systemweit einheitliche Zeitbasis vorhanden sein. Eine *globale Zeit*, also auf eine einheitliche Zeitbasis synchronisierte Systemuhren, ist ein einfaches Mittel eine konsistente Sicht des globalen Systemzustands zu erreichen. In einem, nicht notwendigerweise zeitgesteuerten, System lassen sich die Vorteile davon nutzen, ohne daß neben dem Aufwand der Implementierung weitere Nachteile zu beachten sind. Für den hier angestrebten Einsatz zeitgesteuerter, verteilter Systeme ist die globale Zeit eine grundlegende Voraussetzung.

Eine synchronisierte Uhr ist für die zeitgesteuerte Kommunikation in jedem kommunizierenden Knoten erforderlich, selbst in den skizzierten leistungsschwachen Knoten, wenn sie eine eigene Aktivität besitzen (s. Abb. 3.1). Lediglich die dezentrale I/O ist nicht notwendigerweise betroffen, weil sie nur auf Anfragen reagiert bzw. streng periodisch Netzlast erzeugt. In beiden Fällen ist eine Planung 'von außen' möglich.

Das Thema der Uhrensynchronisation deckt ein weites Feld von unterschiedlichen Mechanismen ab [Cri89, KO87, RKS90, GS94]. Ich werde im Konzept von einer gegebenen globalen Zeit mit maximaler Abweichung  $\Delta GT$  vom Normal ausgehen; zwei Systemuhren weichen höchstens  $2\Delta GT$  voneinander ab.

### 3.2.3 Laufzeitmodell

Eine **Task** besteht aus einer Sequenz von CPU-Instruktionen, die eine logische Einheit bilden und eine bestimmte Aufgabe erfüllen. Über Verkettung und Verschachtelung entstehen Tasks mit umfangreicheren Aufgaben. Eine Unterbrechung einer Task  $T_1$  ist der *spontane* Entzug der CPU durch eine Task  $T_2$ ; der geplante Entzug der CPU bezeichne ich darüber hinaus als Aufteilung, *task splitting*, um den Unterschied zu verdeutlichen.

Ein **Thread** stellt einen logischen Kontrollfluß der CPU dar, in dem nacheinander eine Menge von Tasks ausgeführt werden. Die Prozessorzuweisung an eine Task geschieht über das Starten eines Threads. Mehrere aktive Threads laufen in einem gemeinsamen Adreßraum. In einem **Prozeß** sind ein oder mehrere Threads und ein Adreßraum vereinigt. Die vorliegende Arbeit geht von einem einzelnen Prozeß in jedem Rechner aus, so daß alle Threads des lokalen Systems in demselben Adreßraum laufen.

Echtzeitaufgaben, wie sie in der Einleitung beschrieben wurden, können durch Tasks dargestellt werden. Eine zeitliche Attributierung der Tasks macht sie zu planbaren Einheiten für die CPU-Nutzung. Die geplanten Tasks werden *ununterbrechbar* an einem Stück ausgeführt<sup>2</sup> bzw. die aufgesplittete Teile werden ununterbrechbar ausgeführt. In dem zeitgesteuerten System werden für die Tasks Zeitintervalle für die CPU-Nutzung reserviert (RT – reservation time). Zu Beginn des Intervalls wird einem ausführendem Thread die CPU erteilt und am Ende (zunächst) entzogen.

Von einer weiteren Laufzeitkontrolle gehe ich in der Konzeption nicht aus; nur Abschnitt 4.7 zeigt Ansatzpunkte für eine spätere Erweiterung auf.

## 3.3 Entwurfskriterien

Die folgenden Punkte beschreiben das Maß für die Bewertung des Systems. Der weitere Entwurf muß möglichst viele dieser allgemein gehaltenen Punkte unterstützen, die die Ziele aus Abschnitt 1.3.2 enthalten.

- Verteilte Aufgaben

In dem System arbeiten mehrere Echtzeitaufgaben in einer verteilten Applikation zusammen und mehrere Applikationen nebeneinander. Abhängig-

---

<sup>2</sup>engl.: *non-preemptive* ( $\leftrightarrow$  *preemptive*)

keit und Synchronisation erfordern die Einhaltung zeitlicher Bedingungen für die Kommunikation.

- Vorhersagbarkeit der Planung  
Der Umgang mit harten Zeitbedingungen und HRT-Aufgaben muß möglich sein. Nach der Planung sind die spezifizierten zeitlichen Bedingungen uneingeschränkt einzuhalten.
- Dynamische Planung  
Die Konfiguration des Systems soll zur Laufzeit modifizierbar sein. Die Planungen sind in die laufende Echtzeitverarbeitung zu integrieren und stellen eine besondere Anforderung an den Rechenaufwand dar. Aufwendige Algorithmen für die Suche nach der optimalen Lösung sind meist zu komplex und rechenintensiv, um „nebenher“ abgearbeitet zu werden. Anfragen müssen kurzfristig bearbeitet werden können, um die Reaktivität des Systems zu steigern.
- Knoten geringer Leistung  
Der Einsatz einzelner Rechner mit geringer Leistung muß im verteilten System möglich sein. Diese sollten keine Engpässe in der Gesamtleistung erzeugen.  
Diese Forderung ist für Einsatzbereiche wie Prozeßsteuerung, Robotik und allgemein eingebettete Systeme sinnvoll.
- Abhängigkeiten  
Der Umgang mit Abhängigkeiten in der lokalen und auch der verteilten Verarbeitung von Echtzeitaufgaben erleichtert die Koordinierung komplexer Aufgaben und die gemeinsame Reaktion auf externe Ereignisse.<sup>3</sup> Dieser Punkt ist hauptsächlich eine Verallgemeinerung des nächsten Punktes.
- Enge Kopplung  
Die minimal planbaren Abstände müssen klein sein, um abhängige Aufgaben  $A$  und  $B$  zeitlich nah beieinander ausführen zu können; die Granularität der Planung ist klein (engl. *tightness*). Dieser Punkt dient der Optimierung des Systems wie sie auf Seite 17 dargestellt ist bzw. fordert die Einhaltung des Zieles  $Z2$  der Einleitung.

Es existieren zwei Abstufungen der Anforderung. Erstens lassen sich in der Praxis die Antwortzeiten wesentlich verringern, wenn Abhängigkeiten zur Spezialisierung der Planung genutzt werden können: Eine Planung, die auch harten Zeitbedingungen genügen soll, muß von der maximalen Verzögerung in der Kommunikation ausgehen. Spezifizierte Abhängigkeiten vergrößern

---

<sup>3</sup>Die Aktivität  $A$  muß vor  $B$  ausgeführt werden, oder die Startzeiten von  $A_1$  und  $A_2$  dürfen nicht mehr als  $\Delta t$  auseinanderliegen, ...

das „Wissen“ des Planers. Sie helfen, mit der genaueren Kenntnis der Übertragungsmuster, die maximale Verzögerung  $t_{max}$  für diesen speziellen Fall einzuschränken.<sup>4</sup> Bei gleicher Auslastung der Betriebsmittel ist es möglich, mit einer engeren Kopplung zwischen einzelnen Aufgaben die Antwortzeit zu verkürzen.

Die zweite Anforderung an die enge Kopplung ist wesentlich strenger. Eine gleichzeitige Ausführung der kommunizierenden Aufgaben  $A$  und  $B$  auf unterschiedlichen Rechnern erfordert die gleichzeitige Verfügbarkeit des Kommunikationsmediums. Die Applikation läuft *isoliert* ab und braucht die BM-Zugriffe während ihrer Aktivität nicht mit anderen abzustimmen; vgl. ACID-Modell der Transaktionsverarbeitung, z.B. [BHG87].

- Unterschiedliche Leistungsanforderungen  
Neben der engen Kopplung ist in komplexen Systemen in zweiter Linie auch ein Umgang mit zeitlich weniger engen Leistungsanforderung notwendig, um heterogene Aufgaben zu unterstützen.

Wünschenswert ist darüber hinaus auch die Unterstützung weniger *strenger* Bedingungen, wie sie für SRT-Aufgaben gelten.

### 3.4 Kommunikation

Ein Kommunikationssystem stellt Leistungen zur Verfügung, es bietet einen Kommunikationsdienst an. Dieser Dienst ist über einen Dienstnutzungspunkt mit einer bekannten Schnittstelle zu erreichen (SAP, Service Access Point). Ein Dienstnutzer kommuniziert darüber mit Hilfe von SDUs (Service Data Unit) mit dem Dienst.<sup>5</sup> Zum Erbringen dieses Dienstes tauscht das Protokoll Nachrichten in Form von PDUs (Protocoll Data Unit) aus (s. Abb. 3.2).

Um die umfangreichen Leistungen eines Kommunikationssystems zu spezifizieren, trennt man die Leistungen in aufeinander aufbauende *Schichten*<sup>6</sup>, die jeweils nur ihre direkten Vorgänger bzw. Nachfolger kennen. Ein Dienst der Schicht  $n + 1$  nutzt dann Dienste der Schicht  $n$ , um seine Leistung zu erbringen ( $(n + 1)$ -PDU werden mit  $n$ -SDUs übertragen). Diese Leistung wird jeweils nach oben über einen SAP für die höhere Schicht zur Verfügung gestellt ( $(n + 1)$ -Dienst,  $n$ -SDU,  $n$ -PDU, usw.).

Für die Echtzeitkommunikation sind geplante Leistungen am SAP zu erbringen. Mit der Planung der Kommunikation werden über den SAP Leistungsverhandlungen *zwischen* den Schichten durchgeführt; die geforderten und die an-

---

<sup>4</sup>Das „Wissen“ des Planers über die möglichen BM-Zugriffe ist entscheidend für die Planbarkeit [BKM<sup>+</sup>91].

<sup>5</sup>Je nachdem, ob auf die Schnittstelle zum Dienst wert gelegt wird, spreche ich von SDUs oder allgemein von Paketen.

<sup>6</sup>Man spricht von Schichtenbildung.

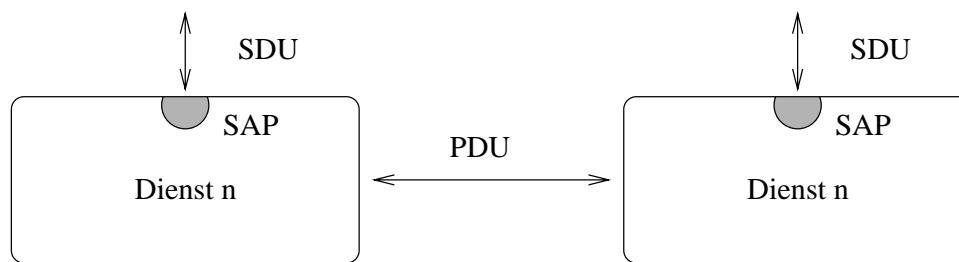


Abbildung 3.2: Dienstmodell

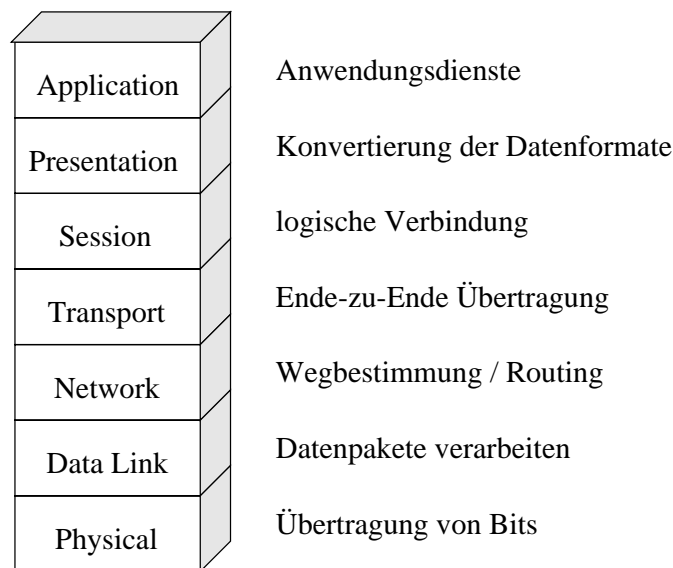


Abbildung 3.3: OSI-Schichtenmodell

gebotenen Leistungsparameter (QoS – Quality-of-Service) der SDUs müssen in der *Leistungsverhandlung* abgestimmt werden (QoS-Negotiation). Dabei geht es um die Abstimmung der konkreten Leistung in dem durch die Spezifikation aufgespannten Raum. In einer Schicht ist die *Leistungsplanung* der PDUs für das Erbringen des verhandelten Dienstes verantwortlich; dies schließt eine Verhandlung über  $(n - 1)$ -SDUs ein.

### 3.4.1 OSI-Referenzmodell

Die ISO (International Standards Organisation) hat ein Schichtenmodell standardisiert, um die Konstruktion von Kommunikationsmechanismen in offenen Systemen zu vereinfachen und zu vergleichen [DZ83, OSI84]. Das OSI-Schichtenmodell ist in Abbildung 3.3 skizziert und unterteilt die Leistung des Kommunikationssystems in sieben aufeinander aufbauende Schichten. Bei der Protokollverarbeitung der zu übertragenden Daten *senken* diese auf der Senderseite von oben durch die

Schichten nach unten bis zur eigentlichen Übertragung der einzelnen Bits auf dem Medium. Auf der Empfängerseite geschieht dies in umgekehrter Reihenfolge. PDUs werden physikalisch nur in der untersten Schicht direkt ausgetauscht, darüber sind jeweils  $(n - 1)$ -SDUs der tieferen Schicht für den Transport der  $n$ -PDUs verantwortlich.

### **Bitübertragungsschicht — Physical layer**

Für die Übertragung einzelner Bits auf dem Medium enthält die unterste Schicht nachrichtentechnische Geräte wie elektrische oder optische Sender und Empfänger. Die Übertragungsverfahren für unterschiedlichen Medien (Funk, Lichtwellenleiter, Kupferkabel) sind hier spezifiziert.

### **Sicherungsschicht — Data Link layer**

In der Sicherungsschicht werden die zu übertragene Bits in Nachrichten – auch Frames genannt – gruppiert (PDU), um sie auf dem reinen Bitstrom der ersten Schicht erkennbar zu machen. Dafür sind die Eingabedaten von der höheren Vermittlungsschicht (SDU) in diese Protokollnachrichten zu unterteilen. Die Übertragungsform kann verbindungslos oder -orientiert, unzuverlässig oder mit Neuübertragung im Fehlerfall erfolgen. Zur Sicherung der Daten werden Prüfsummen an die Nachrichten gehangen mit denen man die Integrität und den korrekten Empfang kontrolliert. In Multiple Access Netzen bildet die Netzzugriffskontrolle (MAC, medium access control) einen Teil dieser Schicht.

### **Vermittlungsschicht — Network layer**

Die Vermittlungs- oder Netzwerkschicht regelt den Versand der Nachrichten über mehrere getrennte Netzsegmente hinweg, die nur an bestimmten Punkten miteinander verbunden sind (Router, Bridge). Diese Schicht vermittelt die Nachrichten an ihre Empfänger, die nicht direkt über die erste Schicht erreichbar sind. Das IP-Protokoll der TCP/IP Familie ist ein Beispiel für diese Schicht.

### **Transportschicht — Transport layer**

Die Transportschicht ist für die korrekte Übertragung von Daten zum Empfängerthread verantwortlich. Beliebige Datenblöcke werden angenommen und evtl. unterteilt, versandt und auf der Empfängerseite wieder zusammengesetzt. Darüberhinaus wird hier erstmals nicht nur ein Rechner sondern die Netzwerkanbindung im Rechner an einzelne Prozesse bzw. Threads als Empfänger (und Sender) vergeben — ein Multiplexing des Netzzugriffs unter den Threads findet statt.

Die weiteren Schichten sind für diese Arbeit nicht interessant und werden zum Beispiel in [Hal95] ausführlich dargestellt. Die Planung der verteilten Applikation und umfangreiche Kommunikationsdienste (-planungen) sind aufbauend den höheren Schichten zuzuordnen.

### 3.4.2 Einbettung des Kommunikationssystems ins OSI-Modell

Die Dienste des zu entwerfenden Kommunikationssystems werden anhand einer Einbettung/Vergleich mit dem OSI-Modell erläutert.

Die *Medium Access Control* (MAC) als Teil der zweiten OSI-Schicht ist der grundlegendste Aufgabenbereich. Das globale Netzmanagement stellt im wesentlichen eine Absprache der einzelnen Rechner über die beantragten Zugriffsmuster bzw. -zeiten der MAC-Schicht dar. Die Planung der Anforderungen und die Vergabe garantierter Zugriffsrechte sichert die Einhaltung der Echtzeitbedingungen des Datenverkehrs (PDUs).

Mit Hilfe der aus der Planung erhaltenen Zugriffsrechte kontrolliert die MAC-Schicht als lokaler Teil des Kommunikationssystems (lokales Netzmanagement) innerhalb jedes teilnehmenden Rechners die korrekte Nutzung dieser Rechte. Damit werden die Planung durchgesetzt und die Echtzeitbedingungen eingehalten. Das lokale Management erweitert also die durch die eingesetzte Standardkommunikationshardware gegebene Sicherungsschicht, um die zeitliche Spezifikation zu garantieren und die Kommunikation vorhersagbar zu machen.

Die *Vermittlungsschicht* lasse ich zunächst außer Betracht und gehe von einem einzigen, physikalisch verbundenen Netzwerksegment aus.<sup>7</sup>

In der *Transportschicht* werden die Dienste des in dieser Arbeit entworfenen Kommunikationssystems bereitgestellt und die Schnittstelle zur Applikation definiert (Transport-SAP). Für den CPU-Bedarf des Zugriffsprotokolls werden Tasks und ein ausführender Thread (Kommunikationsthread, CT) benutzt und automatisch von dieser Schicht beim Verbindungsaufbau geplant. Die Leistungsplanung umfaßt eine Leistungsverhandlung mit der zweiten Schicht. Am T-SAP wird es aber keine Verhandlung mit höheren Schichten geben und ich gehe von einer entsprechend einfachen Schnittstelle aus. Mit den Tasks als Laufzeitrepräsentanten sind die Kanäle der Echtzeitkommunikation für eine Applikation sichtbar.

Mechanismen zur Fehlererkennung, die auf der zweiten und vierten Schicht zu finden sind, werden angeboten. Aufgrund der erkannten Fehler wird aber keine Korrektur vorgenommen; siehe 4.7.

---

<sup>7</sup>Brücken in der zweiten und Router in der dritten Schicht (Vermittlungsschicht) können durch Anwendungen in verbindenden Rechnern realisiert werden; vgl. 4.9.1.2.



# Kapitel 4

## Kommunikationssystem

In diesem Kapitel wird ein Konzept für die Kommunikation in verteilten Echtzeitsystemen und ein Reservierungsprotokoll für die verteilte Planung erarbeitet. Die lokalen Teile des Kommunikationssystems hängen eng mit dem Zugriffsprotokoll zusammen, das den Zugang der Rechner zum Netz regelt und in diesem und dem folgenden Kapitel vorgestellt wird.

Aus der Notwendigkeit einer Planung läßt sich ein grundsätzlicher Umgang mit den daraus erhaltenen Garantien beschreiben (Garantievertrag). Aus diesem Prinzip ist eine Strukturierung des Kommunikationssystems für verbindungsorientierte Datenübertragungen herzuleiten.

### 4.1 Garantievertrag

In einem Echtzeitsystem müssen Aktivitäten ihren Betriebsmittelbedarf anmelden und planen lassen. Einerseits legen sie dadurch ihren Bedarf fest und sichern sich durch die Reservierung die Möglichkeit einer (zeitlich) korrekten Ausführung. Andererseits dürfen sie sich nicht entgegen ihrer eigenen Spezifikation verhalten und andere als vereinbarte und geplante Betriebsmittel benutzen.

Bei der Planung der Echtzeitkommunikation wird ein *Vertrag* zwischen zwei Seiten geschlossen, beide müssen ihren Teil erfüllen! Auf der einen Seite, dem globalen Teil des Vertrages, sind die über eine verteilte Absprache (Leistungsverhandlung) ermittelten Zugriffsrechte auf das Kommunikationsmedium zur Verfügung zu stellen, zu garantieren. Die an der Kommunikation beteiligten Knoten, und insbesondere die Anwendungen, bilden den anderen, lokalen Teil des Vertrages. Es dürfen nur die vertraglichen Leistungen gefordert und keine weiteren Zugriffszeiten genutzt werden, d.h. hier ist das vereinbarte Datenvolumen einzuhalten (vgl. auch [Fer90]). Ich betrachte die vertragliche Vereinbarung zwischen der zweiten und vierten OSI-Schicht.

Die vertraglichen Leistungen der einen Seite bietet der anderen Seite die Grundlagen, die für die vorhersagbare Echtzeitverarbeitung notwendig sind.

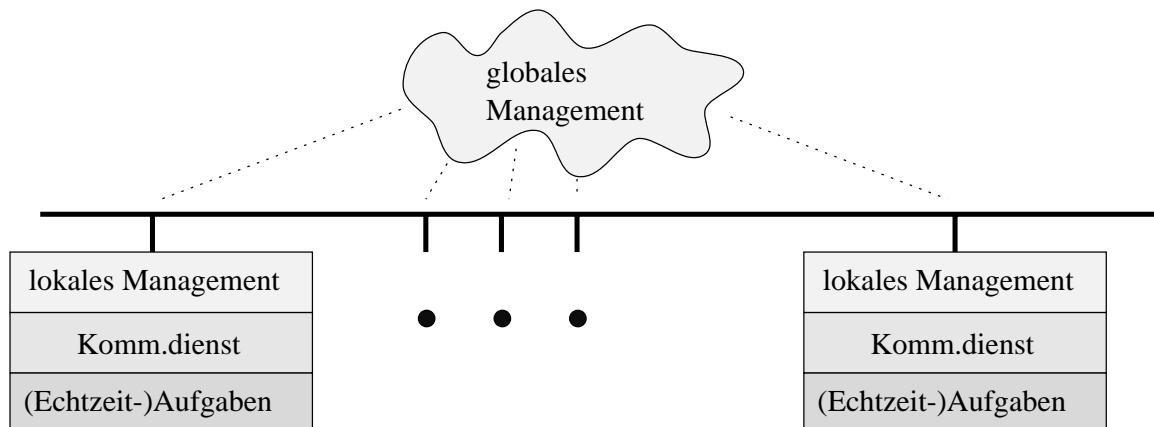


Abbildung 4.1: Aufgabenverteilung des Kommunikationssystems

### Garantierte Verfügbarkeit

Die Anwendung, als Nutzer eines verteilten Kommunikationsdienstes, erhält eine „globale Garantie“ für die Verfügbarkeit des Betriebsmittels Netz. Es ist garantiert möglich, die spezifizierten Daten zu übertragen und die Planung schließt Verstopfungen aus. Die Qualität bzw. Leistung des Dienstes wird mit *Quality-of-Service* bezeichnet.

### Kontrollierte Nutzung

Die Anwendung gibt an das Kommunikationssystem eine „lokale Garantie“ über die anwendungsseitige Einhaltung der Spezifikation. Es findet keine unkontrollierte Nutzung des Kommunikationsmediums entgegen der ursprünglichen Leistungsbeschreibung statt und Kollisionen werden dadurch vermieden. Dies könnte die globale Garantie anderer Kommunikationsteilnehmer stören.

## 4.2 Struktur

### 4.2.1 Globale und lokale Aspekte

Aus der Unterscheidung in globale und lokale Garantien ist eine konzeptuelle Struktur des gesamten Kommunikationssystems ableitbar (s. Abb. 4.1). In allen Knoten ist ein Dienst als Leistung des Kommunikationssystems sichtbar.

Die *globale Garantie* wird durch einen verteilten Mechanismus vergeben. Die Planung muß eine Absprache zwischen allen dazu notwendigen Knoten des verteilten Systems durchführen, um festzustellen, ob die geforderte Leistung neben den anderen Reservierungen kollisionsfrei erbracht werden kann. Auf das sogenannte *globale Netzmanagement* und die Leistungsplanung der zweiten Schicht gehe ich weiter in 4.3 ein.

Um die *lokale Garantie* einzuhalten, ist eine *Flußkontrolle* der übertragenen Daten notwendig, d.h. die bereitgestellten Daten dürfen nur gemäß ihrer ausgehandelten Spezifikation übertragen werden. Zwischen der Anwendung und dem eigentlichen Netzzugriff sichert diese Kontrolle die Einhaltung der vertraglich zugesicherten Leistungen. Eine Überlastung des Netzes wird durch die kontrollierte Nutzung jeder Kommunikationsverbindung verhindert. Darüberhinaus ist die *Integration* der Kommunikation in die Echtzeitverarbeitung des Systems eine wichtige Aufgabe. Der CPU-Aufwand auf der Sender- und der Empfängerseite ist zu planen, damit andere Echtzeitaufgaben nicht durch Interferenzen in der CPU-Nutzung gestört werden; die Verträge über Rechenzeitgarantien müssen respektiert werden. Auf das *lokale Netzmanagement* und die Leistungsverhandlung mit dem globalen Management gehe ich weiter in 4.4 ein.

## 4.2.2 Kommunikationsmodell

### 4.2.2.1 Kommunikationskanäle

In der Einleitung war von einer unidirektionalen, verbindungsorientierten Kommunikation über dynamisch reservierte Kanäle die Sprache. Es soll nun eine genauere Definition vorgestellt und diese gegen alternative Kommunikationsformen abgegrenzt werden.

Ein *Kanal* stellt für die Anwendung eine reservierte, verbindungsorientierte Kommunikationsleitung dar, dessen Qualität durch die Leistungs- oder QoS-Parameter beschrieben ist. Er besteht aus einer Kombination von reserviertem Netzwerkzugriff, einer „Verbindung“, und der notwendigen Rechenzeit der CPU für die Protokollverarbeitung und Flußkontrolle der Datenübertragung in den kommunizierenden Knoten (Ende-zu-Ende Kommunikation). Letztendlich ist jede geplante Kommunikation verbindungsorientiert, da im voraus eine kollisionsfreie Übertragung vom Sender zum Empfänger sichergestellt wird – für *jede* Datenübertragung ist in allen kommunizierenden Knoten der entsprechende Aufwand zu planen.<sup>1</sup> Die Einbeziehung der lokalen CPU-Aktivitäten in die Planung macht also einen verbindungsorientierten Ansatz notwendig.

Zunächst bleibt die Anzahl der Kommunikationsteilnehmer auf zwei pro Kanal beschränkt und ein Kanal ist eine Punkt-zu-Punkt Verbindung zwischen einem Sender und einem Empfänger. Diese Einschränkung ist für das zugrunde gelegte einsegmentige Multiple-Access-Netzwerk eigentlich zu restriktiv und wird später verallgemeinert, bietet aber als konzeptuelle Entscheidung eine minimale Ausgangsbasis für unterschiedliche Ausprägungen und Erweiterungen des Kommunikationssystems.<sup>2</sup> Aus dem gleichen Grund beschränke ich mich zunächst auf eine unidirektionale Datenübertragung (Simplexkommunikation); Sender und

---

<sup>1</sup>NRT-Kommunikation erfordert eine gesonderte Behandlung, s. 4.4.3.2.

<sup>2</sup>Der Einsatz anderer Netzwerktopologien wird erleichtert und die technik- bzw. produktspezifische Erweiterbarkeit gewahrt.

Empfänger sind eindeutig.

Die Kanäle bieten eine unzuverlässige Kommunikation in dem Sinne, daß im Fehlerfall keine automatisch wiederholte Übertragung stattfindet. Diese Problematik bleibt außer Betracht, da erstens moderne Netze nur noch geringe Fehleraten besitzen und zweitens unter Echtzeitbedingungen meist keine Zeit für eine erneute Übertragung bleibt. Hardware- oder Softwaremechanismen der Fehler-toleranz können später ergänzend eingesetzt werden, wie in Abschnitt 4.7 kurz erläutert wird.

Die Verwaltung der Kanäle ist der Dienst des Kommunikationssystems. Als Dienst der Transportschicht ermöglicht ein Kanal die Echtzeitkommunikation zwischen RT-Aufgaben.

#### 4.2.2.2 Aufgaben der Kanalverwaltung

In jedem Rechnerknoten ist eine Instanz des Kommunikationsdienstes aktiv. Sie nimmt für den Kanalaufbau Planungsanfragen aus den (Echtzeit-)Aufgaben entgegen und versucht die Leistungsanforderungen der QoS-Parameter über eine globale Absprache und Reservierung zu planen und zu garantieren. Alle Teilnehmer des Kanals planen den BM-Bedarf der Kommunikation ein und integrieren sie damit in die Echtzeitverarbeitung des Systems; man vermeidet unvorhergesehene Interferenzen mit anderen Aktivitäten. Die Planung der Kanäle wird ohne zeitliche Bedingungen als NRT-Aufgabe ablaufen. Es bleibt einer Erweiterung überlassen, Zeitbedingungen an die Korrektheit der Planung zu binden.

Der Kanal kann ab einem Zeitpunkt in der Zukunft bereitgestellt werden. Ein etablierter Kanal kann beliebig lange (bis auf Widerruf seitens eines Teilnehmers) oder über eine feste Zeitspanne bestehen bleiben. Es ist eine „forward reservation“ über einen festen Zeitraum möglich, die in der Literatur selten berücksichtigt wird. Die Kanäle müssen nicht sofort bis auf unbegrenzte Zeit zur Verfügung stehen!

Rekonfigurationen eines bestehenden Kanals entsprechen einer Änderung der Leistungsparameter und kommen einer erneuten Planung und anschließender Zerstörung des alter Kanals gleich. Umplanungen anderer bestehender Kanäle aufgrund der Rekonfiguration oder Neuplanung eines Kanals betrachte ich nicht, weil durch Abhängigkeiten in den Anwendungen selbst kleine Kanaländerungen zu umfangreichen Planungen in den Knoten führen würden.<sup>3</sup>

Beim Abbau eines Kanals, nach Ablauf des vereinbarten Reservierungszeitraums oder auf Anforderung eines Kommunikationsteilnehmers, müssen im globalen und lokalen Netzmanagement die entsprechenden Betriebsmittel wieder freigegeben werden.

---

<sup>3</sup>Eine QoS-Architektur über alle Schichten des OSI-Modells ist hierfür notwendig, weil mit neuen, umfangreichen Leistungsverhandlungen zwischen großen Teilen des Systems zu rechnen ist.

Für die Nutzung des Kanals wird der entsprechende CPU-Aufwand eingeplant und mit dem Zugriffsprotokoll der MAC-Schicht die Daten übertragen. Für den Netzzugriff werden Zeitintervalle (TDMA-Slots) explizit eingeplant; siehe nächster Abschnitt. Die für den Versand vorgesehenen und bereitgestellten Daten können in diesen Zeiten übertragen werden.

#### 4.2.2.3 Datenübertragung in den Kanälen

Das Kommunikationssystem soll nach den Kriterien aus 3.3 auch auf kleinen, leistungsschwachen Systemen laufen und eine enge Kopplung der verteilten Anwendung erlauben (Tightness). Aus diesem Grund ist es für den Aufbau vor allem des lokalen Kommunikationssystems wichtig, vorab eine Entscheidung über das verwendete Kommunikationsmedium und das Zugriffsprotokoll zu treffen. Für die in dieser Arbeit angestrebte Ende-zu-Ende-Kommunikation mit einer einfachen applikationsseitigen Schnittstelle ist eine solche Entscheidung von zentraler Bedeutung, um eine enge Abstimmung der Leistung der zweiten und vierten OSI-Schicht zu erreichen.

Im Gegensatz dazu wird in vielen Arbeiten der (portable) Entwurf einer minimalen Schnittstelle, zum Beispiel über die Angabe einer maximalen Zugriffszeit  $t_{max}$ , zwischen diesen beiden Schichten gewählt. Dieser Ansatz erleichtert den Top-Down-Entwurf und die Portierbarkeit, aber die Möglichkeit einer engen Kopplung zwischen verteilten, abhängigen Aufgaben wird nicht ausgenutzt. Es besteht die Gefahr, daß die Planungsauslastung niedrig und die Tightness bzw. die Granularität der Planung grob bleibt.

Das Zielsystem ist ein über Multiple-Access-Netzwerk verbundenes zeitgesteuertes, verteiltes System, das aus Einprozessorrechnern besteht. Der Ansatz einer dynamischen Reservierung von TDMA-Zeitintervallen mit alleiniger Sendeberechtigung für einzelne Rechner bzw. Applikationen wird als Basis für das Zugriffsprotokoll verwendet. Die Argumente für diese Entscheidung werden im folgenden zusammengefaßt (vgl. Kapitel 2.1):

- Absprechende Zugriffsverfahren benötigen für den Einsatz als Echtzeitprotokolle ein globales Wissen über die zu erwartenden Kommunikationsmuster und machen eine entsprechende verteilte Planung notwendig. Die Nachteile des ratenbasierten Versands, wie ungenaue zeitliche Vorhersage innerhalb der Spezifikation und lose Kopplung der kommunizierenden Aufgaben, können nicht behoben werden.

Die ungenaue zeitliche Vorhersagbarkeit der Kommunikation wirft das Problem der Interaktion mit laufenden RT-Aufgaben auf. Die Integration in das System ist nicht ohne einen Verlust an Vorhersagbarkeit und eine Reduzierung der Planungsauslastung möglich (vgl. Seite 44 und 84).

- Der alleinige Einsatz von Token-Verfahren zieht die gleichen Nachteile mit sich, aber es ist keine weitere Planung für die Echtzeitfähigkeit des Netzes

notwendig;  $t_{max}$  ist aus der Anzahl der Knoten bekannt. Verbesserungen benötigen eine globale Absprache und können nicht alle Nachteile beheben. Eine solche Absprache kann z.B. die garantierten Parameter der  $VF$  näher an die mittleren heranbringen und damit eine genauere Planung und Auslastung der Betriebsmittel ermöglichen. Trotzdem bleibt eine Ungenauigkeit in der zeitlichen Abstimmung, die insbesondere die Synchronisation verteilter Aufgaben an eine Datenübertragung erschwert. Diese Granularität der Planung beschränkt die garantierbare minimale Ausführungszeit verteilter Aufgaben, wie z.B. das Erzeugen eines Datums, Versand und Verarbeitung in einem anderen Rechner, nach unten.

Die grundlegende Echtzeitfähigkeit der Token-Verfahren ist beim Einsatz leistungsschwacher Knoten ein Vorteil. Aber die verfeinerte Granularität kann man in der Planung nur mit Hilfe einer ständigen Kontrolle bzw. Bestimmung der Netzlast ( $E(VF)$ ) nutzen. Leistungsschwache Knoten können den Kontrollaufwand nicht bewältigen und auch für „normale“ Knoten kann es eine erhebliche Belastung bedeuten, weil jede einzelne Kommunikationsverbindung jedes  $E(VF)$  verändert. Darüber hinaus ist der Erwartungswert nicht ohne weiteres für die HRT-Planung einsetzbar; er ist ein Ansatz für die SRT-Planung.

Ratenbasierte Kommunikation erfüllt die Entwurfskriterien nur unzureichend. Eine zeitlich enge Koppelung verteilter Aufgaben, die durch die Planung garantiert werden kann, und damit die Genauigkeit einer Vorhersage im (zeitgesteuerten) System wird nur unzureichend unterstützt.

- TDMA unterstützt die vorhersagbare Echtzeitkommunikation und enge Koppelung verteilter Aufgaben. Aber der Einsatz in dynamischen Umgebungen erfordert Veränderungen.
- Die Reservierung von Zeitintervallen für den Netzzugriff kann erfolgreich die bisher genannten Nachteile aufheben. Die zeitliche Vorhersagbarkeit ist durch die in der Planung festgelegten Zugriffszeiten für den HRT-Einsatz geeignet. Alle Knoten, auch leistungsschwache, müssen über eine synchronisierte, globale Zeit verfügen und ihre Kommunikation darauf abstimmen. Dies erfordert gegenüber dem ratenbasierten Ansatz einen Mehraufwand, der allerdings konstant und unabhängig von der Netz- und Systemlast ist.
- Die dynamische Planung kann an variierende und neu hinzukommende Aufgaben Garantien vergeben. Die Planung reserviert explizit Zeitintervalle für einzelne Aufgaben, die uneingeschränkt zur Verfügung stehen, aber auch nicht an andere Aufgaben abgegeben werden können. Es bleiben daher unter Umständen reservierte Zeiten ungenutzt und die Planungsauslastung

sinkt, wenn der spezifizierte BM-Bedarf über dem tatsächlich benötigten liegt.<sup>4</sup> Das in Kapitel 5.7 vorgestellte Planungsverfahren geht auf diese Problematik ein und kann eine Lösung vorstellen.

- Die Integration in das laufende Echtzeitsystem ist durch eine feste zeitliche Planung der CPU für die Netzzugriffe sichergestellt. Diese Reservierung stellt eine Laufzeitrepräsentation dar, die für weitergehende Planungen davon abhängiger Aufgaben nutzbar ist; zum Beispiel kann eine Datenverarbeitung zeitlich sehr nah an den Empfang dieser Daten gelegt werden, um die kritische Verarbeitungszeit kurz halten. Mit dieser Integration ist eine enge Kopplung verteilter Aufgaben erreichbar.
- Das TDMA-Verfahren ist (weitgehend) auf jedem MA-Bus realisierbar und bietet mit Hilfe der isolierten Zugriffe innerhalb unterschiedlicher Intervalle eine ideale Voraussetzung zur Kombination mit weiteren Zugriffsprotokollen. Das in der praktischen Anwendung installierte MA-Netz bietet in der Regel ein eigenes Zugriffsprotokoll an, das einerseits mit der zeitlichen Steuerung umgangen und andererseits als zusätzliches Protokoll in einer Kanalüberlagerung genutzt werden kann (s. 4.6.1). Andere Zugriffsprotokolle können innerhalb einzelner Intervalle leicht auf dem MA-Bus in Software realisiert werden.

Eine auf TDMA beruhende Erweiterung erfüllt die Entwurfskriterien aus Abschnitt 3.3 und die in der Einleitung aufgestellten Ziele Z1 bis Z4 können erreicht werden.

#### 4.2.2.4 Kommunikationskanäle in der Applikation

Für eine Applikation stellt sich ein Kanal als ein Puffer dar, aus dem die bereitgestellten Daten zeitgemäß gesendet bzw. in dem sie empfangen werden (vgl. 1.3.1). Die Kanäle stellen eine nicht-blockierende, asynchrone Kommunikation bereit. Die Applikation wird beim Senden (Bereitstellen der Daten) nicht blockiert und erhält nach der Datenübertragung keine synchronisierende Antwort.

Die Integration der Kanäle in die Echtzeitverarbeitung des übrigen Systems ist durch die Reservierung der CPU-Anforderungen für die Protokollverarbeitung sichergestellt. Man kann den Kanal mit dieser Repräsentation in die Applikation integrieren. Mit der Bekanntgabe der TDMA-Zeiten für die Datenübertragung sind neben der abstrakten Sicht des Kanals als Puffer auch die zeitlichen Muster der Zugriffe für die Applikation sichtbar. Die Kommunikation kann mit dieser Information in die Applikationsplanung einbezogen werden. Abhängigkeiten zu der CPU-Reservierung eines Kanals kann man nutzen, um Echtzeitaufgaben vor

---

<sup>4</sup>Der Fall eines Zeitfehlers – der tatsächliche liegt über dem spezifizierten Bedarf – wird hier nicht betrachtet.

und *nach* der Übertragung einzuplanen und die Verzögerungen somit klein zu halten (Tightness).

## 4.3 Globales Netzmanagement

Das globale Netzmanagement hat die Aufgabe, alle Aspekte der Betriebsmittelverwaltung, die über einzelne Rechnergrenzen hinausgehen, zu behandeln und zu koordinieren. Im Aufgabenbereich liegt die Planung der zweiten OSI-Schicht für den Auf- und Abbau und für Rekonfigurationen von Verbindungen. Allgemein ist es für die verteilte Datenverwaltung zuständig, von der Reservierung der TDMA-Slots bis zum Monitoring aller Netzwerkaktivitäten.

### 4.3.1 Verteilte Datenhaltung

Die Verteilung der Datenhaltung ist das zentrale Problem, das aus der Verteilung der Aktivitäten auf mehrere Knoten folgt. Veränderungen dieser Daten dürfen nicht ohne weiteres lokal durchgeführt werden, da ein gemeinsam genutzter Speicher fehlt. Eine Synchronisierung dieser Datenzugriffe ist notwendig, um eine inkonsistente Überschneidung von Veränderungen zu vermeiden. Eine global einheitliche Sicht auf die Daten muß erzeugt werden!

Alle kommunizierende Rechner müssen zumindest die für sie reservierten TDMA-Slots lokal speichern, um die Flußkontrolle darauf abzustimmen. Sie sehen damit nur einen Teil aller globalen Daten (aller TDMA-Reservierungen). Man kann die Datenhaltung über den Umfang der lokal gespeicherten Daten unterscheiden. Hat ein Knoten eine *totale Sicht* auf alle Daten, so können Entscheidungen und Veränderungen darauf lokal vorgenommen und danach für alle anderen sichtbar gemacht werden; dies muß eine Synchronisierung mit einschließen. Bei einer *teilweisen Sicht* aller Daten muß ein Zugriff darauf zusammen mit mehreren Knoten erfolgen. Im allgemeinen Fall sind alle zur Erzeugung einer totalen Sicht notwendigen Knoten am Zugriff beteiligt.

### 4.3.2 Verteilte dynamische Planung

Die verteilte Datenhaltung beschränke ich auf das Problem der (TDMA-)Reservierungen der Kommunikationskanäle und untersuche im folgenden verschiedene Möglichkeiten der Datenhaltung. Der Einfluß der Kanalverwaltung auf die Systemkomponenten kann das zeitliche Verhalten des verteilten Echtzeitsystem beeinflussen. Ungeplante Interferenzen sind unbedingt zu vermeiden und einzuplanende Interaktionen sind zu minimieren, um den Planungsaufwand klein zu halten (low-interference).

#### 4.3.2.1 Beteiligte Komponenten

Die beteiligten Komponenten (Knoten, CPU, Netzwerkcontroller, -medium) einer Planung umfassen, neben der aktiven Beteiligung der Quelle und Senke der Kommunikation, auch solche passiv beteiligten Knoten, die an der Planung teilnehmen (müssen), später aber nicht mehr zu diesem Kommunikationskanal gehören. Die übrigen Komponenten sind unbeteiligt.

In allen beteiligten Komponenten ruft eine Planung eine Last hervor, die für die passiven Teilnehmer unerwartet ist. Mechanismen für einen geregelten, geplanten Umgang mit dieser Last müssen Beeinträchtigungen anderer Echtzeitaufgaben verhindern. Eine Minimierung der Anzahl der passiven Komponenten erspart diesen Aufwand auf Kosten einer Zentralisierung der Planung.

#### 4.3.2.2 Zentrale Reservierung

Bei einer zentralen Reservierung steht ein Server (Reservierungsinstanz, Kommunikationsserver (CS)) mit einer totalen Sicht auf die globale Datenbasis für sämtliche TDMA-Reservierungen bereit. In den Knoten sind nur die eigenen Zugriffszeiten gespeichert und der Aufbau eines neuen Kanals bedarf einer Anfrage an den Server, der die notwendigen Zugriffsrechte und Garantien vergibt.<sup>5</sup>

Seine Clients sind aktive Komponenten, die als Kommunikationsteilnehmer an dem aufzubauenden Kanal teilnehmen werden. Da neben dem Server und den Clients keine weiteren Knoten beteiligt sind, kann die Kommunikationsplanung unabhängig vom restlichen System erfolgen.

Ein Kanalaufbau gliedert sich in zwei Teile, die unter Umständen wiederholt ausgeführt werden müssen. In der zentralen Reservierungsstruktur (vgl. 5.7) wird festgestellt, ob und wie die gewünschten Leistungsparameter im Netz erfüllt werden können (globale Garantie). Dabei ergeben sich mögliche Zugriffszeiten für die TDMA-Reservierung. Mit diesen Ergebnissen muß in den Clients die lokale CPU-Planung (lokale Garantie) der Kommunikation erfolgen. Bei Mißlingen können an den Server abweichende Vorschläge zurückgegeben werden, mit denen eine erneute Runde der Reservierung beginnt.

