Universität Karlsruhe
Fakultät für Informatik
Institut für Telematik
76128 Karlsruhe

Netzwerk-Management und Hochgeschwindigkeits-Kommunikation

Teil XVII

Seminar WS 1997/98

Herausgeber:
Elmar Dorner
Stefan Dresler
Markus Hofmann
Hartmut Ritter
Jochen Schiller
Jochen Seitz

Universität Karlsruhe
Institut für Telematik

Interner Bericht 6/98
ISSN 1432–7864

Zusammenfassung

Der vorliegende Interne Bericht enthält die Beiträge zum Seminar „Netzwerk-Management und Hochgeschwindigkeits-Kommunikation“, das im Wintersemester 1997/98 zum siebzehnten Mal stattgefunden hat.

Die Themenauswahl kann grob in folgende drei Blöcke gegliedert werden:

1. Ein Block beschäftigt sich mit drahtlosen Kommunikationsprotokollen. Der erste Beitrag widmet sich drahtlosem ATM, während ein zweiter Beitrag Möglichkeiten der Satellitenkommunikation vorstellt. Außerdem werden Sicherheitsaspekte im GSM diskutiert.

2. Ein Beitrag stellt das Transis-System zur zuverlässigen, geordneten Gruppenkommunikation über IP vor.

3. Ein dritter Block präsentiert Möglichkeiten der Unterstützung schneller Kommunikation. Hier werden etwa die sogenannten xDSL-Techniken vorgestellt. Außerdem wird der Firewire-Ansatz konventionellen Bussystemen in Rechnern gegenübergestellt. Schließlich widmen sich zwei Beiträge neuartigen Konzepten im Betriebssystem-Bereich, welche eine Kommunikationsunterstützung versprechen.

4. Ein Block mit dem Schwerpunkt auf neuen Netzkonzepten geht in zwei Beiträgen auf sogenannte Aktive Netzwerke ein. Außerdem wird das in den USA betriebene Internet2 vorgestellt.

5. Ein letzter Block befaßt sich mit Fragen des Managements. Hier stehen die DISMAN- und die SNMPv3-Arbeitsgruppe im Vordergrund. Außerdem wird in einem Beitrag das Telecommunication Management Network (TMN) vorgestellt.

Abstract

This Technical Report includes student papers produced within small lessons called seminar of “Network Management and High Speed Communications”. For the seventeenth time this seminar has attracted a large number of diligent students, proving the broad interest in topics of network management and high speed communications.

The topics of this report can be divided into the following areas:

1. One block is devoted to wireless communication technology. First, Wireless ATM is presented, followed by a contribution on ways of satellite communication. Third comes a discussion on the security of the GSM system.

2. One contribution deals with Transis, a communication subsystem for reliable ordered multipeer communication over IP.

3. A third block comprises possibilities for high speed communication, e.g. by using the xDSL technology. Furthermore, the Firewire architecture and protocol are compared to conventional buses commonly found in PCs. Finally, two contributions are devoted to new concepts in operating systems featuring support for efficient communication.

4. One block puts its focus on forthcoming network concepts like Active Networks and the so-called Internet2.

5. Finally, there are three contributions related to network management. They present the DISMAN and SNMPv3 working groups. Furthermore, one talk gives an introduction to the Telecommunication Management Network (TMN).

Inhaltsverzeichnis

Vorwort

Das Seminar „Netzwerk-Management und Hochgeschwindigkeits-Kommunikation“ erfreute sich in den letzten Jahren immer größerer Beliebtheit. Gerade heutzutage sind Stichworte wie „ATM“, „Quality of Service“, „Mobil-Kommunikation“, „Gruppenkommunikation“, „Internet“ und „Management“ in aller Munde. Daher sind die Forschungsgebiete in diesen Bereichen auch von allgemeinem Interesse, so daß sie eine derartige Vielzahl von innovativen Arbeiten aufweisen können, deren Behandlung in anderen Lehrveranstaltungen so detailliert nicht möglich ist.

Jetzt liegt auch der nunmehr siebzehnte Seminarband als Interner Bericht vor. Durch die engagierte Mitarbeit der beteiligten Studenten konnte so zumindest ein Ausschnitt aus dem komplexen und umfassenden Themengebiet klar und übersichtlich präsentiert werden. Für den Fleiß und das Engagement der Seminaristen sei daher an dieser Stelle recht herzlich gedankt.

Die ausgesprochen gute Resonanz bei den Studenten hat uns veranlaßt, auch im Sommersemester 1998 ein derartiges Seminar – natürlich mit geändertem aktuellem Inhalt – durchzuführen, so daß bald ein weiterer Interner Bericht mit neuen Forschungsergebnissen aus innovativen Seminarbeiträgen erscheinen wird. Doch vorerst sollen im vorliegenden Band folgende Themengebiete vorgestellt werden:

Drahtloses ATM

Das B-ISDN-Netz wird als diensteintegrierendes, multimediafähiges Netz verstanden. Das zugrundeliegende ATM-Vermittlungskonzept unterscheidet verschiedene Dienstklassen, die abhängig von den Bedürfnissen der jeweiligen Anwendungen gewählt werden können. Die Vorteile von ATM sollen auch in zukünftige drahtlose Netze Eingang finden. Dieser Beitrag beschreibt das neue Konzept "drahtloses ATM" und stellt verschiedene Architekturen vor, die eine Umsetzung darstellen.

Sicherheitsaspekte in GSM

Mit zunehmender Bedeutung der Kommunikation wächst auch der Wunsch zur Benutzermobilität. Beim Telefon ist dieser Wunsch mittlerweile schon fast zu einer Selbstverständlichkeit geworden. Im vorliegende Beitrag werden neben Struktur und Verwaltungseinheiten des GSM-Netzes auch die Sicherheitsaspekte in der drahtlosen Kommunikation vorgestellt.

Möglichkeiten der Satellitenkommunikation

Als Erweiterung der traditionellen Mobilfunknetze stellt die Satellitenkommunikation ein enormes Zukunftspotential dar. Ein enormer Vorteil beim Einsatz der Satellitentechnik liegt dabei in dem riesigen lückenlos versorgten Gebiet. Anhand von aktuellen Beispielen gibt der Beitrag Einblick in den Aufbau eines Satellitenkommunikationssystems. Dabei werden die einzelnen Komponenten und die eingesetzten Protokolle vorgestellt.

Das Transis System

Anwendungen wie verteilte Datenbanken oder verteilte Simulation können von einem Transportdienst profitieren, welcher Nachrichten mehrerer Sender unter Zusicherung von Garantien hinsichtlich einer einzuhaltenden Ordnung der Nachrichten untereinander an eine Menge von Empfängern ausliefert. Transis ist ein solches Protokoll zur zuverlässigen Gruppenkommunikation bei gleichzeitig hoher Verfügbarkeit des Dienstes. Es wird seit 1991 an der Hebrew University in Jerusalem, Israel entwickelt. Die Ausarbeitung stellt verschiedene Aspekte des Systems dar. Zuerst wird die grundsätzliche Problematik zuverlässiger Gruppenkommunikation motiviert. Es folgt die generelle Darstellung der Architektur. Daran schließt sich eine Vorstellung des Konzepts der Mitgliedschaft innerhalb einer Gruppe sowie der Fähigkeit der Behandlung von Netzpartitionierungen an. Die Darstellung schließt mit einer Bewertung der Leistungsfähigkeit.