Durch diese Struktur des globalen Netzmanagements können auch kleine, wenig leistungsfähige Knoten in ein umfangreiches verteiltes System eingebunden werden! Da sie nur mit ihren eigenen, absehbar vielen Kanälen belastet werden und der Server der einzige passiv beteiligte Knoten ist, können diese Rechner uneingeschränkt auf die zu bearbeitenden Aufgaben zugeschnitten sein, ohne daß sie in Änderungen außerhalb ihres Aufgabenbereichs einbezogen werden.

Daneben existieren aber auch die typischen Nachteile einer Zentralisierung.

---

<sup>5</sup>Nach Fertigstellung des Konzeptes stellte sich heraus, daß [CS97] einen ähnlichen Ansatz vorgeschlagen haben. Allerdings gehen sie nicht auf den Einfluß der Kommunikation auf die übrige Echtzeitverarbeitung der jeweiligen Knoten ein (Integration).

Das System ist fehleranfällig, weil der Ausfall eines einzigen Knotens<sup>6</sup> die weitere Planung verhindert. Leistungsprobleme bei der Skalierung der verwalteten Knoten sind zu erwarten.

#### 4.3.2.3 Dezentrale Reservierung

Bei der dezentralen Reservierung ist kein einzelner Rechner mit einer totalen Sicht auf die Reservierungsdaten für die Planung verantwortlich. Mit einem Verzicht auf die zentrale Stelle der Datenhaltung und Reservierung müssen verteilte Mechanismen die Konsistenz und Korrektheit der Reservierungsdaten aller beteiligten Knoten sichern. Die typischen Nachteile einer Zentralisierung werden für den Preis einer teureren Konsistenzkontrolle umgangen. Es sind immer mehr Komponenten an der Planung beteiligt als am eigentlichen Kommunikationskanal und die passiv beteiligten Komponenten müssen einen Teil des Planungsaufwandes tragen.

Wenn die passiv beteiligten Komponenten nicht wie im zentralen Ansatz im voraus bekannt sind wird der Planungsaufwand in den Knoten eine unerwartete Hintergrundlast erzeugen, die von der Dynamik anderer Systemteile abhängt. Wie viele passive Komponenten einzubeziehen sind und wie groß der jeweilige lokale Aufwand ist, hängt von der Sicht auf die globalen Daten ab – je geringer die Datenhaltung desto umfangreicher ist der erforderliche Kommunikationsaufwand, um eine Planung durchzuführen.

Man kann die Reservierungsprotokolle, die auch für die globale Datenverwaltung (Konsistenz und Korrektheit) zuständig sind, nach dem Umfang der jeweils lokal gehaltenen Daten charakterisieren.

- Alle Rechner haben eine totale Sicht auf alle globalen Daten und können prinzipiell selbständig Planungen durchführen, die sie den anderen Knoten mitteilen. Zur Konsistenzerhaltung muß aber auf diesen globalen Datenbestand kontrolliert, serialisiert zugegriffen werden. Eine Veränderung ist durch eine Transaktion [BHG87] systemweit auf die Planung in jeweils einem Knoten zu beschränken, da die Sperrung des kompletten Datenbestands notwendig ist.

Die gesamte Spezifikation des laufenden Systems ist lokal vorhanden und man ist auf keine Information anderer Knoten angewiesen. Zur Verfügung stehender Freiraum wird erkannt und kann verplant werden. Die erfolgten Änderungen sind aufzuzeichnen und mit dem Commit am Transaktionsende den anderen beteiligten Knoten zur Aktualisierung der globalen Daten vorzulegen.

Dieser Ansatz entspricht der Vorgehensweise in der zentralen Reservierung; jeder Knoten ist sein eigener Server.

---

<sup>6</sup>Single Point of Failure

- Token-Planung

Eine eingeschränkte Sicht auf die globalen Daten erfordert einen dezentralisierten Ansatz. Jeder Knoten muß explizit gefragt werden, ob eine bestimmte Reservierung möglich ist und ein neuer Kanal wird in allen Knoten gegen die bisher aufgebauten Kanäle getestet. Ein Reservierungstoken mit den entsprechenden Parametern stellt eine solche Anfrage dar und wird an alle Knoten gesandt.

In MA-Netzen kann eine Reservierungsanfrage per Broadcast an alle Knoten geschickt werden; alle Knoten sind beteiligt. Diese müssen jeweils ihre Zustimmung geben. Mit einer Ablehnung können Änderungsvorschläge mit zurückgegeben werden, deren Auswertung zu einem erneuten Planungsversuch führt.

Ein anderer Ansatz vermeidet den Broadcast und simuliert ein Token-Verfahren, um die Leistungsparameter an die beteiligten Knoten zu verteilen. Ein einzelnes Reservierungstoken, das unabhängig vom MAC-Protokoll übertragen wird und in einem (logischen) Ring kreist, gelangt mit der Rotation durch alle Knoten.

In kleinen Systemen mit einfachen Kommunikationsmustern ist hiermit eine ausreichend flexible Reservierung realisierbar. Ähnlich der Ringkonstruktion in der Reservierung (s. 5.7) bilden  $n$  aufeinanderfolgende Zeitscheiben, die periodisch wiederholt die gesamte Zeitachse darstellen, die Reservierungsstruktur. Im Token können nun einzelne Zeitscheiben angefordert werden (dynamisches TDMA, z.B. [FH76]). Durch die Einschränkung auf die sehr einfachen Leistungsparameter ist der Planungstest und die Reservierung einfach und schnell durchzuführen (auch für verbindungslose Kommunikation, vgl. [MZ95]).

#### 4.3.2.4 Folgerung

An einigen wichtigen Punkten werden die zwei unterschiedlichen Reservierungsansätze – zentrale  $\leftrightarrow$  dezentrale Reservierung – gegenübergestellt und daraufhin die zentrale Lösung gewählt. Angesprochen werden Fehlertoleranz, Datenintegrität, Leistungsbedarf, Skalierbarkeit.

Unter dem Gesichtspunkt *Fehlertoleranz* wird nur die Anfälligkeit gegen Ausfälle einzelner Komponenten betrachtet, speziell der dauerhafte Ausfall einzelner Knoten in der Planung (siehe 4.7). Ein temporärer Ausfall gefährdet die Uhrensynchronisation und bedarf weiterer Mechanismen, die hier nicht untersucht werden. Generell muß ein ausgefallener passiv beteiligter Knoten erkannt und in der Planung umgangen werden, aber für die Planung ergeben sich daraus keine weiteren Konsequenzen. Ein ausgefallener aktiv beteiligter Knoten muß zum Abbruch der Planung führen. Daneben ist in der zentralen Reservierung der Ausfall des Servers der kritische Punkt, weil er der einzige passiv beteiligte Knoten ist

und nicht umgangen werden kann. Jegliche weitere Veränderung des laufenden Systems würde ohne einen automatischen Ersatz verhindert. Der aktuelle Zustand der Netzwerknutzung bleibt aber erhalten und die laufenden Aufgaben können fortgesetzt werden, solange die globale Zeit erhalten bleibt. Eine Replikation des Servers müßte bei Bedarf für eine Erhöhung der Ausfallsicherheit sorgen.

Die *Integrität der Daten* ist in einer zentralen Reservierung vergleichsweise einfach zu erhalten. Alle globalen Planungsinformationen sind in einem Punkt gespeichert und Änderungen der jeweils lokal in den Knoten duplizierten Teilinformationen geschehen aufgrund einer Absprache in der Planung, die sich immer an der Referenz im Server orientiert. In einem dezentralen Ansatz ist der Aufwand für den Erhalt der Integrität mindestens genauso hoch. Generell modifizieren alle beteiligten Komponenten der Planung ihre Reservierungsdaten und im dezentralen Fall kann ihre Anzahl nur steigen.

Der *Leistungsbedarf* bezeichnet Umfang und Aufwand der Lösung für die beteiligten Komponenten. Entscheidend ist dabei der lokale Aufwand für die Algorithmen und der Umfang des erforderlichen Nachrichtenaustauschs. Es ist eine Frage der Komponente, die die notwendige Leistung erbringt. In einem Server ist eine zentrale Datenhaltung mit wenig Kommunikation verbunden. Die Leistung des Systems wird durch den Server zentralisiert. Bei der dezentralen Datenhaltung bindet das Token-Verfahren mehrere/alle Knoten in die Planung ein und verteilt den Aufwand auf mehrere Komponenten. Das Netz und alle Knoten müssen dabei eine höhere Leistung erbringen als mit einem zentralen Server, der die Hauptlast übernimmt. Bei der Transaktionsplanung ist jeder Knoten sein eigener Server, so daß er nur seinen eigenen Planungsaufwand tragen muß. Darüber hinaus ist die Konsistenz der Datenbasis durch das Protokoll nachzuhalten.

Die Frage der *Skalierbarkeit*, also der Vergrößerung der Anzahl angeschlossener Knoten, ist die traditionelle Schwäche eines zentralen Ansatzes. Mit dem verteilten System muß der Server angepaßt werden. Aber auch beim dezentralen Ansatz wächst mit der Zahl der Knoten der Aufwand für die Kanalverwaltung und -reservierung, die jeden einzelnen Knoten betrifft. Entweder erhöht sich die Dauer der Planung bei konstanter Last in den Knoten (Token-Planung<sup>7</sup>), oder über eine steigende Frequenz der Anfragen steigt auch die Last in den einzelnen beteiligten Knoten (Transaktionplanung<sup>7</sup>).

Schließlich ist das Multiple-Access-Netz selbst eine Invariante im System, das der Skalierbarkeit durch die beschränkte Bandbreite eine obere Grenze setzt.

Aufgrund der Möglichkeit zur Einbindung leistungsschwacher Knoten durch die klar getrennte Aufgabenlast (low-interference) fällt die Entscheidung für eine **zentrale Reservierung**. Geeigneten (Replikations-)Mechanismen müssen in sicherheitskritischen Bereichen die Verfügbarkeit sichern. Eine Skalierung betrifft, neben der physikalischen Leistung des Netzes, nur den Server und ist dadurch unabhängig von den bisherigen Knoten und Aufgaben.

---

<sup>7</sup>Hier sind die obigen Bezeichner aus 4.3.2.3 gemeint.

### 4.3.3 Kommunikationsserver

Der Kommunikationsserver (CS) ist der Koordinator des globalen Netzmanagements und trägt auch den größten Teil der Funktionalität (Abb. 4.1 auf Seite 64): Verwaltung der Kommunikationsverbindungen, der Knotenmenge und die Steuerung der Absprachen zwischen den lokalen Netzmanagern während des Kanalaufbaus.

#### 4.3.3.1 Kommunikationsverbindungen

Im Kommunikationsmodell, Abschnitt 4.2.2, wurden die verbindungsorientierten Kanäle als Grundlage der Kommunikation definiert. Aus der Sicht des Kommunikationsservers (CS) sind nur die globalen Aspekte von Interesse.

Eine *Verbindung* ist der globale Teil eines Kommunikationskanals. Es handelt sich um die reine Netzwerkverbindung und es ist eine Beschreibung der reservierten Zugriffszeiten für die TDMA-MAC sowie der Sender- und Empfängeradresse, ohne Betrachtung der lokalen Aspekte in den Knoten. Das globale Netzmanagement übernimmt durch den Server die Planungen in der zweiten OSI-Schicht. Der Kommunikationsserver verwaltet diese Verbindungen und weist jeder eine eindeutige ID zu, die im weiteren Protokoll zur Benennung dient; die ID des Kanals ist die der Verbindung.

#### 4.3.3.2 Aufbau des Servers

**Verbindungsverwaltung** Die Verbindungsverwaltung entspricht zunächst der Planung eines einfachen, unteilbaren Betriebsmittels, d.h. ein wechselseitiger Ausschluß mit einem serialisierten Zugriff wird realisiert (vgl. 5.3, 5.7). Die Planungsverfahren und die grundlegenden Reservierungsstrukturen sind denen einer CPU-Planung sehr ähnlich. Die Planung weist aber quantitative Unterschiede zur lokalen dynamischen CPU-Planung auf. Die Frequenz der Planungen ist kleiner, aber der Aufwand ist wesentlich größer, da durch die Einbeziehung des lokalen Managements mehrere Knoten beteiligt sind und die verzögernde Netzkommunikation dazwischenliegt. Dies führt zu qualitativen Unterschieden: Auf Umplanungen bestehender Verbindungen wird wegen des großen Aufwands verzichtet.

Die Verbindungsverwaltung ist keine Echtzeitaufgabe und an die Kanalplanungen werden keine zeitlichen Bedingungen gestellt. Überschreitet die Planungs-dauer die vorgesehene Startzeit, gehen die Netzzugriffsrechte der Applikation verloren. Die zeitlichen Parameter werden erst ab Bereitstellung der Verbindung bzw. des Kanals eingehalten.

**Verwaltungskanal** Jeder an der dynamischen Planung teilnehmende Knoten ist über einen bidirektionalen *Verwaltungskanal* permanent mit dem Server verbunden. Er besteht aus zwei unidirektionalen Kanälen, da Bidirektionalität nicht

direkt konzeptuell unterstützt wird. Der mit der Initialisierung aufgebaute Verwaltungskanal regelt die Kommunikation zwischen den Instanzen des Kommunikationssystems: Server  $\leftrightarrow$  lokales Management. Man erhält einen garantierten Durchsatz zum Server, auch wenn dies für die Planung alleine nicht nötig wäre.

Warum werden die übertragenen Verwaltungs- und Planungsinformationen nicht als NRT-Verkehr behandelt? Die Planung läuft zwar nicht als Echtzeitaufgabe, aber es ist für die Konfigurierbarkeit des Kommunikationssystems wichtig, daß minimale Durchsätze für die entsprechenden Daten zur Verfügung stehen, damit, bei einem ausreichend dimensioniertem Server, die Konfigurierbarkeit des Systems gesichert ist. Andernfalls wäre unter Vollast kein Kanalabbau mehr möglich.

**Knotenverwaltung** Alle Knoten im verteilten System müssen dem Server bekannt und jede Kommunikation zwischen ihnen angemeldet sein. Die Verwaltung der Knotenmenge zur Laufzeit ist eine Aufgabe des Servers.

In einer statischen Knotenmenge sind alle Teilnehmer zum Systemstart bekannt und sie bleibt über die gesamte Laufzeit unverändert. Eine manuell erstellte Tabelle kann zum Beispiel diese Menge beschreiben. Eine dynamische Knotenverwaltung erlaubt darüber hinaus eine Modifikation der Knotenmenge zur Laufzeit. Neue Knoten müssen auf eine Sendeaufforderung durch den Server warten und die 'Initialisierung' durchlaufen. Im folgenden wird einer statischen Knotenmenge ausgegangen.

**Initialisierung** Zum Systemstart befindet sich das Netz noch in einem NRT-Zustand. Alle angeschlossenen Knoten sind zu ermitteln (s.o.) und deren Initialisierung zu starten. Dazu gehört die Planung von grundlegenden, im voraus bekannter Kanäle und die Synchronisation der Uhren. Erst wenn *alle* Knoten diese Prozedur abgeschlossen haben, kann das Netz auf Echtzeitbetrieb umgestellt werden.

Die Planung von Kanälen vor dem Echtzeitbetrieb hat zwei Gründe. Zum einen wird jeweils ein Verwaltungskanal zwischen Server und jedem an der dynamischen Planung teilnehmenden Knoten aufgebaut. Zum anderen bietet die Vorabreservierung von Kanälen die Möglichkeit, wichtige Kanäle vor allen anderen aufzubauen. Da Garantien erst *nach* erfolgreicher dynamischer Planung vergeben werden, ist dies der einzige Weg, einen Kanal unbedingt aufzubauen. Dies entspricht einer statischen Offline-Analyse, die für diese Forderung unumgänglich ist (vgl. 1.1.2).

Knoten mit unveränderlichen Kommunikationsanforderungen, welche beim Systemstart (statisch) festgelegt und garantiert werden, kann man von der dynamischen Reservierung ausschließen. Man verbietet damit jegliche (Um-)Planung der entsprechenden Kanäle!

### 4.3.4 Verbindungsaufbau

#### 4.3.4.1 Protokoll

In Abbildung 4.2 beschreibt ein *Petri-Netz* den Ablauf des Protokolls in der globalen Verbindungsreservierung auf dem Server [Bau96]. Es handelt sich um ein System mit individuellen Marken. Die *Stellen* sind in der Abbildung als Kreise dargestellt und in ihnen halten sich die *Marken* auf, die den aktuellen Zustand des Systems beschreiben. Die Marken können beliebige mathematische Konstrukte sein. Die *Transitionen*, als Rechtecke dargestellt, bewegen die Marken durch *Schaltungen* auf den Stellen gemäß der Beschriftung der Kanten. Eine Transition ist so lange aktiv, wie die an den Eingangskanten beschrifteten Marken in den Eingangsstellen zur Verfügung stehen. Eine aktive Transition kann zu einem beliebigen Zeitpunkt schalten, löscht die entsprechenden Marken der Eingangsstellen und legt die Marken, die jeweils an den Kanten angegeben sind, in die Ausgangsstellen.

Die grau gefüllten Stellen bilden den Übergang zum lokalen Teil des Protokolls; sie bilden den Rand des abgebildeten Teilnetzes innerhalb der gesamten Protokollspezifikation, deren weitere Teile in späteren Abschnitten vorgestellt werden. Der Markenaustausch über diese Randstellen beschreibt die Netzwerkkommunikation des Protokolls.

Die Stelle *ID-Vorrat* beinhaltet als Marken eine endliche Untermenge der natürlichen Zahlen  $\mathcal{I} \subset \mathbb{N}_0$ , die in der Variablen *id* eine Verbindung eindeutig bezeichnen. In *KommTeilnehmer* sind die Marken Tupel<sup>8</sup> von Zahlen  $(id, y, z) \in \mathcal{I} \times \mathcal{K} \times \mathcal{K}$ , die eine Abbildung der IDs auf die geordneten Paare der Kommunikationsteilnehmer über der Knotenmenge  $\mathcal{K}$  beschreiben; *y* ist der Initiator der Planung. In *Planungsliste* – die interne Realisierung der Stellen ist nicht dargestellt (z.B. FIFO) – und *Erfolg?* sind nur Tupel  $(id, n) \in \mathcal{I} \times \mathbb{N}_0$  enthalten. Jede Marke beschreibt eine vorläufige Reservierung zu *id*, die sich im *n*-ten Planungsversuch befindet; höchstens einer von zwei Knoten hat bisher zugestimmt. Die *Reservierung* enthält neben der allgemeinen Marke  $\bullet$  auch Marken der Form  $(id, n)$ , die für  $n > 0$  vorläufige Verbindungsreservierungen und für  $n = 0$  feste Reservierungen eines etablierten Kanals darstellen. Die QoS-Parameter kann man sich an die *id* gebunden denken. Der Planungsalgorithmus wird nicht im Netz dargestellt. Die Beschreibung der anderen Stellen und Transitionen ergibt sich aus dem Zusammenhang.

Die folgenden fünf Punkte erläutern den Ablauf des Verbindungsaufbaus:

1. Entgegennahme der Leistungsanforderungen des Senders  
Der Reservierungsauftrag an den Kommunikationsserver für den Aufbau einer neuen Verbindung wird über den Verwaltungskanal gestellt. Neben der Empfängeradresse sind die geforderten QoS-Parameter die Grundlage für die Planung. Der Verbindung wird eine eindeutige ID zugewiesen.

---

<sup>8</sup>Die Abbildung verzichtet auf die Darstellung der Tupelklammern.

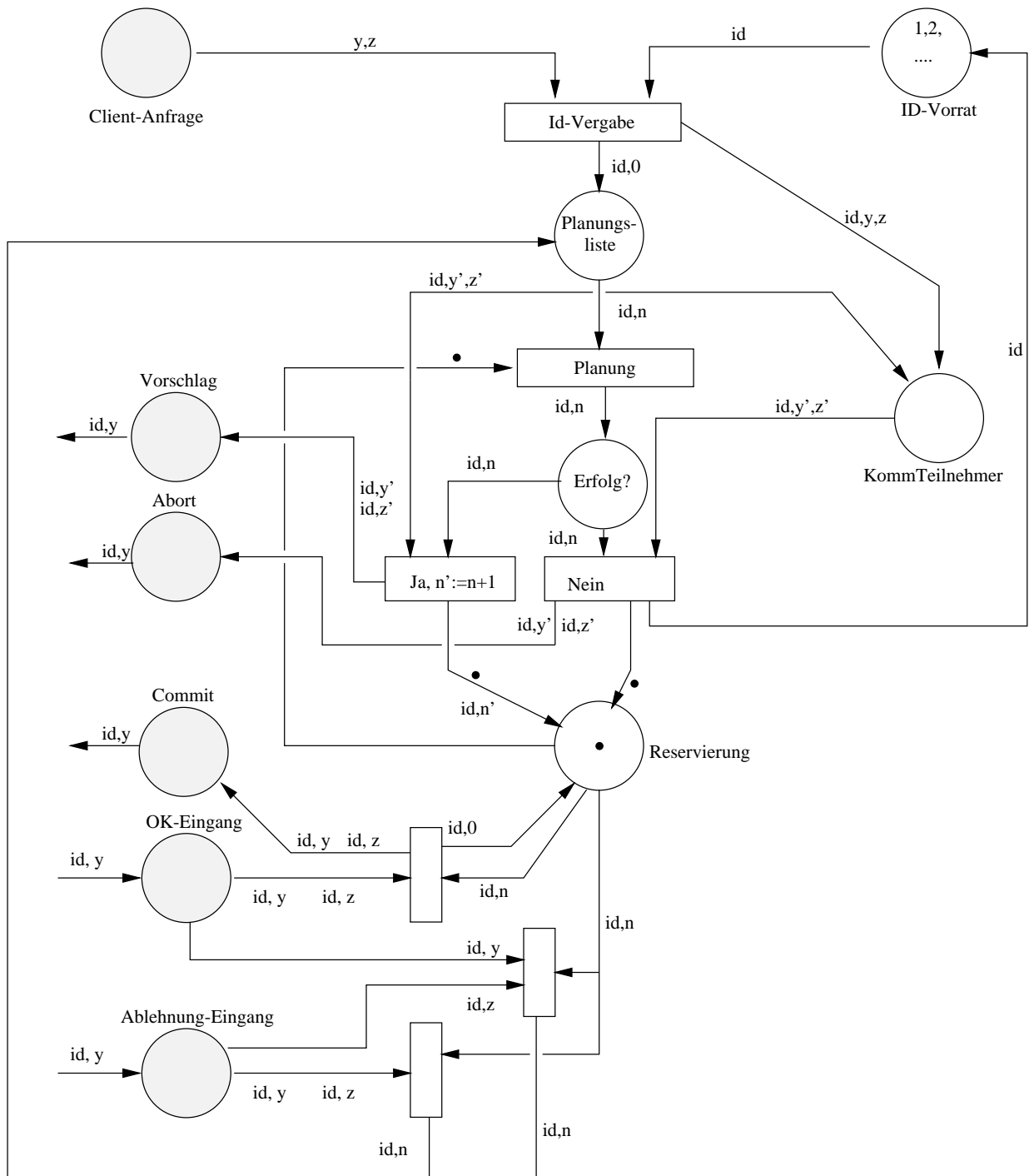


Abbildung 4.2: Petri-Netz des Verbindungsaufbaus

In der Randstelle **Client-Anfrage**, die mit dem Netz des lokalen Managements verbunden ist, werden die Kommunikationsteilnehmer der angeforderten Verbindung als Paar  $(y, z)$  in das globale Protokoll übergeben;  $y$  ist der Initiator. Die Transition **ID-Vergabe** entnimmt ein Paar  $(y, z)$  und eine ID aus den Eingangsstellen und legt neue Marken in die Ausgangsstellen:  $(id, y, z)$  für die Zuordnung von ID und Kommunikationsteilnehmer in **KommTeilnehmer** und  $(id, 0)$  in die Liste der zu planenden Anforderungen **Planungsliste** (es hat bisher kein Planungsversuch stattgefunden).

## 2. Planung in der zentralen Datenbasis

In den zentralen Daten des Servers wird versucht, die notwendigen TDMA-Zugriffszeiten der Verbindung zu reservieren. Ein vorläufiger Eintrag sichert bis zur endgültigen Entscheidung im Punkt 5 (s.u.) die ausgesuchten Zeiten und fährt mit diesem Vorschlag im nächsten Punkt fort.

Die zentrale Reservierung ist der einzige Punkt auf den nebenläufige Verbindungsplanungen jeweils nur serialisiert zugreifen dürfen; alle Zugriffe auf die Datenstruktur finden in einem kritischen Abschnitt statt (engl. *critical region*).

Das Antreffen eines vorläufigen Eintrags einer Verbindung  $v_1$ <sup>9</sup> kann entweder die aktuelle Planung von  $v_2$  abbrechen und zurückstellen, um später einen erneuten Versuch zu unternehmen, oder es wird einfach keine Vorläufigkeit erkannt. Im dem hier dargestellten zweiten Fall wird der Kanalaufbau komplett abgelehnt und es liegt an der Applikation später erneut und ggf. mit anderen Parametern einen Kanal anzufordern.

Im Petri-Netz sorgt die allgemeine Markierung  $\bullet$  in **Reservierung** für das serialisierte Durchlaufen der Planung. **Planung** nimmt darüberhinaus eine Marke aus der **Planungsliste**. Die Markierung  $\bullet$  wird erst nach Prüfung des Planungsergebnisses (**Erfolg?**) wieder in die ursprüngliche Stelle gelegt und damit der kritische Abschnitt verlassen.

Bei einem positiven Ergebnis wird die Anzahl der Planungsversuche erhöht,  $(id, n')$ ,  $n' > 0$ , in der **Reservierung** vermerkt und der Vorschlag an die Teilnehmer  $y'$  und  $z'$  geschickt. Die Kommunikationsteilnehmer erhält man durch nicht-löschendes Lesen (Schlinge<sup>10</sup>) aus **KommTeilnehmer** über die eindeutige  $id$ . Bei einem negativen Ergebnis wird die Marke aus **KommTeilnehmer** entfernt und die ID wieder in den Pool des ID-Vorrats zurückgegeben. Eine **Abort-Nachricht** geht jeweils an die Kommunikationsteilnehmer.

## 3. Kommunikationsteilnehmer fragen

Eine gelungene Reservierung der Verbindung aus dem vorigen Punkt wird

<sup>9</sup>Die Betrachtung der Vorläufigkeit von  $v_1$  sei für den Planungserfolg von  $v_2$  entscheidend.

<sup>10</sup>Eine Schlinge ist eine ungerichtete Kante, oder Pfeile in beiden Richtungen, die eine Marke liest und schreibt und somit nichts an der betroffenen Stelle ändert.

als Vorschlag in die weiteren Absprachen übernommen. Das Angebot des globalen Managements wird über **Vorschlag** an die Kommunikationsteilnehmer geschickt und die Leistungsverhandlung mit dem lokalen Management begonnen bzw. fortgesetzt.

Um die Kanalplanung komplett abzuschließen, müssen die Kommunikationsteilnehmer dem gefundenen Vorschlag zustimmen und durch lokale Planungen für eine konfliktfreie Integration der Datenübertragung in das System sorgen.

In den einzelnen Knoten wird der Vorschlag vom lokalen Netzmanagement bearbeitet (s.u.). Eine Annahme ist wiederum nur vorläufig. Erst der letzte, 5. Punkt der Planung schreibt alles fest.

#### 4. Antwort entgegennehmen

Die Antworten auf die Vorschläge entscheiden über Etablierung oder Verwurf der vorläufigen Verbindung. Im Falle einer Ablehnung der Vorschläge durch einen Knoten muß der Server zum 2. Punkt zurückgehen und die Planung dort erneut ansetzen. Zusätzliche Informationen über das Mißlingen, wie z.B. kollidierende, nicht-verschiebbare lokale Planungen oder alternative Vorschläge, können in einem erweiterten Konzept die Planung vereinfachen.

**OK-Eingang** und **Ablehnung-Eingang** nehmen die Antworten der lokalen Managements entgegen. Bei zwei positiven Antworten erfolgt die Freigabe der Verbindung im nächsten Punkt. Bei mindestens einer Ablehnung muß die Planung durch den Eintrag in die Warteschlange **Planungsliste** in eine weitere Runde gehen und der vorläufige Eintrag in **Reservierung** wird gelöscht.

#### 5. Commit

Erst wenn alle Kommunikationsteilnehmer den Vorschlag akzeptiert haben, wird das Ergebnis durch eine Commit-Nachricht an die Teilnehmer festgeschrieben. Die Planung in der zweiten Schicht und die Leistungsverhandlung zwischen der zweiten und vierten OSI-Schicht über den Verbindungsaufbau ist damit abgeschlossen.

$(id, y)$  und  $(id, z)$  müssen in **OK-Eingang** vorliegen damit die Vorläufigkeit durch Rücksetzen der Marke  $(id, n)$  aus **Reservierung** auf  $(id, 0)$  aufgehoben und die entsprechende Commit-Nachricht über **Commit** an die Teilnehmer gesandt wird.

### 4.3.4.2 Ergänzungen

Eine Planung unter Echtzeitbedingungen kann mit Hilfe der globalen Zeit die Anzahl der notwendigen Nachrichten reduzieren. Die globale Zeit und die RT-Planung in *jedem* Knoten erlauben es, statt der Stellen **Abort** und **Ablehnung-Eingang** Fristen für die positiven Antworten zu setzen [DNS94].

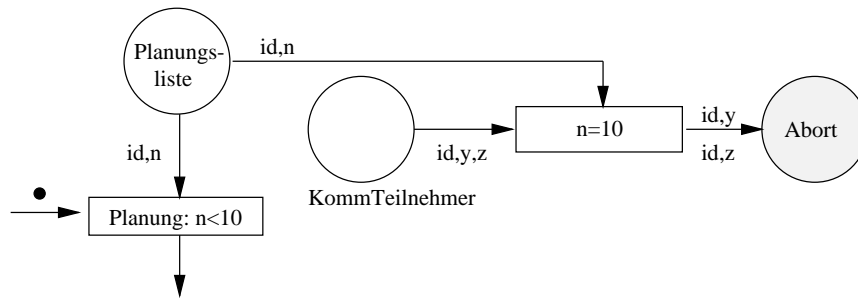


Abbildung 4.3: Erweiterung zum vorzeitigen Abbruch des Protokolls

In dem bisherigen Entwurf ist der Abbruch einer erfolglosen Planung nur durch ein Fehlschlagen des Planungsalgorithmus möglich. Eine maximale Anzahl von erfolglosen Iterationen läßt sich durch eine entsprechende Abbruchbedingung in das Netz integrieren (s. Abb. 4.3). **Planung** schaltet nur noch, wenn es für  $id$  bisher weniger als zum Beispiel 10 Versuche gab. Andernfalls wird ein Planungs-mißerfolg angenommen und entsprechende Nachrichten werden an die Teilnehmer geschickt.

#### 4.3.4.3 Bewertung

Damit ist das Protokoll für den globalen Verbindungsaufbau dargestellt. Es bietet eine verteilte Absprache, bei der mehrere Anforderungen, bis auf den zentralen Zugriff auf die Planungsinformationen, nebenläufig behandelt werden können. Es stellt einen Rahmen zur Verfügung, mit dem in weiteren Arbeiten durch Verbesserungen im Detail (z.B. Hardwareabstimmung), der Untersuchung verschiedener Realisierungsansätze und vor allem unterschiedlicher Planungsalgorithmen die grundlegenden Leistungen optimiert werden können.

Die Randstellen des Petri-Netzes beschreiben als Schnittstelle zum lokalen Protokollteil den Umfang an Netzwerkkommunikation über die Verwaltungskanäle. Eine einleitende Nachricht in **Client-Anfrage** und zwei für das Ergebnis der Planung über **Abort** oder **Commit** sind für jede Kanalanzforderung zu versenden. Beim Abbruch wird unter Berücksichtigung der Annahme des letzten Vorschlags eine Nachricht eingespart. Dazu kommen noch für jeden Planungsvorschlag vier Nachrichten über **Vorschlag**, **OK-Eingang** bzw. **Ablehnung-Eingang**. Der Kommunikationsaufwand ist unabhängig von der Systemgröße, sondern hängt von der Dauer der Leistungsverhandlung und damit von der Auslastung des Netzes und der jeweils lokalen CPUs ab. **Client-Anfrage** und **Vorschlag** transportieren die Spezifikation und sind daher „große Nachrichten“, aber die übrigen beschränken sich auf wenige Bytes. In einer Erweiterung läßt sich der Aufwand über zwei Wege reduzieren: Ein Broadcast des Vorschlags kann eine große Nachricht und die Planung unter Echtzeitbedingungen kann die kleinen Nachrichten **Abort** und **Ablehnung-Eingang** einsparen (s.o.).

Prinzipiell kann man den Vorschlag, wie dargestellt, parallel allen (teilnehmenden) Knoten gleichzeitig unterbreiten und erhält schnellstmöglich eine Antwort: asynchrone Benachrichtigung. Auf der anderen Seite werden u.U. dadurch in den Knoten unnötige Umplanungen lokaler Aktivitäten veranlaßt, wenn der Vorschlag durch die Ablehnung eines Knotens verworfen wird. Ein sequentielles, synchrones Fragen der Teilnehmer dauert entsprechend länger, weil man die Antwort des ersten Knotens abwartet bevor der zweite gefragt wird. Aber man spart im Falle einer Ablehnung im Mittel die Hälfte aller Vorschläge. Daher wird in der Realisierung das sequentielle Vorgehen zu bevorzugen sein, um die Belastung der einzelnen Knoten – die Planung muß zusätzlich zu den geplanten Echtzeitaufgaben stattfinden – möglichst gering zu halten.

### 4.3.5 Kanalabbau

Ein Kanal wird abgebaut, wenn eine zeitliche begrenzte Reservierung abläuft oder wenn ein Kommunikationsteilnehmer dies explizit verlangt. Mit dem Abbau eines Kanals sind TDMA-Reservierungen für die Verbindung im globalen Management und CPU-Planungen im lokalen Management wieder freizugeben. Bei einem vorzeitigen Abbruch sind die anderen Kommunikationsteilnehmer explizit über den Verwaltungskanal zu benachrichtigen.

## 4.4 Lokales Netzmanagement

### 4.4.1 Aufgaben

Das lokale Netzmanagement ist die Schnittstelle von der Applikation zur Verbindungsverwaltung des Kommunikationsservers und stellt entsprechend die lokale Funktionalität der Kanäle, die auf einer Netzwerkreservierung aufbaut, zur Verfügung. Im wesentlichen umfaßt dies drei Aufgaben in der vierten OSI-Schicht:

1. Das lokale Management verwaltet zusammen mit dem globalen die Kanäle, an denen der Knoten teilnimmt. Es nimmt in der Kanalplanung am Reservierungsprotokoll und der Leistungsverhandlung des Verbindungsaufbaus teil, um die Leistungen zwischen der zweiten und vierten OSI-Schicht abzustimmen; es ist eine aktiv beteiligte Komponenten im Sinne von 4.3.2.1.
2. Die kontrollierte Nutzung der vereinbarten Leistungen ist die notwendige zweite Hälfte des Garantievertrags (lokale Garantie). Flußkontrolle und Overload-Verhinderung schützen die Garantien anderer Netzwerkreservierungen, deren konfliktfreier Zugriff andernfalls nicht gesichert ist.
3. Die Integration der Kommunikation in die Echtzeitverarbeitung des Knotens ist eine Voraussetzung für die Vorhersagbarkeit der lokalen Echtzeitaufgaben. Der erforderliche Betriebsmittelbedarf für die Kommunikation,

insbesondere der CPU für die Protokollverarbeitung und Flußkontrolle, ist beim Kanalaufbau zu planen.

Um die beiden letzten Punkte genauer zu analysieren, wird zunächst die von der Kommunikation verursachte sender- und empfängerseitige CPU-Belastung untersucht. Die kontrollierte Nutzung und die Planung der CPU löst die beiden Aufgaben.

#### 4.4.1.1 Senden

Das globale Netzmanagement stellt für einen Kanal reservierte TDMA-Zugriffszeiten zur Verfügung, die der applikationsseitigen Leistungsspezifikation genügen. Die Datenübertragung ist auf diese Zeitintervalle beschränkt, in denen man die Netzwerkbandbreite kollisionsfrei nutzen kann – Senden ist eine Echtzeitaufgabe. Netzzugriffe außerhalb der Intervalle sind nicht erlaubt, denn es könnten andere Reservierungen berührt werden und Kollisionen auftreten (der lokale Knoten hat keine totale Sicht auf die Planungsdaten). Das Kommunikationssystem kontrolliert den Datenversand und man kann eine Realisierung anhand der Hardwareunterstützung unterscheiden.

**Automatisches Senden** Der Applikationsprozessor<sup>11</sup>, auf dem die Echtzeitaufgaben bearbeitet werden, übergibt die sendebereiten Daten frühzeitig an einen Netzwerkprozessor, der, in Kenntnis der zugeteilten Zeitintervalle, den rechtzeitigen Versand übernimmt. Ein dedizierter Prozessor oder ein entsprechend programmierbarer Netzwerkcontroller ermöglicht die komfortabelste Realisierung des lokalen Netzmanagements – je mehr Aufgaben abgetreten werden können, desto kleiner sind Überschneidungen auf dem Applikationsprozessor.

**Direktes Senden** Beim direkten Zugriff auf das Netzwerk übernimmt der Applikationsprozessor die Protokollverarbeitung und Flußkontrolle und realisiert somit den zeitgesteuerten Versand der Daten. An dieser Stelle kann in der Planung ein Konflikt zwischen zwei Betriebsmitteln entstehen: Zu den reservierten TDMA-Intervallen der Netzwerkverbindung ist die CPU zu reservieren. Erst die Kombination der beiden Betriebsmittel bildet einen Kanal und sichert die Integration in das System.

Die Planung ist notwendig, um in einem zeitgesteuerten System die Echtzeitaufgabe ‘Senden’ korrekt zu verarbeiten. Im einfachsten Fall wird die CPU während der gesamten TDMA-Intervalle für die Protokollverarbeitung verplant und dadurch automatisch der Datenfluß kontrolliert. Die Kommunikation wird über die CPU-Reservierung in die Echtzeitverarbeitung integriert.

---

<sup>11</sup>Dieses Synonym für ‘CPU’ verdeutlicht die Rolle in der Echtzeitverarbeitung.

In der Einleitung wurde auf einen dedizierten Prozessor als Grundlage für das Kommunikationssystem verzichtet. Statt dessen wird von der minimalen Ausgangsbasis ausgegangen. Die Entscheidung für das direkte Senden ermöglicht den Einsatz in jedem zeitgesteuerten System. In der Praxis wird eine differenziertere Unterscheidung der Hardwareunterstützung die Realisierung vereinfachen und Erweiterungsmöglichkeiten aufzeigen.

#### 4.4.1.2 Empfangen

Der zweite Punkt ist die Frage der empfängerseitigen Belastung: Was kostet der Empfang? Wann ist eine Planung auf dem Applikationsprozessor notwendig? Das Empfangen von Daten ist weniger kritisch, weil es keinen Einfluß auf die globale Garantie hat. Es geht allein um die Integration des Kommunikationskanals in die Echtzeitverarbeitung des lokalen Knotens. Eine ähnliche Unterscheidung wie beim Senden kann auch für die RT-Aufgabe ‘Empfangen’ getroffen werden.

**Nebenläufiger Empfang** Beim nebenläufigen Empfang werden die einzelnen Nachrichten ohne Belastung des Applikationsprozessors angenommen.<sup>12</sup> Die Daten legt der Netzwerkcontroller in einen eigenen Speicherbereich ab, der nicht zum Hauptspeicher des Rechners gehört. Mit getrennten Speichern geschieht der Empfang der Daten unabhängig von der Echtzeitverarbeitung der CPU. Andernfalls könnten DMA-Zugriffe die CPU „ausbremsen“ ( $\leadsto$  *priority inversion*, [ZA95]), weil DMA und CPU um die Speicherbandbreite konkurrieren. Die WCET-Spezifikationen<sup>13</sup> wären somit vom Kommunikationsaufwand anderer Aufgaben abhängig. Weil sich dieser Aufwand erst zur Laufzeit bestimmen läßt, führt eine Abschätzung u.U. zu geringer Planungsauslastung.

Die CPU-Planung der Echtzeitkommunikation beschränkt sich also auf das zeitgerechte Abholen der Daten aus dem Netzwerkpuffer. Hierfür können weit weniger enge zeitliche Intervalle genutzt werden, so daß auf Kosten der Tightness die Planung über mehr Spielraum verfügt.

Der Pufferspeicher ist ein drittes Betriebsmittel, um das die Kanäle konkurrieren, und das geplant werden muß, damit keine gegenseitigen Störungen auftreten. Man kann den Speicher implizit planen (s. 5.6.1), indem nach Bedarf an jeden Kanal Bereiche vergeben werden und man durch die Tightness sicherstellt, daß es nie zu einem Überlauf des Puffers kommt. Zur Berechnung des Speicherbedarfs eines Kanals ist die Bandbreite mit der Verzögerung der Abholung zu multiplizieren, um so den Einfluß der Tightness auf den Pufferspeicher zu erhalten. Wenn man diesen Bedarf explizit einem

---

<sup>12</sup>Den nebenläufigen Empfang zu untersuchen ist auch in Einprozessorsystemen sinnvoll, weil der Netzcontroller zwar oft für das automatische Senden ungeeignet ist, aber zumindest einen minimalen eigenen Speicher zur Pufferung eintreffender Nachrichten hat.

<sup>13</sup>*worst case execution time*

einzelnen Kanal zuordnet, ist eine gegenseitige Störung ausgeschlossen. Ein Pufferüberlauf, also das Überschreiben älterer Daten durch neuempfangene, wirkt sich nur auf den jeweiligen Kanal aus. In diesem Fall ist die Frage zu stellen, ob die Echtzeitbedingungen sinnvoll spezifiziert waren.

**Direkter Empfang** Beim direkten Empfang muß während der Übertragung einzelner Bytes einer Nachricht der Applikationsprozessor zur Verfügung stehen, andernfalls gehen diese Bytes unbemerkt verloren bzw. laufende Tasks werden durch Interrupts und DMA-Zugriffe, die der Netzcontroller auslöst, verzögert. Die TDMA-Zeitintervalle müssen auch auf der CPU reserviert werden. Bei einem relativ zur CPU langsamen Kommunikationsmedium sinkt dabei die Planungsauslastung erheblich, weil die Datenübertragung nur einen Bruchteil der CPU-Kapazität nutzt.

Das Vorgehen entspricht dem direkten Senden und resultiert in einen ähnlichen Konflikt zwischen Netz und CPU.

Analog zur Senderseite wird zunächst vom direkten Empfang ausgegangen. In der Praxis kann man das Problem der schlechten Planungsauslastung in jedem einzelnen Knoten durch eine Abstimmung der lokalen CPU-Planung auf die Hardware soweit wie möglich entkräften.

#### 4.4.1.3 Dynamische Planung

Die dynamische Planung der Kanäle und die Leistungsverhandlung mit dem globalen Management ist die dritte wesentliche Aufgabe des lokalen Kommunikationssystems. Sie wird in Abschnitt 4.4.3.1 besprochen.

### 4.4.2 Aufbau des lokalen Kommunikationssystems

Die bisherigen Erläuterungen kann man nun zu einem kompletten Modell des lokalen Kommunikationssystems erweitern. In Abbildung 4.4 sind die Grundzüge skizziert und sie werden in den nächsten Abschnitten verfeinert.