Hochgeschwindigkeitskommunikation im privaten Bereich

Während innerhalb der öffentlichen Netze nun schon seit längerem die Möglichkeiten zur Übertragung hoher Datenraten geschaffen sind, sind für den privaten Nutzer immer noch Datenraten von $56 kbit/s$ bis $128 kbit/s$ das Übliche. In letzter Zeit nun werden die Bemühungen um eine kostengünstige Anschlußtechnologie privater Haushalte intensiviert. In erster Linie interessant ist dabei der private Internet-Anschluß, der die Akzeptanz und Nutzung des Internet und der darin angebotenen Dienste fördern soll. Dahinter stehen selbstverständlich auch wirtschaftliche Interessen sowohl der potentiellen Endgeräteanbieter als auch der Diensteanbieter, die sich vom Handel im Internet („e-commerce“) Vorteile versprechen. Dieser Beitrag stellt verschiedene konkurrierende Technologien vor, die teilweise auf bestehender Infrastruktur aufbauen. Als aussichtsreichster Kandidat kann dabei vielleicht die xDSL-Technologie gelten, die erst kürzlich eine massive Förderung erfahren hat: Eine Gruppe von Unternehmen aus dem IT-Bereich, darunter Microsoft, Intel und Compaq, hat sich vor kurzem zusammengeschlossen, um die xDSL-Technologie voranzutreiben und zum praktischen Einsatz zu bringen. Darüber hinaus ist in diesem Bereich aber sicher noch viel Bewegung zu erwarten.

Hochleistungsbusse für die Hochgeschwindigkeitskommunikation

Mit der Entwicklung schneller und leistungsfähiger Lokal- und Weitverkehrsnetze auf der einen Seite und der enormen Leistungssteigerung der Endgeräte auf der anderen Seite ergeben sich neue Hindernisse für eine leistungsfähige Ende-zu-Ende-Kommunikation. Ein immer größer werdendes Hindernis stellt das Bussystem im Endgerät dar, insbesondere im typischen PC — insbesondere deshalb, weil von modernen Bussystemen nicht nur die Übertragung hoher Datenraten sondern auch Dienstgüteunterstützung erwartet wird. Der diesbezügliche Beitrag vergleicht zunächst herkömmliche PC-Bussysteme und gibt dann eine Einführung in das Bussystem nach dem Standard IEEE 1394 („firewire“), das nicht nur im Rechner selbst zum Einsatz kommen soll, sondern auch ein eng begrenztes LAN aufspannen kann.

Neue Betriebssystemkonzepte für die Kommunikation

Im Rahmen der Arbeiten auf dem Gebiet der Hochleistungskommunikation wurde schon bald festgestellt, daß schnelle Netze und leistungsfähige Anwendungen alleine nicht ausreichen, um einem Anwender eine hohe Leistungsfähigkeit zur Verfügung zu stellen. Gefragt sind unter anderem auch Betriebssysteme, die verschiedene Dienstqualitäten eines Netzes an eine Anwendung weiterreichen können. Unterstützende Mechanismen sind unter anderem echtzeitfähige Schedulingmechanismen, Mikrokernel, Protokollmechanismen ohne Kopiervorgänge etc.
Die beiden Beiträge zu dem Thema „Neue Betriebssystemkonzepte für die Kommunikation“ stellen den EXO-Kernel des MIT und das NEMESIS-Projekt aus Cambridge vor. Der Exokernel-Ansatz des MIT, USA, stellt einen relativ radikalen Ansatz dar, der beinahe alle Funktionen eines Betriebssystems außerhalb des Kerns ansiedelt, um sehr schnelle Kontextwechsel zu gewährleisten. Das NEMESIS-Betriebssystem der Universität Cambridge stellt Anwendungen vielerlei Mechanismen zum Multiplexen von Ressourcen zur Verfügung, wie jedoch diese Ressourcen von Anwendungen genutzt werden bleibt letzteren selbst überlassen. Ein NEMESIS-Kern ist lediglich ca. 250 Instruktionen groß!

Active Networks

Active Networks stellen ein Konzept dar, das — im Gegensatz zu traditionellen, rein auf den Datentransport ausgelegten Netzen — innerhalb des Netzes Funktionen zur Verfügung stellt, die beispielsweise den Zustand des Netzes überwachen oder Datenmanipulationen vornehmen können. Der Beitrag „Aktive Netzwerke“ stellt die grundlegenden Ideen und Konzepte aktiver Netzwerke vor, diskutiert an Hand einiger Beispiele die Vor- und Nachteile des Ansatzes und zeigt Möglichkeiten zur Integration aktiver Netze in das Internet der nächsten Generation auf.

Komponenten für Active Networks: Betriebssystem, Sicherheit, Programmiersprachen

Problematisch bei Active Networks ist der Zugriff potentiell sehr vieler Benutzer auf die Netzwerkressourcen, so daß sich hier schnell Fragen der Sicherheit und Betriebsmittelzuteilung ergeben. Da das Netz selbst nun mit Hilfe einer Programmiersprache programmiert werden soll, ist zusätzlich die Nachweisbarkeit bestimmter Eigenschaften von Programmen unabdingbar. Der Beitrag „Komponenten für Active Networks: Betriebssystem, Sicherheit, Programmiersprachen“ geht auf potentielle Probleme von Active Networks ein, wobei der Schwerpunkt auf den Punkten Sicherheit, Betriebssysteme von Netzwerkkommponenten, Ressourcenreservierung und Programmiersprachen für Netzkomponenten liegt.

Internet2: Die neue Generation des Internet

Die ständig zunehmende Anzahl von Kommunikationsteilnehmern im Internet sowie die Entstehung neuartiger Anwendungsfelder im Bereich der Internetkommunikation erfor-

dern die Entwicklung und die praktische Umsetzung fortschrittlicher Kommunikationsprotokolle. Mit der Standardisierung von IPv6 und RSVP wurde die konzeptionelle Grundlage zur Anpassung des Internets an die gewachsenen Anforderungen der Anwender geschaffen. Mit dem Internet 2 entsteht in den USA derzeit ein eigenständiges Forschungsnetz zur Einführung und Erprobung dieser neuen Kommunikationstechnologien. Der Beitrag „Internet2: Die neue Generation des Internet“ stellt sowohl die organisatorischen als auch die technologischen Randbedingungen dieses Projekts vor.

Verteiltes Management im Internet — die "DISMAN Working Group" der IETF

Durch die zunehmende Ausdehnung und Komplexität heutiger Netzwerke ist ein zentrales Management dieser Netze fast nicht mehr praktikabel. Die Struktur heutiger Unternehmen, deren Erfolg auf dem Funktionieren ihrer Kommunikations- und Datennetze beruht, ist ebenfalls Anlaß, die zentrale Managementstruktur durch ein verteiltes Management abzulösen. Erste Standardisierungsbemühungen in diesem Bereich sind bereits im Gange, wie das Beispiel der „DISMAN Working Group“ der Internet Engineering Task Force zeigt. Diese Arbeitgruppe hat sich zum Ziel gesetzt, verteiltes Management für das Internet auf der Basis des „Simple Network Management Protocols“ zu ermöglichen. Die ersten Ergebnisse dieser Arbeitsgruppe werden im Seminarbeitrag vorgestellt.

Management im Internet: die 'SNMPv3 Working Group' der IETF

Für das Management im Internet wurde in den 80er Jahren ein Rahmenwerk entwickelt, welches auf dem Simple Network Management Protocol basiert. Letzteres war zwar einfach zu implementieren und zu benutzen, hatte aber einige gravierende Nachteile, insbesondere bezüglich Sicherheit und Leistungsfähigkeit. Daher wurde das Rahmenwerk – mehr oder weniger erfolgreich – stetig weiterentwickelt, so daß zur Zeit die aktuelle Version 3 kurz vor der Verabschiedung als Internet-Standard steht. Dieses Rahmenwerk ist Thema des Seminarbeitrags.

Telecommunications Management Network: Die Q3-Schnittstelle

Die Verwaltung öffentlicher Netze bedarf anderer Einrichtungen als das Management lokaler Netze. Die von der International Telecommunications Union unter dem Namen Telecommunications Management Network (TMN) standardisierte Architektur umfaßt daher auch mehrere sich ergänzende Sichten auf ein eigens für die Verwaltung definiertes Netzwerk. Dabei existieren ebenfalls Schnittstellen, welche die einzelnen Komponenten des TMN miteinander verbinden. Diese Sichten und Schnittstellen sind Thema des Seminarbeitrags, wobei besonders die Q3-Schnittstelle herausgehoben wird, die in der Standardisierung am weitesten fortgeschritten ist.

Drahtloses ATM

Pawel Wocjan

Kurzfassung

Die Telematik wird in letzter Zeit von den beiden Konzepten der Breitband-Netzwerke, die verschiedene Dienste unterstützen (B-ISDN, Broadband Integrated Services Digital Networks), und der Mobilkommunikation beherrscht. Bis heute ließen sich diese Konzepte nicht zufriedenstellend verbinden. Drahtloses ATM (WATM, wireless ATM) integriert beide Konzepte und bietet damit neue Dienste an, die von keinem Telekommunikations-System bis heute erreicht werden.

1 Einführung

ATM-Technologie (ATM, Asynchronous Transfer Mode) mit der B-ISDN-Signalisierung setzt sich immer mehr als die Transport-Technologie für Festnetze durch, da sie eine flexiblere und effizientere Ausnutzung des Transportmediums als herkömmliche Techniken erlaubt. ATM kann „on-demand“ Bandbreite zur Verfügung stellen und garantiert die Qualität der Verbindung, die durch QoS-Parameter (Quality of Service) festgelegt wird. Erst dadurch werden leistungsfähige Multimedia-Anwendungen möglich.

WATM ist eine zwingende Konsequenz des Erfolgs der ATM-Technologie. Ein kurzer Blick in die Vergangenheit zeigt, daß alle erfolgreichen Festnetze eine drahtlose Erweiterung bekommen haben. Die Benutzer werden drahtlosen Zugang zu ATM fordern, um von überall (mobil) die Vorteile von ATM nutzen zu können. Das Ziel von ATM ist es, überall und jederzeit jedem schnelle und breitbandige Multimedia-Dienste anzubieten. Die bisherigen Systeme der Mobilkommunikation können nicht WATM ersetzen, da sie für spezielle Aufgaben konzipiert wurden und nicht alle Vorteile von ATM in sich vereinen.

WATM ist ein Thema, das auf sehr großes Interesse stößt, und es laufen breits viele Forschungsprojekte. Um den Erfolg vom WATM zu sichern, muß die Technologie standardisiert werden. Diese Aufgabe wird vom ATM-Forum erledigt.

2 WATM Entwurfskriterien

WATM soll mit einer Funkzellenstruktur realisert werden. Jede Zelle wird von einer Basisstation (BS) bedient. Alle BS sind über ein ATM-Netzwerk verbunden. Diese

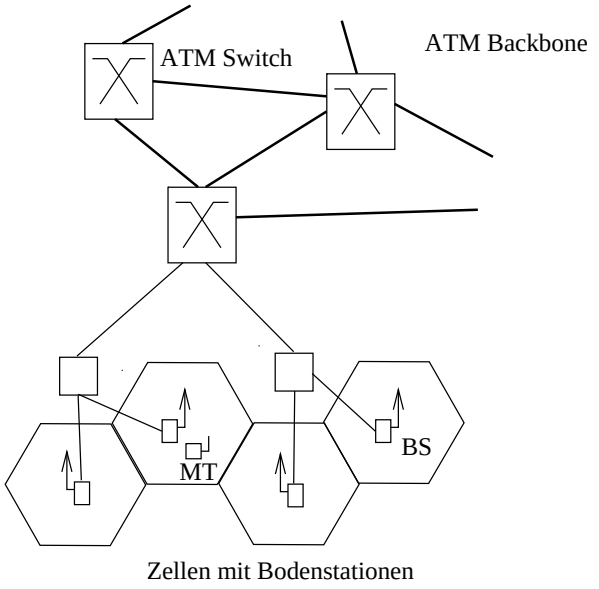

Abbildung 1: WATM-Funkzellenstruktur

Struktur ist in Abbilung 1 dargestellt. Kleine Zellen haben den Vorteil, daß sie Schwierigkeiten wie Verzögerungen, die durch Mehrwegausbreitung entstehen, und große Signalabschwächungen, die auftreten, wenn keine direkte Sichtverbindung besteht, vermindern. Durch Benutzung derselben Frequenz ist kein Handover auf der Bitübertragungschicht notwendig. Die kleine Zellengröße erlaubt es auch, die gleiche Frequenz wiederzuverwenden, so daß die Bandbreite nicht zu knapp wird. Kleine Zellen haben aber auch Nachteile gegenüber großen Zellen. Es ist im Durchschnitt nur eine kleine Anzahl mobiler Teilnehmer (MT) im Service-Bereich einer BS, wodurch die Installationskosten steigen, da mehr BS benötigt werden. Außerdem nimmt die Handover-Rate zu.

Basierend auf dieser Zellenstruktur gibt es zwei grundsätzlich verschiedene Ansätze, WATM zu realiseren. Der erste Ansatz, der schnell und billig realisiert werden könnte, basiert auf der Übertragung der ATM-Zellen mit Hilfe eines bereits bestehenden Funkübertragungssystem. Die ATM-Zellen werden dann als Nutzdaten eines anderen Verbindungsprotokolls übertragen. Sie werden durch eine Konvertierungseinheit, die einen ATM-Knoten emuliert, ver- und entpackt. Diese Vorgehensweise erlaubt es, die bestehenden Protokolle und Systeme für eine Realisierung eines WATM-Systems zu benutzen.

Das Problem ist aber, daß keines der bestehenden drahtlosen Protokolle entworfen wurde, um einen Datenverkehr von kleinen Dateneinheiten wie den ATM-Zellen zu unterstützen. Die Effizienz der Ver- und Entpackung der ATM-Zellen wäre viel zu klein, um praktisch akzeptabel zu sein. Um die MT optimal bedienen zu können, muß die BS die augenblicklichen Datenmengen, die die MT in ihrem Zuständigkeitsbereich übertragen möchten, kennen. Drahtlose Protokolle wie IEEE 802.11 und ETSI HIPERLAN sehen keine Möglichkeit vor, solche Informationen zu übertragen.

Auch wenn geeignete Protokolle existieren würden, wäre die Konvertierung, die bei jeder Übertragung stattfinden müßte, ineffizient. Außerdem wäre mit jeder Änderung der ATM-Protokolle eine Anpassung der Funktionsweise der Konvertierungseinheit notwendig. Ein einfaches Anpassen der Konvertierungseinheit könnte sich als unmöglich

herausstellen, da ihre Funktionen in Hardware oder in Firmware implementiert sein müßten, um den notwendigen Geschwindigkeitsanforderungen zu entsprechen. Eine leicht erweitbare Softwarelösung wäre wahrscheinlich zu langsam. Da ein WATM-System eine transparente, nahtlose und effiziente Erweiterung eines ATM-Netzwerks ermöglichen soll, eignet sich dieser Ansatz nicht.