Jede Datenübertragung läuft über einen Kommunikationsthread (CT), der die Echtzeitaufgabe des direkten Sendens bzw. Empfangens für genau einen Kanal ausführt. Die gepufferten Daten einer Applikation werden dadurch „TDMA-gerecht“ versandt bzw. empfangen. Der CT integriert die Netzwerkverbindung ins System und stellt einen Echtzeitkommunikationsdienst in der vierten OSI-Schicht bereit.

Der Kommunikationsdaemon<sup>14</sup> (CD) repräsentiert die Verwaltung des lokalen Netzmanagements. Er übernimmt die Leistungsverhandlung mit dem globalen Management und erweitert die etablierten Verbindungen lokal zu Kanälen (CT-Planung, Aufbau der Datenstrukturen und Übergabe an die Applikation). Zwei

---

<sup>14</sup>Ich bleibe bei der englischen Schreibweise, um den Leser nicht mit Dämonen zu ängstigen...

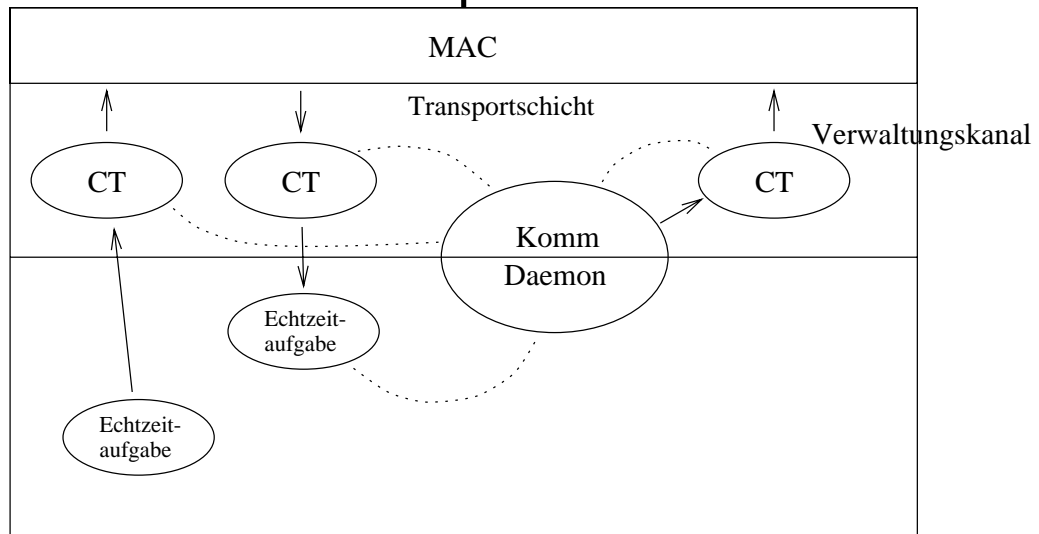


Abbildung 4.4: Aufgabenverteilung des lokalen Kommunikationssystems

unidirektionale Kanäle bilden den bidirektionalen Verwaltungskanal zwischen CD und Kommunikationsserver (in der Abbildung nur einfach dargestellt).

Eine Schnittstelle zwischen den Kommunikationsthreads und dem Daemon dient der Verwaltung bestehender Kanäle (Rekonfiguration, Abbau, o.ä.). An dieser Stelle kann später das Konzept erweitert werden, um zum Beispiel durch eine Abgabe von ungenutzten Reservierungen eine Auflockerung der strikten Zuordnung zwischen CT und Kanal zu ermöglichen (s. NRT-Kommunikation, Abschnitt 4.4.3.2).

### 4.4.3 Kommunikationsdaemon

Der Kommunikationsdaemon (CD) beinhaltet die Verwaltung und Funktionalität des lokalen Netzmanagements und ist auch für die Integration von NRT-Kommunikation in das System zuständig. Jeweils genau eine Instanz ist in jedem Knoten aktiv, der an der dynamisch planbaren Kommunikation teilnimmt. Der Daemon enthält die Schnittstelle des Kommunikationssystems zur Applikation.

#### 4.4.3.1 Planung

Der CD besitzt eine kalenderbasierte Datenstruktur, die eine teilweise Sicht auf die globalen Planungsdaten (TDMA-Zeiten) des Kommunikationsservers bietet. In dieser Struktur sind nur die reservierten TDMA-Zugriffszeiten der Kanäle gespeichert, an denen der Knoten teilnimmt. Es handelt sich zunächst um reine Reservierungsdaten, auf denen kein Planungsalgorithmus läuft. Die Planung der

Verbindungen findet im Kommunikationsserver statt und lediglich die Ergebnisse werden auch jeweils lokal in den aktiv beteiligten Knoten gespeichert.

Der Server bietet im Verbindungsvorschlag eine Leistung an, die der Daemon bewerten muß. Die Leistung wird in der Leistungsverhandlung durch Vorschlag und Ablehnung solange auf die lokalen Anforderung abgestimmt, bis die Kommunikationsteilnehmer sie annehmen oder der Server die Planung abbricht. Die lokalen Anforderungen bestimmt die CPU-Planung, die gemeinsam mit der Netzreservierung die korrekte Ausführung der RT-Aufgaben 'Senden' bzw. 'Empfangen' garantiert: die Verhandlung synchronisiert die Reservierung der beiden Betriebsmittel (engl.: *Multiple Resource Scheduling*).

Ein Scheduler, ein Planer, ist jeweils für ein Betriebsmittel vorhanden. Die Planung des Netzes findet im Server und die der CPU anschließend lokal im Knoten statt. Die einzelnen Planer für Netz und CPU sind zunächst unabhängig voneinander und das lokale Management synchronisiert die CPU-Reservierung der vierten Schicht mit der TDMA-Reservierung der zweiten Schicht. Ein grundlegendes Protokoll ist bereits in 4.3.4 erläutert worden und wird später durch den lokalen Teil vervollständigt.

Die Beschränkung der Verbindungsplanung auf den Server kann zur effizienteren Planung in zwei Ansätzen erweitert werden. Wenn die Planung des Kommunikationsthreads zu den vorgeschlagenen TDMA-Zeiten mißlingt, kann ein (einfacher) Planungsalgorithmus Alternativen in der lokalen CPU- und TDMA-Reservierung suchen und als Gegenvorschlag mit der Ablehnung an den Server schicken. Ein verteilter Algorithmus, erster Teil im Server, zweiter Teil im Knoten, der bei einer Ablehnung aktiv wird und nach einem Alternativvorschlag sucht, der auf dem bisherigen aufbaut, könnte erfolgversprechend in einer zukünftigen Arbeit untersucht werden. Zum Beispiel kann der Server in einem größeren Fenster ein TDMA-Intervall vorschlagen, so daß bei Konflikten in der lokalen CPU-Planung das Intervall verschoben werden kann.

Der zweite Ansatz, mit einer lokalen Planung die strikte Aufgabentrennung zu lockern und damit eine höhere Reaktivität der Kanalverwaltung zu erlangen, beruht auf einer *Kontingentvergabe* reservierter TDMA-Slots (Zeitintervalle) an einzelne Knoten. Der Daemon hat damit einen Vorrat an nicht-benutzten Slots zur Verfügung, um interne Anfragen erfüllen zu können, ohne daß eine Reservierungsanfrage an den Server gestellt werden muß. Diese Slots sind im globalen Management reserviert und stehen dem entsprechenden Knoten allein zur Verfügung. Allerdings muß für einen Kanalaufbau noch immer die Gegenseite in die Planung einbezogen werden. Eine starke Vereinfachung der Planung erreicht man besonders dann, wenn die empfängerseitige Zustimmung (fast) sicher ist. Dieses Konzept wird zum Beispiel eingesetzt, um die Planung einer Broadcastkommunikation zu unterstützen (s. 4.5).

#### 4.4.3.2 NRT-Kommunikation

Die NRT-Kommunikation unterliegt keinen zeitliche Bedingungen und somit eine Planung eigentlich nicht notwendig. Eine Nutzung der Netzleistung *neben* den reservierten Kanälen verlangt aber eine geplante Integration, um die Garantien anderer Echtzeitaufgaben nicht zu gefährden. Die Kommunikation wird zunächst auf verbindungsorientierte NRT-Kanälen beschränkt; Kapitel 4.8 geht auf die verbindungslose Kommunikation ein.<sup>15</sup>

Mit dem vorausgesetzten **direkten Empfang** werden die Möglichkeiten einer flexiblen NRT-Integration stark eingeschränkt. Es bleibt einem nichts anderes übrig, als die CPU der Empfängerseite für die Übertragung zu reservieren und somit einen Kanal aufzubauen.<sup>16</sup> Dabei ist jeweils im Prinzip nur ein Kanal von Knoten *A* nach Knoten *B* notwendig, der die gesamte NRT-Kommunikation zwischen ihnen aufnimmt und die Daten nach FCFS oder Round-Robin Strategie überträgt (vgl. Abb. 4.6); bei stark schwankender Last bieten sich mehrere Kanäle an, die nach Bedarf auf- und abgebaut werden.

Alternativ könnten auch ungenutzte Kapazitäten auf vorhandenen Kanälen für diese Aufgabe genutzt werden. Auf diese Weise sind auch die verplanten und ungenutzten Zeiten nicht verloren, obwohl die Planungsauslastung nicht steigt. Die NRT-Daten müssen in diesem Fall im Daemon gepuffert werden. Der CT meldet das vorzeitige Ende seiner Übertragung über die beschriebene Schnittstelle zum Daemon und greift auf die dort gepufferten NRT-Daten zurück, um die freien Zeiten für die Übertragung zu nutzen. Auf der Gegenseite muß der CT die NRT-Kommunikation von den eigentlichen Kanaldaten unterscheiden und an den Daemon weiterleiten.

Beim direkten Empfang ist die NRT-Kommunikation die alleinige Aufgabe des Daemons und das globale Netzmanagement bleibt davon unberührt. Da der Daemon die freigewordenen Slots verwaltet, muß jede NRT-Kommunikation von ihm gesteuert werden. Ein zentraler Puffer ist für das Sammeln der Daten im Knoten notwendig. Dies entspricht einem einzigen Kanalobjekt auf das die Applikationen zugreifen und dessen CT vom Daemon aktiviert wird. Dies gilt auch für die folgenden Abschnitte.

Erst wenn man auf der Empfängerseite von einem **nebenläufigen Empfang** ausgeht, ergeben sich wesentliche Vereinfachungen für die NRT-Kommunikation. Weil unterschiedliche Empfangsmöglichkeiten gleichzeitig im Netz einsetzbar sind, kann man auf einige Vereinfachungen eingehen, ohne damit die grundsätzliche Entscheidung für den direkten Empfang zu verwerfen. Die mit der Kommunikation gleichzeitige Nutzung der CPU ist auf Empfängerseite nicht mehr notwendig und es bleibt nur die Ermittlung der freien Zeitslots und die darauf beschränkte

---

<sup>15</sup>Die Einheit aus Verbindung und CPU bleibt weiterhin gegeben, ist aber nicht mehr für die einzelne Datentübertragung zwischen bestimmten Aufgaben geplant.

<sup>16</sup>Die Datentübertragung ist ohne eine Reservierung der Gegenseite nicht sinnvoll, weil viele Daten verloren gingen.

Nutzung zu lösen. Auf der Gegenseite können Daten wegen des ungeplanten NRT-Zugriffs verlorengehen. Die Applikation bzw. ein erweitertes Transportprotokoll muß die Nachrichten sequenzieren und ggf. wiederholt übertragen.

Der Daemon verwaltet die freien Slots des lokalen Managements. Lokal geben die aktiven CT ihre ungenutzte Zeiten, die uneingeschränkt für die Übertragung zu jedem Knoten bereitstehen,<sup>17</sup> dem Daemon bekannt; ein freies Kontingent kann natürlich ebenfalls genutzt werden. Zusätzlich kann der Server unverplante Slots an einzelne Knoten vergeben, die dort vom Daemon genutzt werden. Dies kommt im allgemeinen Fall einer unaufgeforderten, aperiodischen Reservierung dieser Slots gleich und schränkt die spätere Planung ein. Um dies zu vermeiden, darf der Server jeweils nur zu Beginn eines ungenutzten Zeitfensters die Länge der freien Nutzbarkeit übertragen und in dem angebrochenen Fenster keine neuen Reservierungen zulassen. Lange freie Fenster können durch entsprechend propagierte Endzeiten unterteilt werden, um die Konfigurierbarkeit des Systems zu erhalten.

Durch die Broadcastfähigkeit des Busses können diese Fenster allen Knoten mitgeteilt werden.<sup>17</sup> Auf diese Weise kann für die NRT-Kommunikation ein anderer MAC-Mechanismus als TDMA für die Zugriffskontrolle bzw. Kollisionsauflösung genutzt werden. Das hier zu wählende Protokoll und eine mögliche Einschränkung des Broadcasts auf ein Multicast an ausgewählte Knoten bieten ein weites Feld für Erweiterungen zur verbesserten Integration der NRT-Kommunikation in das System. Diese *Kanalüberlagerung* kann auch zur Steigerung der Planungsauslastung und für die SRT-Kommunikation eingesetzt werden, siehe 4.6.1 und 5.7. Im nächsten Kapitel ist eine entsprechende Erweiterung der TDMA-Planung einer der zentralen Punkte, der die flexible Einsatz- und Erweiterungsfähigkeit des Kommunikationssystems ermöglicht.

Die Übertragung von NRT-Daten ist auf diese Slots zu beschränken: der Versand von NRT-Daten ist eine Echtzeitaufgabe. Nach Bekanntgabe der freien Fenster muß die Übertragung angestoßen und für eine sofortige Unterbrechung am Ende des Fensters gesorgt werden, damit die globale Garantie anderer Kanäle nicht gefährdet wird. Auf diese Problematik geht Abschnitt 4.7 weiter ein.

#### 4.4.4 Kommunikationsthread

Um die Echtzeitaufgabe des Sendens bzw. Empfangs korrekt zu erfüllen, wird konzeptionell eine Task pro Kanal eingeplant (Kommunikationstask). Sie ist für einen ordnungsgemäßen Zugriff auf das Netz entsprechend der Spezifikation des Garantievertrags zuständig (Flußkontrolle) und integriert den Kommunikationsaufwand in die Echtzeitverarbeitung des Systems (Integration).

Auf der Senderseite findet die Flußkontrolle statt. Die zeitliche Spezifikation der Task leitet sich aus der globalen Garantie der Verbindungsplanung ab. Die

---

<sup>17</sup>...allen Knoten, die über einen nebenläufigen Empfang verfügen.

zeitgesteuerte Ausführung begrenzt beim direkten Senden die Netzzugriffe auf die reservierten TDMA-Zeiten und sichert die lokale Garantie; die Flußkontrolle findet über die CPU-Planung statt. Die Task übergibt die Daten sukzessive an die MAC-Schicht und überträgt sie somit an den Empfänger. Die Datenmenge läßt sich aus der Länge des Intervalls und der Bandbreite des Netzes errechnen. Beim nebenläufigen Senden muß die an die MAC-Schicht übergebene Datenmenge explizit kontrolliert werden. Es darf maximal die aus der Länge des Intervalls errechnete Datenmenge übertragen werden, um die lokale Garantie nicht zu verletzen.

Eine willkürliche Unterbrechung der Sendetask ist technisch nicht immer möglich, weil die Funktionen der Datenübertragung unter Umständen kritische Abschnitte durchlaufen, die ordnungsgemäß abzuschließen sind. Darüber hinaus führt die Unterbrechung einer laufenden Datenübertragung zu einem inkonsistenten Netzwerkzustand und einer fehlerhaften Nachricht. Aus diesem Grund darf die Übertragungsdauer der letzten von der Flußkontrolle zugelassenen Nachricht nicht über das Ende des TDMA-Intervalls hinaus reichen.

Eine (umfangreiche) Datenaufbereitung, die über eine Einteilung in die netztechnisch vorgegebene Nachrichtengröße hinausgeht, müßte in einer eigenen Task vor dem Beginn des TDMA-Intervalls liegen. Dieser Aufwand wird nicht betrachtet und muß in die Echtzeitaufgaben der Applikation verlagert werden oder neue Dienste erweitern den CT entsprechend (vgl. 4.9.2).

Eine Flußkontrolle der Menge der empfangenen Daten findet auf der Gegenseite nicht statt: Die maximale Datenmenge ist beim Kanalaufbau festgelegt worden und verändert sich nicht ohne eine Rekonfiguration des Kanals.

Die eingeplanten Tasks integrieren die Kommunikation in das Echtzeitsystem. Die Tasks sind in der CPU-Planung sichtbar und stellen eine Laufzeitrepräsentation des Kanals dar, die zur Planung weiterer, von der Kommunikation abhängiger Aufgaben genutzt werden kann. Der wichtigere Aspekt der Integration ist es, den BM-Bedarf der Kommunikation zu planen, um die anderen Echtzeitaufgaben nicht zu beeinflussen. Das Problem ist auf der Senderseite mit der eingeplanten Flußkontrolle bereits gelöst. Die Empfangstask plant beim direkten Empfangen den Kommunikationsaufwand ein. Die Task nimmt die ankommenden Daten zeitgerecht an und vermeidet einen Datenverlust.

Jedem Kanal ist genau ein Kommunikationsthread (CT) zugeordnet, der die Kommunikationstask ausführt. Dies kann unter Umständen für das Betriebssystem bei einer großen Zahl von Threads zu einem hohen Verwaltungsaufwand und zu häufigen Kontextwechseln führen. In solchen Fällen kann man mehrere Tasks von einem Kommunikationsthread ausführen lassen, wenn sichergestellt ist, daß eine (Zeit-)Fehlerfortpflanzung zwischen den Kanälen unterbunden wird; Abschnitt 4.7 erläutert dieses allgemeine Problem. Den Vorteil eines gesparten Kontextwechsels durch den Einsatz eines einzigen Threads erhält man nur in dem Fall, daß mehrere Kommunikationstasks *direkt* hintereinander ausgeführt werden. Deshalb wird die Zuordnung eines Threads zu einem Kanal beibehalten.

Bei der Planung der Tasks müssen die TDMA-Zeiten der zugrunde liegenden Verbindung aus einem Vorschlag des Servers bekannt sein, so daß eine vollständige Spezifikation der Tasks möglich ist. In die Spezifikation fließt die Hardwareabhängigkeit der Kommunikation ein. Neben der Unterscheidung nach nebenläufigem und direktem Senden/Empfang muß die Spezifikation an die Genauigkeit der Uhrensynchronisation  $\Delta GT$  angepaßt werden (s. 3.2.2). Streng genommen muß man das TDMA-Intervall  $[t_1, t_2]$  auf  $]t_1 + \Delta GT, t_2 - \Delta GT[$  einschränken, aber abhängig vom Netzwerk entfällt das Problem. Im folgenden sei  $\phi$  die Übertragungsdauer der kleinsten Nachricht. Man kann auf diese Einschränkung verzichten, wenn die MAC-Schicht einen CSMA-Protokoll unterstützt und  $\phi > \Delta GT$  ist. In vielen Fällen ist dies erreichbar, weil eine Uhrensynchronisation in der Größenordnung von Mikrosekunden in lokalen Netzen möglich ist [K<sup>+</sup>89, GS94].<sup>18</sup>

Der Kanalaufbau und die Leistungsverhandlung mit dem Server kann erst dann mit einem Commit positiv fortgesetzt werden, wenn der Kommunikationsthread erfolgreich eingeplant und initialisiert (Puffermanagement u.ä.) ist. Andernfalls kann das lokale Management den vorgeschlagenen Zeiten nicht zustimmen. Die Initialisierung abzuwarten ist sinnvoll, weil erstens ein Initialisierungsfehler die gesamte Kanalplanung verwirft und zweitens an dieser Stelle die Planung neuer Dienste leicht ansetzen kann.

Die Datenübertragung von der Applikation zum CT erfolgt über einen FIFO-Puffer (vgl. 1.3.1). Datenpakete variabler Größe werden in dem Puffer gesammelt und stehen dort für die Übertragung bereit. Die Daten brauchen nicht kopiert werden, wenn in den Puffer nur Referenzen auf die eigentlichen Daten abgelegt werden. Der notwendige Speicherbedarf ist mit dem Aufbau des Kanals anzumelden.

#### 4.4.5 Kanalaufbau

In Abbildung 4.5 beschreibt ein Petri-Netz den lokalen Teil des Kanalaufbaus. Die grau gefüllten Randstellen entsprechen denen des Netzes für den Verbindungsaufbau in Abbildung 4.2 und beide Teilnetze zusammen ergeben das gesamte Reservierungsprotokoll.

In den Stellen werden geordnete Paare von Knoten  $(y, z) \in \mathcal{K} \times \mathcal{K}$ , einzelne Kanal-IDs  $id \in \mathcal{I}$  oder die allgemeine Marke  $\bullet$  gespeichert. Die Stelle ClientID enthält eine Konstante  $x \in \mathcal{K}$ , die nie gelöscht wird und das Netz für den Knoten  $x$  spezialisiert: Der ‘Empfang’ der Marken über die Randstellen wird damit auf das Netz des korrekten Knoten/Empfängers beschränkt und der Inhalt der inneren Stellen (nicht-Randstellen) bezieht sich implizit auf diesen Knoten  $x$ . CPU-Reservierung entspricht der lokalen Planung der CPU und Netz-Reservierung ist die lokale Kopie (Auszug) der globalen TDMA-Reservierung, in der neben den etablierten Kanälen u.U. auch ein Kontingent für die lokale Planung gespeichert

---

<sup>18</sup>Für 10 MBit Ethernet gilt  $\phi = 51.2\mu s$  und für 1 MBit CAN-Bus gilt  $\phi = 54\mu s$ .

ist. K-ID enthält eine Teilmenge von  $\mathcal{I}$  als Kontingent-IDs, die dem Knoten  $x$  zur alleinigen Vergabe bereitstehen. Anforderung und Rückgabe des Kontingents wird hier nicht betrachtet.

Im folgenden wird von einer senderinitiierten Kanalanforderung ausgegangen, um die zwei beteiligten Knoten (Sender und Empfänger) zu unterscheiden. Dies stellt keine Einschränkung dar.

Den Kanalaufbau im lokalen Netzmanagement kann man in zwei Teile trennen. Zum einen stellt auf der Senderseite die Applikation eine interne Anfrage an den Daemon, der erste Schritt einer Applikation zum Aufbau eines Kanals. Zum anderen erreichen Verbindungsvorschläge des Servers als externe Anfragen über den Verwaltungskanal sowohl den Sender als auch den Empfänger.

#### 4.4.5.1 Interne Anfrage

Die interne Kanalanforderung wird in folgenden Schritten auf der Senderseite bearbeitet:

1. QoS-Umsetzung

Eine Applikation fordert mit einer Leistungsspezifikation (QoS) einen Kanal beim CD an. Daraus werden die erforderlichen TDMA-Parameter für die Verbindungsplanung ermittelt. Diese QoS-Umsetzung wird in 4.4.7 kurz dargestellt, aber die Problematik nicht weiter verfolgt, weil die einfache Applikationsschnittstelle eine direkte TDMA-Spezifikation benutzt.

QoS-Umsetzung legt eine Marke  $(y, z)$  in die Warteschlange K-Liste;  $y$  ist der eigene Knoten, der Sender, und  $z$  der Empfänger. Auf eine explizite Adressierung der Empfängerapplikation verzichtet die Darstellung und in der Realisierung sind zusätzliche Information darüber notwendig.

2. Kontingentplanung

Die aus der Leistungsspezifikation ermittelten TDMA-Parameter sind der Verbindungsplanung zu übergeben. Falls im Daemon ein Kontingent von Slots für lokale Planungen bereitsteht, wird zunächst hier versucht, eine Verbindungsreservierung durchzuführen und vorübergehend zu markieren (Kontingentplanung). Wenn dies gelingt, wird die Verbindungsanforderung mit diesem Vorschlag und einer eindeutigen ID an den Empfänger weitergeleitet, so daß die Planung im Server entfallen kann. Die ID wird aus K-ID gelöscht, um die Eindeutigkeit zu wahren.

Für die Weiterleitung können die jeweiligen Verwaltungskanäle über den Server benutzt werden. Damit hätte man den Planungsaufwand im Server verringert, aber nicht den Kommunikationsaufwand zwischen den Knoten. Um dies zu verbessern, muß der Empfänger externe Anfragen auch über eine andere Schnittstelle entgegennehmen können. Es ist entweder ein Kanal

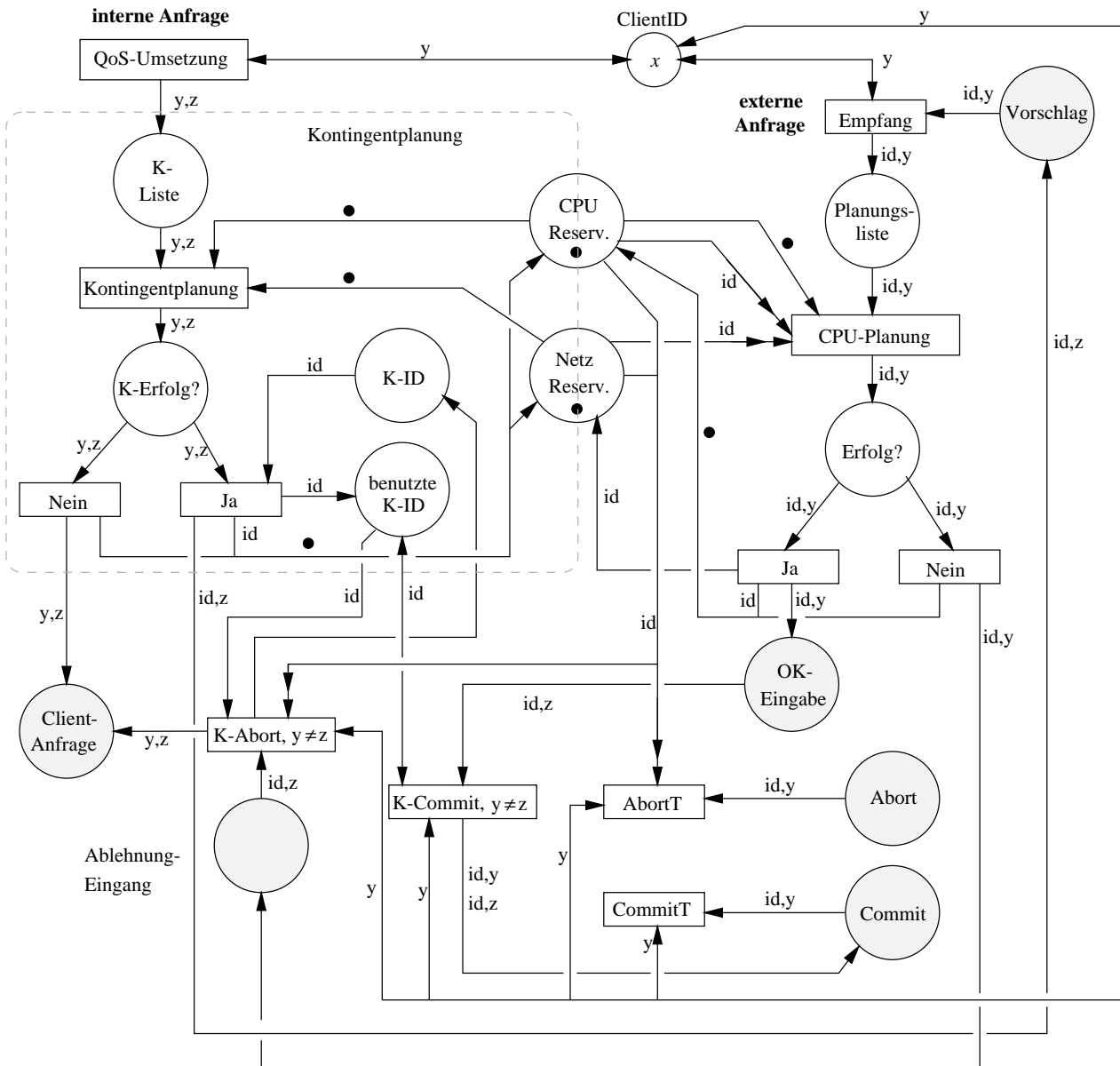


Abbildung 4.5: Petri-Netz des Kanalaufbaus für Client  $x$

zum Empfänger nötig, oder man verwendet verbindungslose Kommunikation (s.u.), um zusätzliche Verwaltungskanäle zwischen den einzelnen Knoten zu vermeiden. Im abgebildeten Netz ist diese Unterscheidung nicht sichtbar. Die Randstelle *Vorschlag* wird nicht nur für die Client→Server Kommunikation, sondern auch für Client→Client benutzt.

Auf demselben Weg nutzt der Initiator, wie der Server, die Stellen *OK-Eingabe* und *Ablehnung-Eingang* als Eingabe, um die Antwort des Empfängers zu erhalten. Eine positive Antwort läßt die Transition *K-Commit* schalten und zwei Marken in *Commit* einfügen. Im Petri-Netz gleicht dieser Abschluß der Kanalplanung der serverbasierten Planung; in der Realisierung wird nur eine Nachricht tatsächlich übers Netz versandt. Die ID bleibt bei der Kontingentsplanung in *benutzte K-ID* erhalten. Bei einer negativen Antwort entfernt eine *Abräumkante*<sup>19</sup> (Doppelpfeil) alle vorläufigen Einträge in der CPU- und der Kontingentsreservierung des verworfenen Planungsversuchs und gibt die ID wieder ins Kontingent zurück. Das Protokoll fährt im nächsten Punkt wie im kontingentfreien Fall fort.

### 3. Server-Anfrage

Wenn man lokal keine freien Slots findet, wird die Verbindungsanforderung über *Client-Anfrage* zur Planung an den Server geschickt. Die Marke  $(y, z)$  enthält den Planungsauftrag mit den erforderlichen TDMA-Parametern.

Das weitere Vorgehen liegt zunächst im Bereich des globalen Netzmanagements, bis von dort Vorschläge an den Sender und Empfänger als externe Anfragen ausgehen.

#### 4.4.5.2 Externe Anfrage

Eine externe (Reservierungs-)Anfrage wird vom Server über den Verwaltungskanal oder im Falle einer Kontingentsplanung unter Umgehung des Servers über eine weitere Kommunikationsschnittstelle (extra Verwaltungskanal oder verbindungslose Kommunikation) an den Kommunikationsdaemon versandt (im Netz in der Stelle *Vorschlag* verborgen, „Vergrößerung“). Dies ist der erste Schritt in der Kanalplanung auf der Empfängerseite und der zweite auf der Senderseite, der die interne Anfrage fortsetzt (s.o.).

### 1. CT-Planung

Die Leistungsverhandlung zwischen der MAC- und der Transportschicht beginnt mit einer externen Anfrage über *Vorschlag*. Anhand der vorgeschlagenen, vorläufigen TDMA-Zeiten der zugrunde liegenden Verbindung muß

---

<sup>19</sup>Eine *Abräumkante* entscheidet nicht, ob eine Transition aktiv ist, sondern löscht beim Schalten die spezifizierten Marken der Eingangsstelle.

der Kommunikationsthread geplant werden (**CPU-Planung**). Eine Abräumkante löscht eine evtl. vorhandene vorläufige Reservierung zu *id*, die zu einem Vorschlag der vorangegangenen Planungsrunde aufgebaut wurde und mit dem neuen Vorschlag überflüssig wird, aus **CPU-Reservierung** bzw. **Netz-Reservierung**.

Die CPU-Anforderungen des CT werden aus dem geforderten QoS bestimmt. Im einfachsten Fall des direkten Sendens und Empfangens ohne Vorverarbeitung der Daten sind sowohl für die Sende- als auch für die Empfangstask die vorgeschlagenen TDMA-Zeiten auf der CPU zu reservieren.

#### 2. Ergebnis zurückmelden

Mit einer erfolgreichen CT-Planung wird die Annahme des Vorschlags über **OK-Eingabe** an den Server zurückgemeldet und im nächsten Punkt auf den Abschluß oder eine neue Runde der Planung gewartet. Wenn die CT-Planung keinen Erfolg hat, wird eine Ablehnung des Vorschlags an den Server geschickt (**Ablehnung-Eingang**).

#### 3. Ende der Planung

Ein von allen akzeptierter Vorschlag wird vom Server mit einem Commit bestätigt und damit ist der Kanal etabliert (**Commit**). Mit einer Bestätigung der bisherigen Planung wird der CT gestartet und eine Datenstruktur als Referenz auf den Kanal an die Applikation zurückgegeben (**CommitT**).

Andernfalls erhalten die Kommunikationsteilnehmer eine **Abort** Meldung, um Reservierungen aufzuheben und um insbesondere die Applikation über den Planungsmißerfolg zu benachrichtigen (**AbortT**).

Im Petri-Netz ist die Bearbeitung der externen Anfrage einer Server- und einer Kontingentsplanung identisch. Die Unterschiede der Verwaltungskommunikation sind in den Randstellen verborgen und in der Abbildung nicht sichtbar.

### 4.4.6 Kanalnutzung – das Kanalobjekt

Kanäle werden in einem objekt-orientierten Entwurf durch eine Klasse<sup>20</sup> beschrieben. Der Applikation steht ein instanziiertes Objekt dieser Klasse pro Kanal für die Kommunikation zur Verfügung: das Kanalobjekt.

Das Kanalobjekt besteht aus dem notwendigen Pufferspeicher, dem Kommunikationsthread und der Task des Sendens/Empfangens (Abb. 4.6). Die Kanäle erstrecken sich in ihrer Funktionalität über mehrere Schichten. Der Pufferspeicher ist auch ein Teil der Applikation und wird von dort gefüllt bzw. geleert. Der Kommunikationsthread und die -tasks steuern in der Transportschicht die Vermittlung der eingehenden Daten zur Applikation und den Zugang der Daten zum

---

<sup>20</sup>C++-Notation

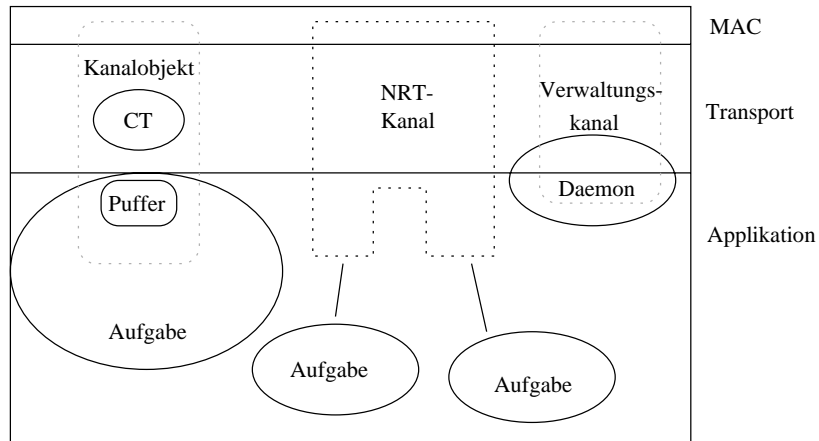


Abbildung 4.6: Die Kanalnutzung über das Kanalobjekt.

Netz innerhalb eines Knotens. Somit ist durch den zeitgesteuerten Zugriff das Objekt auch ein Teil der MAC-Schicht, obwohl in einer Realisierung die Nutzung von Standardkomponenten und -treibern eine direkte Programmierung auf dieser Ebene ausschließt.

Der Verwaltungskanal ist ein Kanal, über den die Verwaltungsinformationen übertragen werden. Die NRT-Kommunikation erfordert ein anderes Konzept für die interne Realisierung des Kanalobjekts, nach außen bleibt aber die gleiche Darstellung erhalten, auf die die Applikation zugreift (s. 4.4.3.2).

Die Instanziierung erfolgt bei der vorläufigen Planung und entscheidet mit über deren Erfolg. Somit bietet sich die Möglichkeit, einen detaillierteren Planbarkeitstest als die reine CPU-Nutzung oder einen neuen Dienst über die Ableitung der Klasse flexibel in das System einzubinden.

Nicht-blockierende Methoden für das Senden bzw. Empfangen übernehmen den Transport der Daten von der Applikation zum Kommunikationssystem und umgekehrt. Dabei werden die Daten in den objektinternen Puffer geschrieben und spezifikationsgemäß durch den CT versandt. Die Tasks und deren Parametrisierung sind sichtbar und somit für die Applikationsplanung nutzbar.

#### 4.4.7 QoS-Umsetzung

Die Schnittstelle der lokalen Kanalverwaltung zur Applikation liegt oberhalb der Transportschicht. Auf die nötige Leistungsverhandlung und Umsetzung der applikationsseitigen QoS-Parameter wird in diesem Abschnitt kurz eingegangen. Im übrigen ist diese Schnittstelle nicht Gegenstand der Arbeit, es findet keine Verhandlung mit höheren Schichten statt und ich gehe von einer reinen TDMA-Spezifikation aus (siehe 5.4).

Aus der applikationsseitigen QoS-Spezifikation der Echtzeitkommunikation wird die Spezifikation der TDMA-Slots der zugrunde liegenden Verbindung und

der Sende- bzw. Empfangstask erstellt. Erweiterungen einer reinen TDMA-Spezifikation durch die Umsetzung der QoS-Parameter können z.B. einen Kontrollkanal für Metadaten oder eine zeitlich redundante Übertragung zur Fehlertoleranz umfassen; eine Auflistung von QoS-Parametern folgt im Kapitel 'Planung', 5.5.1. Zur Umsetzung der Parameter in eine TDMA-Spezifikation kann in diesen Beispielen der Durchsatz vergrößert werden, um die durchschnittliche Datenmenge des Kontrollflusses bzw. die redundanten Informationen neben der eigentlichen Datenübertragung aufzunehmen. Gleichzeitig muß man den Kommunikationsthread anpassen, damit der Durchsatz entsprechend genutzt wird. An dieser Stelle kann auch eine Zeitfehlerbehandlung zur Unterstützung harter oder weicher Zeitbedingungen nachträglich in das Konzept integriert werden.

Die reine Umsetzung der Parameter reicht aber nicht aus.<sup>21</sup> In der Planung müssen diese QoS-Parameter in die Leistungsverhandlung eingehen, die in einem kompletten System zwischen *allen* Schichten des OSI-Modells stattfindet. Die Abstimmung zwischen Server und Daemon über die Parameter der Verbindung und des Kommunikationsthreads muß in analoger Weise auch in den höheren Schichten fortgesetzt werden. Insbesondere der Aspekt der Kanalplanung ist nur im ganzen zu lösen, da auch die Abhängigkeiten der Applikation einzubeziehen sind; siehe zum Beispiel [NS95, CCH94, MIS96a].

Eine Absprache mit den höheren Schichten entfällt, wenn keine Informationen über die bei der Planung erreichte Konkretisierung der Parameter *innerhalb* der Spezifikation aus dem Kommunikationssystem nach oben gegeben wird. Bei der TDMA-Planung folgen daraus aber unnötige Verzögerungen (mangelnde Tightness), weil die genauen Zeiten nicht mehr sichtbar sind. Das vorliegende Konzept unterstützt diese Absprache beim Kanalaufbau, indem die erfolgreiche Initialisierung des Kommunikationsthreads, die sich über mehrere Schichten erstrecken kann, in der Planung abgewartet wird. Weitere Absprachen werden nicht unterstützt; dies resultiert aus der fehlenden Rekonfigurationsmöglichkeit.

## 4.5 Gruppenkommunikation

In diesem Abschnitt werden die Broadcasteigenschaften des Busses in das Konzept integriert und Kanäle mit mehreren Teilnehmern untersucht; weitere Aspekte wie Gruppenmitgliedschaft, Nachrichtenreihenfolge und atomarer Broadcast werden nicht betrachtet. Nach der Beibehaltung der Simplexkommunikation werden zwei Punkte unterschieden.

---

<sup>21</sup>In 5.5.5 wird die Umsetzung einzelner Parameter ergänzend zur TDMA-Spezifikation konkretisiert, aber sie wird nicht in die Planung einbezogen.

### 4.5.1 Broadcastkanal

Bei einem *Broadcastkanal* wird beim Kanalaufbau nur der Sender ohne Berücksichtigung eines Empfängers geplant. Der Sender plant die Kommunikation „ins Leere“ und der Empfänger plant die Teilnahme an dem Kanal später. Der empfängerinitiierte Kanalaufbau übernimmt für die lokale Planung die feststehenden Parameter des etablierten Broadcastkanals. Bei Konflikten in der lokalen Planung findet keine Rekonfiguration des Kanals statt; die Parameter können nur unverändert ohne Leistungsverhandlung übernommen werden.

Dies ist ein sehr einfacher Mechanismus, dessen Erfolg von der Planbarkeit der lokalen Knoten abhängt. Zusammen mit einer Kontingentplanung des Senders ist ein Broadcastkanal ohne den zentralen Server aufbaubar.

Der empfängerinitiierte Abbau von Broadcastkanälen muß nur die lokalen Reservierungen aufheben. Aus dem Abbruch des Senders resultiert wie bisher der Abbau aller betroffenen Reservierungen der beteiligten Knoten.

### 4.5.2 Aktivität

Die bisherige Einschränkung auf die Simplexkommunikation ist für ein (einsegmentiges) Broadcastnetz nicht notwendig. Zu den reservierten Zugriffszeiten der Verbindung muß nicht unbedingt *die* Sendetask und *die* Empfangstask geplant werden, wenn man die Zugriffsrechte *innerhalb* der Verbindung anders zwischen dem ursprünglichen Sender und Empfänger aufteilen möchte. Im folgenden wird die *Aktivität* als planbare Einheit des Kommunikationssystems eingeführt (vgl. 5.2), die eine Verallgemeinerung des Kanalbegriffs ist.

Die TDMA-Reservierung einer Aktivität grenzt sie in gewohnter Weise gegen andere Kanäle bzw. Aktivitäten ab. Die Gruppe von Tasks und CTs der Kommunikationsteilnehmer, die in einer Aktivität zusammengefaßt sind, nutzen die für sie reservierten Verbindungszeiten. In der Verantwortung der Gruppe liegt es, mit Hilfe einer Absprache die zugestandenen Zugriffszeiten kollisionsfrei zu nutzen. Dafür bietet das Kommunikationssystem keine Unterstützung an. Es schränkt lediglich die Kommunikation auf die äußeren zeitlichen Grenzen der Aktivität ein und erfüllt somit die lokale Garantie gegenüber dem Kommunikationssystem.

Zum Beispiel ist bisher eine bidirektionale Kommunikation nur durch eine Kombination von zwei unidirektionalen Kanälen möglich. Es spricht aber nichts dagegen, eine Verbindung in der ersten Hälfte der erteilten Slots für eine Kommunikation  $A \rightsquigarrow B$  und anschließend für  $B \rightsquigarrow A$  zu nutzen, wenn dieser Unterteilung beide Seiten folgen. Ein solches RPC-ähnliches Kommunikationsmuster kann als eine Aktivität geplant werden: Auf jeder Seite ist die Sendetask auch Empfangstask, und umgekehrt. Der konkrete Verlauf der Kommunikation braucht nach außen nicht sichtbar zu sein.

## 4.6 Weitere Zugriffsprotokolle

### 4.6.1 Kanalüberlagerung

Die in 4.5.2 vorgestellten *Aktivitäten* beruhen auf einer gemeinsamen Spezifikation, innerhalb der sich die Teilnehmer die BM-Zugriffsrechte eigenständig aufteilen. Dieses Konzept kann man verallgemeinern, wenn man die (implizite) Aufteilung von verschiedenen, einzeln geplanten Aufgaben zuläßt und in einem TDMA-Intervall mehrere Aufgaben „gleichzeitig“ das Zugriffsrecht besitzen können: die Kanäle sind überlagert. Die Idee ist wesentlich für die Lösung der Probleme, die die TDMA-Planung aufwirft (siehe 5.7.1), und bedarf eines über die zeitlichen Parameter erweiterten Planungsmodells.<sup>22</sup>

Zunächst beschränke ich die Erläuterungen auf die Vergabe von gleichzeitigen Zugriffsrechten in einem Knoten. Das zeitgesteuerte Zugriffsrecht des Knotens kann man über eine implizite Planung der Kommunikationsthreads an die entsprechenden Kanäle verteilen. Der verplante und ungenutzte Overhead im Netzwerkzugriff kann dadurch verkleinert werden. In der Praxis durchläuft nämlich ein einzelner Sender pro TDMA-Slot alle Verzögerungen, die das Betriebssystem respektive Treiber, bietet und kann somit keine 100%ige Auslastung des Netzes erreichen. Verschachtelte, implizit durch die Verzögerungen gesteuerte Zugriffe mehrerer Sender können dieses Problem (teilweise) umgehen, indem sie eine ununterbrochene Übertragung der Daten aus den Applikationen an den Netzcontroller sichern; das Problem und die vorgeschlagene Lösung hängen wesentlich von den eingesetzten Komponenten ab und müssen in jeder Implementierung extra untersucht werden!