Stattdessen wird ein Ansatz verwendet, der im Hinblick auf die Bedürfnisse der ATM-Operationen entworfen wurde. Die zugrundeliegende Informationseinheit ist die ATM-Zelle und der drahtlose Zugangspunkt stellt eine Erweiterung eines ATM-Switchs dar, der die MT in einer optimalen Reihenfolge bedient. Das drahtlose Subsystem operiert auf den ATM-Zellen und erlaubt es, sowohl die bestehende als auch neue ATM-Software zu benutzten. Konzepte wie der ATM-Protokollstapel, Kontrollmechanismen wie E.164 oder IP-over-ATM-Adressierung, VC-Multiplexing, Cell Prioritization (CLP), Explicit Congestion Notification (ECN), Q.2931-Signalisierung usw. bei dem WATM-Entwurfskonzept werden beibehalten. Der ATM-Protokollstapel wird direkt übernommen und um neue funkspezifische Schichten ergänzt. Das ATM-Konzept muß um eine Mobilitätunterstützung erweitert werden, da Funktionen wie Location Management, Hand-over usw. im Festnetz nicht notwendig sind.

2.1 WATM Protokoll Architektur

Eine allgemeine WATM-Protokoll-Architekur ist in Abbilung 2 gezeigt. In der Kon-

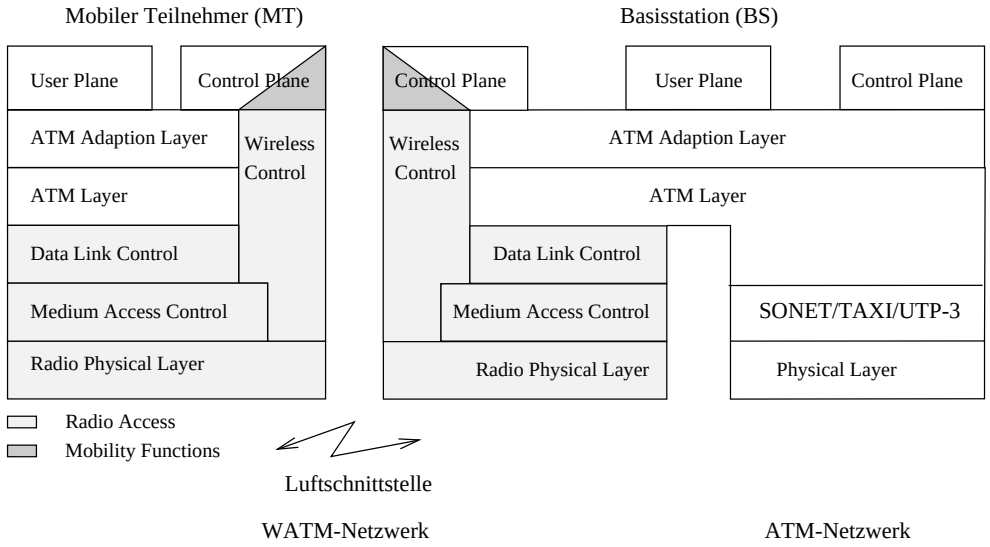

Abbildung 2: WATM Protokoll-Architektur für MT und BT

trollsäule ist die Mobilitätsunterstützung untergebracht.

2.1.1 Bitübertragungschicht (PHY, Physical Layer)

Während ein ATM-Switch im Festnetz eine ATM-Verbindung mit 25-155 Mbit/s anbietet, ist eine 25 Mbit/s Funkverbindung bis heute schwierig zu implementieren. Für eine Hochgeschwindigkeitsfunkverbindung sind Frequenzen im Bereich von einigen GHz notwendig. Obwohl 155 MBit/s aufgrund der Beschränkungen der heutigen Technik

nicht realisiert werden können, nimmt man allgemein an, bald eine Übertragungsgeschwindigkeit von 622 MBit/s im 60 GHz-Band zu erreichen. Es wurden bis jetzt zwei verschiedene Spezifikationen für PHY vom ATM-Forum vorgeschlagen.

	Low Speed	High Speed
Frequenzband	5.15 - 5.35 GHz, 5.725 - 5.875 GHz	59 GHz - 64 GHz
Zellenradius	80 m	10 - 15 m
Leistung	100 mW	10-20 mW
	bis 12	7
Kanalbandbreite	30 MHz	150/700 Mhz
Datenrate	25 Mbit/s	155/622 Mbit/s
Modulation	16 tone DQPSK	32 tone DQPSK
MAC Schnittstelle	parallel, 3.127 Mbyte/s	parallel, 87.5 Mbyte/s
Packetlänge	PHY-Kopf & MAC-Kopf & 4 ATM Zellen	

2.1.2 Mediumzugangschicht (MAC, Medium Access Control)

Die Mediumzugangsschicht (MAC) muß die unterschiedlichen Dienstkategorien wie CBR (constant bit rate), VBR (variable bit rate) und ABR (available bit rate) unterstützen, wobei die QoS-Parameter eingehalten werden müssen. Eine allgemeine MAC-Schicht mit dem dynamischen TDM/TDMA-Rahmenformat (siehe Abbilung 3) wurde in [Rayc96] vorgeschlagen. Die Downlink-Übertragungen werden in einem einzel-

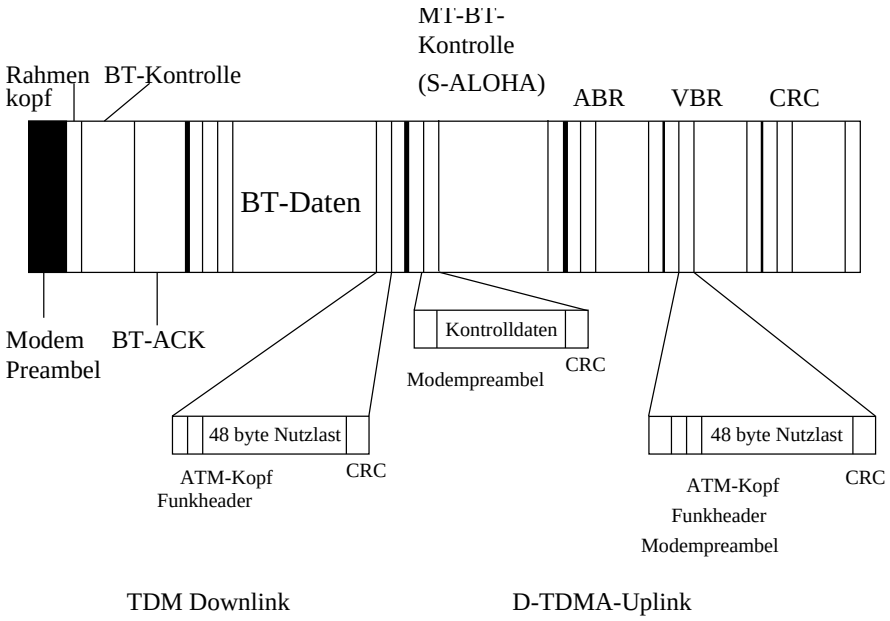

Abbildung 3: Dynamischer TDMA-Rahmen für WATM

nen Burst von der BT zu den MT zeitlich gemultiplext (TDM). Die Uplink-Übetragungen benutzen dynamisches Zeitmultiplexverfahren (TDMA). Die Grenzen zwischen den Subrahmen können sich verschieben, um sich besser den jeweils vorliegenden Verkehrsanforderungen anzupassen. Eine Gruppe mehrerer Rahmen bildet einen Superrahmen.