Ein implizites (ratenbasiertes) Planungsverfahren muß innerhalb der expliziten Planung den Zugriff auf den Netzcontroller steuern und die Vorhersagbarkeit der Ausführung garantieren. Das gilt für jede Form der Kanalüberlagerung und muß zusammen mit den Möglichkeiten des generischen Planungsmodells (Abschnitt 5.7) untersucht werden. Aus Kenntnis der Verzögerungen im Betriebssystem und im Netzcontroller, die für einfache Modelle empirisch bestimmbar sind, ist eine Kanalüberlagerung in einem Rechner leicht zu planen.

Für die Überlagerung unterschiedlicher Sendeknoten und die damit verbundene implizite<sup>23</sup> Vergabe des Zugriffsrechts auf das Netz muß man ratenbasierte Zugriffsverfahren, die *innerhalb* des der Aktivität zugeteilten TDMA-Intervalls auf dem Netz genutzt werden, in die Planung einbeziehen. Wie in Abschnitt 2.1.1.3 erläutert, können auch harte Zeitbedingungen eingehalten werden; aber in erster Linie ist eine Kommunikation mit weichen Echtzeitbedingungen *innerhalb* des

---

<sup>22</sup>Im nächsten Kapitel werde ich ab Seite 127 ausführlich auf das Problem eingehen. Die Kenntnis der dortigen Erklärungen sind hier teilweise notwendig, weil hier nur die Problemlösung und nicht die -stellung besprochen wird.

<sup>23</sup>Mit der impliziten Vergabe des Zugriffsrechts ist hier die Vergabe ohne explizite Kontrolle durch den Kommunikationsserver gemeint.

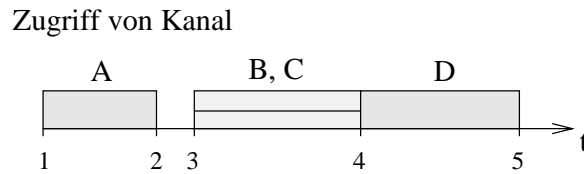


Abbildung 4.7: Kanalüberlagerung und alternative Zugriffsverfahren

überlagerten TDMA-Intervalls möglich; vgl. Ziele Z3 und Z4 der Einleitung.

## 4.6.2 SRT-Kommunikation

Die im letzten Abschnitt eingeführte Kanalüberlagerung bietet eine Möglichkeit, die SRT-Kommunikation über zusätzliche Zugriffsprotokolle in das Kommunikationssystem zu integrieren, ohne die Vorhersagbarkeit der anderen reservierten TDMA-Intervalle einzuschränken. Die Abgrenzung ist in Abbildung 4.7 skizziert. Die TDMA-Intervalle  $[t_1, t_2]$ ,  $[t_3, t_4]$  und  $[t_4, t_5]$  werden wie bisher vom Server geplant, aber der Zugriff erfolgt in  $[t_3, t_4]$  von zwei Kanälen  $B$  und  $C$ . Der Netzzugriff innerhalb des überlagerten TDMA-Intervalls geschieht mit einem anderen, nicht-zeitgesteuerten Protokoll, das spätestens zu  $t_4$  abgebrochen werden muß und das die Grundlage für die weitere Planung bildet.

Ein ratenbasiertes Zugriffsverfahren kann zum Beispiel eingesetzt werden, um mit statistischem Multiplexing stark variierende Datenströme unter weichen Echtzeitbedingungen zu übertragen (s. 5.3.3). Die Planung der CPU und die Inhalte des Reservierungsprotokolls müssen natürlich den Anforderungen angepaßt werden.

Die CPU-Planung bietet einen weiteren Ansatz, weiche Zeitbedingungen in das System zu integrieren; z.B. [MIS96b]. Neben der erwähnten Veränderung des Zugriffsprotokolls kann die Planung eines unterbrechbaren Kommunikationsthreads, der die Flußkontrolle und den Zugriff auf den NIC steuert, die Kommunikation weiter flexibilisieren. Eine prioritätenbasierte Ausführung kann verschiedene Wichtigkeits- oder Dringlichkeitsstufen in die lokale Kanalüberlagerung und zusammen mit einem entsprechenden Zugriffsprotokoll (siehe z.B. adreß-/prioritätsbasierte Auflösung auf Seite 40) auch in die globalen Überlagerung verschiedener Sendeknoten einführen.

## 4.7 Fehlertoleranz

Dieser Abschnitt geht auf drei mögliche Fehler in der Kommunikation im verteilten Echtzeitsystem ein: Übertragungsfehler, Zeitfehler in der Verarbeitung von Kommunikationstasks und der Ausfall einzelner Knoten des Systems.

### 4.7.1 Übertragungsfehler

Der Begriff 'Übertragungsfehler' wird als Synonym für die Bitfehlerwahrscheinlichkeit, mit der einzelne Bits einer Nachricht kippen, genutzt.<sup>24</sup> Wenn nicht bereits aus der verwendeten Technologie für die Datenübertragung vorhanden, kann das Kommunikationssystem durch automatisch generierte Prüfsummen (CRC) versuchen, solche Fehler zu erkennen. Ist die Korrektur anhand der Prüfsumme nicht möglich, kann eine Fehlermeldung an die Applikation weitergegeben werden. Erkennt das Kommunikationssystem den Fehler nicht, liegt die Fehlerbehandlung im Aufgabenbereich der Applikation.

Prüfsummen und die Übertragung redundanter Informationen, die der Fehlererkennung und -behebung dienen, reduzieren die effektive Bandbreite und die Planungsauslastung, aber ermöglichen dadurch eine Fehlerbehandlung.

### 4.7.2 Zeitfehler

#### 4.7.2.1 Unterbrechung der Sendetask

In der Realisierung geht der CT von einer festen Datenmenge pro Intervall aus, die sukzessive an die MAC-Schicht zur Übertragung gegeben wird. Temporäre Fehler in der Übertragung verzögern die Sendetask und die Datenmenge kann nicht komplett *innerhalb* des TDMA-Intervalls übertragen werden. In der Sendetask kann ein Zeitfehler entstehen, weil sie nicht jederzeit unterbrechbar ist.

Übertragungsfehler, die aus der Unterbrechung einer laufenden Datenübertragung resultieren, muß man in diesem Fall akzeptieren. Aber kritische Abschnitte der Datenübertragung kann man nicht unterbrechen, zumindest nicht ohne einen inkonsistenten Zustand den anderen Kanälen zu überlassen, der die nachfolgende Datenübertragung stört. Der Zeitfehler kann sich auf Tasks im gleichen Knoten fortpflanzen und erst an anderer, scheinbar unabhängiger Stelle auftreten (Dominoeffekt). Letztlich ist die lokale und globale Garantie aller Kanäle gefährdet.

Die Lösung des Problems ist dessen Einschränkung: Die unzuverlässige Kommunikation akzeptiert die Verletzung der globalen Garantie wegen einer fehlerhaften Übertragung in dem betroffenen Kanal.<sup>25</sup> Es werden so viele Daten wie möglich übertragen, aber es dürfen auf keinen Fall die lokale Garantie verletzt und andere Kanäle beeinflusst werden. Das Problem der Zeitfehler reduziert sich damit auf die Vermeidung einer Zeitfehlerfortpflanzung und des Einflusses auf nicht betroffene Kanäle.

Der rechtzeitige Abbruch der Tasks und der Datenübertragung muß die Einhaltung der lokalen Garantie erzwingen. Dies geschieht am besten direkt beim Netzzugriff in der MAC-Schicht, aber in der Praxis verhindert der Einsatz von

---

<sup>24</sup>Verlust oder Duplikation von Paketen zu erkennen und darauf zu reagieren liegt in der Verantwortung der Applikation.

<sup>25</sup>Redundante Informationsübertragung kann die mittlere Bitfehlerwahrscheinlichkeit auf Kosten der Planungsauslastung beliebig reduzieren.

Standardkomponenten und -treibern, die keinen Eingriff innerhalb dieser Schicht erlauben, die zeitliche Kontrolle einzelner Byteübertragungen und macht einen Ansatz in der Transportschicht notwendig. Für die Sendetask muß eine WCET bestimmt bzw. durchgesetzt werden, die Zeitfehler aufgrund von Übertragungsfehlern verhindert.

#### 4.7.2.2 TaskPair im Kommunikationsthread

Die **TaskPair**-Konstruktion (TP) hilft, die Schwierigkeit der WCET-Bestimmung und -Einhaltung in den Griff zu bekommen [Str95]. Man kann die Sendetask, deren WCET man bisher allein aus der Datenmenge und der Bandbreite ohne Betrachtung von Übertragungsfehlern ermittelte, mit einer WCET im Fehlerfall versehen, indem man den rechtzeitigen Abbruch *vor* einem Zeitfehler mit einplant. Die Übertragung der kompletten Datenmenge in diesem Intervall ist dann nicht mehr gesichert. Das entspricht der obigen Aussage über die globale Garantie einer unzuverlässigen Kommunikation.

**Definition** Ein TP besteht aus einer Task MT (MainTask) mit beliebigen zeitlichen Spezifikationen und einer harten Task ET (ExceptTask), die zusammen eine Einheit mit einer festen maximalen Laufzeit bilden. Ein unterstützendes Laufzeitsystem gewährt der MT entsprechend ihrer Spezifikation Rechenzeit. Ist die MT beendet, ist das TP beendet. Andernfalls wird die MT *vor* Verbrauch ihrer gemeinsamen WCET so abgebrochen, daß die ET ihre vereinbarte WCET-Rechenzeit erhält und das TaskPair mit einer vorhersagbaren maximalen Ausführungszeit beendet wird.

**Einsatz** Im Kommunikationsthread ist die Sendetask die MainTask eines TP. Sie überträgt nach FCFS alle Daten des Kanalpuffers. Die ExceptTask unterbricht sie vor dem Überschreiten des TDMA-Intervalls. Die ExceptTask ET kann für ein Rücksetzen der Werte, die von MT unvollständig bearbeitet wurden, für die Behebung eines Fehlers oder für das Liefern eines Default-Wertes benutzt werden. Das TaskPair kann für eine Integration von Tasks mit unbekannter WCET in eine harte Echtzeitumgebung benutzt werden. Nach außen präsentiert sich ein TaskPair wie eine harte Task, die eine vorgegebene Aufgabe sicher erfüllt.

#### 4.7.3 Knotenausfall

Der Ausfall einzelner Knoten kann temporär oder dauerhaft geschehen. Ein temporärer Ausfall ist wegen der in der Planung festgelegten Übertragungszeiten der globalen Garantie für die restlichen Knoten ohne Bedeutung (von den betroffenen Applikationen abgesehen). Ein längerer Ausfall unterbricht die Uhrensynchronisation und beim Wiederanlauf ist die Einhaltung der lokale Garantie nicht mehr

gewährleistet. Aus diesem Grund sollte man von einem *fail-silent*-Verhalten ausgehen: Im Fehlerfall trennt sich ein Knoten für immer vom Netz.

Die Auswirkungen auf den Kanalaufbau sind bereits in 4.3.2.4 auf Seite 73 besprochen worden; aber das Problem der zugeteilten und nach dem Ausfall ungenutzten Reservierungen ist noch offen. Einerseits können die Knoten dem Server durch periodische Nachrichten ihre fehlerfreie Funktionalität beweisen, weil im fail-silent-Modell sonst keine Nachrichten mehr von ihnen kommen würden. Das Kommunikationssystem erkennt einen Ausfall und reagiert mit dem Abbau der entsprechenden Verbindungen. Andererseits können in komplexen Systemen einzelne Applikationen als Nutzer der Kanäle, auch ohne einen Ausfall des gesamten Knotens, ausfallen.

Man kann zur Erkennung dieser Ausfälle die periodischen Nachrichten auf einzelne Kanäle ausdehnen. Einen anderen Ansatz bietet die Idee, das Nutzungsrecht und die -dauer über *Tickets* einzuschränken (vgl. Kerberos [SNS88, NT94]). Für eine Kanalreservierung wird ein Ticket mit einer begrenzten Lebensdauer vergeben, das man regelmäßig über den Server verlängern muß. Es ist die Aufgabe des lokalen Managements, die Kanäle zu überprüfen und die Verlängerung zu beantragen. Falls ein Ticket nicht explizit verlängert wird, wird der Kanal vom Server abgebaut.

## 4.8 Verbindungslose Kommunikation

Das Problem der verbindungslosen NRT-Kommunikation kann ähnlich dem verbindungsorientiertem Fall bearbeitet werden (siehe 4.4.3.2): Es muß lediglich jeder Nachricht die Sender- und Empfängeradresse vorangestellt werden. In expliziten NRT-TDMA-Intervallen, extra reserviert oder als solche vom Server freigegeben, findet in der Regel sowieso ein Multiplexing mehrerer NRT-Kanäle statt, so daß hier nur kleine Änderungen vorzunehmen sind. Wenn zusätzlich ungenutzte, aber reservierte Intervalle etablierter Kanäle genutzt werden, verteilt sich die Datenübertragung auf verschiedene Reservierungen und kommt auch bisher einer verbindungslosen Kommunikation gleich. Insgesamt ist die fehlende eindeutige Kanal-ID, die den Sender und den Empfänger beschreibt, der einzige Unterschied zur verbindungsorientierten NRT-Kommunikation.

Die verbindungslose Kommunikation unter Echtzeitbedingungen versendet einzelne Nachrichten unter zeitlichen Bedingungen: RT-Datagramme. Freie Zugriffszeiten können wiederum wie in der NRT-Kommunikation ermittelt werden. Die SRT-Kommunikation kann in diesen Zeiten leicht über ein alternatives Protokoll auf das Netz zugreifen; siehe Abschnitt 2.1.1.1 „absprechende Zugriffsverfahren“. Innerhalb eines Knotens kann der Verkehr über (Prioritäts-)Warteschlangen gesteuert werden. Beide Ansätze sind Beispiele einer Kanalüberlagerung.

Für die verbindungslose HRT-Kommunikation sind grundsätzlich eigene Kanäle einzuplanen (vgl. „periodic server“ [MZ95], „deferrable server“ [SM89]), an

denen im Sender eine Liste mit ausstehenden Datagrammen hängt  $(d_i)_{i \in \{1, \dots, n\}}$ . Die Planung fügt ein Datagramm  $D$  an der Stelle  $j$  so in die Liste ein, daß die Übertragungszeiten der  $(d_i)_{i < j}$  eine zeitfehlerfreie Übertragung von  $D$  und  $(d_i)_{i > j}$  erlauben (siehe auch EDF-Planung in 5.6.4.2). Die Planung der reinen Datenübertragung einer HRT-Kommunikation ist damit leicht möglich.

Die Datenverarbeitung der Gegenseite muß mit eingeplant werden, um harte Zeitbedingungen der Kommunikation 100%ig einhalten zu können. Zwischen jedem Paar verbindungslos kommunizierender Knoten muß ein Kanal die jeweiligen „periodic server“ verbinden. Darüber hinaus muß in den meisten Fällen auch sichergestellt sein, daß die Applikation die Daten rechtzeitig verarbeiten kann. Dies macht eine verteilte Planung einzelner Datagramme notwendig und ist aufgrund des hohen Aufwands nur in seltenen Situationen sinnvoll. Nur wenn auf der Empfängerseite konstruktionsbedingt eine korrekte Verarbeitung sichergestellt ist, kann auf die Planung der Gegenseite verzichtet werden. Aber dafür ist eine statische Analyse der Kommunikationsteilnehmer erforderlich.

## 4.9 Erweiterungsmöglichkeiten

### 4.9.1 Planung komplexer Netze

Eine steigenden Anzahl von Knoten und sehr unterschiedliche QoS-Anforderungen in Teilen des Netzes machen eine Unterteilung der Netzplanung sinnvoll. Man kann die Planung in einem einzigen Segment auf mehrere Server oder die Netz- und Planungslast in mehrere Netzsegmente aufteilen.

#### 4.9.1.1 Verteilung der Reservierung

In einem Netzsegment, in dem sehr viele kleine, leistungsschwache Knoten und einige leistungsstarke Knoten mit einem hohen variablen Kommunikationsbedarf existieren, ist der Server einer großen differenzierten Planungslast ausgesetzt. Viele kleine Anfragen sind zu bewältigen, die die Reaktivität des Servers erheblich einschränken.

Der Einsatz mehrerer Server, die jeweils die Kommunikation einiger Knoten des Netzes planen, verteilt die Last. Die Server verplanen getrennte TDMA-Bereiche und können somit im wesentlichen autonom arbeiten. Die dynamische Planung erfordert auf größeren Zeitskalen eine Absprache zwischen den Servern, um die verwalteten Bereiche den Anforderungen anzupassen. Hierzu sind Mechanismen ähnlich der dezentralen Reservierung in 4.3.2.3 einsetzbar. Man verbindet die beiden unterschiedlichen Planungsansätze miteinander und kann sowohl den Einsatz leistungsschwacher Knoten unterstützen als auch die Vorteile einer dezentralen Reservierung nutzen.

Die Knoten lassen sich funktionell gruppieren und einzelnen Servern zuordnen, so daß die Planungslast auf die jeweiligen Verwaltungsbereiche beschränkt bleibt.

Ein Server und eine Menge leistungsschwacher Knoten präsentieren sich so nach außen als eine Einheit. Die strenge zeitliche Trennung der TDMA-Bereiche beibehaltend, können über die getrennten Server unterschiedliche MAC-Protokolle eingesetzt werden, die den gruppierten Funktionen unter Umständen besser entsprechen als die TDMA-Planung.

#### 4.9.1.2 Weitere Topologien

Mit den Überlegungen des letzten Abschnitts gelangt man direkt zur weiteren Strukturierung des physikalischen Netzes. Der Einsatz mehrerer, physikalisch getrennter Netzsegmente unterteilt nicht nur die Planung, sondern auch die Datenübertragung in unterschiedliche Verantwortungsbereiche. Brücken oder Router müssen die Segmente verbinden. Die in 2.2 dargestellten Arbeiten bearbeiten teilweise das Problem der kombinierten Planung von CPU und *zwei* Netzzugriffen.

Die Verbindungsorientierung der Kommunikation ist eine Voraussetzung für die Planung einer „Segmentüberschreitung“. Die Planung einer Kommunikation aus Segment  $A$  nach Segment  $B$  erfolgt im verbindenden Rechner (Brücke) über zwei Kanäle, von denen einer nach  $B$  sendet und einer aus  $A$  empfängt. Eine Task  $T$  und die zwei Kanäle müssen so eingeplant werden ( $A \rightarrow T \rightarrow B$ ), daß sie die Daten aus  $A$  (dem Puffer des entsprechenden Kanals) nach  $B$  unter Einhaltung der maximalen Verzögerung (Delay) vermitteln kann (siehe auch 5.5.1). Die Forderung kann durch Abhängigkeiten in der CPU-Planung modelliert werden.

Die Vermittlung ist in ihrer Leistung sehr stark eingeschränkt, wenn nur die direkten Kommunikationsprimitiven (Senden/Empfangen) zur Verfügung stehen. Die planbar gleichzeitige Kommunikation mit beiden Segmenten ist dann in der Brücke nicht möglich.

#### 4.9.2 Neue Dienste

Die automatisierte Einplanung des Kommunikationsthreads kann man ergänzen, um weitere Dienste anzubieten, die etwa eine Datenvorverarbeitung, -verschlüsselung oder Redundanz (Fehlererkennung, -toleranz) in die Kommunikation einführen. Eine CT-Anpassung – der Thread kann mehrere Tasks abarbeiten – sorgt dafür, daß der zusätzlichen Aufwand eingeplant ist. Die Klasse der Kanalobjekte kann durch eine Ableitung um neue Dienste einfach erweitert werden.

Man kann die automatische Planung auch ganz unterbinden und es der Applikation überlassen, die verfügbaren TDMA-Zeiten korrekt zu nutzen. Bei einer engen Verzahnung von Kommunikation und Applikation kann man damit zusätzlichen Aufwand durch Kontextwechsel vermeiden. Aber die Planungsauslastung sinkt, wenn das Zugriffsrecht nicht ununterbrochen genutzt wird; es sei denn der Kanal ist überlagert...



# Kapitel 5

## Planung

In diesem Kapitel gehe ich auf die Planung der Netzzugriffe ein. Es wird erläutert, *für wen, was, wie* geplant wird: Aktivitäten, Betriebsmittel (BM), Planungsalgorithmen – Echtzeitplanung. Dabei werde ich die Planung der Netzzugriffe in einen generischen, zeitgesteuerten Ansatz einbetten, der auch für andere Betriebsmittel geeignet ist (z.B. CPU).

Im vorangegangenen Kapitel wurden das Kommunikationssystem und das Protokoll zur dynamischen Planung der notwendigen Betriebsmittel (Netz und CPU) vorgestellt. Die lokale und globale Planung setzt eine kalenderbasierte, dynamische Planung des zeitgesteuerten Netzzugriffs voraus, die ich in diesem Kapitel vorstellen werde.

### 5.1 Garantie

In einem dynamischen System muß man sich ab dem Zeitpunkt der erfolgreichen Planung auf die Verfügbarkeit der Betriebsmittel verlassen können, um die Vorhersagbarkeit des Systems zu gewährleisten.

Für eine erfolgreich eingeplante Betriebsmittelanforderung wird eine *Garantie* vergeben, die die korrekte Bereitstellung der Betriebsmittel beschreibt – die angeforderte Leistung wird garantiert [SR91]; der in 4.1 dargestellte Garantievertrag kann auf jede BM-Nutzung erweitert werden. Eine vergebene Garantie wird nicht zurückgezogen und sichert die korrekte Ausführung einer geplanten Echtzeitaufgabe.

Es ist die Aufgabe der Planung, die Garantien zu vergeben und durchzusetzen. Die Betriebsmittelanforderungen müssen bei Neuplanungen gegeneinander verglichen werden, damit die garantierte Verfügbarkeit nicht durch BM-Konflikte gefährdet wird.

## 5.2 Aktivitäten

Ich betrachte *Aktivitäten* als die planbaren Einheiten (PE), mit deren Spezifikation die Planbarkeit gegenüber den bestehenden Garantien getestet wird. Eine Aktivität besteht aus einer Gruppe von BM-Nutzern, die als Ganzes operieren und eine gemeinsame Spezifikation besitzen. Die garantierte Leistung stellt ein Nutzungsrecht für die Gruppe dar, das sie von den anderen Aktivitäten des Systems abgrenzt. Die Nutzung *innerhalb* der Gruppe kann die Planung nicht garantieren.

Die Abstraktion von dem inneren Aufbau einer Aktivität erleichtert die Planung von Echtzeitaufgaben, die in einer wohldefinierten Weise miteinander arbeiten, so daß eine kombinierte Spezifikation für die vorhersagbare Ausführung ausreicht und eine Betrachtung feinerer Details entfallen kann. Zum Beispiel können in der CPU-Planung damit abwechselnd aktive Threads erfaßt werden und in der Datenübertragung löst man sich von der Simplexkommunikation (s. 4.5.2); ein RPC mit einer dezentralen I/O entspricht einer solchen Aktivität.

Im Bereich der Echtzeitsysteme betrachtet man in der Planung meist Aktivitäten, die aus einer einzelnen Task oder einer Menge einzeln spezifizierter Tasks bestehen. Ich habe den Begriff, wie oben beschrieben, auf eine Gruppe mit *einer* Spezifikation erweitert, um Nachteile zeitgesteuerter Systeme abzuschwächen (siehe 5.7); in den folgenden Erläuterungen spielt diese Erweiterung aber noch keine Rolle.

## 5.3 Betriebsmittel

### 5.3.1 Vielfachheit

Ein Betriebsmittel ist *einfach*, wenn zu einem Zeitpunkt  $t$  seine Funktionalität von höchstens einer Aktivität genutzt werden kann. Andernfalls ist es *mehrfach* vorhanden. Die CPU in einem Einprozessorsystem und ein Multiple-Access-Bus<sup>1</sup> sind einfache BM, der Hauptspeicher eines Rechners ist ein mehrfaches BM.

### 5.3.2 Teilbarkeit

Auf einem höheren Abstraktionsniveau betrachtet man die Vielfachheit innerhalb eines beliebigen Zeitintervalls  $\Delta t$ , die Teilbarkeit. Kann ein Betriebsmittel in einem Zeitintervall prinzipiell von mehreren Aktivitäten genutzt werden, so ist es *teilbar*. Ein mehrfaches BM ist inhärent teilbar und bei einem einfachen BM kann man die Teilbarkeit in  $\Delta t$  durch wechselseitige Nutzung simulieren. Die Teilbarkeit eines Betriebsmittels kann man implizit oder explizit simulieren (s. 5.6.1). Bei der expliziten Teilbarkeit werden die Zugriffsrechte direkt an die

---

<sup>1</sup>Eine Funkstrecke ist ein Multiple-Access-Netz, das über die Nutzung verschiedener Frequenzen zu einem mehrfachen BM wird.

Aktivitäten vergeben. Im impliziten Fall erfolgt die Vergabe nach einem indirekten Verfahren, wie etwa das der Prioritätensteuerung oder des Round-Robin mit einer entsprechend begrenzten Nutzungsdauer.

Die CPU und der MA-Bus sind teilbar, wobei CSMA/CD die Teilbarkeit des Buszugriffs während einer einzelnen Nachrichtenübertragung einschränkt. Ausgabegeräte wie Lautsprecher o.ä. sind unteilbare BM, die ich im weiteren nicht mehr betrachte.

### 5.3.3 Zugriff

Die Vergabe von Zugriffsrechten auf Betriebsmittel muß geplant werden, wenn die nebenläufige Nutzung andernfalls zu Fehlern führen würde; dies gilt natürlich insbesondere für die zeitbeschränkte Nutzung von CPU und Netz.

Der unteilbare, *ausschließliche Zugriff* auf ein einfaches BM ist die einfachste Methode, die Vorhersagbarkeit und Einhaltung der Garantie zu sichern.

Die simulierte Teilbarkeit eines einfachen Betriebsmittels realisiert eine *anteilige Nutzung*; z.B. 20% der in einem Intervall zur Verfügung stehenden BM-Leistung. Die daraus resultierende „gleichzeitige“ Nutzung eines Betriebsmittels innerhalb eines Intervalls wird durch das Multiplexing der Zugriffsrechte auf mehrere Aktivitäten erreicht.

Beim **Zeitmultiplexing** werden prozentuale Anteile am BM in zeitliche Anteile an einem Intervall übersetzt. Diese Unterteilung wird in der Planung *explizit* gespeichert, indem man das Aktivitätsintervall in feste Teile unterteilt. Eine anteilige Vergabe von z.B. 50% des Betriebsmittels wird abgebildet auf eine ausschließliche Reservierung von 50 Zeiteinheiten in einem Intervall der Größe 100 (s. Abb. 5.1), so daß man im Mittel die Hälfte der Intervalllänge nutzen kann. Mit einem beliebig kleinen Multiplexintervall würde man als Grenzwert eine anteilige Nutzung eines inhärent teilbaren Betriebsmittels erreichen. Die Größe des Intervalls als Bereich der Mittelwertbildung spielt eine sehr wichtige Rolle im Zeitmultiplexverfahren (siehe 'Durchsatz' auf Seite 115). Die Notwendigkeit einer ausschließlichen Nutzung kann man mit dem expliziten zeitlichen Multiplexing umgehen, ohne auf die Vorteile für die Vorhersagbarkeit verzichten zu müssen.

Eine implizite Unterteilung der Intervalls, indem man überprüft, ob die zeitlichen Bedingungen eingehalten werden können und dann z.B. prioritätengesteuert zur Laufzeit die Zugriffsrechte bestimmt (s. 5.6.1), liefert den gleichen Mittelwert, schränkt aber die Vorhersagbarkeit auf die Länge des Multiplexintervalls ein.

Das **statistische Multiplexing** ist eine Erweiterung des zeitlichen Multiplexings. Verschiedene, über statistische Angaben spezifizierte, variable Aktivitäten werden nebenläufig verarbeitet. Wenn man zum Beispiel zwei Angaben der Form hat, daß jeweils im Mittel 20% des BM genutzt wird und nur *ungefähr* alle  $x$  Zeiteinheiten eine höhere Anforderung von bis zu 60% besteht (ein *Burst*), können diese beiden Nutzungen, abhängig von dem Wert  $x$  und der Länge des Bursts, mit einer bestimmten, errechenbaren Wahrscheinlichkeit nebeneinander, konflikt-

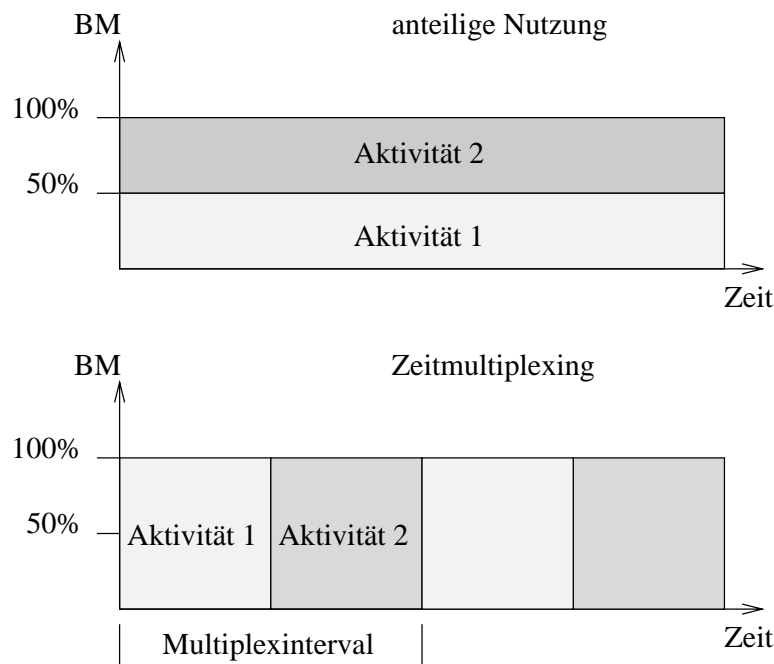


Abbildung 5.1: Zeitmultiplexing als Realisierung einer anteiligen Nutzung

frei arbeiten. Mehrere verschiedene, durchaus sehr differenzierte Spezifikationen lassen sich möglicherweise nebeneinander einplanen. Die Zugriffsrechte, die über die Grundlast hinausgehen, müssen in einer impliziten Absprache zwischen den Aktivitäten vergeben werden (s. 4.6.2 und 5.7.6).

Man erhält statistische Garantien, die die konfliktfreie Übertragung mit einer bestimmten Wahrscheinlichkeit zusichern ( $p$ %ige Garantie). Das statistische Multiplexing ist somit nur für den Einsatz mit weichen Zeitbedingungen geeignet. Für den Einsatz neben einer HRT-Kommunikation müssen die Zugriffe der verschiedenen Echtzeitklassen getrennt werden. Alle Aktivitäten mit einer statistischen Last werden in einer Aktivität zusammengefaßt und erhalten einen festen, explizit reservierten Anteil der BM-Leistung. Interferenzen sind dann durch das Zeitmultiplexing ausgeschlossen.

## 5.4 Zeitliche Spezifikation

Folgende grundlegenden zeitlichen Parameter beschreiben die Leistungsspezifikation einer ausschließlichen Nutzung eines BM, die für die Netzkommunikation im nächsten Unterkapitel erweitert wird:

- **Aktivitätsdauer, Ausführungszeiten**  
Die Ausführungszeit (execution time, ET) beschreibt die Länge der für die Aufgabenerfüllung notwendigen BM-Nutzung.

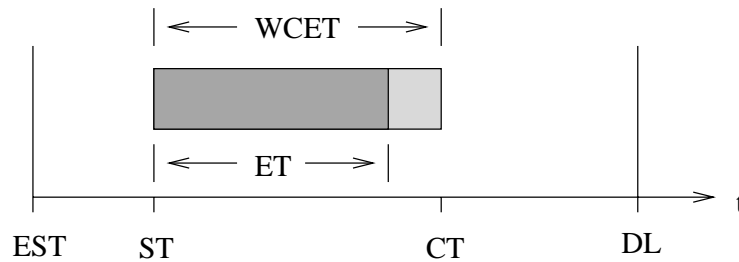


Abbildung 5.2: Zeitliche Parameter einer Aktivität

Dieser Wert ist von der aktuellen Laufzeitumgebung abhängig und entscheidet sich erst *nach* der Laufzeit. Aufgrund von Abhängigkeiten, Interferenzen und variierenden Eingaben benötigt man bei wiederholten Ausführungen in der Regel unterschiedliche ET. Angaben, die nicht nur für eine einzige Ausführung gelten, sind die erwartete ET (EET), die kürzeste ET und die maximale ET (WCET).<sup>2</sup> Die Angabe der maximalen Ausführungszeit WCET (Worst-Case-ET) ist für die genaue Planung und Vorhersagbarkeit der Aktivität wichtig und unter HRT-Bedingungen sogar notwendig.

Für eine Aktivität  $A$  schreibe man kurz  $et(A)$ ,  $eet(A)$  und  $wcet(A)$ .

- Start, Abschluß

Die Startzeit  $st(A)$  beschreibt den Beginn der Verarbeitung, der Zeitpunkt des ersten BM-Zugriffs. Die Abschlußzeit  $ct(A)$  (completion time) ist der Zeitpunkt des letzten BM-Zugriffs. Diese Werte ergeben sich aus der Planung.

- Bereitzeit, Frist

Diese Parameter beschreiben die frühestmögliche Startzeit  $est(A)$  (earliest start time) und die spätestmögliche Abschlußzeit  $dl(A)$  (deadline);  $\Delta dl(A)$  bezeichnet einen relativen Offset zur Startzeit. Die Ankunftszeit (arrival time)  $at(A) \leq est(A)$  ist der Zeitpunkt des Planungsbeginns.

Der Spielraum (laxity) einer Aktivität ist die maximale Länge einer Unterbrechung, die ein fristgerechtes Beenden nicht stört. Zur Zeit  $st(A)$  gilt:  $l(A) = dl(A) - st(A) - wcet(A)$ .

- Periode, Ende

Eine Periode  $per(A)$  beschreibt eine regelmäßige Wiederholung der Aktivität  $A$ . Die einzelne Aktivität bezeichne ich als Instanz, die durch einen Index identifiziert wird:  $est_i(A)$ ,  $st_i(A)$ , ...

<sup>2</sup>Aus Gründen der Vollständigkeit führe ich hier und im folgenden auch Parameter auf, die eine Interpretation der reservierten Zeiten erfordern. Das in Abschnitt 5.7 vorgestellte Planungsverfahren wird sich auf die nicht-interpretierte Reservierung von Ausführungszeiten beschränken, siehe auch 5.8.

Weiterhin gilt:  $st_{i+1}(A) = st_i(A) + per(A)$  und  $per(A) \geq \Delta dl(A) = const$  und  $wcet(A) = const$ .

Eine periodische Aktivitäten besitzt auch eine letzte Instanz: nach  $end(A)$  wird keine weitere Instanz gestartet. Diese Information ist in dynamischen Systemen wichtig, um Planungen in der Zukunft durchführen zu können, wenn das Ende der Aktivität bekannt ist. Diese grundlegende Voraussetzung für die sinnvolle Planbarkeit zeitlich begrenzter periodischer Aktivitäten wird meist in der Literatur vernachlässigt.

- Jitter

Mit Jitter bezeichnet man maximale Variationen einer nicht-konstanten Periode. Der Parameter kann in zwei Richtungen interpretiert werden. Einerseits können variierende Zugriffsanforderungen, die zur Laufzeit auftreten und akzeptiert werden müssen, damit beschrieben werden. Vergrößert man den Jitter, steigt der BM-Bedarf einer vorhersagbaren Planung und die Planungsauslastung sinkt (die Ausführungszeit bleibt unverändert). Andererseits bietet sich für den beabsichtigten TDMA-Einsatz eine andere Interpretation an: eine Variation der Startzeiten erhöht die Erfolgchancen der Planung (s. Abb. 5.4 auf Seite 116).

Für eine statistische Reservierung müßten Ausführungsdauer, Periode (Jitter) und ggf. Deadline um entsprechende Wahrscheinlichkeitsverteilungen bzw. -werte ergänzt werden. Die Leistungsspezifikation für die Nutzung der CPU wird durch die oben angegebenen allgemeinen Parameter bestimmt.

Die Leistungsspezifikation des SAP der MAC-Schicht erfolgt in der TDMA-Planung ebenfalls über diese Parameter.

## 5.5 Leistungsspezifikation für Netzzugriffe

Die Kommunikation im verteilten Echtzeitsystem muß je nach Applikation verschiedenen Qualitäten genügen. Im folgenden stelle ich daher eine allgemeine Sammlung von Parametern für die Leistungsspezifikation am SAP der Transportschicht vor. Mit ihnen können die Leistungen der Kanäle beschrieben werden: man spricht auch von einer *Quality-of-Service* (QoS) Beschreibung.

### 5.5.1 QoS-Parameter

Allgemeine Parameter zur Beschreibung eines Kommunikationsdienstes müssen die folgenden Bereiche abdecken: Kommunikationsform, Paketgröße, Verzögerung, Periode, Jitter, Durchsatz, Burstphasen, Güte der Garantie, Verlustwahrscheinlichkeit, Sicherheit, Verhalten zwischen Verbindungen; siehe zum Beispiel [Bö95, Fer90]. Diese gehen teilweise über die betrachtete Umgebung einzelner MA-Netzsegmente hinaus und sind vollständigkeithalber kurz erwähnt.

Die **Form der Kommunikation** ist in Echtzeitsystemen eingeschränkt, um die Vorhersagbarkeit zu wahren. Die grundsätzlichen (Kombinations-)Möglichkeiten sind in der Abbildung 5.3 aufgeführt. Grundlegend kann zwischen

	verbindungsorientiert	verbindungslos
unidirektional	×	×
bidirektional	×	

Abbildung 5.3: Kommunikationsformen

*verbindungsorientierter* (VO) und *verbindungsloser* (VL) unidirektionaler Kommunikation unterschieden werden. Verbindungsorientierte Kommunikation kann auch bidirektional erfolgen.

In der **verbindungsorientierten** Kommunikation wird eine logische Verbindung zwischen Sender und Empfänger aufgebaut. Beim Aufbau der Verbindung wird deren Leistung festgelegt bzw. abgestimmt. Mit der Festlegung eines Kommunikationweges sind alle erforderlichen Betriebsmittel zu planen, so daß die zeitlichen Bedingungen für die Echtzeitkommunikation garantiert werden können.

**Unidirektionale** Verbindungen (Simplexverbindung) erlauben einen Datenfluß nur in einer Richtung von einer vorher festgelegten Quelle zu einer festgelegten Senke. Zwei unidirektionale Verbindungen können eine Bidirektionalität emulieren (Halbduplexverbindung), z.B. für einen RPC.

Ein Multicast läßt hier mehrere Senken als Empfänger zu und über weitere Gruppenkommunikationsmechanismen kann dieses Modell bis auf eine  $n$ - $m$  Kommunikation mit  $n$  Sendern und  $m$  Empfänger erweitert werden. Dies entspricht den Aktivitäten, wobei hier keine Unterstützung der Gruppenkommunikation betrachtet wird.

Anwendungsgebiete für diese Form der Kommunikation sind überall dort, wo erzeugte Daten von einem oder mehreren Verbrauchern genutzt werden; als Beispiel hierfür ist die Übertragung von Uhrzeiten *eines* Zeitgebers und von Sensorwerten an die Steuerung und zusätzliche Visualisierung zu nennen.

**Bidirektionale** Verbindungen (Duplex- oder Halbduplexverbindung) fassen die Anforderungen eines Hin- und Rückweges zusammen und erlauben einen Datenfluß in beide Richtungen.

Im Fall der **verbindungslosen** Kommunikation wird jedes Paket (Datagramm) mit seinen eigenen Ziel- und Leistungsbeschreibungen einzeln übertragen. Eine Planung zur Einhaltung zeitlicher Bedingungen ist zwar

grundsätzlich möglich, eignet sich aber wegen einer im allgemeinen kurzfristigen Benutzung von Datagrammen und des relativ großen Aufwands der Planungsverfahren schlecht für einen praktischen Einsatz.

Die **Verkehrcharakteristik** beschreibt das grundsätzliche Merkmal einer verbindungsorientierten Kommunikation: kontinuierliche, periodische oder variable Datenrate. In dieser Reihenfolge wird der Aufwand der Spezifikation größer, d.h. die erforderlichen Parameter nehmen zu und das Datenaufkommen wird weniger vorhersagbar und ist schwieriger zu planen.

Die **kontinuierliche Datenrate** beschreibt konzeptionell ein gleichmäßiges „Versickern“ der Daten. Zum Senden bereitgestellte Daten werden in einem kontinuierlichen Strom übertragen und unabhängig von der Ankunftszeit läßt sich die Abschlußzeit der Übertragung angeben. Man ist hierbei nicht auf feste Sendezeitpunkte o.ä. angewiesen und es steht (theoretisch) ständig ein Teil der Netzkapazität für die Verbindung bereit – anteilige Nutzung.

In der Praxis ist dabei die Größe des Multiplexintervalls des zeitlichen Multiplexings entscheidend, deren untere Grenze von der Netzwerkarchitektur und Verarbeitungsgeschwindigkeit der Rechner abhängt. Letztlich muß mit diskreten Blöcken gearbeitet werden und die entsprechenden SAPs erhalten komplette SDUs.

Die erforderlichen Parameter sind: Bandbreite, Durchsatz (s.u.).

Die **periodische Datenrate** einer Aktivität/Netzreservierung  $A$  beschreibt periodisch wiederholte Übertragungen. Alle  $per(A)$  Zeiteinheiten wird ein Paket für den Versand bereitgestellt. Die Anforderungen an die anschließende Übertragung sind noch zu spezifizieren: die Daten können mit einer kontinuierlichen Datenrate oder jeweils innerhalb einer Deadline  $\Delta dl(A)$  gesendet werden. Die Angaben sind äquivalent und können mit dem ‘Durchsatz’ (Intervalllänge) verfeinert werden.

Die Diskretisierung der kontinuierlichen Kommunikation innerhalb des Multiplexintervalls stellt einen Übergang zur periodischen Datenrate dar.

Mit festen Perioden und Datenmengen läßt sich diese Form der Kommunikation einfach planen und besitzt eine hohe Vorhersagbarkeit [LL73, K<sup>+</sup>89]. Periodisches Abfragen einer Sensorinformation ist ein Anwendungsbeispiel. Die erforderlichen Parameter sind: Bandbreite, Durchsatz, Periode, Jitter, Paketgröße.

Bei der **variable Datenrate** handelt es sich um eine weitere Abschwächung der Spezifikationsgenauigkeit und sogar um eine eigene Klasse von Anforderungen. Es wird nur noch eine statistische Angabe über die zu erwartende

Auslastung gemacht, so daß eine Planung mittels statistischem Multiplexing (s.S. 109) erfolgt. Man erhält eine statistische,  $p\%$ ige Garantie.

Die erforderlichen Parameter sind: Bandbreite, Durchsatz, Burstphasen.