Die Downlink-Preambel ermöglicht die Rahmensynchronisation und dient als a traning sequence for the purposes of equalization. Die BS benutzt den Rahmenkopf,

um ihre Identifikation (ID, identification) und Informationen über die Positionen und Längen der Subrahmen den MT mitzuteilen. Die Downlink-Kontrollnachrichten, die im BS-Kontrollbereich übertragen werden, enthalten Information über Reservierung, BS-MT-Bestätigungen (ACK) und andere Kontrollinformationen des Funkprotokolls. Der BS-Datenbereich enthält Downlink-Datenzellen. Der MT-BS-Kontrollbereich wird von den MTs benutzt, um Kontrollnachrichten in einem Slotted-ALOHA-Modus an die BS zu übertragen. Die Uplink-Daten und MT-BS-Bestätigungen werden im MT-BS-Datenbereich übertragen. Die Übertragungen in diesem Bereich werden entweder auf einer Rahmen-zu-Rahmen-Basis (bei ABR, CBR/VBR-Wiederholungen und MT-BS-Bestätigungen) reserviert oder fest vorbelegt (bei CBR).

CBR-Daten werden in jeweils gleichen Slot-Sequenzen in einem oder mehreren Rahmen eines Superrahmens übertragen. Die Positionen der CBR-Slots sind relativ statisch in jedem Rahmen. Den VBR-Daten werden mit Hilfe eines auf UPC (Usage Parameter Control) basierenden statistischen Multiplexalgorithmus Slots zugewiesen. Die ABR-Slots werden dynamisch reserviert, wobei spezielle ABR-Slots oder unbenutzte CBR/VBR-Slots in jedem Rahmen ausgenutzt werden.

Die MAC-Schicht besteht aus den beiden Komponenten S-MAC (Supervisory MAC) und C-MAC (Core MAC).

- Die **Supervisory MAC**-Komponente kann als die Intelligenz der MAC-Schicht angesehen werden. Sie verarbeit die Kontrollinformationen und stellt einen Ablaufplan (schedule table) für die C-MAC-Komponente für jeden Rahmen auf. In der BS ist die S-MAC-Komponente für die Zuteilung des Kanals (downlink und uplink) und für alle virtuellen Verbindungen (VC) (ABR/CBR/VBR) verantwortlich. Sie empfängt Anforderungen der VCs im System zur Slotreservierung. Damit die QoS-Parameter eingehalten werden, muß sie einen Ablauf für die Datenübertragung festlegen. Die Call Admission Control (CAC) für die Funkverbindung gehört auch zu den Aufgaben der S-MAC-Komponente in der BS. Beim MT verarbeitet die S-MAC-Komponente die Kontrollinformationen, die sie während der BS-MT-Phase empfangen hat, und stellt einen Ablaufplan für die C-MAC-Komponente auf. Sie ist auch für die Steuerung der Übertragung der MT-BS-Kontrollinformation während der Slotted-ALOHA-Phase verantwortlich.

- Die **Core MAC**-Komponente kann als die Schnittstelle zwischen der DLC- und der Bitübertragungschicht aufgefaßt werden. Entsprechend des Ablaufplans, der durch die S-MAC-Komponente erstellt wurde, multiplext/demultiplext die C-MAC-Komponente Senden und Empfangen für die korrespondierenden VCs.

2.1.3 Sicherungschicht (DLC, Data Link Control)

ATM wurde unter der Annahme entworfen, daß die Bitfehlerrate (BER, bit errror rate) klein ist und daß die Fehler unabhängige Einzelfehler sind. Diese Annahme stimmt für Übertragungssysteme, die über Glasfaser vernetzt sind. BER ist bei der Funkübertragung viel größer und es treten sehr häufig Bündelfehler, wenn ein Kanalschwund (Fading) auftritt. Deswegen wird ATM viele seiner Vorteile einbüßen, wenn die ATM-Zellen und -Mechanismen der Festnetze für die Funkübertragung ohne Modifikationen übernommen werden.

Einer der Gründe für die Schnelligkeit der ATM-Zellenvermittlung besteht darin, daß keine (bei Nutzdaten) bzw. eine sehr einfache (bei HEC) Fehlerkontrolle vorgenommen wird. Im HEC können 1-Bitfehler korrigiert werden. ATM nimmt an, daß die Fehlerrate des Übertragungsmediums sehr klein ist. Die Fehlerkorrektur wird in höheren Schichten des Protokollstapels durchgeführt. Oft wird Transmission Control Protocol (TCP) verwendet, das annimmt, daß alle Fehler durch Überlastung (congestion) verursacht werden. Bei Glasfaserverbindungen ist dies der Fall, da die Anzahl der Fehler, die durch Übertragung selbst (aufgrund der Eigenschaften des Mediums) verursacht werden, kleiner ist als die Anzahl der Fehler, die durch eine Netzüberlastung verursacht werden. Bei der drahtlosen Übertragung treten viele Fehler aufgrund des unzuverlässigen Luftmediums auf. TCP nimmt dann an, daß die Fehler durch eine Überlastsituation hervorgerufen wurden, und wird deswegen eine Sequenz von Maßnahmen zur Behebung der Überlastung einleiten, die gar nicht eingetreten ist. Beim drahtlosen ATM wird dadurch die Effizienz erheblich beeinträchtigt.

Die Lösung besteht darin, Fehlererkennung, -korrektur und automatisches transparentes Wiederholen verlorengegangener Informationen zu implementieren. Die Fehler bleiben für die höheren Schichten unsichtbar. Dadurch werden die negativen Effekte auf TCP reduziert. Die Frage, wieviel Overhead durch diese zusätzliche Fehlerkorrektur verursacht wird, ist noch nicht vollständig geklärt. In [CaMc97] wird eine Kombination verschiedener Maßnahmen vorgeschlagen. Sie beinhalten:

- Durch Interleaving (Verschachtelung) erscheinen Bündelfehler wie quasi statistisch unabhängige Einzelfehler,
- Verringerung der Fehlerrate durch Fehlerkorrektur (FEC, Forward Error Correction) und
- Wiederholen verlorengegangender Informationen (ARQ, Automatic Repeat Request).

Die Aufgabe der DLC-Schicht ist es, die Fehler vor den höheren Schichten zu verbergen. Deswegen sind Fehlererkennung, -korrekur (FEC) und automatisches Wiederholen (ARQ) notwendig. Dabei sind die Anforderungen der unterschiedlichen Dienstkategorien (ABR, VBR und CBR) zu beachten. Bei ABR gibt es kein Zeitlimit für die Fehlerbehebung. Die Verzögerung bei der Fehlerbehebung von VBR- und CBR-Daten darf nicht lang sein, weil sonst die synchrone bzw. isochrone Übertragung der Daten nicht mehr gewährleistet ist.

Eine Protokoll-Simulation für ABR und CBR wurde in [Rayc96] durchgeführt.

2.2 Mobilitätsunterstützung

Um Mobilität zu unterstützen, reicht die ATM-Signalisierung (Q.2931) nicht aus, weil darin keine Funktionen wie Handover, Location Management, Re-routing, Registrierung usw. vorgesehen sind.

2.2.1 Konzepte zur Mobilitätsunterstützung

Um Verbindungen mit einem MT jederzeit aufbauen zu können, muß sein Aufenthaltsort bekannt sein. Im Festnetz sind die Teilnehmer fest und können durch ihre Identifikationen und die Routinginformationen der ATM-Switches lokalisiert werden. Sobald die Teilnehmer mobil werden, sind die Routen zu den MT nicht mehr statisch. Es werden drei verschiedene Konzepte zur Mobilitätunterstützung in WATM-Systemen vorgestellt.

Die hierarchische Architekur eines *virtuellen Verbindungsbaumes (VCT, virtual connection tree)* in [AcNa97] ist in Abbildung 4 dargestellt. Der Wurzelknoten (RN, root node) des VCT ist ein ATM-Switch, der an das ATM-Backbone-Netzwerk angeschlossen ist. Die Knoten unterhalb des RN besitzen eine baumartige Verbindungsstruktur. Die Blätter sind BSen, die möglicherweise nach einem Handover die Verbindung mit dem MT übernehmen werden müssen.