Die **Bandbreite**  $B$  beschreibt, wie viele Bytes durchschnittlich pro Zeiteinheit verschickt werden (Bits/s). Es handelt sich hierbei um einen Mittelwert und die Größe des gemittelten Zeitintervalls ist nicht festgelegt. Wann welche Datenmenge transportiert wird, ist offen.

Der **Durchsatz**<sup>3</sup>  $D$  erlaubt im Gegensatz zur Bandbreite eine genauere Spezifikation der geforderten und reservierten Übertragungsgeschwindigkeit, obwohl rechnerisch immernoch gilt:  $B = D$ . Zusätzlich zu einer gegebenen Datenmenge wird auch das Multiplexintervall – die genaue Zeit, die das Versenden maximal dauern darf – angegeben. Dadurch sind die Kommunikationsmuster durch eine Verfeinerung der Granularität genauer bestimmbar. Während für die Bandbreite kein Unterschied zwischen den Angaben 100KB/s und 1MB/10s<sup>4</sup> besteht, liegt ein solcher beim Durchsatz durchaus vor. Bei 100KB/s werden in jeder Sekunde, egal wann, 100KB Daten bewegt. Mit der Angabe 1MB/10s allerdings wird nur sichergestellt, daß nach zehn Sekunden eine von vornherein bereitgestellte Datenmenge von 1MB am Endpunkt der Übertragung bereit steht. Es wird nicht bestimmt, wann innerhalb dieser Zeitspanne die Übertragung erfolgt. Man kann hiermit die Kontinuität der Verbindung beschreiben.

Die **Paketgröße**  $S$  beschreibt die maximale Größe eines Paketes (SDU) in Bytes. Ein Paket ist hierbei eine Einheit für den Versand von Daten und spezifiziert die Datenmenge der einzelnen Instanz einer periodischen Datenrate.

Erweiternd kann man die Kommunikationsmuster detaillierter spezifizieren, indem man mehrere Pakete pro Instanz zuläßt.

Die **Verzögerung** (Delay) beschreibt die maximale Verzögerung beim Senden, d.h. die Zeit zwischen der Übergabe des Paketes an die Transportschicht und der Entgegennahme auf der Empfängerseite durch die Applikationen. Diese Verzögerungen resultieren zum einen aus technisch bedingten Verarbeitungszeiten in der Netzkommunikation oder aus notwendigen Bearbeitungszeiten der Protokollverarbeitung. Zum anderen kann aber auch aus taktischen Gründen eine Verzögerung an einer oder an mehreren Stellen für einen kollisionsfreien Ablauf sorgen.

---

<sup>3</sup>‘Durchsatz’ benutzt man in der Systemanalyse auch häufig zur Beschreibung einer Leistung pro Zeiteinheit. Aber ich definiere ihn in dieser Arbeit zur Abgrenzung als Leistung pro wählbarem Zeitintervall.

<sup>4</sup>Kilobyte pro Sekunde bzw. Megabyte pro Sekunde

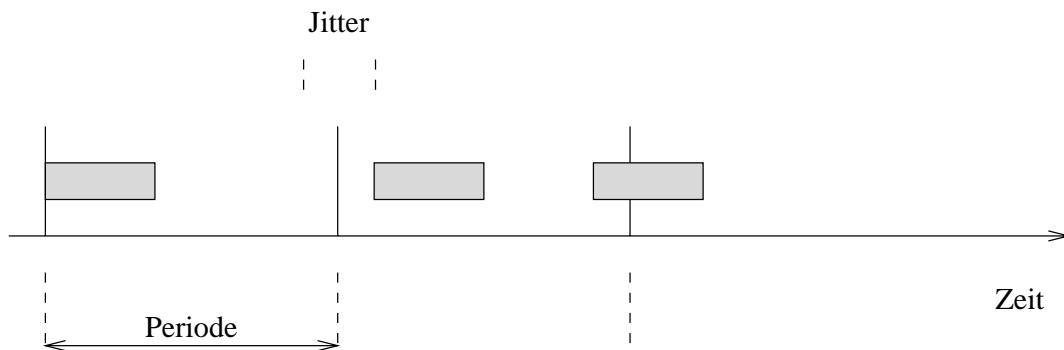


Abbildung 5.4: Jitter einer periodischen Aktivität

Die **Periode**  $P$  beschreibt den Zeitabstand, mit dem Pakete gesendet werden.

Mit **Jitter**  $J$  bezeichnet man mögliche Abweichungen von der Periode bzw. maximale Variationen der Verzögerung, wie sie vom Empfänger empfunden werden (s. Abb. 5.4).

**Burstphasen** (Länge, Frequenz) beschreiben für eine variable Datenrate eine statistische Last, die neben einer kontinuierlichen oder periodischen Grundlast zusätzliche Phasen erhöhter Auslastung besitzt. Die *Bursts* sind durch die minimale Interburstzeit (max. Frequenz), Burstdurchsatz und maximaler Länge beschrieben.

Einsatz findet diese Spezifikation überall dort, wo in mehr oder weniger regelmäßigen Abständen eine größere Datenmenge anfällt. Um diese Lastspitzen zu umgehen, müßten ausreichend dimensionierte Puffer zum Glätten eingesetzt werden. Wenn die Verzögerung klein gehalten werden soll oder kein Platz für die Zwischenspeicherung zur Verfügung steht, ist diese Spezifikation mit den Vorteilen des statistischen Multiplexings für Planung und Versand zu benutzen (s. Seite 109).

Die **Güte der Garantie** beschreibt Abstufungen in der geforderten bzw. erhaltenen Wichtigkeit oder die Qualität der Planung: NRT, SRT, HRT.

Der **Start** und das **Ende** einer Reservierung begrenzen die Kommunikation in der Zeit. Dies entspricht  $est(A)$  und  $end(A)$  der zeitlichen Spezifikation.

Folgende Parameter erwähne ich der Vollständigkeit halber. Sie werden aber im Rest der Arbeit nicht betrachtet.

Die **Verlustwahrscheinlichkeit** gibt die Anfälligkeit des Kanals gegenüber Paketverlusten oder -veränderungen aufgrund von Übertragungs- oder Zeitfehler an. Die Wahl einer Netzwerkverbindung (in einem WAN), die Größe

des bereitgestellten Pufferspeichers und der geplante Umfang des CPU-Aufwands, als Methode der Flußkontrolle in SRT-Kommunikation, können von dieser Angabe abhängen.

Die **Sicherheit** einer Leitung kann die Übertragung sensibler Daten schützen. Hier könnten Parameter einfließen, die eine Verschlüsselung oder Wahl einer Leitung beeinflussen (z.B. Funkstrecken im WAN).

Über das **Verhalten zwischen Kanälen** kann z.B. eine Synchronisation beschrieben werden, wie sie im Multimedia-Bereich (Bild und Ton) zwingend erforderlich ist. Entweder als Leistung der tatsächlich Netzkommunikation oder durch ausreichende Puffer als Zwischenspeicher des empfangenden/verarbeitenden Rechners kann diese Kontrolle direkt ins RT-Protokoll einfließen oder in einem aufsetzenden Dienst zur Verfügung gestellt werden.

### 5.5.2 Aspekte der harten Echtzeitkommunikation

Welche der oben angegebenen Parameter sind für eine genaue Spezifikation von harten Echtzeitanforderungen notwendig?

- **Kommunikationsform**

Eine verbindungsorientierte Kommunikation ist keine unbedingte Voraussetzung für die Einhaltung harter Zeitbedingungen, aber die notwendige Planung und der Verwaltungsaufwand können dadurch auf ein verträgliches Maß reduziert werden.

Der Versand verbindungsloser Datagramme unter harten Zeitbedingungen erfordert eine Reservierung für jedes einzelne Paket, um die Kommunikation in die Applikation – in die höheren OSI-Schichten – zu integrieren (siehe 4.8); auf die Planung solcher Datagramme gehe ich nicht ein.

- **Verkehrscharakteristik**

Die Planung einer kontinuierlichen oder periodischen Datenrate erlaubt die Vergabe von Garantien, die harten Echtzeitbedingungen genügen. Mit einem konstanten, zeitlichen Multiplexing lassen sich diese korrekt planen.

Der Einsatz einer variablen Datenrate in der harten Echtzeitkommunikation ist nicht ohne weiteres möglich. Für die Vergabe 100%-iger Garantien muß der gesamte maximal benötigte Betriebsmittelbedarf reserviert werden, so daß man eine sehr geringe Planungsauslastung und keinen Vorteil gegenüber einer kontinuierlichen oder periodischen Datenrate mehr hat.

- **Bandbreite, Durchsatz, Paketgröße, Periode**

Dies sind notwendige Parameter für die Bestimmung der Datenrate.

Für einen TDMA-Zugriff können sie in eine rein zeitliche Spezifikation (*est, st, wcet, dl*) überführt werden und genügen aufgrund ihrer Planbarkeit den Anforderungen eines HRT-Einsatzes.

- Verzögerung und Jitter

Diese Parameter können zusätzlich benutzt werden, um die Planbarkeit zu verbessern. Die ‘Verzögerung’ ist für die Verbindung unterschiedlicher Netzsegmente und die damit erforderliche Abstimmung des Kommunikationsverkehrs wichtig. ‘Jitter’ in der Kommunikation – Jitter im Datenerzeuger kann geglättet werden – hilft die Planbarkeit zu verbessern, weil konkurrierende Netzzugriffe gegeneinander verschoben werden können.

### 5.5.3 Andere QoS-Spezifikationen

Alternativ zu den oben vorgestellten Parametern werden hier Spezifikationen anderer Kommunikationssysteme erläutert und ihre Eignung für den Einsatz unter harten Zeitbedingungen untersucht.

#### 5.5.3.1 ATM

Mit dem Asynchronous Transfer Mode (ATM) ist auch eine Spezifikation des Leistungsmodells definiert worden ([LB92, DP91]), an der auch noch weiter gearbeitet wird. Die Technik und die Spezifikation sind besonders für den Transport von Multimedia-Daten geeignet.

Eine ganze Reihe von Parametern zur Beschreibung des Verkehrs stehen zur Verfügung, die man für die Implementierung von Dienstklassen nutzen kann. Nur wenige werden als Parameter der Dienstklassen bis zum Nutzer gereicht: Die *Peak Cell Rate* (PCR) beschreibt die maximale Frequenz, mit der Zellen (Pakete mit 48 Byte Nutz- und 5 Byte Kontrolldaten) gesendet werden. Zusammen mit der *Cell Delay Variation Tolerance*, die angibt, um wieviel diese Zellen vor ihrem eigentlichen Sendetermin abgeschickt werden, kann man Burstphasen modellieren.

Die durchschnittliche Zellenrate wird mit der *Sustainable Cell Rate* festgelegt und der *Burst Tolerance* Parameter beschreibt die Häufigkeit von Bursts, die dann mit der PCR bearbeitet werden. Dies entspricht der oben angegebenen Beschreibungen von Burstphasen und Durchsätzen.

Zusammen mit *Delay*, *Jitter* und verschiedenen Möglichkeiten für die Fehlerbeschreibung (Zellenverlust, Markierung unwichtiger Zellen für die Löschung im Konfliktfall, ...) spiegeln diese Parameter insgesamt die bereits unter 5.5.1 angegebenen QoS-Parameter wieder.

Verschiedene Dienstklassen (ATM Adaptation Layer, AAL) werden aus diesen Parametern gebildet und sind vom Dienstanutzer parametrisierbar, so daß nur ein Teil der Werte direkt sicht- und bestimmbar ist (s. Abb. 5.5).

Klasse A	Klasse B	Klasse C	Klasse D
zeitliche Abhängigkeiten		keine zeitl. Abhäng.	
CBR	VBR		
verbindungsorientiert – VO			VL

Abbildung 5.5: ATM-Dienstklassen

- Die Klasse A stellt eine kontinuierliche Datenrate zur Verfügung. Anwendungen sind z.B. Videoübertragung oder Netzwerkpartitionierung (virtual networks, Netzwerkemulation); kontrollierter (die Zellen sind mit einer Sequenz-Nummer versehen), konstanter Durchsatz.
- Eine variable Datenrate (VBR) mit zeitlicher Spezifikation stellt die Klasse B bereit. Statistisches Multiplexing wird für die Planung dieser Klasse (VBR) eingesetzt.
- Die Klasse C beschreibt eine variable Datenrate ohne zeitliche Bedingungen, die auch verlässliche, fehlerfreie Kommunikation anbietet. Anwendung: Datentransfer (mit gewünschtem Durchsatz, aber nicht garantiert).
- Einen verbindungslosen Kommunikationsdienst stellt die Klasse D zur Verfügung, der ebenfalls keinen zeitlichen Bedingungen unterworfen ist.

Die vordefinierten Verkehrsklassen schränken die erforderlichen Parameter sinnvoll ein und vereinfachen somit die Spezifikation der Verbindungen.

### 5.5.3.2 Tenet

In der Tenet-Gruppe der Universität von Kalifornien in Berkeley und des International Computer Science Institute wird die Tenet protocol suite entworfen [BF<sup>+</sup>94]. Die Arbeit umfaßt die Leistungen der Transport- und der Netzwerkschicht und stellt verbindungsorientierte, unidirektionale, unzuverlässige Kanäle mit garantierten Leistungscharakteristika zur Verfügung.

Das System besteht aus mehreren Komponenten, u.a. RTIP (RT Internet Protocol) in der Netzwerkschicht, RMTP (RT Message Transport Protocol), CMTTP (Continuous Media Transport Protocol) und zwei weitere Management Protokolle, die den Auf- und Abbau von Verbindungen und das Monitoring des Datentransfers übernehmen.

Das Real-Time Message Transport Protocol (RMTP) beschreibt einen Dienst zur Übertragung von Echtzeitdaten. Maximaler und mittlerer Durchsatz, Größe des Intervalls zur Durchschnittsbildung, Paketgröße, maximale Verzögerung, Jitter, Verlustwahrscheinlichkeit ( $\rightarrow$  Puffergrößen) sind die Parameter der Spezifikation.

Das Continuous Media Transport Protocol (CMTP) beschreibt eine periodische Datenrate, mit der eine kontinuierliche Verbindung simuliert wird [WM91]. Der Durchsatz bestimmt sich aus der Periode und der mittleren und maximalen Paketgröße. Neben dem Delay existieren noch Parameter für Verlustbehandlung (Verlustwahrscheinlichkeit, Austauschindikator, Defaultpaket) und Puffergrößen beim Sender und Empfänger, um Schwankungen der Verarbeitungsgeschwindigkeit auszugleichen (Jitter wird nicht benutzt, das Protokoll glättet).

### 5.5.3.3 RFC 1363 ‘Flow Spec’

Der als RFC (Request For Comments) von Partridge und der IRTF (Internet Research Task Force) veröffentlichte Vorschlag zielt deutlich auf eine Erweiterung der bestehenden Internettechnologie [Par92]. Lokal in dem sendenden Rechner wird eine Abwandlung des Token-Verfahrens für die Modellierung der Datenübertragung genutzt.

Die *Dual leaky bucket* Flußkontrolle beschreibt die mittlere und die höchste kurzfristige Datenrate über zwei Eimer (Buckets), die Zugriffsrechte in Form von Tokens enthalten.

Pakete können nur versandt werden, wenn man im Besitz einer entsprechenden Menge Senderechte, *Tokens*, ist, die man dem Token-Bucket entnimmt. Der Eimer der Größe Token Bucket Size füllt sich kontinuierlich mit Token Bucket Rate. Sendebereite Daten legt man im Leaky Bucket ab, der den maximalen Zugriff auf das Netz steuert: Pakete im Leaky Bucket werden mit Maximum Transmission Rate versendet (Leaky Bucket Size = Token Bucket Size). Weitere Parameter sind Maximum Transmission Unit (MTU), Jitter, minimaler Jitter (der bemerkt wird), Verlustwahrscheinlichkeit, Burstverlustwahrscheinlichkeit, Qualität der Garantie (6 Stufen).

Als ratenbasiertes Verfahren ist diese Flußkontrolle nicht für den TDMA-Einsatz geeignet; vgl. auch *linear bounded arrival process* in [MIS96b]: maximum message size, maximum message rate, maximum burst size.

## 5.5.4 Bewertung

Mit den QoS-Parameter von ATM, Tenet und RFC 1363 ist die Spezifikation von HRT-Kommunikation möglich: Ihre Hauptanwendungen liegen zwar in der Spezifikation von Multimedia- bzw. SRT-Datenströmen, aber mit einer Verlust- bzw. Zeitfehlerwahrscheinlichkeit von 0% lassen sich auch harte Bedingungen spezifizieren. Wie gut in dem jeweiligen System die Einhaltung unterstützt wird, ist eine andere Frage!

Die Parameter des RFCs beschreiben einen ratenbasierten Versand und sind für eine TDMA-Spezifikation nicht interessant. Die Parameter von ATM und Tenet sind für eine zeitliche HRT-Spezifikation geeignet, weil eine zum Abschnitt 5.5.2 äquivalente Darstellung möglich ist.

Die in 5.5.1 angegebenen QoS-Parameter bilden eine allgemeine Grundlage, die teilweise auf die Spezifikationen in ATM, Tenet und RFC 1363 abzubilden ist. Besonders die zusätzliche Möglichkeit, eine zeitlich begrenzte Reservierung in der Zukunft über *Start* und *Ende* einer Betriebsmittelnutzung durchführen zu können, erleichtert den praktischen Einsatz in dynamischen Systemen. Wenn eine Reservierung immer nur zum aktuellen Zeitpunkt beginnt und ohne Beschränkung ist, kann die Planung mit zeitlich begrenzten Aufgaben nicht umgehen und reduziert dadurch die Planbarkeit erheblich; dieses Versäumnis betrifft alle alternativ vorgestellten Spezifikationsparameter und wird auch in der Literatur meist ausgelassen.

Mit den oben abgegrenzten Parametern ist die Spezifikation einer harten Echtzeitkommunikation in einem zeitgesteuerten System möglich. Die dafür notwendige Umsetzung in die TDMA-Parameter der zeitlichen Leistungsspezifikation aus 5.4 erläutere ich im nächsten Abschnitt. Mit den allgemeinen QoS-Parametern kann darüber hinaus das statistische Multiplexing einer variablen Datenrate beschrieben werden.

### 5.5.5 Unterstützung der QoS-Parameter

An der applikationsseitigen Schnittstelle (SAP) in der vierten OSI-Schicht erwartet das vorgestellte Kommunikationssystem eine TDMA-Spezifikation und bietet keine Verhandlungen über die Parameter an (vgl. 4.4.7). In der Applikationsentwicklung sind die in 5.5.1 angegebenen QoS-Parameter aber einfacher zu benutzen und ich gebe hier eine Umsetzung dieser Parameter auf die zeitliche Spezifikation, die als Eingabe für die Planung dient, an. Die Umsetzung ist kein Teil des Kommunikationssystems, sondern muß in der Applikation durchgeführt werden.<sup>5</sup>

Im Hinblick auf den beabsichtigten Einsatz in Multiple-Access-Netzen kann die Planung die geforderten Leistungen teilweise nur angenähert erreichen. Zum Beispiel ist das Betriebsmittel MA-Netz nicht teilbar, so daß eine kontinuierliche Datenrate nur durch zeitliches Multiplexing angenähert werden kann, dessen Genauigkeit von der Größe des Multiplexintervalls abhängt (Parameter 'Durchsatz').

Die *Kommunikationsform* hat auf die Planung/Reservierung der Datenübertragung im Server keinen Einfluß, weil nur die Zugriffszeiten berücksichtigt werden. Auch eine Halbduplexkommunikation, die über eine Aktivität realisiert wird, ist im Server nicht sichtbar. Sie ist eine Erweiterung der zeitlichen Spezifikation, die direkt in die Parametrisierung des Algorithmus' der lokalen CT-Planung einfließen muß. Brücken und Router sind auf den Parameter der Kommunikationsform in der Planung der verschiedenen angeschlossenen Netzsegmente angewiesen (vgl. 4.9.1.2).

---

<sup>5</sup>Die Reservierungsstruktur kann nur die zeitliche TDMA-Spezifikation aufnehmen und mit der Umsetzung unterstützt die Planung indirekt auch die allgemeinen QoS-Parameter.

Die am einfachsten zu unterstützende *Verkehrscharakteristik* ist die periodische Datenrate, deren Parametrisierung direkt in die Planung übernommen werden kann. Sei  $B_{net}$  die physikalische Bandbreite des Kommunikationsmediums,  $D$  der Durchsatz,  $S$  die maximale Paketgröße und  $M$  die periodische Aktivität. Dann gilt:

$$\begin{aligned} est(M) &= \text{Start} \\ end(M) &= \text{Ende} \\ wcet(M) &= \frac{S}{B_{net}} \\ \Delta dl(M) &= \frac{S}{D} \end{aligned}$$

Jitter  $J$  läßt sich beliebig festsetzen und direkt in die Planung übernehmen. Die Angabe einer Bandbreite statt des Durchsatzes verändert die Parameter. Es gilt zwar immer  $B = D$ , aber:

$$\begin{aligned} \Delta dl(M) &= P \\ J &= P \end{aligned}$$

Der Jitter führt nicht zu Überlappungen oder falschen Nachrichtenreihenfolgen, weil nur Zugriffszeiten reserviert werden.

Eine kontinuierliche Datenrate einer Bandbreite  $B$ <sup>6</sup> muß über die Periode bzw. den Durchsatz angenähert werden. Je kleiner die Periode ist, desto kleiner ist der Unterschied zur periodischen Datenrate mit

$$S = B \times P = D \times P.$$

Die variable Datenrate spielt mit dem statistischen Multiplexing eine eigene, für die SRT-Kommunikation wichtige Rolle. Die notwendige klare Trennung zu den anderen Datenraten ergibt noch keinen Vorteil für eine einzelne Reservierung. Im Gegenteil, um eine Beeinflußung sicher auszuschließen, sind (pessimistische Worst-Case-) Annahmen über das Datenvolumen zu treffen. Man muß aus der Spezifikation eine periodische Datenrate formen und diese wie oben beschrieben umsetzen. Erst wenn mehrere variable Datenraten geplant werden, ist das statistische Multiplexing anzuwenden und man erhält eine verkürzende Überlagerung der Aktivitäten.

Die *Verzögerung* spielt erst bei einem Einsatz mehrerer getrennter Segmente eine Rolle. Verzögerungen in den verbindenden Brücken und in größeren Topologien auch in den zum Einsatz kommenden Routern müssen berücksichtigt werden. Ich beschränke mich hier auf lokale Netze mit einem einzelnen Segment und betrachte diesen Parameter nicht weiter.

---

<sup>6</sup>Einen Durchsatz anzugeben ist nicht sinnvoll, weil die Bandbreite im Idealfall ständig zur Verfügung steht.

Die Planung der CPU erfolgt auf analoge Weise; die Datenstrukturen und Parameter lassen sich sowohl für die Netz- als auch für die CPU-Planung einsetzen (siehe 4.4).

## 5.6 Echtzeitplanung

### 5.6.1 Planungsproblem

Sei  $\mathcal{A} = \{A_1, A_2, \dots, A_n\}$  eine Menge von ununterbrechbaren, periodischen Aktivitäten.<sup>7</sup> Die zeitlichen Parameter für den Zugriff auf ein Betriebsmittel sind durch  $est_j(A_i)$ ,  $dl_j(A_i)$ ,  $wcet(A_i)$  und  $per(A_i)$  gegeben. Die Echtzeitplanung legt die Startzeiten  $st(A_i)$  und damit eine Reihenfolge in der Art fest, daß keine Konflikte in den Zugriffen auf das BM auftreten:

$$\begin{aligned} st_j(A_i) &\in [est_j(A_i), dl_j(A_i) - wcet(A_i)] \\ &\text{und } \forall i_1 \neq i_2, j_1, j_2: \\ st_{j_1}(A_{i_1}) < st_{j_2}(A_{i_2}) &\Rightarrow st_{j_2}(A_{i_2}) - st_{j_1}(A_{i_1}) > wcet(A_{i_1}) \end{aligned} \quad (5.1)$$

Die Startzeiten  $st(A_i)$  beschreiben einen Plan. Man bezeichnet einen Plan als *brauchbar*, wenn er die korrekte, spezifikationsgemäße Ausführung aller Aktivitäten ermöglicht.

Die Planung erfolgt in mehreren Schritten:

#### 1. Test auf Planbarkeit von $\mathcal{A}$

Es ist zu testen, ob für eine gegebene Menge  $\mathcal{A}$  von Aktivitäten die Bedingungen (5.1) erfüllbar sind. Man unterscheidet zwischen notwendigen, exakten und hinreichenden Tests.

Ein notwendiger Test ist erfüllt, wenn ein brauchbarer Plan existiert. Für die Planbarkeit der Menge  $\mathcal{A}$  ist es z.B. notwendig, daß für die Auslastung gilt

$$\mu = \sum \frac{et(A_i)}{per(A_i)} \leq 1$$

Wenn ein hinreichender Test erfüllt ist, existiert ein brauchbarer Plan. Es kann aber zu einer Aktivitätsmenge  $\mathcal{A}$  ein brauchbarer Plan existieren, obwohl der hinreichende Test fehlgeschlagen ist.

Ein exakter Test bewertet eine Menge genau dann positiv, wenn es einen brauchbaren Plan gibt.

---

<sup>7</sup>Die Planung der Netzkommunikation kann man auf ununterbrechbare Aktivitäten beschränken, weil eine Unterbrechung höchstens an vereinbarten Punkten geschehen kann (Splitting  $\leftrightarrow$  Unterbrechung).

## 2. Planerstellung

Die Startzeiten  $st(A_i)$ , und damit die Reihenfolge, muß so festgelegt werden, daß ein brauchbarer Plan entsteht.

## 3. Plandurchsetzung

Die reine Betriebsmittelzuteilung (Dispatching) der Plandurchsetzung geschieht zur Ausführungszeit der Aktivitäten. Das Wissen über die ausgeübte Kontrolle der Betriebsmittelnutzung fließt in die Planungerstellung ein.

Die ersten beiden Punkte werden in der expliziten Planung oft zusammengefaßt. Der Test auf Planbarkeit kann durch das Erstellen eines expliziten, brauchbaren Planes erfolgen. Bei der impliziten Planung ergibt sich aus dem erfolgreichen Test eine Vorschrift für die sukzessive Erstellung eines brauchbaren Planes zur Ausführungszeit.

### **Impliziter Plan**

Ein impliziter Plan legt keine Startzeiten  $st(A_i)$  im voraus fest; sie sind implizit gegeben. Ausgehend von einem Zeitpunkt  $t$ , bis zu dem die Startzeiten bekannt sind, werden induktiv die Zeiten bis  $t + \Delta t$  bestimmt. Durch den vorangegangenen Test ist die Brauchbarkeit des sich ergebenden Planes gesichert. Das ratenbasierte Verfahren der Echtzeitkommunikation entspricht diesem Ansatz.<sup>8</sup>

### **Expliziter Plan**

Ein expliziter oder „kalenderbasierter“ Plan führt die Startzeiten der einzelnen Aktivitäten explizit auf. Für einen beliebigen Zeitpunkt  $t$  kann aus dem Plan das Zugriffsrecht direkt ermittelt werden. Einen Plan in der expliziten Planung nenne ich daher auch Reservierung<sup>9</sup>. Das planbasierte Verfahren der Echtzeitkommunikation entspricht diesem Ansatz.

Ein explizites Planungsverfahren ist meist nicht vollständig explizit! Allein die Verwaltung einer unendlichen periodischen Aktivität verbietet dies schon. Es müssen also implizite Vorschriften als Teil der expliziten Planung benutzt werden. Um die potentiell unendliche Zeitachse zu verwalten, können die Zeiten mit einer Hashfunktion auf eine Tabelle abgebildet werden oder eine Liste als expliziter Plan (Teilabschnitt der Zeitachse) wird nur bis zu einem notwendigen Punkt in der Zukunft aufgebaut. Im zweiten Fall nähert sich die Planerstellung der des impliziten Plans an, aber der Plan muß nicht unbedingt fortlaufend aufgebaut werden: die Kenntnis der BM-Nutzung zum Zeitpunkt  $t$  erfordert keine zum Zeitpunkt  $t - \Delta t$ .

---

<sup>8</sup>Hier wird nicht weiter darauf eingegangen, da die Argumentation in 2.1.1.3 im Kontext der Echtzeitkommunikation für diese Arbeit ausreicht.

<sup>9</sup>Die bisherige Verwendung von „Reservierung“ erfolgte synonym.

### Optimales Planungsverfahren

Für die Bewertung eines Planungsverfahrens führt man das *Garantieverhältnis* zwischen der Zahl der erfolgreich eingeplanten Aktivitäten einer Menge  $\mathcal{A}$  und der Zahl der theoretisch aus dieser Menge annehmbaren Aktivitäten ein. Ein optimales Verfahren erreicht ein Garantieverhältnis von eins und entspricht einem exakten Planbarkeitstest.

### Komplexität

Das oben angegebene Planungsproblem entspricht, eingeschränkt auf nicht-periodische Aktivitäten ( $per(A) = 0$ ), dem aus der Komplexitätstheorie bekannten Problem der Intervallsequenzierung und ist damit NP-vollständig [GJ79].

## 5.6.2 Abhängigkeiten

In der Planung können Abhängigkeiten zwischen kommunizierenden Aufgaben  $A$  und  $B$  für eine Koordinierung der Netzzugriffe genutzt werden (vgl. 3.3). Eine planbar schnelle Antwort einer RPC-Kommunikation erfordert zum Beispiel die abhängige Planung zweier Simplexverbindungen (Tightness); ähnlich wie auch die Datenverarbeitung vom Datenempfang abhängig ist und eine Integration in die Planung der Applikation erleichtert.

In einer zeitgesteuerten Planung sind solche Abhängigkeiten einfach zu berücksichtigen. Im Plan werden die Reihenfolge und die Abstände der Aufgaben explizit festgelegt und die parametrisierten Abhängigkeiten von  $A$  und  $B$  und evtl. anderer durch Umplanungen betroffener Aufgaben können direkt überprüft werden [Bak91, Law83].

Abhängigkeiten in der impliziten Planung kann man auf modifizierte zeitliche Parameter abbilden [Bla76], die den Aktivitäten jeweils Zeitfenster zuweisen. Diese Parameter dienen einem Algorithmus für die Planung unabhängiger Aktivitäten als Eingabe. Die Einbeziehung weiterer Betriebsmittel bedarf einer aufwendigeren Planung, die NP-hart ist.

Für eine weitergehende Diskussion von Abhängigkeiten siehe auch [SSNB95, Mü97].

## 5.6.3 Dynamische Planung

Bei der dynamischen Planung sind Besonderheiten zu beachten. Aufgrund der Komplexität des ursprünglichen Problems können zur Laufzeit meist nur hinreichende Planbarkeitstests durchgeführt werden, insbesondere gilt dies für explizite Planung ununterbrechbarer Aktivitäten. Der Aufwand für die Planung steigt (weit) mehr als linear mit der Anzahl der Aktivitäten. Selbst vermeintlich kleine Rekonfigurationen in einem expliziten Plan, wie die Verschiebung einer Startzeit, können umfangreiche Arbeiten nach sich ziehen, wenn ein hohes Garantieverhältnis angestrebt wird (neue Konflikte treten auf und erfordern weitere Umplanun-

gen). Wenn solche (umfangreichen) Umplanungen unterbunden werden, ist die effiziente dynamische Planung nur ein hinreichender Test.

Die explizite Planung unterstützt kurzfristige Entscheidungen. Beim Planbarkeitstest muß nicht immer gegen alle Aktivitäten verglichen werden, wie es bei der impliziten Planung notwendig ist. Der explizite Plan erlaubt einen einfachen Planbarkeitstest durch simples Nachschauen in den gespeicherten Strukturen, bei dem *schnell* ein hinreichender Test durchgeführt wird; möglicherweise ungerechtfertigt schlecht, wenn Umplanungen eine Ablehnung vermeiden könnten. In negativen Fall wird eine Garantie verweigert und die Anfrage muß später noch einmal gestellt werden.

Folglich entspricht die kalenderbasierte, explizite Planung den in 3.2.2 angegebenen Kriterien und unterstützt die gewählte zeitliche Steuerung des gesamten Systems.

## 5.6.4 Betriebsmittelvergabe

Die Vergabe der Zugriffsrechte für ein Betriebsmittel spiegelt die Unterscheidung in implizite und explizite Pläne wieder und ist der wichtigste Teil der Plandurchsetzung.

### 5.6.4.1 Zeitgesteuerte Vergabe

Die Zugriffsrechte werden zu festgesetzten Zeiten an die Aktivitäten vergeben und ihnen entzogen. Ein expliziter Plan speichert diese zeitgesteuerte Betriebsmittelvergabe.

### 5.6.4.2 Prioritätengesteuerte Vergabe

Bei der prioritätengesteuerten Vergabe erhält diejenige Aktivität das Zugriffsrecht, die die höchste Priorität hat. Es handelt sich um ein implizites Verfahren. Bei der Planerstellung werden Prioritäten an die verschiedenen Aktivitäten vergeben, um damit zur Laufzeit die Startzeiten festzulegen. Bei einer dynamischen Veränderung des Plans zur Laufzeit muß die Vergabe der Prioritäten *komplett* neu erfolgen.

*Konstante Prioritäten* werden bei Planerstellung vergeben und bleiben bis zu einer Umplanung unverändert. Die Rate Monotic (RM) Planung ist ein statisches Verfahren, das beim Einsatz konstanter Prioritäten optimal ist. Einer Menge periodischer Aktivitäten mit  $\Delta dl(A_i) = per(A_i)$  werden jeweils konstante Prioritäten umgekehrt proportional zur Länge der Periode zugewiesen [LL73].

*Dynamische Prioritäten* werden während der Laufzeit der Aktivität verändert. Die Planung nach Fristen (Earliest Deadline First, EDF) ist das bekannteste Beispiel, das aber von unterbrechbaren Aktivitäten ausgeht. Der Algorithmus vergibt zur Laufzeit die höchste Priorität an die Aktivität mit der kürzesten Frist

und hat einen Aufwand von  $O(n^2)$ . Die Fristenplanung ist optimal für einfache Betriebsmittel und der hinreichende und notwendige Planbarkeitstest ist [LL73]:

$$\mu = \sum \frac{et(A_i)}{per(A_i)} \leq 1$$

Eine zeitgesteuerte Nutzung kann man simulieren, indem die Prioritäten zeitgesteuert verändert werden.

## 5.7 Generisches Planungsmodell

In diesem Abschnitt stelle ich mein Konzept zur zeitgesteuerten Planung und kalenderbasierten Reservierung eines Betriebsmittels vor. Sowohl im globalen als auch im lokalen Management des Kommunikationssystems wird diese Reservierung zur Planung der TDMA-Zugriffe verwendet. Ausgehend von der reinen TDMA-Planung isolierter Zugriffszeiten werden auf den nächsten Seiten am Beispiel der HRT-Kommunikation<sup>10</sup> die Nachteile dieses Verfahrens erörtert und die Kanalüberlagerung als eine wesentliche Verbesserung vorgestellt. Mit diesem Schritt erreiche ich zunächst das Ziel Z3 der Einleitung (s. S. 31). Daraus ergibt sich als direkte Konsequenz die Möglichkeit, innerhalb einer Überlagerung ein anderes Zugriffsprotokoll zu nutzen (Ziel Z4).

### 5.7.1 Problematik der TDMA-Planung

Die TDMA-Planung der Netzzugriffe sichert durch die Vergabe fester Zeiten für die alleinige Nutzung die Verfügbarkeit des BM Netz. Die Datenübertragung ist allein auf diese garantierten Zeiten beschränkt. Weil aber die Spezifikation einer Datenübertragung zunächst eine Spezifikation der Datenmenge ist, muß diese auf die rein zeitliche Beschreibung der TDMA-Intervalle abgebildet werden, die in der Planung verglichen werden.

Die Leistungsanforderung muß für die **HRT**-Kommunikation dementsprechend die maximale Datenmenge bzw. die WCET der Sendetask, und letztlich die Länge des TDMA-Intervalls, beschreiben. In sie fließt die Unsicherheit sowohl über die maximale Datenmenge als auch über die Verzögerungen der Kommunikation, die trotz alleinigem Netzzugriff auftreten (z.B. Fehlererkennung in der physikalischen Schicht, Verzögerungen innerhalb des Netzadapters oder MAC-Schicht auf die man keinen Einfluß hat).

---

<sup>10</sup>Die Planung wird zwar auf die reinen Reservierungszeiten beschränkt, aber um die Problematik zu verdeutlichen, ist eine Betrachtung harter Zeitbedingungen sinnvoll. Die Probleme ergeben sich, unter Umständen weniger deutlich, in jeder Art der reinen TDMA-Planung und die mögliche Einsatzfähigkeit auch unter harten Bedingungen ist ein Ziel dieser Arbeit.

Eine Worst-Case-Spezifikation wird in der Praxis über der im Mittel benötigten Leistung liegen und somit bleiben immer reservierte Zeiten ungenutzt (geringe Planungsauslastung).

Diesen Nachteil muß man bei der HRT-Planung prinzipiell in Kauf nehmen, aber drei Punkte verstärken das Problem einer geringen Auslastung. Erstens ist die WCET-Spezifikation ungenau. Selbst bei einfachen Kommunikationsmustern (periodische Datenübertragung) ist das System meistens nicht mit der Genauigkeit zu analysieren, daß ein Maximum angegeben werden könnte; lediglich obere Schranken sind spezifizierbar. Die Sendetask ist noch einfach zu analysieren, aber das Zusammenspiel mit dem Netztreiber und der darunterliegenden Hardware ist, gerade auch beim Einsatz von Standardkomponenten, oft nur noch nach oben abschätzbar und die WCET „künstlich“ vergrößert.

Zweitens hat die Planung eine minimale, zeitliche Granularität, die die Spezifikationsmöglichkeiten einschränkt. Unterhalb einer zeitlichen Auflösung  $\Delta t$ , die der Slotgröße entspricht, unterscheidet die TDMA-Planung verschiedene Startzeiten  $t_1$  und  $t_2$  nicht voneinander. Eine untere Schranke für  $\Delta t$  ergibt sich aus der Genauigkeit der Uhrensynchronisation  $\Delta GT$  und aus der Übertragungsdauer der kleinsten Nachricht. Zusätzlich kann die Schranke wegen des notwendigen Verwaltungsaufwandes vergrößert werden: Jede explizit geplante Aktivität wird in einem zeitgesteuerten System explizit gestartet. Es ist daher nicht sinnvoll, Startzeitdifferenzen und Ausführungszeiten unterhalb von  $\Delta t$  zu spezifizieren.

Die Wahl von  $\Delta t$  ist eine Abwägung zwischen der erreichbaren Granularität und des zu jeder Startzeit notwendigen Verwaltungsaufwandes. Eine Erhöhung der Rechenleistung hilft hier nur bedingt weiter. Die Zeit für die Verwaltung sinkt zwar, aber auch  $|t_1 - t_2|$  verkleinert sich und das gleiche Problem stellt sich mit einer anderen Granularität erneut. Selbst wenn eine beliebig genaue Analyse möglich ist, geht also in die WCET die Granularität  $\Delta t$  ein und schränkt die Spezifikationsmöglichkeiten einzelner Aktivitäten ein.

Drittens gibt es darüber hinaus eine weitere, wichtigere Granularität in der *dynamischen* Planung. Die einzelne Spezifikation einer Aktivität läßt sich in der Regel selbst oberhalb von  $\Delta t$  weiter verfeinern, um so die obere Schranke für die WCET zu verbessern. Die daraus entstehenden Sequenzen zeitlicher Spezifikationen (aktive und inaktive Phasen) vergrößern die Eingabe der dynamischen Planung und stellen an die ‘einfachen’, hinreichenden Tests eine hohe Anforderung. Mit zunehmender Genauigkeit der Spezifikation (und damit Länge der Sequenz) steigt der Aufwand exponentiell und ein hinreichender Test lehnt zunehmend häufiger die Garantievergabe ungerechtfertigt ab: Die geplante Auslastung bleibt klein. Die drei Punkte zusammengenommen schränken die Möglichkeiten der dynamische Planung ein und führen zu niedrigen Planungsauslastungen:

Worst-Case-Spezifikationen sind oft in ihrer zeitlichen Abschätzung zu ungenau oder erfordern aufgrund ihres Umfangs eine sehr aufwendige dynamische Planung.

Als Beispiel seien zwei Aktivitäten  $A_1$  und  $A_2$  angenommen, die ununterbrechbar eingeplant werden müssen (s. Abbildung 5.6). Aus einer hinreichend genauen Analyse ist jeweils die aufgezeigte Nutzung eines (implizit) teilbaren Betriebsmittels bekannt:  $wcet(A_i) \leq 2\Delta t$ , aber die Zugriffe beschränken sich auf jeweils zwei Hälften mit einem minimalen Abstand. Eine verfeinerte Spezifikation kann  $A_{11}, A_{12}$  und  $A_{21}, A_{22}$  beschreiben, aber in der „herkömmlichen“ Planung bleibt es bei einer WCET-Summe von  $\leq 4\Delta t$ . Wie in der Abbildung aufgezeigt ist es möglich, eine Verschränkung (Überlagerung) der Ausführung für eine Reduzierung der *kombinierten* WCET zu nutzen und die Planungsgranularität  $\Delta t$  zu umgehen.<sup>11</sup> Teile der künstlich erhöhten WCET werden eingespart und man erhält  $wcet(A_1 \cup A_2) \leq 3\Delta t$  und mit einem kleinem maximalen Abstand zwischen den Aktivitätsteilen sogar  $wcet(A_1 \cup A_2) \leq 2\Delta t$ , wie in der Abbildung dargestellt.

Beim Applikationsentwurf können solche Kombinationen u.U. in Aktivitäten zusammengefaßt werden. In der dynamischen Planung treten diese Kombinationsmöglichkeiten erst zur Laufzeit auf und sind in der Planung auf ihre Konfliktfreiheit zu überprüfen. Dieser *Konflikttest* prüft die Verträglichkeit der Überlagerung – auch wenn  $\sum et(A_i) \leq \Delta t$  gilt, kann nicht jede beliebige Menge von Aktivitäten innerhalb eines Slots auf das BM zugreifen, weil keine explizite Kontrolle darüber ausgeübt wird (vgl. priority inversion, [ZA95]) – und bedarf dazu einer über die rein zeitlichen TDMA-Parameter hinausgehenden Spezifikation.

Ein Konzept, das die vorhersagbare Planung nicht auf die zeitlichen TDMA-Parameter beschränkt, hebt einige Nachteile des TDMA-Verfahrens auf. Einfache Spezifikationen, die zwar nur eine grobe zeitliche Genauigkeit besitzen, aber mit erweiterten Konflikttests eine Überlagerung ihrer spezifizierten Ausführungszeiten erlauben, reduzieren den Aufwand des Planungsalgorithmus' und erhöhen die Planbarkeit. Die Planungsauslastung in ihrem bisherigen Sinn bleibt zunächst wegen der groben zeitlichen Spezifikation niedrig, aber die nicht genutzten, reservierten Zeiten können von anderen Aktivitäten *geplant* genutzt werden. Die Überlagerung von Aktivitäten bietet die Möglichkeit, die Planungsauslastung über das Niveau der reinen TDMA-Planung zu erhöhen.

Ein Konflikttest mit erweiterbaren Parametern kann das Garantieverhältnis erhöhen, ohne daß dafür eine komplexe zeitliche Spezifikation notwendig ist.

## 5.7.2 Konzept

Für die angesprochenen Probleme der dynamischen TDMA-Planung erarbeite ich in diesem Kapitel die grundlegenden Lösungen, die teilweise erst in Erweiterungen des Kommunikationssystems zum Einsatz kommen können, aber die Erweiterbar-

---

<sup>11</sup>Die Aktivitäten werden nicht unterbrochen, sondern geben ihrerseits die Kontrolle ab: kooperative Unterbrechung (engl. cooperative preemption).