Wenn eine Verbindung zu einem MT angefordert wird, wird ein VCT aufgebaut. Dabei wird zwischen jedem der Blätter und dem RN eine Verbindung vorbereitet. Es ist immer nur eine BS für die Übertragung der Daten zuständig. Nach einem Handover wird eine neue BS aktiv und die Verbindung wird transparent umgeschaltet. Die *„mobile virtual circuit"-Architektur* (MVC) in [YuLe96] funktioniert ähnlich wie

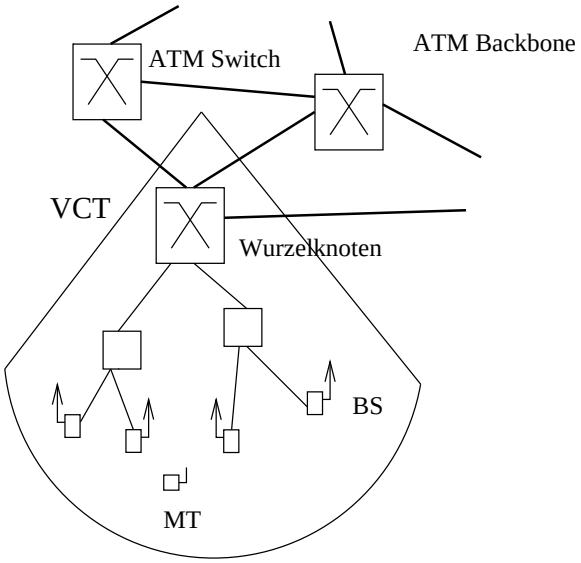

Abbildung 4: VCT-Architektur

die VCT-Architektur, wobei der Baum aber dynamisch konfiguriert wird. Der Kettenpunkt (TP, tether point) entspricht dem RN beim VCT. Es werden alle unmittelbar nach einem Handover potentiellen BSen durch das Netzwerk erkannt und unter dem TP vorbereitet. Die Bandbreite wird nicht sofort für die potentiellen Handover-BSen reserviert und die Anzahl der Knoten unter dem TP ist bei MVC kleiner als unter dem RN bei VCT.

Die *Zwei-Datenbanken-Architektur (two-tier database architecture)* in [AkCo96] (siehe Abbilung 5), partitioniert das WATM-Netzwerk in Zonen. Jede Zone enthält eine bestimmte Anzahl an BSen, die jeweils eine Funkzelle abdecken. In jeder Zone gibt es

eine Datenbank, in der Informationen über die permanenten Benutzer und die augenblicklichen Besucher gespeichert sind. Die Zwei-Datenbanken-Architekutr ist lokalisierungsabhängig. Die Teilnehmer-Identifikation (UID, user identification) spezifiziert eindeutig die Heimzone des Teilnehmers, wo er als permanenter Benutzer eingetragen ist. Für jeden permanenten Benutzer wird in der Datenbank seiner Heimzone die Information, in welcher Zone er sich gerade befindet, verwaltet. Jede Zonen-Datenbank enthält Informationen, welche BT für die besuchenden MT zuständig ist, eingetragen.

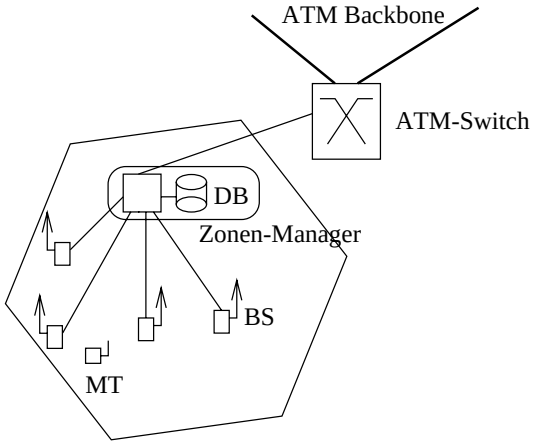

Abbildung 5: Zwei-Datenbanken-Architektur

2.2.2 Location Management

Location Management wird durch die gewählte Architekur bestimmt.

In der Zwei-Datenbanken-Architektur wird ein MT in zwei Schritten lokalisiert. Zuerst wird die für den MT im Augenblick zuständige Zone (serving zone) ermittelt. Sie ist in Datenbank der Heimzone (home zone) des MT gespeichert. Dann wird in der Datenbank der zuständigen Zone nachgesehen, welche BS für den MT zuständig ist.

Bei den hierarchischen Architekturen wird die Information durch das Netzwerk geroutet, bis sie einen RN bzw. TP erreicht. Bei VCT wird die ATM-Zelle vom RN zu der zuständigen BS mit Hilfe eines Mechanismus (VCN-monitor/translator) geleitet. Bei MVC leitet der TP die ATM-Zelle zu der zuständigen BS mit Hilfe eines virtuellen Gruppenkanals (GVC, group virtual channel). Der GVC bleibt während der Verbindung konstant, während sich die Pfade je nach der BS ändern.

2.2.3 Handover

In einem drahtlosen Netzwerk ist die Handover-Funktion sehr wichtig. Sie ermöglicht es, einem Teilnehmer sich auch außerhalb einer Funkzelle einer bestimmten BS zu bewegen. Die Übergabe der Verbindung zwischen den BSen muß ohne Unterbrechung der Verbindung und möglichst ohne Qualitätverlust stattfinden.

Bei VCT werden die Handovers durch die MT ausgelöst. Ein Handover wird eingeleitet, indem der MT ATM-Zellen der neuen BS übermittelt. Die neue BS wird durch eine

Verbindungsnummer identifiziert, die beim Aufbau der Verbindung vom RN zugewiesen wurde. Die ATM-Zellen werden werden von der neuen BS über den RN zum Ziel übertragen. Der RN erkennt die Verbindungsnummer des MT von der neuen BS und ändert entsprechende Tabellen, so daß an den MT gerichtet Zellen an die neue BS übermittelt werden.

MVC-Rerouting während eines Handovers wird durch die alte BS eingeleitet, indem sie dem TP signalisiert, daß der GVC den jetztigen virtulellen Pfad (VP, virtual path) ändern wird. Für die Dauer einer Verbindung bleibt der GVC gleich, während sich die VPs mit der zuständigen BS ändern. Die neue BS teilt den neuen VP mit. Bei MVC wird eine dynamische Rekonfiguration des Baums (siehe Abbilung 6) durchgeführt. Ein dynamischer Baum enthält wenige BSen und unterstützt Handovers in einer beliebigen Umgebung während einer Verbindung. Beim VCT ist der Baum dagegen statisch und erlaubt ein Handover nur zwischen den BSen des Baums. Ein Handover an eine BS außerhalb des Baums ist nicht möglich.

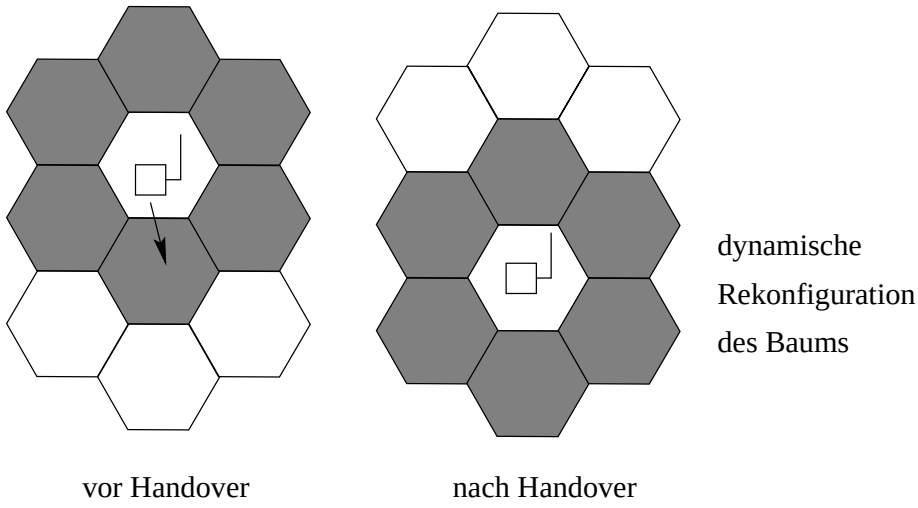

Abbildung 6: MVC-Handover

3 WATM-Projekte

- Magic WAND (Wireless ATM Network Demonstrator)

  http://www.tik.ee.ethz.ch/ wand/

  Magic WAND ist ein europäisches Projekt, das eine Demonstration mobiler Terminals für Multimediainformationszugang mit einem schnellen drahtlosen ATM-Netz erreichen will. Es wird das 5-GHZ-Band benutzt, um die Kommunikation zwischen den mobilen Geräten und den ATM-Switches zu realisieren.

- MEDIAN (Wireless Broadband CPN/LAN for Professional and Residential Multimedia Applications)

  http://www.imst.de/mobile/median/median.html

  Das Hauptziel von MEDIAN ist es, ein Pilotsystem für drahtloses High-Speed PLAN (premises local area network) für Multimedia-Anwendungen zu implemen-

tieren. Das System soll eine drahtlose ATM-Netzwerkerweiterung unterstützen. Es wird das 60-GHz-Band benutzt, um Bitraten bis 155 Mbit/s zu erreichen.

- AWACS (ATM Wireless Access Communication System)

  http://www.uk.infowin.org/ACTS/RUS/PROJECTS/ac228.htm

  Das Hauptziel von AWACS ist es, ein Systemkonzept und einen Demonstrationsprototypen für den drahtlosen öffentlichen Zugang zu den B-ISDN-Diensten zu entwickeln. Es wird das 19-GHz-Band benutzt, um Bitraten bis 34 Mbit/s zu erreichen.

- ORL Radio ATM

  http://www.cam-orl.co.uk/radio/

  ORL (The Olivetti & Oracle Research Laboratory) entwickelt seit 5 Jahren ein drahtloses ATM-System. Das System soll als eine Erweiterung eines bestehenden ATM-LANs dienen. Es basiert auf Pico-Zellen. Jede Zelle besitzt eine Basisstation. Die mobilen Teilnehmer kommunizieren miteinander über die Basisstationen, die als ein ATM-Netzwerk konstruiert sind.

- RDRN (Rapidly Deployable Radio Networks)

  http://www.tisl.ukans.edu/RDRN/

  Das Ziel von RDRN ist es, drahtlose auf ATM basierende Architekturen und Protokolle für Hochgeschwindigkeitsnetzwerke zu entwerfen, die auf der Verbindungs- und Netzwerkebene adaptiv sind, um einen schnellen Aufbau und eine automatische Rekonfiguration in einer veränderlichen Umgebung zu ermöglichen.

- ETSI BRAN (Broadband Radio Access Networks)

  http://www.etsi.fr/bran/

  ETSI (European Telecommunications Standards Institute) entwickelt Spezifikationen für einen drahtlosen Zugang zu Breitbandnetzwerken mit einer Bitrate von 25 Mbit/s oder mehr. BRAN ist nicht exakt ein WATM-System, es wird aber eine Schnittstelle für auf ATM und TCP/IP-basierende Festnetze anbieten.

Literatur

[AcNa97] A. Acampora und M. Naghshineh. An Architecture and Methology for Mobile-Executed Handoff in Cellular ATM Networks. *IEEE Journal on Selected Areas in Communications* 12(8), 1997, S. 1365–75.

[AkCo96] B.A. Akyol und D.C. Cox. Handling Mobility in a Wireless ATM Network. *Wireless Networks* 2(3), August 1996, S. 163–171.

[CaMc97] B.J. Cain und D.N. McGregor. A Recommended Error Control Architecture for ATM Networks with Wireless Links. *IEEE Journal on Selected Areas in Communication* 15(1), Januar 1997, S. 16–27.

[Rayc96] D. Raychaudhuri. Wireless ATM: An enabling technology for multimedia personal communication. *Wireless Networks* 2(3), August 1996, S. 163–171.

[YuLe96] O.T.W Yu und V.C.M Leung. Connection architecture and protocols to support efficient handoff over an ATM/B-ISDN personal communications network. *Mobile Networks and Applications* Band 1, 1996, S. 457–473.

Sicherheitsaspekte in GSM

Thomas M. Schoch

Kurzfassung

GSM ist der europäische Standard für Mobilfunknetze, deren wichtigster Dienst die Sprachkommunikation ist. Diese Arbeit untersucht Sicherheitsaspekte von GSM, worunter hier der Schutz vor Angriffen verstanden wird. Die Übertragung auf der Luftschnittstelle als sicherheitstechnisch schwächstem Medium wird mit kryptographischen Methoden gesichert. Der Zugang zum Netz an der Luftschnittstelle wird durch ein Authentikationsverfahren nur registrierten Teilnehmern gestattet. Pseudonyme als Teilnehmeradressen sollen verhindern, daß einzelne Teilnehmer beim Abhören der Funkkanäle identifiziert werden können. Außerdem erschweren sie die Erstellung von Bewegungsprofilen. Diese Sicherheitsmaßnahmen sind ein Anfang. Die hohen Anforderungen des technischen Datenschutzes werden jedoch nicht erfüllt. So besteht beispielsweise eine Sicherheitslücke darin, daß die Authentikation einseitig erfolgt, d. h. das Netz authentisiert sich nicht gegenüber dem Teilnehmer, der ihm seine Identität und vertraulich zu behandelnde Daten übergibt. Desweiteren findet nur eine Verschlüsselung auf der Luftschnittstelle statt, es gibt keine Ende-zu-Ende-Verschlüsselung.

1 Einleitung

Unsere Gesellschaft hat sich nachhaltig gewandelt. In der Vergangenheit war Finanzkraft der bestimmende Faktor für Erfolg, heutzutage ist es Information und Kommunikation. Daher wird Datensicherheit ein zunehmend wichtigerer Faktor zum Erfolg jedes Unternehmens. Der drahtlose Zugang zu Kommunikationsnetzen ruft neue Probleme im Bereich der Sicherheit hervor, weil Abhören und aktive Beeinflussung hier leichter fallen als in Festnetzen. [Crui96]

GSM ist der europäische Standard für die mobile Kommunikation, hauptsächlich genutzt für das mobile Telefonieren, der sich in den letzten Jahren großer Popularität erfreut. Die Forderung nach ständiger Erreichbarkeit mobiler Teilnehmer hat zur Folge, daß die Kommunikation über verschiedenartige Medien, insbesondere eine Funkverbindung geführt werden muß. Die Funkverbindung, auch Luftschnittstelle genannt, kann nicht physikalisch gesichert werden. Dieser Unterschied zur Kommunikation über ein Festnetz impliziert besondere Anforderungen an die Sicherheit eines Mobilnetzes.