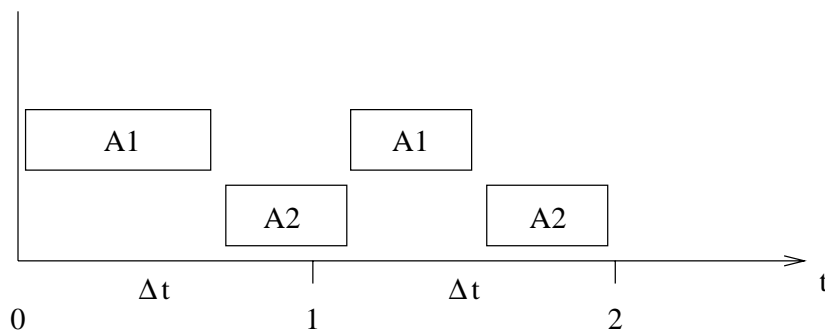


Abbildung 5.6: WCET-Reduzierung durch Nebenläufigkeit

keit und Einsatzfähigkeit des Entwurfs demonstrieren. Folgende Punkte sollen in den Entwurf des Planungsmodells einfließen:

1. Die Planung präsentiert sich nach außen als eine explizite Verwaltung der linearen Zeitachse mit einer Sicht auf freie und belegte Slots.
2. Reservierungen auf der Zeitachse besitzen jeweils eine eigene Spezifikation und einen Konflikttest, der bei der Planung Kollisionen im BM-Zugriff feststellt.
3. In ein bestehendes System sollen neue Aufgaben mit neuen Konflikttests, die über die einfachen TDMA-Parameter hinaus gehen, integriert werden können. Um eine hohe Erweiter- und Planbarkeit zu erreichen, können Intervalle mehrfach genutzt werden (Überlagerung).

Die Punkte beschreiben die zu erfüllenden Aufgaben. Punkte 1 und 3 erfordern eine möglichst allgemeine Verwaltung der Zeitachse, die die Parameterwahl nicht konstruktionsbedingt einschränkt: keine Beschränkung der zulässigen Perioden oder der Planung in der Zukunft (zeitlich begrenzte Reservierungen).

Die im zweiten Punkt geforderten individuellen Konflikttests lockern die oben angesprochenen Einschränkungen der TDMA-Planung. Sie ermöglichen die Überlagerung von Aktivitäten und die Kombination verschiedener Leistungsparameter in *einer* Reservierungsstruktur (Punkt 3). Es können feinere zeitliche Parameter mit einem anderen Algorithmus (Konflikttest) oder applikationsseitige, semantische Parameter in die Planung einbezogen werden. In der Netzwerkplanung sind innerhalb überlagerter Intervalle Konflikttests einsetzbar, die die Planung eines alternativen Zugriffsprotokolls übernehmen. Jede Überlagerung ist von allen beteiligten Aktivitäten genau zu prüfen, um die Garantien der Planung nicht zu gefährden.

### 5.7.2.1 Verwaltung der Zeitskala

Die Zeit ist in Zeitscheiben konstanter Länge, *Slots*, unterteilt, die die Granularität der Reservierung bzw. Slotverwaltung darstellt. Wenn man zunächst davon

ausgeht, daß einem Slot höchstens eine Aktivität zugeordnet werden kann, ist die Zeitskala als eine unendlich große Tabelle  $T$  anzusehen, in der zu jedem Slot ein Verweis auf eine Aktivität stehen kann.  $T$  ist der explizite Plan.

Sei  $\mathcal{A} = \{\varepsilon, A_1, A_2, \dots\}$  eine Menge von Aktivitäten (einschließlich einer leeren Aktivität  $\varepsilon$ ) und die Slots  $\mathcal{S} = \mathbb{N} = \{1, 2, 3, \dots\}$  werden mit den natürlichen Zahlen numeriert.  $T = \pi = (a_i)_{i \in \mathcal{S}} \in \mathcal{A}^{\mathcal{S}}$  ist eine explizite Beschreibung eines Planes  $\pi$ , eine Abbildung

$$\begin{aligned} \pi : \mathcal{S} &\rightarrow \mathcal{A} \\ t &\mapsto e_t \cdot \pi = a_t \end{aligned}$$

wobei  $e_i$  der kanonische  $i$ -te Basisvektor  $(0, \dots, 1, 0, \dots)$  ist (siehe z.B. [Bri83]). Die Funktion ist zu beliebigen Zeitpunkten auswertbar;  $\pi$  ist auf ganz  $\mathcal{S}$  definiert.

Es ist nicht möglich, die komplette Zeitskala  $T$  explizit im Computer zu speichern, selbst wenn man sich auf  $\mathcal{A} \setminus \{\varepsilon\}$  beschränkt, da eine einzige unendlich periodische Aktivität  $A$  ( $end(A)$  ist nicht spezifiziert) die Speicherung von unendlich vielen einzelnen Instanzen erfordern würde, wie bereits auf Seite 124 angesprochen.  $\pi^{-1}$  ist aus gleichem Grund genauso unhandlich und bietet auch nicht die Unterstützung für die dynamische Planung, wie in 5.6.3 gefordert.

Für die Implementierung expliziter Planungsalgorithmen, also solchen, die einen expliziten Plan erstellen, benötigt man eine Beschreibung von  $\pi$ , die eine schnelle Auswertung erlaubt und eine effiziente Speicherung und Verarbeitung im Computer ermöglicht. Ein Planungsalgorithmus muß auf  $\pi(t) \stackrel{?}{=} \varepsilon$  testen<sup>12</sup> und ggf. die Startzeit von  $\pi(t) = A_i$  verändern, um Platz für einen neuen Eintrag zu machen, oder an einer anderen Stelle  $t'$  testen; diese Konflikterkennung und -behebung ist der wichtigste Teil des Algorithmus'.

Folgende Strukturen kommen für die Verwaltung der Zeitskala in Betracht:

- lineare Liste

Eine lineare, geordnete Liste mit den Instanzen der Aktivitäten  $A_i$  als Knoten und den jeweiligen Startzeiten als Sortierungsschlüssel kann verwendet werden. Die leere Aktivität  $\varepsilon$  muß natürlich nicht gespeichert werden.

Das sukzessive Ermitteln der *nächsten* Aktivität in der Liste kann mit konstantem Aufwand  $O(1)$  geschehen. Aber der zufällige Zugriff mit  $O(n)$  muß beim Planen durchaus öfters benutzt werden.

Dazu kommt noch die Schwierigkeit, periodische Aktivitäten zu speichern. Es kann nicht für jede Instanz ein Eintrag in die Liste genommen werden. Die Liste kann nur bis zu einem bestimmten Zeitpunkt in der Zukunft vollständig sein, die Beschreibungen periodischer Aktivitäten können nur

---

<sup>12</sup>Ein allgemeiner Konflikttest kann hier die Verträglichkeit mit evtl. vorhandenen Aktivitäten prüfen.

Schritt für Schritt aufgelistet werden (z.B. kann die Liste bis zum kleinsten gemeinsamen Vielfachen der Perioden vervollständigt werden). Andere Mechanismen für den Planungstest sind hier einfacher (siehe Punkt „Ring“ unten).

- Baum  
Um die Zugriffszeiten der Listenstruktur zu verbessern, kann die Sortierung der Einträge z.B. in einer Baumstruktur helfen. Damit hat man die Probleme verkleinert, aber nicht behoben.
- Hashing  
Grundsätzlich kann man Hashingmethoden anwenden, um  $\mathcal{S}$  auf einen algorithmisch vernünftigen Rahmen zu beschränken. Wenn man nicht von einer einheitlichen Ausführungsdauer ausgeht, ergeben sich allerdings Schwierigkeiten in der Reservierung aufeinanderfolgender Slots. Eine Hashfunktion, die die Ordnung von  $\mathcal{S}$  (teilweise) bewahrt, umgeht diese Probleme. Zum Beispiel bieten Ringe diese Möglichkeit.

- Ring  
Ein periodisch wiederholter Bereich von  $n$  Slots bildet einen Ring der Größe  $n$ , so daß  $\mathcal{S} = \{i + kn \mid k \in \mathbb{N}_0\}$  ist. Dies entspricht einer Hashingfunktion  $t \mapsto i, i = t \bmod n$ . Solche Ringe beschreiben mit einem Eintrag pro Slot Tabellen  $T$ , die eine Teilmenge aller möglichen Pläne  $\mathcal{A}^{\mathcal{S}}$  darstellt.

Die Planung unendlicher, periodischer Aktivitäten wird durch diese Struktur gut unterstützt. Die einzelnen Instanzen sind im Ring einzutragen, bis man auf einen belegten Slot oder, im Falle einer erfolgreichen Planung, auf eine Instanz derselben Aktivität stößt. Die Anzahl der erforderlichen Umläufe  $k$  im Ring ergibt sich aus:  $p \mid kn$ , die Periode  $p$  ist ein Teiler von  $kn$ .

Der Aufwand eines Zugriffs auf die Ringeinträge (Slots) ist konstant. Die Planung kann kurzfristige Entscheidungen sehr schnell treffen, wenn keine Umplanung bestehender Einträge vorgenommen wird.

Eine sehr einfache Konfiguration ergibt sich aus der Beschränkung aller Perioden auf Teiler von  $n$ . Zusammen mit einer linearen Liste für aperiodische Aktivitäten ist dies die einfachste Lösung für eine komplette Verwaltung von  $\mathcal{S}$ .

Wenn die Perioden nicht derart eingeschränkt werden sollen oder die gesamte Planung (auch der aperiodischen Reservierungen) in dieser Struktur erfolgen soll, müssen die Ringeinträge mehrere Slots aufnehmen.

Ich werde den Ansatz verfolgen, die Zeitachse mittels Hashings auf einen Ring  $\mathcal{R}$  abzubilden. Jeder Slot  $t$  wird nach  $t' = t \bmod n$  abgebildet;  $n$  ist dabei die Ringgröße, die periodisch wiederholt die gesamte Zeitskala abdecken kann. Die

Ringeinträge  $\mathcal{R}[i]$  entsprechen den Slots  $\{i, i + n, i + 2n, \dots\} = \{i + kn\}$ ;  $i$  nennt man Ringoffset oder -index. Die Mehrfachbelegung von  $\mathcal{R}[i]$  ist durch einen Ver-

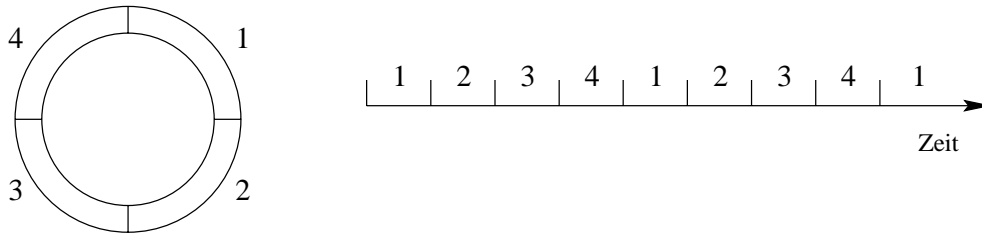


Abbildung 5.7: Ring als Zeitskala

gleich der genutzten Slots – der jeweiligen Ringrunden  $k$  – auf gegenseitige Verträglichkeit zu testen.

### 5.7.2.2 Reservierungseinheiten – Konflikttest

Die Hauptaufgabe der Reservierung ist die zeitliche Festlegung der Zugriffe auf bereitgestellte Betriebsmittel und damit einer Positionierung der Zugriffe innerhalb der Leistungsspezifikation der Aktivität (PE). Die Möglichkeiten, die durch die Spezifikation aufgespannt werden, konkretisieren sich in der expliziten Reservierung: die Startzeit innerhalb eines möglichen Bereichs wird festgelegt, die Gesamtausführungszeit wird mit Einträgen auf der Zeitachse dargestellt usw. Die Parameter aus 5.4 werden für die zeitliche Spezifikation der Einträge genutzt, wobei die Ausführungszeiten nicht interpretiert und allgemein nur Reservierungszeiten  $rt(A)$  statt  $et(A)$ ,  $eet(A)$  oder  $wcet(A)$  verwendet werden.

Die Einträge auf der Zeitskala, die für eine Reservierung vorgenommen werden müssen, sind die Einheiten der Reservierung, die die Slots der Zeitachse belegen. Aus solchen *Reservierungseinheiten* RE sind die Einheiten der Planung PE zusammengesetzt.  $\pi(t)$  gibt also genauer eine RE zurück, über die auf die Planungseinheit (Aktivität) geschlossen werden kann.

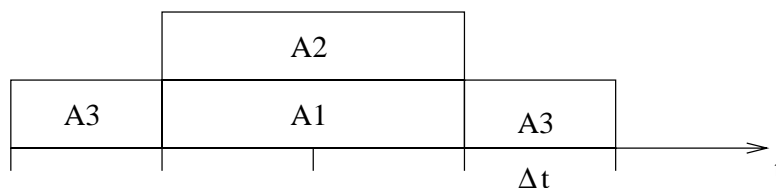


Abbildung 5.8: Reservierungseinheiten beschreiben geplante Aktivitäten

In Abbildung 5.8 sind zwei Beispiele verdeutlicht.  $A_1$  und  $A_2$  sind die oben erwähnten, überlagerten Aktivitäten. Sie verwenden für die Planung ununterbrechbar zwei Slots der Länge  $\Delta t$ , die sie aber aufgrund eines negativen Konflikttests nebeneinander nutzen können. Sie werden jeweils durch eine RE der Länge

2 aufgebaut, die das BM nur zu 50% anteilig nutzen. Die Zusammensetzung einer gesplitteten Aktivität  $A_3$  aus zwei REen wird ebenfalls in dieser Abbildung veranschaulicht.  $st(A_3)$  ist der Start der ersten RE und  $ct(A_3)$  ist das Ende der zweiten.

Der *Konflikttest* ist Teil der jeweiligen Reservierungseinheit und entscheidet während der Planung, ob eine Reservierung in einem bestimmten Slot möglich ist. Neben der Einhaltung der „äußeren“ zeitlichen Spezifikation der RE – die Ausführungszeit muß zwischen  $est(RE)$  und  $dl(RE)$  liegen – ist insbesondere festzustellen, ob ein Konflikt um das BM mit einer im gleichen Slot liegenden RE auftritt. Der Test stellt damit einen integralen Bestandteil des Planungsalgorithmus' dar. Die Tests arbeiten über eine festgelegte Schnittstelle, so daß unterschiedliche Versionen und unterschiedliche Algorithmen in der Reservierungsstruktur kombiniert werden können.

### 5.7.2.3 Planungsalgorithmen

Die dynamische Planung in der expliziten Datenstruktur fügt inkrementell neue Reservierungen neben den bestehenden ein. Die Aufgabe des Planungsalgorithmus' ist es, die zu planenden REen über der Zeitachse zu positionieren und die Verträglichkeit mit evtl. bestehenden Reservierungen durch die Konflikttests zu sichern.

Die RE verfügen über eine einheitliche Schnittstelle zu den Leistungsinformationen und dem dahinterliegenden Algorithmus; sie haben ein „Verhalten“ innerhalb ihrer Spezifikation (Verschieben, Unterbrechen, allgemein: Konflikttests). Es existieren also keinen einheitlichen Einträge mit einem passenden Algorithmus, sondern hinter jedem Eintrag steht ein (individueller) Algorithmus, der die Navigation innerhalb der Spezifikation übernimmt. Dieser Ansatz ist für eine Spezialisierung der Konflikttests notwendig, um sie an die Anforderungen verschiedener Aktivitäten anzupassen, und erlaubt den Einsatz unterschiedlicher Planungsalgorithmen auf disjunkten Slotintervallen.

Die Handhabung und Erweiterung unterschiedlicher Leistungsparameter unter Beibehaltung einer strikten zeitlichen Abgrenzung ist deshalb leicht möglich, weil die Kontrolle einer eingeplanten RE bei dieser selbst bleibt und nicht von außen verändert wird: eine unabdingbare Forderung für die Integration verschiedener Ansätze in einem Echtzeitsystem. Neue Konflikttests werden in der Regel daher mit alten kollidieren, aber gerade das sichert die Vorhersagbarkeit der Planung.

Diese Vorgehensweise wirft eine neue Frage in der Planung auf: Wie verträglich sind die Kombinationen von unterschiedlichen Algorithmen und Konflikttests? Es kann zu Einbrüchen im erreichbaren Garantieverhältnis kommen, wenn die Abstimmung zwischen den Algorithmen schlecht ist. Diese Problematik liegt außerhalb der Diplomarbeit und sollte in weiterführenden Arbeiten untersucht werden.

### 5.7.3 Aufbau der Reservierung

Zu der Reservierung in einer Betriebsmittelplanung – Planung *eines* Betriebsmittels – gehören folgende Teile:

1. Nach außen präsentiert sich die Reservierung als eine lineare Zeitskala, die die Navigation zwischen den geplanten Aktivitäten und den beanspruchten Slots erlaubt. Man erhält eine abstrakte Sicht auf einen unendlich langen Zeitstrahl und die darauf platzierten reservierten Aktivitäten. An dieser Stelle ist die Verwaltung der Slots nicht sichtbar.

Die Reservierung verwaltet die Informationen eines expliziten Plans, der für die Planung und deren Durchsetzung notwendig ist. Diese Reservierungen haben keine eigenständige Aktivität und sind auf Mechanismen zur Plandurchsetzung angewiesen.

Die Klasse `ResSched` wird den Plan repräsentieren.<sup>13</sup>

2. Für die interne Realisierung der unendlich langen Zeitskala wird ein Ring benutzt. Der Ring  $\mathcal{R}$  deckt  $n$  Slots ab, die sich periodisch wiederholen. Um den Abstraktionsgrad nicht zu schnell abfallen zu lassen, setzte ich an dieser Stelle *Manager* für die Slotverwaltung ein (`ResSchedEntryManager`). Damit bleibt man eine Abstraktionsstufe über den einzelnen Slots, so daß man durch die Manager Gruppen von Slots betrachten kann, ohne mit diesen direkt umzugehen. Die über den Slots gelegene Verwaltungsstruktur enthebt `ResSched` von dem zugehörigen Aufwand und erlaubt eine feinere Abstimmung auf die Anforderungen des Systems (heuristische Verfahren können die durchschnittliche Auslastung der Manager berücksichtigen, bei unterschiedlichen Auslastungen können entsprechende Slotverwaltungen innerhalb der Manager die Zugriffe beschleunigen und letztlich können diese bei Bedarf geteilt bzw. zusammengefaßt werden).

Die `ResSchedEntryManager` registrieren die reservierten Aktivitäten in „ihrem“ Verwaltungsintervall, für das sie jeweils zuständig sind. Die Zuordnung von Reservierungen und Slots wird in diesen Managern erledigt und aus diesen ist der Ring aufgebaut<sup>14</sup>.

3. Die zu reservierende Instanz einer Aktivität wird durch die Reservierungseinheiten RE beschrieben. Diese Einheit der Reservierung, dargestellt durch die Klasse `ResSchedEntry` (oder `RSE`), besitzt nicht nur eine Beschreibung der eigenen aktiven Phasen, die im nächsten Punkt erläutert wird, sondern bietet auch eine Schnittstelle zu dem eigentlichen Planungsalgorithmus.

---

<sup>13</sup>Zur Bezeichnung verwende ich hier Begriffe der Sprache C++.

<sup>14</sup>Insbesondere können diese Manager auch für die Realisierung einer dünnbesetzten Liste als Alternative zum Ring benutzt werden; s. S. 124.

Die bisherige Struktur ist für die reine Verwaltung der Information geeignet. Auf dieser Datenstruktur können durchaus verschiedene Algorithmen für die Platzierung der Reservierungseinheiten genutzt werden, die über ihr (gegenseitiges) Verhalten integriert werden. Dies resultiert in unterschiedlichen Implementierungen der abstrakten Klasse `RSE`.

4. Jede RE besitzt eine Slotspezifikation, die die Slots, in denen sie aktiv ist, und damit die Konkretisierung der Aktivitätsspezifikation beschreibt.

Zusammen mit dieser Spezifikation, die über die reine TDMA-Spezifikation hinausgehen kann, wird ein Teil des Planungsalgorithmus' realisiert. Der Konflikttest wird hier durchgeführt: Kontrolle der Einhaltung der zeitlichen Parameter, der BM-Konflikte mit anderen REen und somit auch die Kontrolle der Mehrfachbelegung der Ringeinträge  $\mathcal{R}[i]$ .

Die zeitlichen Leistungsparameter aus 5.4 bilden die minimale Grundlage für gegenseitige Interaktion, mit der jede RE umgehen kann. Weitergehende Parameter sind mit neuen bzw. erweiterten Implementierungen, sowohl der entsprechenden Aktivitätsspezifikation als auch der entsprechenden RE mit dem dazugehörigen Planungsalgorithmus, verbunden. Im Zweifelsfall bilden die zeitlichen Parameter aber die Basis für alle Absprachen.

5. Ein weiterer Teil, der etwas außerhalb dieser Reservierungsstruktur liegt, ist der Planungsauftrag (`RSchedCommission`). Die Applikation stellt damit die Planung einer Aktivität in Auftrag. Der Auftrag ist ein Teil des Planungsverfahrens und muß die applikationsseitige Spezifikation des QoS in die entsprechenden Reservierungseinheiten und Aktivitätsspezifikationen für die Reservierung übersetzen. So werden die Spezifikationen durch die obige Hierarchie sukzessive bis in den Punkt 4 auf die Slot- bzw. Ringebene umgesetzt.

#### 5.7.4 Reservierung

Die Klasse `ResSched` präsentiert den Plan nach außen als lineare Liste von Reservierungen. Alle Reservierungen lassen sich anhand ihrer Startzeiten ordnen und erfüllen damit die Voraussetzung für die Darstellung in der linearen Liste. Die Hauptaufgabe besteht in der Bereitstellung von Navigationsmöglichkeiten durch die Einträge und in der internen Verwaltung der `ResSchedEntryManager`, die ihrerseits die Zuordnung von Reservierungseinheiten zu Slots durchführen. Alle Slots der Zeitachse sind ansprechbar und die Planbarkeit in die Zukunft ist nicht eingeschränkt. Reservierungen unterliegen in der Wahl der Slots keinen Beschränkungen und weichen somit nach außen nicht wegen Implementierungsfragen von der abstrakten Sicht einer linearen Zeitachse ab.

### 5.7.4.1 Navigation

Für die Navigation durch die Reservierungen benutze ich das Konzept der Iteratoren (siehe Anhang A). Der bidirektionale Iterator als verallgemeinerter Zeiger erlaubt den Zugriff auf einen Eintrag bzw. eine Reservierung auf die gezeigt wird. Die möglichen Schritte zur nächsten bzw. zur vorhergehenden Reservierung liefern auf kanonische Weise die Sicht auf eine lineare, geordnete Liste.

Es stehen folgende Funktionen zur Verfügung, um über die Klasse `ResSched` einen Iterator zu initialisieren:

- `begin()`, `begin(n)` und `end()` Die beiden Funktionen liefern einen Iterator auf den frühesten Eintrag nach Null bzw. nach einem vorgegebenen Slot `n`. `end()` liefert einen Iterator auf ein vordefiniertes letztes, leeres Element, das allerdings aus der Liste nur erreichbar ist, wenn keine unendlich periodische Aktivität geplant ist.

Dieser Abschluß der Reservierung birgt noch einige Unstimmigkeiten, die noch zu untersuchen sind. Von diesem letzten Element sollte eigentlich ein Übergang zu dem Vorgänger möglich sein, aber wie verhält es sich mit den unendlich periodischen Aktivitäten in diesem Fall? In der Praxis macht hier ein Zurückgehen keinen Sinn und es müssen andere Mechanismen gefunden werden. Wenn man davon ausgeht, daß man nur zwischen bekannten Iteratoren vor *und* zurück geht, tritt dieser Fall allerdings erst gar nicht auf.

- `find(ResSchedEntry[,slot])` und `find(RSchedCommission[,slot])`  
Das Finden eines Eintrags zu einem vorgegebenen RSE bzw. Planungsauftrag, evtl. nach einem Zeitpunkt `slot`, gehört ebenfalls zu den bereitgestellten Möglichkeiten. Diese sind insbesondere für die eigentlichen Algorithmen notwendig.

Über das Iteratorobjekt selber ist der Vorgänger und Nachfolger erreichbar.

- `++iterator`, `--iterator`  
In Anlehnung an die C++-Zeiger, kann mit dem Postinkrement und -dekrement der Iterator auf die nächste bzw. vorherige RE in der Liste gesetzt werden; oder auf `end()` falls diese nicht existieren.

Die Planungsalgorithmen können ihre Slotzugriffe auf diese Funktionen beschränken, wenn sie aperiodische Aktivitäten reservieren. Die Planung periodischer Aktivitäten erfordert Kenntnis über den internen Aufbau der Slotverwaltung, weil die REen nicht vollständig explizit gespeichert werden können.<sup>15</sup> Hier ist die Kenntnis der Ringgröße  $n$  und ein Zugriff auf die Ringeinträge  $\mathcal{R}[i]$  erforderlich.

---

<sup>15</sup>Es sind an dieser Stelle sicher weitere Abstraktionen der Verwaltung möglich, aber der Overhead steigt u.U. erheblich.

### 5.7.4.2 Interna

Intern müssen Datenstrukturen die Einträge auf der beliebig lange Zeitskala verwalten. Der beabsichtigte Ring der Größe  $n$  wird durch eine Tabelle von  $\frac{n}{m}$  Slotmanagern `ResSchedEntryManager` gebildet, die zunächst jeweils einen festen, gleich großen Bereich von  $m$  Slots verwalten.

Der Zugriff auf einen Slot kann somit durch eine einfache modulo-Operation an den entsprechenden Manager weitergegeben werden. Ein Iterator wird analog in den Managern definiert und wird zur Adressierung eines `ResSchedEntry` benutzt. Beim linearen Durchlaufen der Reservierungen muß zunächst innerhalb eines Managers und ggf. im nächsten Manager der Iterator weitergesetzt werden.

### 5.7.5 Slotverwaltung

Die Klasse `ResSchedEntryManager` verwaltet als Slotmanager die Slots  $[\nu, \nu + m) + kn$ ;  $n$  ist die Ringgröße,  $k \in \mathbb{N}_0$ ,  $m|n$ ,  $m|\nu$ ,  $\nu < n$ . Ein Slot ist eindeutig durch den Ringoffset  $i$  und den Zähler für den Ringumlauf  $k$  bestimmt. Zu jedem  $i$  gehört eine Liste von Reservierungseinheiten, die zu den Zeitpunkten  $t \equiv i \pmod{n}$  aktiv sind. Bei der Einplanung müssen die Aktivitätsspezifikationen auf einen BM-Konflikt in diesen  $t$ 's getestet werden.

Die Verwaltung der  $m$  Ringeinträge, also der Abbildung  $\pi$  im Bereich  $[\nu, \nu + m) + kn$ , kann z.B. durch eine Liste oder ein Array erfolgen, das mehrere Einträge pro Slot erlaubt. Bei der Verwaltung wird generell nur mit Ringoffsets gearbeitet, Slotnummer modulo Ringgröße. Daher müssen die Konflikttests zwischen den REen die Slots  $\mathcal{S}$  betrachten, damit nicht fälschlicherweise zeitliche Konflikte für unterschiedliche Ringumläufe  $k$  erkannt werden.

Reservierungseinheiten, die sich über die Grenzen von Slotmanagern erstrecken, erhalten einen Eintrag in allen beteiligten Managern. Für die Navigation in den gespeicherten Einträgen werden Iteratoren in analoger Weise zu `ResSched` genutzt.

### 5.7.6 Reservierungseinheit

Die Reservierungseinheit (RE) umfaßt eine Beschreibung der Aktivitätszeiten bzw. der aktiven Slots und eine Schnittstelle zum Planungsalgorithmus. In Abschnitt 5.7.2.2 wurden bereits die grundlegenden Konzepte eingeführt: Mechanismen zur Planung und Verwaltung der Reservierungseinheiten, die auf einem Planungsalgorithmus aufbauen, sind über eine Schnittstelle der RE verfügbar.

Mit der Bindung des Planungsalgorithmus' an die einzelnen planbaren Einheiten erhält man eine Art „selbständige“ Einheiten. Statt nur eine Spezifikation und von außen *einen* Planungsalgorithmus an diese zu binden, erhält man durch diesen Ansatz die Möglichkeit, mehrere Algorithmen in einem Planungsverfahren einzusetzen. Die Handhabung und Erweiterung wird erleichtert. Die Kontrolle

einer eingeplanten RE bleibt bei dieser selbst und kann nicht von außen gestört werden: eine unabdingbare Forderung für die Integration verschiedener Ansätze in einem Echtzeitsystem.

Die Schnittstellen zum Planungsalgorithmus der Reservierungseinheit umfassen konkret folgende Aufgaben, die zum Plazieren auf der Zeitachse benutzt werden bzw. das „Verhalten“ bereits geplanter REen beschreiben:

- Ein Überschneidungs- oder *Konflikttest* entscheidet, ob die zeitlichen Bedingungen eingehalten werden und ob zwei gegenübergestellte RE in Konkurrenz um die Betriebsmittelzugriffe treten. Aktivitäten, die im gleichen Ringoffset liegen (sollen), müssen diesem Test genügen (Konflikttest negativ). Sie sind entweder in verschiedenen Ringumläufen aktiv oder ihre Nebenläufigkeit kann von *beiden* REen geduldet werden (Überlagerung). Bei einem BM-Konflikt kann mit den unten aufgeführten Funktionen versucht werden, diesen zu lösen.

Die Konflikttests sind nicht symmetrisch! In zwei REen können verschiedene Tests implementiert sein. Wenn unterschiedliche Algorithmen die Überschneidung in einem Slot bewerten sollen, können sie zu unterschiedlichen Ergebnissen kommen. Zwei gegenübergestellte Reservierungseinheiten müssen also immer mit zwei Tests auf Konflikte überprüft werden.

Aus der Sicht der Anwendung entscheiden die an dieser Stelle eingesetzten Algorithmen über die Teilbarkeit des Betriebsmittels (vgl. 5.3.2). Die Algorithmen können bestimmen, ob ein Slot oder ein Intervall von mehreren Aktivitäten genutzt werden kann. Man erhält eine implizite Teilbarkeit des Betriebsmittels, wenn die Summe der Ausführungszeiten in den Slot bzw. in das betrachtete Intervall der Überschneidung „paßt“. Hiermit ergeben sich Möglichkeiten, die in 5.3.3 angegebenen Multiplexverfahren implizit zu realisieren, ohne daß eine genaue zeitliche Festlegung der Aktivitätsabfolge notwendig ist. Die Kooperation der REen in der impliziten Absprache während der Überlagerung beschränkt sich auf die RE-seitige Einhaltung der Spezifikation (insbesondere beim statistischen Multiplexing). Der Konflikttest muß wie in der impliziten Planung sicherstellen, daß keine BM-Konflikte die Einhaltung der zeitlichen Bedingungen gefährdet.

Zur Lösung der in 5.7.1 erarbeiteten TDMA-Problematik setzt man eine implizite Planung auf kleinen Zeitskalen ein.

Der Konflikttest ist (im weiteren Sinne) zur Konfliktvermeidung geeignet. Erkennt der Test aber einen Konflikt, so kann man folgende Funktionen der RE zur Lösung einsetzen; sie dienen allgemein zum Plazieren und Navigieren der REen:

- Das *Verschieben* der Startzeit einer Planungseinheit bzw. der dazugehörigen Reservierungseinheit(en) stellt die grundlegendste Methode zur Planerstellung dar.

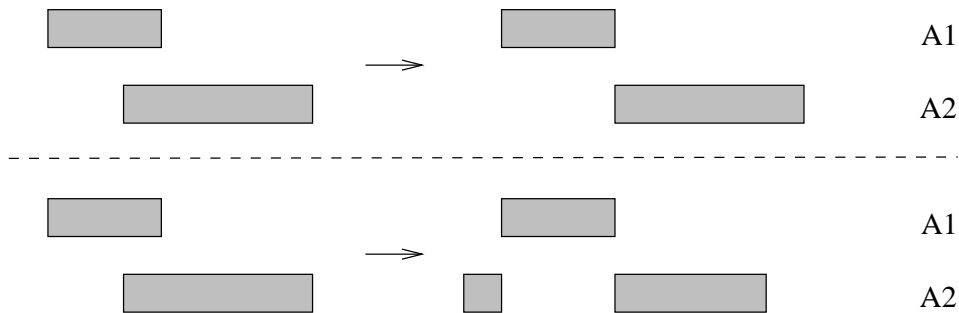


Abbildung 5.9: Verschiebung und Aufteilung als Konfliktlösung

lung dar. Die RE wird absolut in einem Slot positioniert, um die Spezifikation zu konkretisieren. Das Verschieben einer bereits eingeplanten PE (Umplanung durch Neuplanung und Löschen des alten Eintrags) geschieht relativ zu ihrem bisherigen Startpunkt  $st(RE)$ <sup>16</sup> auf der Zeitachse.

Abhängigkeiten speichert man in den RE und kontrolliert deren korrekte Einhaltung vor einer Verschiebung.

- Das *Aufteilen* oder Splitten einer RE in einzelne, kleinere RE beschreibt eine geplante Unterbrechung der Planungseinheit und simuliert die Teilbarkeit des BMs explizit. In zeitgesteuerten Systemen beschränkt sich die Unterbrechungsbehandlung auf solche geplanten Aufteilungen in ununterbrechbare REen. Die implizite Absprache in einer Überlagerung ist eine gewollte Ausnahme, um die Nachteile zu reduzieren.
- Weitere Verfahren zur Konfliktlösung können in das System eingeführt werden. Einzelne Reservierungseinheiten können zusätzliche Funktionen erhalten. Will man neue Funktionalität in die gesamte Planung konsistent einführen, müssen bestehende REen unabhängig von ihrer bisherigen Leistung um geeignete Funktionen mit sinnvollen Defaultergebnissen erweitert werden.

### 5.7.7 Aktivitätsbeschreibung

Die Aktivitätsbeschreibung bzw. die Slot- oder Aktivitätsspezifikation ist ein eigenständiger Teil der Reservierungseinheit und bietet die Funktionalität für den Umgang mit zeitlichen Beschreibungen der aktiven Phasen einer Planungseinheit. Durch diese zeitliche Spezifikation wird der Raum für die Planungsmöglichkeiten aufgespannt, in dem die RE bei einer erfolgreichen Planung liegen muß.

Zunächst wird durch die allgemeinen Parameter aus 5.4 eine einheitliche Schnittstelle definiert, die um weitere Parameter ergänzt werden kann. Dieser

<sup>16</sup>*est, st, wct, dl* benutzt man zur Bezeichnung von Slots und Zeiten gleichermaßen.

kleinste gemeinsame Nenner erlaubt eine Integration verschiedener Ansätze, aber erweiterte Konflikttests benötigen weitere Parameter. Ein deutliches Beispiel ist das statistische Multiplexing: Mit den allgemeinen Parametern umfaßt der aktive Bereich das gesamte in Frage kommende Intervall und läßt keinen Platz für andere Aktivitäten. Mit der Kenntnis der speziellen Parameter für das statistische Multiplexing ist eine genauere Beschreibung der aktiven Phasen möglich; wenn auch in diesem Fall nur statistische Werte zur Verfügung stehen.

Die Aktivitätsbeschreibung spannt den Raum der Planungsmöglichkeiten auf. Indem man die RE auf der Zeitachse plaziert und damit die Startzeit *st* festlegt, plant man die BM-Zugriffe ein und kann eine Garantie darüber vergeben.

### 5.7.8 Reservierungsauftrag

Ein Reservierungsauftrag nimmt eine applikationsseitige Beschreibung einer Echtzeitaufgabe entgegen und initiiert, nach einer Vorverarbeitung der Parameter, den eigentlichen Planungsalgorithmus.

Eine anwendungsbezogene Parametrisierung einer Aufgabe (vgl. 5.5.1) wird in eine Folge von Reservierungseinheiten mit einer TDMA-Parametrisierung auf der Ebene der Ringverwaltung übersetzt. Die REen werden dann, über die ihnen eigene Funktionalität (s.o.), in den Ring eingefügt.

Ein Teil des Planungsalgorithmus' ist im Reservierungsauftrag enthalten. Die Spezifikation der REen ist der erste Schritt der Planung. Eine applikationsseitige TDMA-Spezifikation muß man nicht interpretieren, aber bereits die QoS-Parameter aus 5.5.1 und insbesondere weitere Parameter zur Kontrolle von Überlagerungen erfordern an dieser Stelle eine Parameterumsetzung, die zum Beispiel auf die aktuelle Auslastung oder auf Überlagerungsmöglichkeiten in bereits geplanten Aktivitäten<sup>17</sup> eingehen kann. Spätestens im zweiten Schritt, der die Suche im Raum der Planungsmöglichkeiten startet, muß der Planungsalgorithmus ansetzen. Über die Iteratoren der Reservierung bzw. über einen direkten Ringzugriff mittels der Slotverwaltung `ResSchedEntryManager` wird die Zeitachse untersucht und eine freie Position für die Konflikttests gesucht. Bei einem Fehlschlag (positiver Test und keine Umplanung möglich) verschiebt man sukzessive die RE im Planungsraum, bis man einen Erfolg oder Mißerfolg an die Applikation melden kann. Diesen einfachen Algorithmus werde ich in der Planung einsetzen. Es ist eine Ausgangsbasis für Erweiterungen, sowohl des Reservierungsauftrags als auch des Konflikttests in den Reservierungseinheiten.

Der Reservierungsauftrag bleibt als Repräsentant der planbaren Einheit weiter für die REen sichtbar und kann zum Beispiel synchronisierende Maßnahmen zwischen ihnen unterstützen. Die Reservierungseinheiten sind über den Auftrag miteinander verbunden.

---

<sup>17</sup>Erweiterte Konflikttests sind besonders erfolgversprechend, wenn sie auf „gleichartige“ REen angewandt werden, weil eine sinnvolle Koordinierung sichergestellt ist.

### 5.7.9 Zusammenfassung

Ich fasse die wichtigen Punkte der Reservierung noch einmal zusammen:

- In einem Ring mit mehrfach belegten Elementen werden sowohl die periodischen als auch die aperiodischen Aktivitäten gespeichert.

Neben der Datenhaltung in *einer* Struktur und den kurzen Zugriffszeiten ist eine einfache Darstellung der Reservierungen nach außen als lineare Liste möglich; Laufzeitsystem und Planungsalgorithmen werden sinnvoll unterstützt. Einfache, hinreichende Planungstests können konstruktiv durchgeführt werden und sind gerade bei dynamischen Planungen zur Laufzeit einsetzbar.

- Die Realisierung des Ringes, also die tatsächliche Zuweisung von Aktivitäten zu den Ringelementen (Ringoffsets), geschieht mit Hilfe der Slotmanager, aus denen der Ring aufgebaut wird.

Dadurch, daß man die Slotverwaltung in diesen Managern durchführt, bewahrt man sich die Möglichkeit, diese Einzelheiten unabhängig vom Rest der Reservierungsstruktur aufzubauen. Die interne Verwaltung kann, je nach Auslastung, in verschiedenen Weisen implementiert sein und sich sogar (zur Laufzeit) im Ring abschnittsweise verändern. Mit dem Konzept der Manager bleibt man sogar auf dieser Ebene offen für andere grundlegende Strukturen als Ringe.

- In den Ring werden Reservierungseinheiten RE eingetragen, aus denen jede planbare Aktivität (planbare Einheit) zusammengesetzt ist. Sie reservieren Zugriffszeiten die nicht interpretiert werden (s.u.). Sämtliche (Um-)Planungen in der Struktur nutzen ausschließlich die von den REen bereitgestellte Funktionalität zum Plazieren der Einheiten.

Konflikttests, Verschiebungen, Unterbrechungen (Aufteilung) als wesentliche Punkte der Planung auf der Ringebene sind über geeignete Schnittstellen verfügbar.

- Zusätzliche Planungsalgorithmen lassen sich durch neue Implementierungen dieser Schnittstellen in die Struktur integrieren, ohne daß die bestehenden Verfahren geändert werden müssen. Dadurch eröffnet sich ein weites Feld an Erweiterungsmöglichkeiten, innerhalb isolierter Zeitintervalle unterschiedliche Planungs- und damit BM-Zugriffsverfahren einzusetzen.

## 5.8 Echtzeitklassen

Die in der Einleitung unterschiedenen Klassen der Echtzeitverarbeitung müssen bei der Planung ebenfalls unterschieden werden. Es ergeben sich allerdings unterschiedliche Aspekte in der in 5.6.1 angegebenen Unterteilung (Planbarkeitstest,

Planerstellung und Plandurchsetzung). Die Plandurchsetzung hat letztlich dafür zu sorgen, daß harte Echtzeitbedingungen eingehalten werden und daß für weiche Echtzeitaufgaben eine geeignete Zeitfehlerbehandlung ausgeführt wird.

Auf der Ebene der Reservierung sind nur Zeiten für die jeweiligen Aktivitäten gespeichert; mehr nicht! Das heißt insbesondere, daß keine Interpretation in WCET, EET oder ähnliches erfolgt. Die reservierten Zeiten sind den Aktivitäten zugeordnet und, ob sie im ungünstigsten oder im durchschnittlichen Fall aufgebraucht werden, ist hier ohne Belang. Eine Reservierungseinheit wird nicht nach harten oder weichen Zeitbedingungen unterschieden. Der Reservierungsauftrag hat die Aufgabe, für eine planbare Einheit Reservierungseinheiten zu erzeugen, die eine etwaige Fehlerbehandlung einschließen.

Folgende Punkte muß das Laufzeitsystem zur Plandurchsetzung unterstützen<sup>18</sup>:

- Harte Echtzeitbedingungen werden durch die rigorose Einhaltung der zeitlichen Bedingungen der Betriebsmittelzugriffe, die in der Reservierung gespeichert sind, eingehalten.

Alle Startzeiten hält man ein, wenn Aktivitäten, die ihren zugesagten zeitlichen Rahmen zu überschreiten drohen (Zeitfehler), rechtzeitig abgebrochen werden. Da die Worst-Case-Spezifikationen der HRT-Aufgaben die Planungsgrundlage bilden, wird definitionsgemäß ein ausreichender Zugriff auf das verwaltete Betriebsmittel ermöglicht.

- Für weiche Echtzeitaufgaben stehen zunächst analog die parametrisierten Zugriffsmöglichkeiten bereit. Beim Einplanen der Reservierungseinheiten muß eine Zeitfehlerbehandlung vorgesehen werden, die eine Propagierung und Störung der anderen Aktivitäten verhindert (vgl. TaskPair-Konzept des Abbrechens *vor* einer Deadlineverletzung, s. 4.7.2.2).

Darüber hinaus ist bei weichen Zeitbedingungen auch ein vom System verursachter Fehler möglich. In gewissem Umfang kann man die zugestandene Ausführungszeit kürzen oder die Startzeit verschieben und trotzdem eine gemäß der Spezifikation korrekte Ausführung erreichen. Den Spielraum eines mittels statistischem Multiplexing geplanten SRT-Netzzugriffs kann man (teilweise) für eine andere Übertragung nutzen, ohne die  $p$ %ige Garantie des SRT-Zugriffs zu verletzen. Der maximale Wert der Aufgabe wird nicht mehr erreicht, weil nicht 100%ig alle Daten übertragen werden, aber der Wert fällt nicht wie bei den harten Zeitbedingungen auf 0 oder  $-\infty$ . Diese Möglichkeiten kann man nur in einer Überlagerung ausnutzen: Die Zugriffe einer isolierten RE sind garantiert, erst ein erweiterter Konflikttest kann die Zugriffsrechte innerhalb eines überlagerten Intervalls nach anderen Maßstäben (implizit) verteilen.

---

<sup>18</sup>vgl. Definitionen in 1.1.3.

Eine anschließende Zeitfehlerbehandlung kann zum Beispiel versuchen, nach einem temporären Abbruch einer SRT-Aufgabe, diese später in den ermittelten NRT-Slots fortzusetzen (s.u.), um noch einen möglichst großen Wert für ihre Berechnung zu erlangen. Hier liegt ein großes Anwendungsgebiet der (kurzfristigen) dynamischen Planung, die die Auslastung bzw. die erreichten Werte erhöhen kann.

Im Gegensatz zu den zwingenden Worst-Case-Angaben der harten Echtzeit sind ungenaue Spezifikationen für SRT-Aufgaben erlaubt. Die vorgesehene Fehlerbehandlung und evtl. sogar Fehlerbehebung durch eine fortgesetzte Ausführung ermöglicht eine vereinfachte Spezifikation der Anwendung.

- Für eine Integration von Nicht-Echtzeit Verarbeitung in das Echtzeitsystem kommen zwei Lösungen in Frage. Entweder sind diese Aufgaben jederzeit unterbrech- und verdrängbar und können somit 'ständig im Hintergrund laufen', oder für die NRT-Verarbeitung sind vom System extra Zeitbereiche zu reservieren. Diese Alternativen sind bereits in 4.4.3.2 für die Netzkommunikation diskutiert worden.

# Kapitel 6

## Implementierung

In diesem Kapitel soll die beispielhafte Implementierung der vorgestellten Konzepte beschrieben und anhand einer Applikation zur Steuerung eines innovativen Roboters demonstriert werden. Der umfangreiche Aufbau der Beispielanwendung wurde durch viele Vorarbeiten im Institut SET, Gruppe Responsive Systeme, der GMD, St. Augustin, in der diese Diplomarbeit entstand, ermöglicht. Der vorausgesetzte CPU-Planer, die Software der Zeitsynchronisation, das Visualisierungstool Jewel und die erste Version der Robotersteuerung sind dort entwickelt worden. Die Implementierung des Kommunikationssystems und der Steuerung erfolgte in der Sprache C++ und ist portabel aufgebaut. Der Einsatz in beliebigen zeitgesteuerten Systemen, die über einen Multiple-Access-Bus kommunizieren, ist ohne großen Portierungsaufwand möglich.

### 6.1 Aufbau des Zielsystems

#### 6.1.1 Hardware

Der Aufbau des verteilten Systems ist in Abbildung 6.1 skizziert. Es werden nur Standardkomponenten eingesetzt und keine Änderungen am Betriebssystem oder den Netzcontrollern (Treibern) durchgeführt. In den Knoten steht jeweils ein Motorola 68020 mit 4 MB Hauptspeicher zur Verfügung (MVME 68020), der an das Kommunikationsmedium angeschlossen ist. Als Multiple-Access-Netz kommt der CAN-Bus zum Einsatz; siehe [Bos91] und Abschnitt 2.1.1.1.

Der CAN-Bus bietet eine maximale Bandbreite von 1 MBit/s, die in der Beispielanwendung aber aufgrund limitierender Komponenten auf 100 KBit/s beschränkt wird. Einzelne Nachrichten können bis zu 8 Bytes Nutzdaten enthalten (plus 44–56 Bits Verwaltungsinformationen und Bitstuffing) und werden prioritätenbasiert versandt (absprechendes Zugriffsverfahren). 11 Bit beschreiben die Priorität, die in der Kollisionsauflösung verzögerungsfrei bestimmt, welche Nachricht übertragen wird. Es dürfen nie zwei gleiche Prioritäten kollidieren.

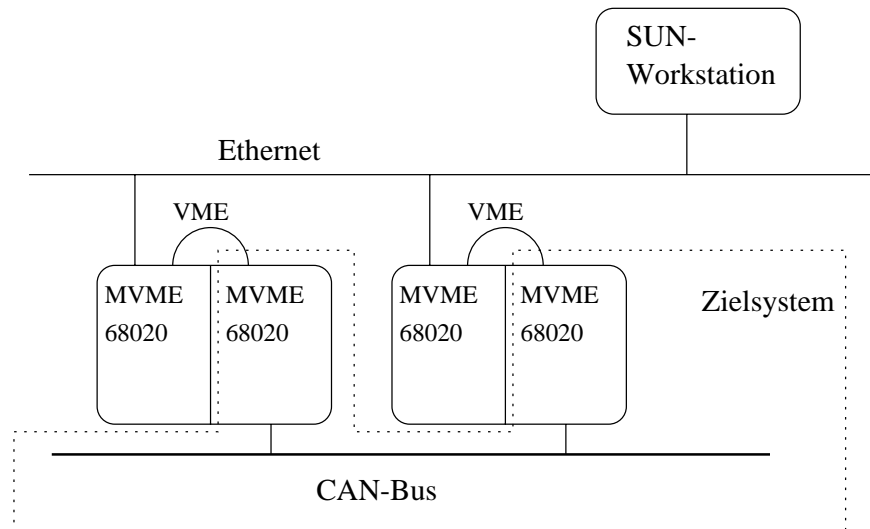


Abbildung 6.1: Aufbau der Testumgebung

Knoten wählen einzelne Prioritäten aus, deren Nachrichten sie empfangen wollen, und somit nutzt man diese zur Adressierung sowohl der Sender als auch der Empfänger (Kanal- oder CAN-ID). Der verwendete Netzcontroller (NIC) verfügt über ausreichenden eigenen Speicher, um zu jeder Priorität die letzte Nachricht aufzubewahren: Der Empfang ist nebenläufig. Der im NIC ausgelöste Interrupt beim Nachrichtenempfang wird in allen Empfängern und dem Sender gleichzeitig ausgelöst; die Laufzeitverzögerung  $\tau$  kann hier vernachlässigt werden.

In dieser Testumgebung arbeiten in den Knoten zwei Prozessoren, aber für das Kommunikationssystem und die Applikation wird nur einer genutzt. In einen über den VME-Bus erreichbaren gemeinsamen Speicher schreibt der Applikationsprozessor (SUT<sup>1</sup>) zur Laufzeit Ereignisdaten, die vom zweiten Prozessor ausgewertet und an die zentrale Visualisierung über das Ethernet geschickt werden [LKG92]. Man erreicht eine möglichst geringe Belastung des SUT (low-interference). Am Ethernet ist eine SUN-Workstation angeschlossen, die die Visualisierung und Applikationskontrolle übernimmt.

### 6.1.2 Software

In den Knoten kommt jeweils das Echtzeitbetriebssystem VxWorks mit prioritäten-gesteuerter Prozessorzuteilung zum Einsatz. Darauf baut eine zeitgesteuerte CPU-Nutzung auf, die die dynamisch veränderbaren Prioritäten zur Durchsetzung eines expliziten Planes nutzt (vgl. 5.6.4). Der explizite CPU-Plan wird durch einen bereits existierenden Planer aufgebaut [Mü97].

Die Zeitsteuerung der CPU erfolgt durch den Clock-Interrupt mit einer vor

---

<sup>1</sup>engl.: *system under test*

Systemstart wählbaren Frequenz von bis zu 1000 Hz; darüber ist der erzeugte Overhead für diese Hardware zu groß. Auf diesen Wert legt man die Granularität sowohl der CPU- als auch der Netzplanung fest.

Die Uhrensynchronisation erfolgt in Software und erreicht über ein CAN-Protokoll eine maximale Abweichung in der Größenordnung von  $\Delta GT = 10\mu s$  [GS94]. Der *Zeitmaster* schickt eine Nachricht, die in allen Knoten gleichzeitig einen Interrupt, in dem die lokale Zeit genommen wird, auslöst (auch im Master). In die nächste Nachricht schreibt der Master seinen letzten, im Interrupt genommenen Zeitstempel, der in den Empfängern zur Korrektur der lokalen Uhr genutzt wird. Der Kommunikationsbedarf des Protokolls wird nach der Initialisierung des Systems als „normaler“ Kanal eingeplant.

## 6.2 Kommunikationssystem

### 6.2.1 Datenübertragung

Das Kommunikationssystem greift zeitgesteuert auf den CAN-Bus zu: dynamisches TDMA. Die Prioritäten werden nicht genutzt, weil zwar eine Verzögerungsfunktion  $VF(t)$  mit bekanntem  $t_{max}$  angegeben werden kann [KT95], aber dieser ratenbasierte Ansatz nicht den aufgestellten Entwurfskriterien entspricht. Die Prioritäten können gut in einer Erweiterung in Kanalüberlagerungen und Konflikttests eingesetzt werden. Eine sinnvolle Kombination dieser Ansätze ist möglich.

Es geht keine effektive Bandbreite der Datenübertragung aufgrund der Unsicherheit  $\Delta GT$  der globalen Zeit verloren, weil zwischen TDMA-Intervallen keine „Leerzeit“ zur Wahrung des alleinigen Zugriffsrechts eingefügt werden muß. Die Länge  $\phi = 540\mu s$  der kürzesten Datenübertragung bei 100KBit/s und der CSMA-Zugriff verhindern mit  $\Delta GT < 50\mu s$  Kollisionen an diesen Grenzen.

Ich gehe von einer fehlerfreien Übertragung aus (vgl. 4.7), weil die Signalübertragung des CAN-Busses durch eine Prüfsumme (CRC) sehr gut gegen temporäre Fehler geschützt ist; Hamming-Distanz 5. Im Fehlerfall versucht der NIC *automatisch* eine erneute Übertragung und gefährdet durch die Verletzung der lokalen Garantie die globale Garantie anderer Kanäle. Daher müßte eine als fehlerhaft markierte Nachricht verworfen werden<sup>2</sup> und es liegt dann an der Applikation, mit diesen Fehlern umzugehen.

Idealerweise müßte die zeitgesteuerte Datenübertragung als Leistung der zweiten OSI-Schicht in das Betriebssystem bzw. die entsprechenden Treiberprogramme des NIC integriert werden. In der Implementierung besteht kein Zugriff auf diese Systemteile, aber die Schnittstelle zum NIC bietet die Möglichkeit direkt

---

<sup>2</sup>Zu einzelnen Nachrichten kann ein Error-Interrupt ausgelöst werden, in dem die Übertragung abubrechen ist. Dies ist nur über eine direkte Programmierung des CAN-Controllers möglich, der hier aber nicht adressierbar ist.

auf den Sende-, Empfangspuffer und die dazugehörigen Verwaltungsinformationen zuzugreifen und die Interruptsteuerung läßt sich weitgehend anpassen.

### 6.2.2 Kanal

Der Kommunikationskanal wird in einer Klasse `ChannelBase` implementiert, von der bei jedem Kanalaufbau ein Objekt instanziiert und der Applikation als Referenz übergeben wird – das Kanalobjekt. Die Klasse besitzt keinen eigenen Kommunikationsthread und greift direkt auf den Bus zu. Als Basisklasse ist sie für die CT-Anpassung vorgesehen. Die virtuelle Memberfunktion `TDMA suggestion` übernimmt nach der Verbindungsplanung die Koordinierung der lokalen BM-Planung und gibt eine boolesche Antwort an den aufrufenden Daemon zurück; in der Basisklasse immer `true`.

Die virtuellen Memberfunktionen `send(dataPtr, size)` und `recv(dataPtr, size)` sind definiert, die die Schnittstelle zur Datenübertragung darstellen. In der Basisklasse wird nur die unidirektionale Kommunikation unterstützt, aber Erweiterungen lassen sich in abgeleiteten Klassen implementieren.

Die Klasse `Channel` ist von der Basisklasse `ChannelBase` abgeleitet und implementiert ein Kanalobjekt, wie es im Konzept vorgestellt wurde (s. 4.4.6). Der Konstruktor der Klasse initialisiert die Puffer der applikationsseitigen Schnittstelle. Die überladene virtuelle Funktion `TDMA suggestion` läßt den CT von der CPU-Planung reservieren. Der bereitgestellte Pufferspeicher wird von der Applikation über `send` und `recv` erreicht und intern vom CT mit den entsprechenden Funktionen der Basisklasse geleert bzw. gefüllt.

Ein unidirektionaler IPC-Kanal für die lokale Kommunikation ist ebenfalls von `ChannelBase` abgeleitet worden. Lokale Kanäle und Kanäle zu entfernten Kommunikationsteilnehmern müssen in der Applikation nicht unterschieden werden, weil bei beiden der Zugriff über die gleiche Basisklasse erfolgt. Das Puffermanagement, das auch in `Channel` eingesetzt wird, erlaubt hier die Kommunikation über einen gemeinsamen Speicher.

Erweiterte Dienste des Kommunikationssystems sind konzeptionell in einer höheren Schicht auf die vorhandenen Dienste aufzubauen. In der Implementierung kann eine von `ChannelBase` oder `Channel` abgeleitete Klasse weitere Funktionalität in das bestehende Konzept integrieren. Die CT-Anpassung ist sehr einfach möglich, weil `TDMA suggestion()` nach der Verbindungsplanung die lokale Planung übernimmt.

### 6.2.3 Lokales Management

Die Klasse `CDaemon` implementiert den Kommunikationsdaemon. Sie wird genau einmal in jedem Knoten instanziiert und arbeitet zunächst in einem NRT-Modus, um die Initialisierung durchzuführen. Das Uhrenprotokoll muß eine einheitliche Zeitbasis herstellen und gleichzeitig werden die Verwaltungskanäle der statischen

Knotenmenge aufgebaut. Die Daemonen können bereits die Kanäle und das Reservierungsprotokoll benutzen, um die TDMA-Zeiten der Verwaltungskanäle auszuhandeln. Der Zeitmaster reserviert einen Broadcastkanal für das Uhrenprotokoll, so daß die Umstellung in den Echtzeitmodus fließend erfolgen kann. Der Server startet den Echtzeitmodus explizit mit einem Broadcast auf einem bekannten Kanal, wenn alle Daemonen ihre erfolgreiche Uhrensynchronisation gemeldet haben.

Der Daemon erzeugt einen eigenen NRT-Thread für die Kanalverwaltung. Dieser kommuniziert über den Verwaltungskanal mit dem Server und über einen IPC-Kanal mit den lokalen Aufgaben. Applikationen können einen Kanal mit einer TDMA-Spezifikation `ActiveSlotInfo` über die Memberfunktionen `inputChannel()` oder `outputChannel()` anfordern. Die Funktion blockiert und wartet auf ein Ergebnis des Daemon-Thread. Wahlweise erhält man eine Referenz auf den Kanal über `ChannelBase` oder `Channel`, sofern die Planung erfolgreich war.

```
ActiveSlotInfoPreempt asi (0, 1.5, 80, 30, 2);
// est, et, period, delay, dl => 75% usage of active interval

CDaemon* d = CDaemon::getDaemon();
ChannelBase* ch = new ChannelBase (50, 1);

ch = d->outputChannel (asi, 50, 1, ch);
// call with ChannelBase* to prevent automatic CPU-scheduling
if (ch) {
    // daemon has scheduled the bus access
    // now schedule CPU manually
    const ActiveSlotInfoPreempt* asi = ch->TDMAspec();
    // get the bus TDMA slots

    thr = new Thread ("ThreadName");
    t = new Task(taskBodyProc, thr,
                asi->startTime, ...);

    cpuScheduler->insert(t);
    ...
}
```

### 6.2.4 Globales Management

Die Klasse `CServer` implementiert den Kommunikationsserver. Sie ist von der Klasse `CDaemon` abgeleitet, um einen kombinierten Einsatz im gleichen Rechner zu erleichtern – der Server ist sein eigener Daemon. Es ist nur eine Reservierungsstruktur notwendig und der Zugriff der lokalen Aufgaben erfolgt wie bisher über die `CDaemon`-Superklasse. Die Klasse startet nach wie vor nur einen Thread,

der hier die Aufgaben des globalen *und* des lokalen Managements übernimmt (s. Abb. 6.2).

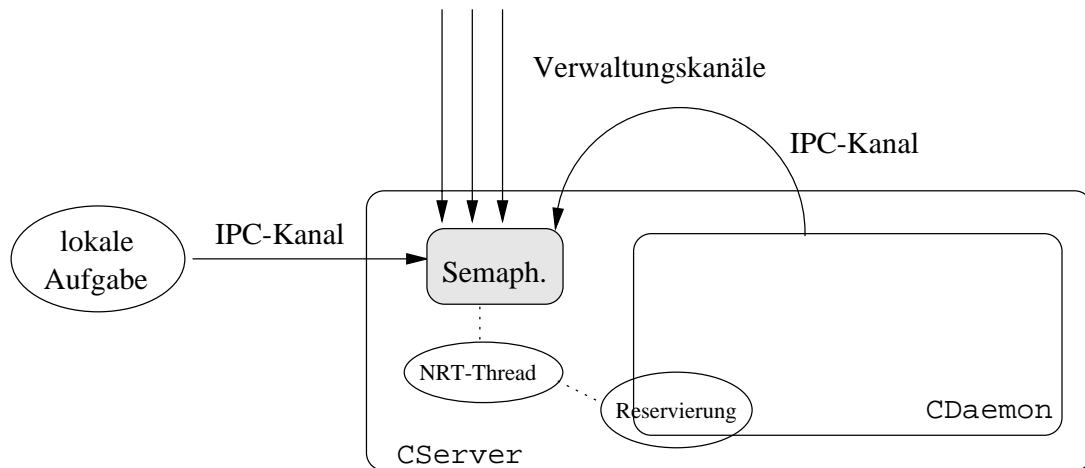


Abbildung 6.2: Aufbau von CServer

Die eingehenden Verwaltungskanäle setzen eine (zählende) Semaphore, wenn sie Daten empfangen haben und starten damit den NRT-Thread des Servers. Applikationsanfragen über den IPC-Kanal werden direkt an CDaemon weitergegeben. Diese Klasse wird unverändert weiter genutzt, weil die Funktionalität als Teilnehmer des Reservierungsprotokolls unverändert ist und von der lokalen „Anwesenheit“ des Servers nicht abhängt. Die Kommunikation zum Server läuft über die gleichen Schnittstellen, weil bei der Initialisierung der Verwaltungskanal als IPC-Kanal erzeugt wird. Dadurch wird die lokale Kommunikation in die Liste der anderen Verwaltungskanäle eingefügt und eine Bevorzugung vermieden.<sup>3</sup>

Das Reservierungsprotokoll übermittelt beim Kanalaufbau und der Leistungsverhandlung reine TDMA-Parameter, die direkt in die zentrale Planung übernommen werden. Der Zugriff auf die Reservierungsstruktur ResSched der Klasse CDaemon erfolgt nur durch den einen Thread, der sequentiell die Planungsaufträge bearbeitet. Eine weitere Synchronisation der Zugriffe ist daher nicht notwendig.

## 6.3 Planung

Die Applikation spezifiziert einen Reservierungsauftrag über eine Instanz der Klasse RSchedCommission. Ein Teil des Planungsalgorithmus' ist in der Memberfunktion schedule() enthalten, die zu Beginn der Planung von der Reservierungsklasse ResSched aufgerufen wird. Darin werden die zur Beschreibung der

<sup>3</sup>Eine anfängliche Implementierung rief direkt die entsprechenden Funktionen des Daemons bzw. Servers auf, so daß der Kontrollfluß des Threads (Server→Daemon→Server) ununterbrochen lokale Anfragen bearbeitete und das System externe Anfragen lange unbeantwortet ließ.

Planungseinheit notwendigen Reservierungseinheiten instanziiert. Die Positionierung der REen innerhalb der Spezifikation übernimmt aus Kenntnis der internen Verwaltung bisher ein einfacher Algorithmus in `ResSched` (s.u.). Der letzte Teil des Planungsalgorithmus' ist in den REen codiert.

```
// call reservation from application
rsc = new t_RSchedCommissionPreempt(TDMAasi);
ResSched::iterator it = reservation.insert(rsc);
// start reservation

if (it.defined()) {
    // successful insertion, get scheduled starttime
    startTime = rsc->asi()->startTime; // == it.start()

    // example navigation
    ++it; // next activity
    nextStart = it.start(); // start time of next activity
}
```

Die Klasse `ResSched` implementiert die explizite Slotverwaltung, die genau einmal pro verwaltetem Betriebsmittel instanziiert wird. Die Klasse definiert Memberfunktionen und Iteratoren, die die Slots als lineare Liste darstellen. Die Funktion `search_free(...)` wird von `insert()` aufgerufen und implementiert einen sehr einfachen Algorithmus zum Positionieren von REen auf der Zeitachse, dessen Funktion in Abbildung 6.3 skizziert ist. Es wird versucht, einen (konflikt-)

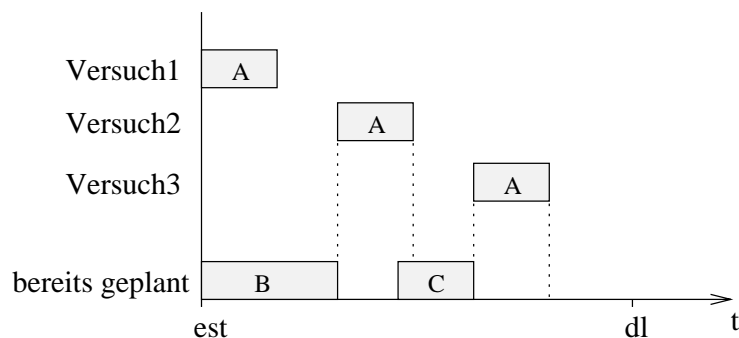


Abbildung 6.3: Positionierung auf der Zeitachse

freien Platz gemäß der Spezifikation auf der Zeitachse zu finden, indem, beginnend bei  $st(A) = est(A)$ , Konflikttests mit evtl. bereits verplanten REen durchgeführt werden. Im positiven Fall startet man die Konfliktauflösung der REen. In der Skizze schlägt der erste Planungsversuch, die erste Positionierung, fehl; es sind hier nur rein zeitliche Konflikttests dargestellt. Die Konfliktauflösung von  $A$  setzt

die neue Startzeit hinter das belegte Intervall. Dieser Vorgang iteriert bis zum Erfolg oder dem Erreichen der Deadline  $dl(A)$ . Eine Verschiebung der bereits geplanten Aktivitäten  $B$  und  $C$  ist prinzipiell durch deren Konflikttest möglich, wird aber vom Kommunikationssystem explizit verboten.

Zur Verwaltung der Slots sind ringförmig angeordnete Slotmanager, **ResSchedEntryManager**, vorhanden, die ihrerseits den Zugriff auf die lineare Liste belegter Ringoffsets über Iteratoren ermöglicht. Die Anzahl  $m$  der vom Manager verwalteten Ringoffsets ist konstant und ergibt sich aus der vorgegebenen Anzahl der Manager  $M$  und der Ringgröße  $n$ :  $m = \frac{n}{M}$ . Die Manager haben nur eine sehr eingeschränkte Information über die Positionierung einer RE auf der Zeitachse, denn die Einträge sind nach dem Ringoffset  $i$  der Startzeiten sortiert:  $s = st(RE) \bmod n$ .

Die Implementierung dieser Liste ist außerhalb der Manager nicht sichtbar und baut auf einer Listenverwaltung auf. Die Liste enthält pro Reservierungseinheit, die im verwalteten Slotintervall liegt, einen Eintrag und diese sind nach aufsteigendem Ringoffset des Starts sortiert. Vorgänger und Nachfolger sind somit im Ring  $\mathcal{S} \bmod n$  leicht zu finden. Vorgänger und Nachfolger auf der Zeitachse  $\mathcal{S}$  müssen von **ResSched** u.U. in mehreren Slotmanagern gesucht werden, wenn die Nachfolger im Ring zu anderen Ringumläufen  $k$  aktiv sind ( $st(RE) = i + kn$ ).

Die Liste muß linear abgearbeitet werden, um zu bestimmen, ob ein Slot  $t$  leer ist. Der im Vergleich zu einer Arrayimplementierung erhöhte Suchaufwand hält den Speicherbedarf gering und dessen Verwaltung einfach. Dieser Ansatz bietet sich bei einer dünnen Besetzung der Einträge an. In einer alternativen *Arrayverwaltung* ist jedem der  $m$  verwalteten Ringoffsets ein Arrayeintrag zugeordnet. Folglich muß eine RE, die sich über mehrere Slots (Ringoffsets) erstreckt, in mehreren Arrayelementen eingetragen werden. In jedem Arrayelement können wiederum mehrere REen eingetragen sein: Teilbare Betriebsmittel können mehrere Aktivitäten im gleichen Slot erlauben, zu einem Ringoffset  $i$  können mit verschiedenen Ringumläufen  $k, k'$  verschiedene Slots  $i + kn, i + k'n$  gehören.<sup>4</sup> Der Zugriff auf einen Ringoffset geschieht über das Array in konstanter Zeit. Um auf einen bestimmten Slot zuzugreifen, müssen darüber hinaus die Einträge in dem entsprechenden Arrayelement auf eine Belegung durchsucht werden.

Eine eigene Klasse **ActiveSlotInfo** existiert für die Aktivitätsbeschreibungen, weil sie auch unabhängig von den Reservierungseinheiten (s.u.) benutzt werden kann. Diese Basisklasse definiert rein virtuelle Memberfunktionen, die erst in den abgeleiteten Subklassen implementiert werden. Auf folgende Funktionalität kann zugegriffen werden:

- Start und Ende einer (periodischen) Aktivität sowie deren Ausführungszeit und Periode kann abgefragt werden.

---

<sup>4</sup>Dies ist der Mechanismus, die unendlich lange Zeitskala im Rechner verwalten zu können.

- Die nächste aktive Phase nach einem bzw. die letzte vor einem Zeitpunkt (Slot) kann erfragt werden.
- Ein Konflikttest gliedert sich in zwei Teile. Der erste wird von `intersected()` ausgeführt und prüft auf rein zeitliche Überschneidungen der Slots zweier `ActiveSlotInfo`-Spezifikationen. Der zweite Teil des Tests findet in der Reservierungseinheit `ResSchedEntry` statt (s.u.).

Implementiert sind Beschreibungen und REen aperiodischer, periodischer und unterbrechbarer Aktivitäten. Die unterbrechbaren Aktivitäten demonstrieren die Erweiterbarkeit der Planung über die allgemeinen Konflikttests und die Kanalüberlagerung. Sie nutzen das Betriebsmittel innerhalb ihrer aktiven Phasen anteilig. Sie sind zwischen  $est(A)$  und  $dl(A)$  aktiv, nutzen aber anteilig maximal  $\frac{rt(A)}{\Delta dl(A)}$ ; periodische Aktivitäten werden analog behandelt. Der zeitliche Konflikttest ist in den aktiven Phasen positiv, aber der zweite Teil des Tests kann in `ResSchedEntry` den Spielraum  $dl(A) - est(A) - rt(A)$  für eine **Überlagerung** nutzen (s.u.). In dieser Überlagerung entscheiden die CAN-IDs über die Vergabe der Zugriffsrechte.

Die Implementierungen der Reservierungseinheiten leiten sich aus der Basisklasse `ResSchedEntry` ab: aperiodische, periodische und unterbrechbare REen. Die Basisklasse besitzt einen Zeiger auf den Reservierungsauftrag und eine Schnittstelle zur Aktivitätsbeschreibung. Die virtuelle Memberfunktion `intersect(...)` bildet den zweiten Teil des Konflikttests. In ihm kann nach der festgestellten zeitlichen Überlagerung eine genauere Untersuchung des BM-Konflikts erfolgen. In der Implementierung unterbrechbarer Aktivitäten wird die Leistung kurz demonstriert: Die Überlagerung der aktiven Phasen schließt die konfliktfreie Planung nicht aus, wenn durch eine Reduzierung des Spielraums ausreichend zusätzlicher Platz geschaffen werden kann. Die RE speichert in einer Membervariable ihren Spielraum der im Test bis auf 0 reduziert werden kann.

Dieser gegenüber der reinen TDMA-Planung erweiterte *Konflikttest* erlaubt eine Überlagerung von Aktivitäten mit der einfachsten Verträglichkeitsprüfung: Bleibt die Summe der anteiligen Nutzungen im überlagerten Intervall kleiner als 1? Die kritische Aufgabe, die Verträglichkeit der BM-Zugriffsmuster zu garantieren, muß der Applikationsentwurf erfüllen. Diese Realisierung reicht aber bereits aus, um anhand der Beispielapplikation die grundlegenden Ideen und die Leistungsfähigkeit des Konzepts zu demonstrieren:

Der kombinierte zeit- und eventgesteuerte CAN-Zugriff bildet eine flexible Basis, die unterschiedlichste Leistungsanforderungen der Applikationen erfüllen kann!

Eine BM-Teilbarkeit kann man auch simulieren, indem man die RE mittels `split()` aufteilt und ein zweites Objekt, eine zweite RE, instanziiert. Die Funktion `resolve()` wird zur Lösung von bestehenden Konflikten verwendet (Verschieben, automatisches Splitting, Laufzeitverkürzung (vgl. graceful degradation, z.B.

[Mul93])). Einzuplanende REen verändern darüber ihre Parameter und starten die Planung über einen rekursiven Aufruf von `ResSched::search_free()` erneut. Implementiert ist für bisher eine einfache Verschiebung der Startzeit, die von dem Planungsalgorithmus bei neu einzuplanenden REen verwendet wird.

Die Funktionalität des Planungsalgorithmus' ist über die virtuellen Funktionen auch von anderen REen zur Konfliktlösung nutzbar. Weitere Algorithmen, die eine zusätzliche Funktionalität bieten, können in Subklassen implementiert werden und auch die Basisklasse der bestehenden Implementierung erweitern, wenn entsprechende Funktionen mit sinnvollen Defaultwerten den Einsatz der alten RE-Implementierungen weiterhin erlaubt. `movable()` und `splittable()` geben zum Beispiel über die Möglichkeiten zur Konfliktlösung Auskunft bzw. `move()` und `split()` werden ggf. mit der Rückgabe einer Fehlermeldung abgelehnt.<sup>5</sup>

Stilisiert folgt die Suche nach freien Slots dem folgendem Schema:

```
ResSchedEntryManager::search_free(RSchedEntry& re, int st)
{
    for (re2=firstManagedRE(); re!=0; re=nextManagedRE(re)) {
        if (re->intersected(re2, st))
            return re2;
    }
    return 0;
}
```

```
ResSched::search_free(RSchedEntry& re, ....) // stilisiert
{
    int st;

    if (re.startTime<0) // initialze
        st = re.startTime = re.est;

    ResSchedEntryManager* man = get_manager(st);
    int intersectedAt;

    do {
        collidedRE = man->search_free(re);
        if (collidedRE != 0) {
            intersectedAt = re->intersect(collidedRE);

            if (intersectedAt >= 0) {
                if (re->resolve(collidedRE, intersectedAt)) {
```

---

<sup>5</sup>Mit dem ANSI-C++-Standard und den darin verfügbaren RTTI (run time type information) ist eine Abfrage der gebotenen Leistungen auch über Informationen der Klassenhierarchie möglich.

```

        // try to modify specification to avoid the conflict

        return search_free(re, ...);
        // call recursively with modified parameters
    }
}

if (man->is_managed(st + re.active_length(st)))
    // we have reached the managed end of the re
    man=0;
else
    // the end of re lies in the next manager
    man = get_next_manager(man);
} while (man);

return iterator(re)
}

```

## 6.4 Beispielapplikation – Robotersteuerung

In diesem Abschnitt wird die Beispielapplikation vorgestellt. Sie basiert auf dem implementierten Kommunikationssystem und steuert einen innovativen Roboter, an dem neue Methoden und Werkzeuge für verteilte Echtzeitsysteme getestet werden. Mit Hilfe des Kommunikationssystems kann die früher zentral erfolgte Steuerung verteilt werden, ohne die Bewegungsmuster wesentlich zu beeinflussen. Bei niedrigerer CPU-Auslastung kann eine höhere „Steuerungsleistung“ erbracht werden: Sensoren häufiger abfragen; Skalierbarkeit durch freie CPU-Zeit und zusätzliche Rechner; hohe Planungsauslastung mit überlagerten Buszugriffen.

### 6.4.1 Schlangenroboter

Der schlangenartige Roboter besteht aus einem *Kopf* und mehreren *Sektionen*, die jeweils gleich aufgebaut und hintereinandergeschaltet sind [PDK96]. Alle Teile sind an einen CAN-Bus angeschlossen und bilden zusammen mit der Konfiguration aus Abbildung 6.1 das Gesamtsystem. Der Kopf enthält vier Lichtsensoren und vier Leuchtdioden (LED), die ausgelesen bzw. geschaltet werden können. Eine Sektion enthält einen Mikroprozessor mit CAN-Schnittstelle, SLIO, und besteht aus zwei beweglichen *Joins*. Ein Join kann mit zwei Motoren in zwei orthogonalen Richtungen gebogen werden, um die Beweglichkeit einer Schlange nachzuahmen. Ein aktiver Motor verändert zweimal pro Umdrehung den Status

eines Wellensensors.

Die Mikroprozessoren nehmen Steuerkommandos auf einer festen CAN-ID entgegen, führen sie aus und senden ggf. auf einer anderen festen ID ein Ergebnis zurück. In der jetzigen Ausbaustufe nehmen sie nur sehr einfache Kommandos von den externen Steuerungsrechnern entgegen. Der Kopf verarbeitet:

- Lese Lichtsensor {1|2|3|4}  
Auf dieses Kommando sendet der SLIO den entsprechenden Wert zurück.
- Setze LED {1|2|3|4}, {An|Aus}

Die Steuerkommandos für die Sektionen sind:

- Starte Motor {1|2|3|4}, {links|rechts}
- Stoppe Motor {1|2|3|4}  
Zwei Nachrichten müssen dafür an den SLIO über den CAN-Bus übertragen werden, auf die er jeweils eine Antwort schickt.
- Lese Wellensensor  
Auf dieses Kommando sendet der SLIO den entsprechenden Wert der vier Motoren einer Sektion in einer Nachricht zurück. Die Software muß über den kumulierten Wert die Position bestimmen.

An den SLIO muß regelmäßig eine Kalibrierungsnachricht fester ID gesandt werden, um ein automatisches Abschalten zu verhindern.

### 6.4.2 Steuerung

Die Steuerung der Schlange erfolgte bisher durch einem einzigen Rechner [NS97]. Eine erweiterte Version des Roboters wird in Zukunft in jeder Sektion einen programmierbaren Prozessor enthalten, der eine wesentlich größere Zahl von Sensoren steuert und die lokalen Aufgaben der Sektion vor dem restlichen System verbirgt (vgl. Darstellung in der Einleitung). Das Kommunikationssystem soll in dieser Umgebung eingesetzt werden. Weil aber die Entwicklung noch nicht abgeschlossen ist, erfolgt die Demonstration mit der oben beschriebenen Schlange und der folgenden Steuerung.

Komplexe Bewegungen werden aus Sequenzen von Steuerkommandos zusammengesetzt. Ich gehe hier nur auf die einzelnen Kommandos ein, deren Übertragung zeitlichen Bedingungen unterworfen ist. Die Kalibrierungsnachricht muß alle 40ms versandt werden. Für den Start eines Motors ist in den bisherigen Bewegungsmustern keine Zeitbeschränkung notwendig, aber sobald er gestartet wurde, muß der Wellensensor mit einer durch die maximale Motorgeschwindigkeit vorgegebenen Frequenz (12,5Hz) abgefragt werden und die Übertragung der

Stoppnachricht muß nach Erreichen der Sollposition innerhalb einer festen Frist erfolgen.

Die Steuerungssoftware ist ursprünglich für einen einzelnen Rechner entworfen worden (siehe [PDK96, NS97, Mü97]) und ich habe sie zur Demonstration für die Mehrrechnernutzung umgeschrieben. Zwei VME-Rechner *A* und *B* der Testumgebung werden zur Steuerung eingesetzt. Auf beiden ist ein Kommunikationsdaemon und auf *A* auch der Server aktiv. Die Aufgabenverteilung zwischen den Rechnern ist in Abbildung 6.4 dargestellt. Jeder Rechner übernimmt die Steuerung einer Sektion der Schlange.

Rechner A	Kommunikation <i>et, Δdl, p</i> in ms	Rechner B	Kommunikation <i>et, Δdl, p</i> in ms
Server	8, 8, 80	Daemon	8, 8, 80
Daemon		Calibrator	2, 4, 40
Globale Zeit	6, 8, 320		
Kopf	12, 12, 80		
Kommandobroadc.	4, 4, 80		
Sektion 1	18, 36, 80	Sektion 0	18, 36, 80

dynamisch  
geplant

Abbildung 6.4: Aufgabenverteilung zwischen den Steuerrechnern

*A* erhält über das Ethernet Steueranweisungen von der Workstation und sendet die nicht nur für die Sektion 1 bestimmten Befehle über einen Broadcastkanal an alle weiteren Steuerrechner. Diese Anweisungen bestehen aus Sollpositionen der Motoren und einem *CHECK*-Kommando, das den Start der Bewegung initiiert. Nach dem Erhalt eines *CHECKs* plant die jeweilige Sektionssteuerung dynamisch einen bidirektionalen Kanal zum Sektions-SLIO, über den die Steuerkommandos (StartMotor, StoppMotor, LeseWellensensor) und die automatisch generierten Antworten des SLIOs übertragen werden.

Neben den einmaligen StartMotor-Befehlen an die maximal vier Motoren einer Sektion, muß die periodische Steuerung regelmäßig die Wellensensoren der Motoren abfragen und ggf. die Motoren stoppen. Für die Kommandos und Antworten wird ein ausreichend großes, zusammenhängendes TDMA-Intervall reserviert. Der Sliokanal wird zunächst ohne CPU-Reservierung geplant. Der CPU-Bedarf der Kommunikation wird zusammen mit der Steuerungsapplikation in einer Task reserviert, weil diese Task hauptsächlich kommuniziert und eine eigene Kommunikationstask (CT) einen unnötigen Overhead erzeugen würde. Die TDMA-Zeiten der Verbindung werden, wie bei der CT-Planung, auch für diese CPU-Planung benutzt, so daß die Einhaltung der lokalen Garantie gesichert ist. Die Busauslastung dieser Task ist kleiner als 60%, weil zwischen den Nachrichten kurze, lokale Verarbeitungszeiten und kurze Wartezeiten auf die SLIO-Antwort liegen. Eine Spezifikation und Planung der einzelnen Befehle ist aufgrund der Granularität der CPU- und Bus-Planung (4ms, 250Hz) nicht möglich.

Das Konzept der Kanalüberlagerung kann man in dieser Situation anwenden, um die Planungsauslastung über 60% zu heben. Beide Rechner können die Sektionssteuerung gleichzeitig ausführen, wenn man die zu planende Ausführungszeit geringfügig erhöht und damit die Verschränkung der Buszugriffe der einzelnen Tasks ermöglicht (siehe Abb. 6.5).<sup>6</sup> Die Verträglichkeit der Überlagerung ist hierbei durch den Applikationsentwurf sichergestellt: Die Zugriffsmuster enthalten konstruktionsbedingt Wartezeiten, die von einer zweiten Steuerung genutzt werden können. Andere Aktivitäten können, selbst wenn sie eine geringere anteilige Nutzung des BMs haben, nicht ohne weiteres mit einer Sektionssteuerung überlagert werden; im überlagerten Intervall findet keine explizite Kontrolle der Buszugriffe statt – der CAN-Bus ist kollisionsfrei – und eine Datenübertragung von mehr als ca. 1ms würde die ET der Steuerung verändern. Eine Kanalüberlagerung ist in der Beispielanwendung nur zwischen verschiedenen Rechnern möglich. Der benutzte CPU-Scheduler kann einen Zeitslot nur für eine einzige Task reservieren und deswegen ist die Überlagerung des CPU-Bedarfs zweier Tasks nicht möglich.

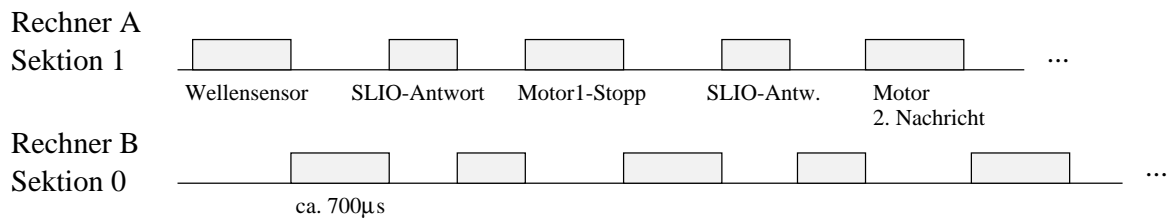


Abbildung 6.5: Zugriffsmuster der Sektionssteuerung

Die Kommunikationskanäle des Uhrenprotokolls und des Calibrators können auf ähnliche Weise überlagert werden. An ihnen läßt sich der Einsatz eines weiteren Zugriffsprotokolls demonstrieren, obwohl es für diese Applikation nicht existenziell ist. Das Uhrenprotokoll sendet immer zwei Nachrichten kurz nacheinander und der Calibrator immer nur eine einzelne, die eine höhere CAN-ID und damit eine niedrigere Priorität besitzt. Abbildung 6.6 zeigt die Ausführungszeiten der entsprechenden Bus-Zugriffe, die explizit geplant zur gleichen Zeit auf Rechner A und B beginnen. Das Zugriffsprotokoll des CAN-Busses steuert implizit die Zugriffe innerhalb des reservierten Intervalls. Der Calibrator wird zunächst verzögert, weil die Uhren-Nachricht eine höhere Priorität besitzt. Man erhält eine kombinierte Ausführungszeit, die nur unwesentlich größer als die des nicht überlagerten Uhrenprotokolls ist.

<sup>6</sup>Die vergrößerte WCET senkt die Planungsauslastung der lokalen CPU-Planung. Aber Überlagerungsmöglichkeiten würden die ohnehin wegen der groben Planungsgranularität schlechte CPU-Auslastung wesentlich verbessern.

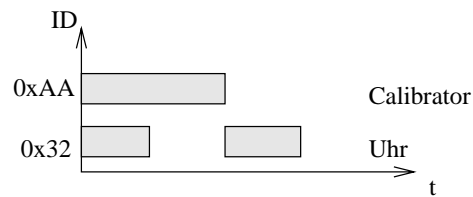


Abbildung 6.6: Überlagerte Zugriffszeiten von Uhrenprotokoll/Calibrator

Die Steuerung des Kopfes ist immer aktiv und schaltet die LEDs immer so, daß sie die Richtung der hellsten Lichtquelle anzeigen. Die Verwaltungskanäle, der Broadcastkanal der Schlangensteuerung und die Kommunikation für das Uhrenprotokoll, den Calibrator und die Kopfsteuerung werden beim Systemstart einmal geplant und bleiben unverändert bestehen.

Abbildung 6.7 zeigt beispielsweise die verplanten Buszugriffe, wenn beide Sektionssteuerungen aktiv sind.

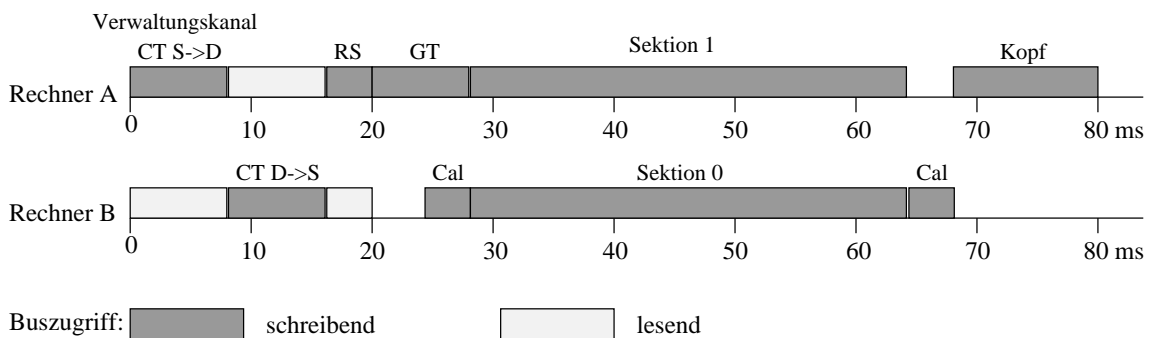


Abbildung 6.7: Geplante Auslastung des Busses

Die portable, objekt-orientierte Implementierung führt zu einem umfangreichen Quelltext, vor allem auch, weil die Erweiterbarkeit des Entwurfs berücksichtigt wurde. Der Implementierungsumfang ist in folgender Tabelle dargestellt:

Komponente	Zeilen
Planung	4000
CAN	1500
Server	1100
Daemon	1300
Kanal	2200
Utilities	~2500
Summe	ca. 12500

Abbildung 6.8: Umfang der Implementierung

# Kapitel 7

## Zusammenfassung und Ausblick

In diesem Kapitel möchte ich die wesentlichen Punkte der Arbeit zusammenfassen.

### 7.1 Was leistet das Konzept?

In der Diplomarbeit ist ein Konzept für die zeitbeschränkte Kommunikation in verteilten Echtzeitsystemen entworfen worden. Die garantierte Vorhersagbarkeit der Betriebsmittelzugriffe auf Netzwerk *und* CPU und die Fähigkeit, mehrere Zugriffsprotokolle dynamisch nebeneinander zu planen und einzusetzen, sind die wichtigsten erreichten Ziele. Die Einsatzfähigkeit in der Robotik und Automatisierungstechnik ist ein weiterer Aspekt des Entwurfs, welcher aber nicht auf diese Problemstellung beschränkt ist. Ebenso wichtig ist die durchgängige Erweiterbarkeit des Kommunikationssystems: An vielen Stellen ist der speziellen eine allgemeine Lösung vorgezogen worden, die den Einsatz in vielen unterschiedlichen Situationen und Umgebungen ermöglicht und eine flexible Ausgangsbasis für weitere Untersuchungen bildet.

Das entworfene Kommunikationssystem bearbeitet die systemnahen Probleme der Netzwerkkommunikation auf einem Multiple-Access-Bus, unter Ausschluß von Fehlern. Auf eine Fehlerbehandlung, die nicht Teil des Entwurfs ist, wird an kritischen Stellen hingewiesen. Die Schnittstelle zur Applikation ist sehr einfach und besteht aus Sende- und Empfangspuffern. Die betrachtete Problematik liegt in den OSI-Schichten zwei und vier und umfaßt die Planung der Netzzugriffe und der CPU. Die minimalen Voraussetzungen für einen praxisnahen Einsatz sind damit erfüllt.

#### 7.1.1 Kanäle

Die Kommunikation erfolgt konzeptionell zunächst über verbindungsorientierte, unidirektionale *Kanäle*, die zur Laufzeit auf- und abgebaut werden können. Die

dynamische Planung der Kanäle besteht aus zwei Teilen: Das globale Netzmanagement reserviert die Netzzugriffe der Datenübertragung und das lokale Management reserviert lokal den für die Datenübertragung und Flußkontrolle erforderlichen CPU-Aufwand. Ein Reservierungsprotokoll ist für die Absprache zwischen diesen Teilen des Systems entworfen worden. In der Applikation repräsentiert ein instanziiertes Objekt einer 'Kanalklasse' einen Kanal und erlaubt über einen Puffer den Zugriff auf dessen Leistung. Jedem Objekt ist ein Thread zugeordnet, der die Daten zeitgerecht überträgt bzw. empfängt.

### 7.1.2 Globales Management

Das *globale Netzmanagement* erfolgt zentral in einem Server und entlastet somit die Knoten des verteilten Systems. Im wesentlichen führt der Server eine Planung der Netzzugriffe durch und reserviert explizit TDMA-Intervalle für einzelne Rechner (nicht unbedingt exklusiv, s.u.). Das Reservierungsprotokoll beginnt den Kanalaufbau mit einer TDMA-Planung im Server und übermittelt die vorläufigen Ergebnisse als Vorschläge an die Kommunikationsteilnehmer. Der Kanalaufbau ist abgeschlossen, wenn der Vorschlag von allen akzeptiert ist; andernfalls geht das Protokoll mit einem neuen Vorschlag in eine weitere Planungsrunde. Es steuert die Leistungsverhandlung zwischen dem globalen und dem lokalen Management und ist die Grundlage für die Absprache der Netz- und der CPU-Planung.

Das System skaliert, indem der Rechner der zentralen Planung angepaßt wird. Auf diese Weise können die anderen Knoten des Netzes unverändert bleiben und die Broadcast-Fähigkeiten des MA-Netzes können für kurze Kommunikationswege zur Planung genutzt werden.

### 7.1.3 Lokales Management

Das *lokale Netzmanagement* ist in allen Knoten des verteilten Systems aktiv und führt die Leistungsverhandlung zwischen Netz und CPU durch. Das Management ist aufgrund des zentralisierten Servers nur an der Kanalplanung beteiligt, bei der es selber Kommunikationsteilnehmer ist. Die vom Reservierungsprotokoll vorgeschlagenen TDMA-Zugriffszeiten sind von der Flußkontrolle des Kanals einzuhalten und der Aufwand der Datenübertragung ist in die Echtzeitverarbeitung des Systems zu integrieren. Zu diesem Zweck wird die CPU sowohl auf der Sender- als auch auf der Empfängerseite explizit für jeden Kanal eingeplant. Der Einsatz synchronisierter Uhren und die zeitgesteuerte CPU-Nutzung garantieren die Vorhersagbarkeit des Zugriffsprotokolls.

### 7.1.4 Datenübertragung

Ich gehe von einem Multiple-Access-Bus aus, der zunächst keine Echtzeiterstärkung bietet. Das Zugriffsprotokoll ist nach den aufgestellten Entwurfskri-

terien schrittweise entworfen worden: Die Forderungen nach einer dynamischer Planung, die auch harten Zeitbedingungen genügen kann, nach der Einsatzfähigkeit kleiner, leistungsschwacher Rechner und nach der planbar engen Kopplung abhängiger Aufgaben werden zunächst mit einem zeitgesteuerten Zugriff (dynamisches TDMA) voll erfüllt.

In komplexen Systemen werden aber meist heterogene Leistungsanforderungen an die Kommunikation gestellt und eine rein zeitbasierte Lösung bietet dafür nicht genügend Flexibilität. Den Vorteil einer solchen Lösung, innerhalb eines Zeitintervalls isoliert von anderen Applikationen auf das Netz zugreifen zu können, nutze ich für den Einsatz zusätzlicher Zugriffsprotokolle. Innerhalb eines Intervalls kann man mehreren Nutzern den Zugriff erlauben (Kanalüberlagerung) und die BM-Vergabe nach anderen, auch ereignisgesteuerten, Verfahren planen und steuern. Die Vorteile eines zeitgesteuerten und eines ereignisgesteuerten Ansatzes können in einem System kombiniert werden! Neben einer zeitlich sehr genau vorhersagbaren Kommunikation ist die störungsfreie Integration weniger strenger Anforderungen möglich.

Das dynamische TDMA-Verfahren kann man auf jedem Multiple-Access-Netz einsetzen, wenn sich die Uhren hinreichend genau synchronisieren lassen. Das MA-Netz bildet eine minimale Ausgangsbasis, auf der sich daneben viele weitere Zugriffsprotokolle, in Hard- oder Software, aufsetzen lassen, wenn sie der Zeitsteuerung nachgeordnet werden können. Das entworfene Planungsmodell läßt eine Kombination verschiedener Algorithmen und Parametrisierungen in *einer* Struktur zu (s.u.).

### 7.1.5 Planung

Die Planung der Netzzugriffe im Server gleicht einer CPU-Planung und bestimmt die Eigenschaften des dynamischen TDMA für den Netzzugriff in den Kanälen. Die reservierten Zeiten werden nicht interpretiert, so daß es keine Unterscheidung zwischen WCET oder EET (zwischen HRT und SRT) gibt; dies ist einer höheren Ebene vorbehalten. Der Einsatz unter harten Zeitbedingungen ist prinzipiell möglich, weil der gesamte Entwurf die Verfügbarkeit der reservierten Zeiten garantieren kann. Der hohen Vorhersagbarkeit der expliziten Planung stehen auch einige Nachteile gegenüber, wie zum Beispiel die schlechte Auslastung ungenutzter, verplanter TDMA-Intervalle.

Diese Problematik ist analysiert worden und Konzepte zur Lösung sind in den Entwurf eines generischen Planungsmodells eingeflossen, das die explizite Planung und Reservierung von Betriebsmitteln unterstützt und die Anwendung alternativer Planungsverfahren innerhalb bestimmter Zeitintervalle zuläßt. Das Modell beschränkt keine zeitlichen Parameter und erlaubt sogar eine zeitlich begrenzte Reservierung mit Start und Ende in der Zukunft. Die wesentliche, neue Idee gegenüber einer „herkömmlichen“ TDMA-Planung besteht darin, daß man mehreren Nutzern gleichzeitig innerhalb eines festen Zeitintervalls den BM-Zugriff

erlaubt. *Konflikttests*, die den Reservierungseinträgen zugeordnet sind, kontrollieren, ob diese *Überlagerung* nach der Spezifikation möglich ist, und erlauben damit den Einsatz weiterer Planungsalgorithmen innerhalb bestimmter Zeiten der TDMA-Planung. Die sehr große Erweiterbarkeit der Planung eröffnet viele unterschiedliche Einsatzmöglichkeiten, die über die explizite Planung wechselseitig ausschließender Zugriffszeiten hinausgehen. Die kombinierte Planung unterschiedlicher Zugriffsprotokolle erschließt ein breites Einsatzfeld, das derart bisher kaum untersucht worden ist.

## 7.2 Was leistet die Implementierung?

Die Implementierung demonstriert die grundsätzliche Realisierbarkeit des Konzepts. Es ist ein funktionsfähiges Kommunikationssystem erstellt worden, das in einer Beispielapplikation genutzt wird. Die Beispielapplikation zeigt den dynamischen Auf- und Abbau von Kanälen und demonstriert einfache Überlagerungsmöglichkeiten. Die Verfügbarkeit der Zugriffszeiten ist bei einer exklusiven Reservierung garantiert, wenn man von einem fehlerfreien Netz ausgeht. Temporäre Übertragungsfehler können auf den entsprechenden Kanal beschränkt werden und sind durch die Applikation zu behandeln.

Die Implementierung ist, genau wie das Konzept, eine Basis für die Untersuchung verschiedener Lösungsansätze und Erweiterungen.

## 7.3 Ausblick

Zukünftige, komplexe Anwendungen in verteilten Systemen erfordern eine flexible Planung unterschiedlicher Leistungsparameter. Die Probleme der HRT-Planung und -Spezifikation und die Integration von SRT-Kommunikation in *ein* Kommunikationssystem müssen gelöst werden.

Das Konzept bietet dazu Lösungsmöglichkeiten an. In zukünftigen Arbeiten sind vorrangig die Kanalüberlagerungen zu untersuchen. Sie heben einige gravierende Nachteile der rein zeitgesteuerten Planung auf und erlauben sogar die Kombination unterschiedlicher Zugriffsprotokolle und damit den Einsatz unter weichen Zeitbedingungen. Die Vorhersagbarkeit der HRT-Kommunikation bleibt wegen der strikten zeitlichen Trennung in unterschiedliche TDMA-Intervalle erhalten. Die Vorteile der Zeitsteuerung können für die HRT-Kommunikation und die Vorteile der impliziten, ratenbasierten Steuerung (zum Teil) für die SRT-Kommunikation genutzt werden.

Ein nächster Schritt für die Erweiterung dieser Diplomarbeit könnten Routingalgorithmen in der Netzwerkschicht oder umfangreichere Protokolle in der Transportschicht sein. Für eine vollständige Integration des hier vorgestellten, grundlegenden Kommunikationsdienstes in ein verteiltes Echtzeitsystem wird der

Entwurf einer umfassenden Leistungsverwaltung notwendig. Die Leistungsverhandlungen dürfen nicht nur unterhalb der vierten OSI-Schicht stattfinden, sondern müssen darüber hinaus durch alle Schichten bis zur Applikation reichen, um Veränderungen in allen Teilen des Systems bzw. der Applikationen zu erfassen.



# Anhang A

## Iteratoren

Im Konzept und in der Implementierung von Klassen benutze ich *Iteratoren* für den Zugriff auf und die Navigation durch die internen Datenbestände; siehe z.B. [SL94], dem diese Darstellung folgt.

### A.1 Datenzugriff

Alle „Containerklassen“, also solche, die Daten verwalten, müssen eine Möglichkeit anbieten, auf diese Daten zuzugreifen. Iteratoren sind verallgemeinerte Zeiger, die einen Verweis auf ein Datum darstellen und damit den Zugriff darauf ermöglichen. Ein Iterator ist, als Teil der Containerklasse, eine Klasse, die den Zugriff auf die Containerdaten regelt.

Eine Abstraktion der gespeicherten Daten von den bereitgestellten Daten ist durch den Weg über den Iterator möglich. Das interne Format der Datenspeicherung muß also nicht nach außen gegeben werden oder sichtbar sein. Der Zugriff auf die Daten kann beliebig kontrolliert werden: (i) nur das ist sichtbar, was man möchte und (ii) Veränderungen der Daten können unterbunden oder nachverfolgt werden. Den Datenzugriff nennt man wie bei den Zeigern *dereferenzieren*.

### A.2 Navigation

Neben dem Zugriff auf einzelne Daten ist die Navigation durch die Daten einer Containerklasse zu regeln. Funktionen für das Setzen und das Versetzen der Iteratoren müssen zur Verfügung stehen. Auch eine Kontrolle über die Dereferenzierbarkeit muß entweder automatisch oder von Hand durchgeführt werden. Ich gehe nicht von einer automatischen Prüfung aus.

Das Setzen der Iteratoren wird konzeptionell durch eine Funktion der Instanz der entsprechenden Containerklasse durchgeführt. Dazu gehören, Funktionen die das erste, letzte, das n-te Element oder das erste Element größer 42 zurückgeben, also einen Iterator der auf das entsprechende Element oder auf ein ungültiges

verweist. Die Funktion `end()`, die einen Iterator auf das letzte Element liefert, benutzt hierfür oft ein vordefiniertes leeres Datum. Die Technik, ein vordefiniertes letztes Element im Container zu haben, bietet dabei eine Reihe von Möglichkeiten. Man umgeht die Sonderbehandlung des leeren Containers und vermeidet so zusätzliche Bedingungen in der Verarbeitung. Als vereinbarter ungültiger Iterator kann die Prüfung auf Dereferenzierbarkeit vereinfacht werden.

Für das Versetzen des Iterators braucht man nicht unbedingt die zugehörige Containerklasse. Von einem bekannten Datum aus kann man den Iterator auf ein neues relativ zum alten versetzen. Hierbei unterscheidet man unterschiedliche Typen<sup>1</sup>:

- Ein *forward iterator* erlaubt nur das Weitersetzen des Iterators auf *das nächste* Datum. Ein Vergleich mit `end()` beschränkt die Suche am Ende des Containers.
- Ein *bidirectional iterator* ist ein forward iterator, der auch auf das vorhergehende Datum zurückgesetzt werden kann. Man kann also beliebig in beiden Richtungen zwischen erstem und letztem Element wandern.
- Ein *random access iterator* erlaubt neben dem bidirektionalen auch den beliebigen Datenzugriff über ein Nummerierungsschema. Arrays mit einer Indizierung über die natürlichen Zahlen bieten diese Möglichkeit zum Beispiel an.

Alle Datenbestände, die aufzählbar sind, lassen sich damit abdecken. Von der einfach verketteten, doppelt verketteten Liste und einem Array — entsprechend den obigen drei Iteratortypen — über mehrdimensionale Listen und Bäume, bei denen die Funktionen für das Versetzen der Iteratoren komplizierter sind und evtl. zusammen mit der zugehörigen Containerklasse durchgeführt werden.

### Navigation ohne Container

Ein großer Vorteil bei der Verwendung in Standarddatenstrukturen wie Liste, Array, Heap,... ist die Möglichkeit, mit den Iteratoren auf den Daten unabhängig von der zugrundeliegenden Datenspeicherung zu arbeiten. Sogar die Verarbeitung zwischen unterschiedlichen Typen von Strukturen ist möglich. Ein bidirektionaler Iterator in einem Array von Zahlen ist nicht von einem in einer Liste von Zahlen zu unterscheiden!

Für die Schnittstelle der in dieser Arbeit entworfenen Datenstrukturen spielt dieser Aspekt keine Rolle. Aber bei der Implementierung ist es wichtig, daß man die verwendeten Datentypen in ihrer Realisierung frühzeitig abstrakt hält. Wenn mit einem bidirektionalen Iterator umgegangen wird, ist es zunächst für die Implementierung nicht erheblich, ob die Daten in einem Array, einer Liste oder einem

---

<sup>1</sup>Die Unterscheidung ist aus [SL94] übernommen und reicht für die in dieser Arbeit untersuchten Datenstrukturen aus.

Baum gespeichert sind. Der Einsatz in Echtzeitsystemen erfordert natürlich eine genaue Analyse des Aufwandes, die aber durch den Gebrauch der Iteratoren nicht erschwert wird.



# Literaturverzeichnis

- [Abr70] N. Abramson. The ALOHA system — another alternative for computer communications. In *Proceedings of the AFIPS Fall Joint Computer Conference*. AFIPS Press, Montvale, N.J., November 1970.
- [B<sup>+</sup>93] A. Burns et al. Allocating and scheduling hard real-time tasks on a point-to-point distributed system. In *Proceedings of The Workshop on Parallel and Distributed Real-Time Systems*, pages 11–20, April 1993.
- [Bak91] T.P. Baker. Stack-based scheduling of realtime processes. *Journal of Real-Time Systems*, 3, 1991.
- [Bau96] B. Baumgarten. *Petri-Netze – Grundlagen und Anwendungen*. Spektrum Akademischer Verlag GmbH, 1996.
- [BF<sup>+</sup>94] A. Banerja, D. Ferrari, et al. The Tenet real-time protocol suite: Design, implementation and experiences. Technical Report TR-94-059, University of California at Berkeley, 1994.
- [BHG87] P.A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley, Reading, MA, USA, 1987.
- [BKM<sup>+</sup>91] S. Baruah, G. Koren, D. Mao, et al. On the competitiveness of on-line real-time task scheduling. In *Proceedings of the Real-Time Systems Symposium*, pages 106–115, 1991.
- [Bla76] J. Blazewicz. Scheduling dependant tasks with different arrival times to meet deadlines. In E. Gelenbe and H. Beilner, editors, *Modelling and Performance Evaluation of Computer Systems*. North-Holland, 1976.
- [BMR96] Roberto Baldoni, Achour Mostefaoui, and Michel Raynal. Causal delivery of messages with real-time data in unreliable networks. *Real-Time System*, 10:245–262, 1996.

- [Bö95] Stefan Böcking. Communication performance models. Technical report, International Computer Science Institute, Berkeley, 1995.
- [Bos91] Bosch. CAN Specification, Ver. 2.0. Robert Bosch GmbH, Stuttgart, 1991.
- [Bri83] E. Brieskorn. *Lineare Algebra und Analytische Geometrie*. Vieweg, 1983.
- [Cap79] J.I. Capetanakis. Tree algorithms for packet broadcast channels. *IEEE Transactions on Information Theory*, 25:505–515, Sept. 1979.
- [CAZ92] B. Chen, G. Agrawal, and W. Zhao. Optimal synchronous capacity allocation for hard real-time communication with timed token protocol. In *Proceedings of the Real-Time Systems Symposium*, pages 198–207, 1992.
- [CCH94] A. Campbell, G. Coulson, and D. Hutchison. A quality of service architecture. *Computer Communications Review*, 24(2), April 1994.
- [Cri89] F. Cristian. Probabilistic clock synchronization. *Distributed Computing*, 3:146–158, 1989.
- [CS97] Chih-Che Chou and Kang G. Shin. Statistical Real-Time Channels on Multiaccess Bus Networks. *IEEE Transactions on Parallel and Distributed Systems*, 8(8), August 1997.
- [DNS94] M. Di Natale and J.A. Stankovic. Dynamic end-to-end guarantees in distributed real-time systems. In *Proceeding of the Real-Time Systems Symposium*, volume 15. IEEE, 1994.
- [DP91] M. De Prycker. *Asynchronous Transfer Mode*. Prentice Hall, 1991.
- [DZ83] J.D. Day and H. Zimmerman. The OSI Reference Model. In *Proc. of IEEE*, volume 71, pages 1334–1340, Dec 1983.
- [Fer90] D. Ferrari. Client Requirements for Real-Time Communication Services. *IEEE Communications Magazine*, 28(11):65–72, Nov 1990.
- [FH76] M.J. Fischer and T.C. Harris. A model for evaluating the performance of an integrated circuit and packet switched multiplex structure. *IEEE Transactions on Communication*, 24(2):195–202, Feb 1976.
- [Gal78] R.G. Gallager. Conflict resolution in random access broadcast networks. In *Proceedings of the AFOSR Workshop in Communication Theory and Applications*. IEEE, Piscataway, N.J., September 1978.

- [GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, San Francisco, 1979.
- [Gro82] R.M. Grow. A timed token protocol for local area networks. In *Proceedings of Electro/82, Token Access Protocols*, 1982.
- [GS94] M. Gergeleit and H. Streich. Implementing a Distributed High-Resolution Real-Time Clock using the CAN-Bus. In *1st Intl. CAN-Conference*, Erlangen, September 1994. Can in Automation e.V.
- [Hal95] Fred Halsall. *Data Communications, Computer Networks and Open Systems*. Addison-Wesley, 1995.
- [IEEa] IEEE 802.3 Carrier Sense Multiple Access with Collision Detecion. IEEE Publication Service.
- [IEEb] IEEE 802.4 Token Bus. IEEE Publication Service.
- [IEEc] IEEE 802.5 Token Ring. IEEE Publication Service.
- [ISO92] ISO. Road Vehicles — Interchange of Digital Information — Controller Area Network (CAN) for High Speed Communication. ISO DIS 11898, February 1992.
- [Jef85] D.R. Jefferson. Virtual time. *ACM Transactions on Programm. Lang. Syst.*, 7(3), 1985.
- [K+89] H. Kopetz et al. Distributed fault-tolerant real-time systems: The mars approach. *IEEE Micro*, pages 25–41, 1989.
- [KG94] H. Kopetz and G. Grünsteidl. TTP — a protocol for fault tolerant real-time systems. *IEEE Computer*, 27(1):14–23, 1994.
- [KGM96] B. Kao and H. Garcia-Molina. Scheduling soft real-time jobs over dual non-real-time servers. *IEEE Transactions on Parallel and Distributed Systems*, 7(1):56–68, January 1996.
- [KO87] H. Kopetz and W. Ochenreiter. Clock synchronization in distributed real-time systems. *IEEE Transaction on Computers*, 36(8):933–940, 1987.
- [Kop91] H. Kopetz. Event-triggered versus time-triggered real-time systems. Technical Report 8/91, Technische Universität Wien, Vienna, Austria, 1991.
- [Kop93] H. Kopetz. Scheduling. In S. Mullender, editor, *Distributed Systems*, pages 491–509. Addison-Wesley, 1993.

- [KS83] J.F. Kurose and M. Schwartz. A family of window protocols for time-constraint applications in CSMA networks. In *Proceedings of the INFOCOMM 83*. IEEE, Piscataway, N.J., April 1983.
- [KS90] D.D. Kandlur and K.G. Shin. Traffic routing for networks with virtual cut-through capability. In *Proc. 10th International Conference on Distributed Computing Systems*, pages 398–405. IEEE, May 1990.
- [KSF94] D.D. Kandlur, K.G. Shin, and D. Ferrari. Real-time communication in multi-hop networks. *IEEE Transactions on Parallel and Distributed Systems*, 5(10), 1994.
- [KSY84] J.F. Kurose, M. Schwartz, and Y. Yemini. Multiple-access protocols and time-constrained communication. *ACM Computing Surveys*, 16(1), 1984.
- [KT95] A. J. Wellings K. Tindell, A. Burns. Calculating Controller Area Network (CAN) Message Response Times. *Control Engineering Practice*, 3, 8 1995.
- [Kur84] J.F. Kurose. *Time-constraint Communication in Multiple Access Networks*. PhD thesis, Columbia University, New York, 1984.
- [Lam78] L. Lamport. Time, Clocks, and the Ordering of Events in a Distributed Environment. *Communications of the ACM*, 21:558–564, July 1978.
- [Law83] E.L. Lawler. Recent results in the theory of machine scheduling. In A. Bachem, editor, *Mathematical Programming: the State of the Art*. Springer-Verlag, 1983.
- [LB92] J.Y. Le Boudec. The Asynchronous Transfer Mode: a tutorial. *Computer Networks and ISDN Systems*, 24:279–309, 1992.
- [LeL87] G. LeLann. The 802.3D protocol: A variation of the IEEE 802.3 standard for real-time LANs. Technical report, INRIA, France, 1987.
- [LKG92] F. Lange, R. Kroger, and M. Gergeleit. JEWEL: Design and implementation of a distributed measurement system. *IEEE Transactions on Parallel and Distributed Systems*, Dez 1992.
- [LL73] C.L. Liu and J.W. Layland. Scheduling Algorithms for Multiprogramming in a Hard Real Time Environment. *Journal of the ACM*, 20(1):46–61, 1973.

- [MB76] R.M. Metcalfe and D.R. Boggs. Ethernet: Distributed packet switching for local computer networks. *Communications of the ACM*, 19(7):395–403, July 1976.
- [MIS96a] A. Mehra, A. Indiresan, and Kang G. Shin. Design and evaluation of a QoS-sensitive communication subsystem architecture. Technical Report CSE-TR-280-96, University of Michigan, 1996.
- [MIS96b] A. Mehra, A. Indiresan, and Kang G. Shin. Resource Management for Real-Time Communication: Making Theory meets Practice. In *Proc. IEEE Real-Time Technology and Applications Symposium*, June 1996.
- [MK85] M.L. Molle and L. Kleinrock. Virtual time CSMA: Why two clocks are better than one. *IEEE Transactions Comm.*, 33(9), September 1985.
- [ML81] B. Maglaris and T. Lissack. A priority TDMA protocol for satellite data communication. In *Proceedings of the International Communications Conference*. IEEE, Piscataway, N.J., June 1981.
- [MSST93] S. Mukherjee, D. Saha, M. Saksena, and S. Tripathi. A bandwidth allocation scheme for time constrained message transmission on a slotted ring lan. In *Proceedings of IEEE RTSS*, pages 44–53, 1993.
- [Mü97] Gero Mühl. Dynamische Planung abhängiger Tasks. Master’s thesis, FernUniversität Hagen, 1997.
- [Mul93] S.J. Mullender, editor. *Distributed Systems*. Addison-Wesley, 2nd edition, 1993.
- [MZ95] N. Malcolm and W. Zhao. Hard Real-Time Communication in Multiple-Access Networks. *Real-Time Systems*, 8:35–77, 1995.
- [Net97] E. Nett. Real-time behaviour in a heterogeneous environment. In *Proceedings of WORDS’97*, Newport Beach, USA, Feb 1997.
- [NS95] K. Nahrstedt and J.M. Smith. The QoS Broker. *IEEE Multimedia Magazine*, 2(1), 1995.
- [NS97] E. Nett and H. Streich. The GMD-Snake – Real-Time Scheduling of a Flexible Robot Application at Run-Time. In *Intl. Workshop on Parallel Computation and Scheduling*, Ensenada, Mexiko, Aug 1997.
- [NSB<sup>+</sup>96] E. Nett, H. Streich, P. Bizzari, A Bondavalli, and F. Tarini. Adaptive software fault tolerance policies with dynamic real-time guarantees. In *Proceedings of WORDS’96*, pages 78–85. IEEE Computer Society Press, 1996.

- [NT94] B.C. Neuman and T. Ts'o. Kerberos: An Authentication Service for Computer Networks. *IEEE Comm. Magazine*, 32:33–38, 1994.
- [OSI84] Basic Reference Modell for Open Systems Interconnection. ISO 7498, 1984.
- [Pal92] B.G. Palmer. A comparison of three protocols supporting time-dependant and time-independent. Master's thesis, Worcester Polytechnic Institute, 1992.
- [Par92] C. Partridge. A proposed flow specification. RFC 1363, September 1992.
- [PDK96] K.L. Paap, M. Dehlwisch, and B. Klaasen. GMD-Snake: a semi-autonomous Snake-like Robot. In *3rd Intl. Symposium on Distributed Autonomous Robotic Systems*, 1996.
- [Ple92] P. Pleinevaux. An improved hard real-time scheduling for the IEEE 802.5. *Journal of Real-Time Systems*, 4(2):99–112, 1992.
- [RKS90] P. Ramanathan, D.D. Kandlur, and K.G. Shin. Hardware assisted software clock synchronization for homogeneous distributed systems. *IEEE Transaction on Computers*, 39:514–524, April 1990.
- [Ros89] F. Ross. An overview of FDDI: the Fiber Distributed Data Interface. *IEEE Journal on Selected Areas in Communication*, 7(7), 1989.
- [SdA94] M. Saksena, J. da Silva, and A. Agrawala. Design and implementation of maruti-ii. In Sang H. Son, editor, *Principles of Real-Time Systems*. Prentice Hall, 1994.
- [Shi91] K. G. Shin. HARTS: A distributed real-time architecture. *IEEE Computer*, 24(5):25–35, 1991.
- [SL94] A.A. Stepanov and M. Lee. The Standard Template Library. Technical Report HPL-94-34, Hewlett-Packard Laboratories, April 1994.
- [SM89] Strosnider, J.K. and Marchok, T.E. Responsive, deterministic IEEE 802.5 Token Ring scheduling. *Journal of Real-Time Systems*, 1(2):133–158, 1989.
- [SNR91] J.A. Stankovic, D. Niehaus, and K. Ramamritham. Springnet: a scalable architecture for high-performance, predictable, and distributed real-time computing. Technical Report 91-74, Department of Computer Science, University of Massachusetts at Amherst, 1991.

- [SNS88] J.G. Steiner, C. Neuman, and J.I. Schiller. Kerberos: An Authentication Service for Open Network Systems. In *Proceedings of the Usenix Winter Conference*, pages 191–201, 1988.
- [SR91] J.A. Stankovic and K. Ramamritham. The Spring Kernel: a new paradigm for real-time operating systems. *IEEE Software*, 8(3), 1991.
- [SRL90] L. Sha, R. Rajkumar, and J. Lehoczky. Real-Time Scheduling Support in Futurebus+. In *Proc. 11th Real-Time Systems Symposium*, pages 331–340. IEEE, 1990.
- [SRLR89] L. Sha, R. Rajkumar, J. Lehoczky, and K. Ramamritham. Mode change protocols for priority-driven preemptive scheduling. *Real-Time Systems*, 1989.
- [SRN91] J. Stankovic, K. Ramamritham, and E. Nahum. Predictable inter-process communication for hard real-time systems. In *Proceedings of the 10th IFAC Workshop on Distributed Computer Control Systems*. Pergamon Press, 1991.
- [SS93] L. Sha and S.S. Sathaye. A systematic approach to designing real-time systems. *IEEE Computer*, 26(9):68–78, September 1993.
- [SSL89] Sprunt, B., Sha, L., and Lehozky, J. Aperiodic task scheduling for hard real-time systems. *Journal of real-Time Systems*, 1(1):27–60, 1989.
- [SSMT94] D. Saha, M. Saksena, S. Mukherjee, and S. Tripathi. On guaranteed delivery of time-critical messages in dqdb. In *Proceedings of IEEE Infocomm*, pages 272–279, 1994.
- [SSNB95] J. Stankovic, M. Spuri, M. Di Natale, and G. Buttazzo. Implications of classical scheduling results for real-time systems. *IEEE Computer*, 28(6):16–25, 1995.
- [Sta88] J.A. Stankovic. Misconceptions about Real-Time Computing: A Serious Problem for Next-Generation Systems. *IEEE Computer*, 21:10–19, 1988.
- [Str95] H. Streich. TaskPair-Scheduling: An approach for dynamic real-time systems. *Intl. Journal of Mini & Microcomputers*, 17(2):77–83, 1995.
- [Tan95] Andrew S. Tanenbaum. *Verteilte Betriebssysteme*. Prentice Hall, 1995.
- [Tan96] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall, 1996.

- [Tin94] K. Tindell. Analysis of hard real-time communication. Technical Report YCS-222, University of York, England, Jan 1994.
- [VC95] C. Venkatramani and T. Chiueh. Design, Implementation, and Evaluation of a Software-based Real-Time Ethernet Protocol. In *Proceedings of SIGCOMM '95*, 1995.
- [vEBBV95] T. von Eiken, A. Basu, V. Buch, and W. Vogels. U-Net: a user-level network interface for parallel and distributed computing. *Operating Systems Review*, 29(5), December 1995.
- [Wei78] Joseph Weizenbaum. *Die Macht der Computer und die Ohnmacht der Vernunft*. Suhrkamp Verlag, 1978.
- [WM91] B. Wolfinger and M. Moran. A continuous media data transport service and protocol for real-time communication in high speed networks. In *Proc. 2nd Intl. Workshop on Network and Operating System Support for Digital Audio and Video*, Nov 1991.
- [ZA95] Dieter Zöbel and Wolfgang Albrecht. *Echtzeitsysteme*. Intern. Thomson Publishing, 1995.
- [ZK91] H. Zhang and S. Keshav. Comparison of rate-based service disciplines. In *Proceedings of SIGCOMM '91*, ACM SIGCOMM Computer Communication Review, September 3–6 1991.
- [ZR87] W. Zhao and K. Ramamritham. Virtual time CSMA protocols for hard real-time communication. *IEEE Transactions on Software Engineering*, 13(8), 1987.

# Index

## A

Abhängigkeit, 43, 111  
Abräumkante, 80  
Aktivität, 84, 94  
Ankunftszeit, 27  
arrival time, *siehe* Ankunftszeit  
ATM, 33, 104  
Aufgabe, *siehe* Echtzeitaufgabe  
Aufteilung, *siehe* Splitting  
Ausführungszeit, *siehe* execution time

## B

Bandbreite, 70, 101  
Beteiligte Komponenten, *siehe* Komponenten  
Betriebsmittel, 94  
-konflikt, *siehe* Konflikt  
-vergabe, 112  
einfaches, 94  
Leistungsspezifikation, *siehe* Leistungsspezifikation  
mehrfaches, 94  
Teilbarkeit, *siehe* Teilbarkeit  
Betriebssystem, 9  
Binary Exponential Backoff, 26  
Brücke, 47  
Broadcast  
-netzwerk, 23  
Broadcastkanal, 84  
Brücke, 34  
Burst, 95

## C

CAN, 18, 26, 131  
CD, *siehe* Kommunikationsdaemon

Commit, 66  
CRC, 19, 87  
critical region, 65  
CS, *siehe* Kommunikationsserver  
CSMA, 25  
CT, *siehe* Kommunikationsthread

## D

Datenhaltung  
teilweise Sicht, 56  
totale Sicht, 56  
Datenrate, 100  
kontinuierliche, 100  
periodische, 100  
variable, 100  
Datenübertragung, 14  
Deadline  
-verletzung, 129  
Delay, *siehe* verzögerung  
Dienst, 9, 91  
verteilter, 10  
Dienstnutzungspunkt, *siehe* SAP  
Dispatcher, 110  
Dominoeffekt, 8, 87  
Dual leaky bucket, 106  
Duplexkommunikation, 99  
Durchsatz, 101

## E

Echtzeit  
-aufgabe, 2, *siehe* Task  
-betriebssystem, *siehe* Betriebssystem  
-klasse, 6  
-kommunikation, 10, 12  
-system, 2

hart, 129  
 weich, *siehe* SRT, 129  
 Echtzeitkommunikation  
   weiche, 8  
 EDF, 90, 112  
 embedded systems, 13, 43  
 Empfang  
   direkter, 71  
   nebenläufiger, 70  
 Ende-zu-Ende Kommunikation, 16  
 Ende-zu-Ende Kommunikation, 51  
 Ethernet, 26  
 execution time  
   WCET (worst case), *siehe* WCET  
 execution time, 96  
   EET (expected case), 97  
 Externe Anfrage, 80  
**F**  
 FDDI, 33  
 Fehler  
   -behandlung, 129  
   -fortpflanzung, *siehe* Zeitfehler  
   -modell, 8, 86  
   -toleranz, 8, 10, 59, 86  
     CRC, *siehe* CRC  
   erkennung, 47  
   korrektur, 47  
**G**  
 Garantie, 3, 93  
   globale, 50  
   Güte, 102  
   lokale, 50  
   statistische, 4, 96, 101  
 Garantievertrag, 49  
 Granularität, *siehe* Tightness  
 Gruppenkommunikation, 83  
**H**  
 Halbduplex, 99  
 Hardware, 39, 131  
 HARTS, 35  
 Hochgeschwindigkeitsnetz, 33

HRT (hard real-time), 6

## I

Integration, 14, 19, 35, 51, 52, 68  
 Interne Anfrage, 78  
 Isolation, 44  
 Iterator, 123, 153  
   bidirectional, 154  
   forward, 154  
   random access, 154

## J

Jitter, 98

## K

Kalender, *siehe* Plan, explizit  
 Kanal, 15, 51  
   -ID, 61, 63, 132  
   -abbau, 52, 68  
   -aufbau, 52, 77  
     global, 63  
   -nutzung, 52, 55, 81, *siehe* Kanal-  
     objekt  
   -objekt, 81, 134  
   -überlagerung, *siehe* Überlagerung  
   -umplanung, 52  
   statischer, 62  
 Knotenverwaltung, 62  
 Kollision, 13, 23, 50  
 Kollisionszeit, 25  
 Kommunikation, *siehe* Echtzeitkom-  
   munikation  
   -NRT, 74  
   HRT-, 103  
   Punkt-zu-Punkt, *siehe* Punkt-zu-  
     Punkt  
   SRT-, 86  
   verbindungslose, 89  
 Kommunikationsdaemon, 71, 72  
 Kommunikationsform, 99, 107  
 Kommunikationsserver, 57, 61  
 Kommunikationstask, 75  
 Kommunikationsthread, 47, 71, 75  
   -planung, 80

- Kommunikationsvertrag, *siehe* Garantievertrag
- Komponenten  
  aktiv beteiligte, 57  
  passiv beteiligte, 57  
  unbeteiligte, 57
- Konflikt, 3
- Konflikttest, 115, 120, 124, 125, 139
- Kontingenz, 73, 75, 78
- Kontrollfluß, 42
- L**
- LAN, 23
- Laufzeitrepräsentation, 14
- Laxity, *siehe* Spielraum
- Leistungsspezifikation, *siehe* QoS, 96
- Leistungsverhandlung, 15, 37, 45, 49, 73, 83, 151
- low-interference, 56
- M**
- MAC, 24, 46, 47
- MARS, 37
- Medium Access Control, *siehe* MAC
- Multiple Resource Scheduling, 73
- Multiplexing, 127  
  statistisches, 86
- Multiplexverfahren, 95  
  statistische, 95, *siehe* Garantiezeit-, 95
- N**
- Netzmanagement  
  globales, 18, 56  
  Komponenten, *siehe* Komponenten  
  ten  
  lokales, 19, 68  
  Planung, 56  
  dezentrale, 58  
  externe Anfrage, 80  
  interne Anfrage, 78  
  zentrale, 57  
  Verteilung, 56
- Netzsegment, 19, 23, 34
- Netzzugriffskontrolle, *siehe* MAC
- NIC, 33, 132
- NRT-Kommunikation, 74
- O**
- OSI, 45, 47, 61, 71
- P**
- PDU (protocoll data unit), 44
- PE, *siehe* Planungseinheit
- periodic server, 89
- periodisch  
  unendlich, 117
- Petri-Netz, 63, 77
- Plan, 109  
  brauchbar, 109  
  explizit, 110  
  implizit, 110
- Planbarkeit  
  Test auf, 109
- planbasiert, 32, 110
- Planung, 3, 40, 72, 109  
  dynamische, 5, 12, 111  
  Granularität, 114, *siehe* Tightness  
  Reservierung, *siehe* Reservierung  
  statisch, 4  
  Modi, 5
- Planungsalgorithmus, 120, 124
- Planungsauslastung, 5, 114
- Planungseinheit, 94, 119
- Preemption, *siehe* Taskunterbrechung
- Priorität, 112  
  dynamische, 112  
  konstante, 112
- Prozeß, 42
- Punkt-zu-Punkt, 34, 51
- Q**
- QoS, 12, 45, 50, 51, 63, 107, 122  
  -Parameter, 98  
  -Umsetzung, 78, 82
- QoS-Negotiation, *siehe* Leistungsverhandlung

**R**

ratenbasiert, 31, 110  
 Redundanz, 8  
 Rekonfiguration, 3, *siehe* Planung, 12,  
*siehe* Kanalumplanung  
 reservation time, *siehe* RT  
 Reservierung, 110, 121, 122  
   Slotmanager, 124  
 Reservierungsauftrag, 127  
 Reservierungseinheit, 119, 124  
 Ring, 118, 124  
   -größe, 118  
   -offset, 119, 138  
   Mehrfachbelegung, 119, 122  
 RM, 112  
 Robotik, 11, *siehe* embedded systems,  
   43, 141  
 Router, 34, 47  
 RPC, 16, 99  
 RT, 21

**S**

SAP (service access point), 44  
 Scheduling, *siehe* Planung  
 Schichtenbildung, 44, *siehe* OSI  
 Schlange, 141  
 SDU (service data unit), 44  
 Segment, *siehe* Netzsegment, 46  
 Senden  
   automatisches, 69  
   direktes, 69  
 Sicht, *siehe* Datenhaltung  
 Simplexkommunikation, 51, 84, 99  
 Single Point of Failure, 58  
 Single Point of Failure, *siehe* Zentra-  
   lisierung  
 Skalierung, 58, 60  
 Slot, 27, 116  
 Software, 40, 132  
 Spielraum, 97  
 Splitting, 42, 109, 126  
 Spring, 36  
 SRT, 86

SRT (soft real-time), 7

System  
   verteiltes, 10  
 Systemstart, 62

**T**

Task, 42, 75  
   -splitting, *siehe* Splitting  
   -unterbrechung, 42, 87  
 TaskPair, 88, 129  
 TDMA, 20, 28, 37  
   -Intervalle, 28, 53  
   -Slots, *siehe* TDMA-Intervalle  
   -dynamisches, 53, 59  
   dynamisches, 28  
 Teilbarkeit, 94, 125  
   implizite, 94, 125  
 Tenet, 34, 105  
 Test  
   Planbarkeits-, 109  
 Thread, 42  
 Tightness, 43, 54, 70  
 Timed Token, 33  
 Token Bus, 29  
 Token Ring, 29, 32  
 Token-Verfahren, 29, 59, *siehe* Timed  
   Token  
 Transaktion, 58  
 Transparenz, 10  
 Transportschicht, 47

**U**

Überlagerung, 20, 75, 85, 108, *siehe*  
   Konflikttest, 125, 144  
 Überlastung, 6  
 Uhr  
   logische, 41  
   Synchronisation, 41  
 unidirektional, 99  
 Unterbrechung, 109

**V**

Verbindung, 51, 61

Verbindungsaufbau, *siehe* Kanalaufbau,global  
verbindungsorientiert, 99  
Verbindungsverwaltung, 61  
Verkehrsscharakteristik, 100, 108  
Verstopfung, 13  
Verstopfungen, 50  
Verteilung, 10  
Vertrag, *siehe* Garantievertrag  
Verwaltungskanal, 61  
verzögerung, 101  
Verzögerungsfunktion, 24, 133  
virtual time, 27  
Vorhersagbarkeit, 3, 93  
VxWorks, 132

**W**

WAN, 34, 91, 102  
WCET, 7, 97  
window protocol, 27  
Worst-Case, 70  
-Analyse, 7  
Ausführungszeit, *siehe* WCET

**Z**

Zeit  
  globale, 41, 62  
Zeitachse, 116  
Zeitfehler, 6, 87, 129  
  -fortpflanzung, 8, *siehe* Dominoeffekt  
Zeitscheibe, *siehe* Slot  
Zeitscheibenverfahren, *siehe* TDMA  
Zeitsteuerung, 41  
Zentralisierung, 57  
Zugriffsmuster, 20  
Zuverlässigkeit, 10