1.1 Grundrechte und das Freiheitsprinzip

In modernen Rechtsstaaten sind gewisse Grundrechte als Abwehrrechte gegenüber staatlicher Gewalt garantiert. Sie einzuschränken bedarf legitimer Ziele und des Nachweises, daß die eingesetzten Mittel tatsächlich geeignet sind, diese Ziele zu erreichen (Verhältnismäßigkeit). Ein Beispiel ist das informationelle Selbstbestimmungsrecht, das „Befugnis des Einzelnen, grundsätzlich selbst über die Preisgabe und Verwendung seiner persönlichen Daten zu bestimmen“ (Bundesverfassungsgericht). Ein weiteres Beispiel ist das Fernmeldegeheimnis, das in Artikel 10 GG ausdrücklich geschützt ist. [GG10]

Zentrale Grundannahme beider Rechte ist die Überzeugung, daß eine unbeobachtete und vertrauliche Kommunikation eine wesentliche Voraussetzung für die persönliche Selbstbestimmung ist. Diese ist zugleich konstitutiv für die Handlungs- und Mitwirkungsfähigkeit der Bürger und damit für das freiheitlich-demokratische Gemeinwesen.

Allerdings hat vertrauliche Kommunikation immer schon das Interesse staatlicher Sicherheitsbehörden geweckt, weil die Entfaltungsmöglichkeiten der Bürger in einem Spannungsverhältnis zu den Sicherheitsinteressen Dritter und des Staates gesehen werden.

Mit der Entwicklung moderner Kommunikationstechnik und der allgemeinen Verfügbarkeit starker Verschlüsselungsverfahren zum Schutz elektronischen Datenaustausches hat diese Auseinandersetzung an Aktualität und Brisanz gewonnen. [BiFo97a]

Kryptographie stellt Methoden zur Verfügung, die jetzt erstmals die Möglichkeit bieten, daß der Bürger durch eigene technische Maßnahmen in eigener Verantwortung sein Kommunikationsgeheimnis sicherstellen kann. Er muß sich nicht mehr auf den Staat verlassen. [Gerl97]

Auf der einen Seite steht das verfassungsrechtlich verbürgte Recht der Bürger auf unbeobachtete und vertrauliche Kommunikation. Auf der anderen der öffentliche Auftrag der Sicherheitsbehörden, einer Gefährdung von Gesellschaft und Staat entgegenzuwirken. Die sogenannte Kryptokontroverse ist eine Debatte über den Ausweg aus dem Dilemma, daß die Bürger ein Interesse an absoluter Unentschlüsselbarkeit haben, die technisch auch möglich, jedoch nicht zugunsten der Interessen von Sicherheitsbehörden relativierbar ist, ohne daß die ermöglichte Öffnung der geheimen Inhalte auch wieder von Unbefugten genutzt werden könnte. [Hamm97]

Seit einiger Zeit wird von der Bundesregierung geprüft, ob das Erfordernis einer rechtlichen Regelung des Einsatzes von Verschlüsselungsverfahren besteht. Die Sicherheitsbehörden haben beispielsweise ein Interesse an der Verpflichtung zur Benutzung genehmigter Verschlüsselungsverfahren und Hinterlegung der geheimen Schlüssel bei einer vertrauenswürdigen Instanz. Es gibt mehrere Ansätze für solche *key-recovery*-Systeme, von denen jedoch jeder Probleme beinhaltet. Nicht zuletzt ist eine Instanz, die geheime Schlüssel in großer Zahl speichert, ein besonders interessantes Objekt für Angriffe. [Fox97b]

Der hinterlegte Schlüssel kann außerdem zum Schlüsselaustausch für ein zusätzliches, verbotenes Verschlüsselungsverfahren benutzt werden, dessen Anwendung und damit die Übertretung des Kryptoverbots aber erst „zu spät“ bemerkt wird. Eine Durchsetzung einen Kryptoverbots setzt also die weitgehende Aushöhlung des Fernmeldege-

heimnisses voraus. Doch selbst dann können sich Kommunikationspartner steganographischer Verfahren bedienen, bei denen die Verwendung eines Verschlüsselungssystems praktisch nicht nachweisbar ist. [HuPf96]

Mit Steganographie (*Verdecktes Schreiben*, Kunstwort aus dem Griechischen) wird eine Nachricht versteckt, so daß ihre Existenz oder der Übertragungszeitpunkt dem Uneingeweihten unbekannt bleibt; mit Kryptographie bleibt dies erkennbar. Man kann beides kombinieren. Steganographische Methoden werden dann interessant, wenn z. B. Verschlüsselung verboten ist und man das Verbot unentdeckt übertreten will. [Riha94a]

Der Aufwand für die Infrastruktur zur Erreichung von Minimalergebnissen einer Kryptoreglementierung stünde in keinem Verhältnis zu dem geringen Ertrag angesichts der technischen Möglichkeiten, jegliche Überwachung durch verschiedene Formen der Verschlüsselungstarnung zu unterlaufen. [Hamm97]

Eine Regulierung kryptographischer Techniken zur Konzelation (*to conceal*, engl.: verbergen, verheimlichen) mit dem Ziel der Verbrechensbekämpfung ist somit nicht sinnvoll, da ein Kryptoverbot nicht wirksam durchsetzbar ist, aber die Sicherheitsinteressen von Bürgern, Wirtschaft und Gesellschaft empfindlich beeinträchtigt. [HuPf96]

1.2 Bewegungsprofile

Ein Teilaspekt der Kommunikation ist der Aufenthaltsort der Kommunikationsteilnehmer. Das Fernmeldegeheimnis schützt diesen neben den Kommunikationsinhalten, da er einen „näheren Umstand der Kommunikation“ ausmacht. [TKG]

Mobilität und Erreichbarkeit eines Mobilteilnehmers zählen zu den Eigenschaften, die die Mobilkommunikation zu dem attraktiven Dienst gemacht haben, der sie heute ist. Um diese Eigenschaften zu gewährleisten, speichern die Vermittlungssysteme der Netzbetreiber den aktuellen Aufenthaltsort jedes Teilnehmers. Diese Datenspuren, die die Mobilstationen hinterlassen, verraten Aufenthaltsort und Aktivität eines Mobilteilnehmers.

Beispiel: Die Computer der schweizerischen Telefongesellschaft Swisscom speichern im Minutentakt sechs Monate lang, in welcher Zelle die Telephone gerade funken. In Deutschland dürfen solche Bewegungsdaten nicht aufgezeichnet werden. Das verbietet die Fernmeldeüberwachungsverordnung. Verbindungsdaten - Informationen, wer wann mit wem telephoniert hat - werden in Deutschland 80 Tage lang gespeichert und dürfen nur preisgegeben werden, wenn ein richterlicher Beschluß vorliegt. [Rubn97]

Verbindungsdaten von in Fremdnetzen registrierten Teilnehmern werden im D1-Netz 6 Monate lang gespeichert. [Luka97]

Inhaltsdaten können von den Teilnehmern durch eigene Verfahren verschleiert werden, auf die Verkehrsdaten haben die Kommunikationspartner jedoch keinen Einfluß. Dies stellt eine Verletzung des Rechts auf informationelle Selbstbestimmung und eine Gefährdung der Privatsphäre dar, die aus den in Kapitel 1.1 geschilderten Gründen unerwünscht ist.

In bestehenden Netzen wäre eine Anonymisierung der Teilnehmer gegenüber dem Netzbetreiber durch die Verwendung vorausbezahlter *debit*-Karten möglich. Die Strafverfolgungs- und Sicherheitsbehörden haben jedoch anonyme Lösungen verhindert: Eine

befristete Lizenz für die Betreiber des D1-Netzes, eine *prepaid*-Karte einzusetzen, wurde mit der Begründung, das Verfahren erschwere die Verfolgung von Tätern, nicht verlängert. Dennoch gibt es eine ISDN-Richtlinie der EG, die den Mitgliedsstaaten die Verpflichtung auferlegt, den anonymen Zugang zu Telekommunikationsdiensten ermöglichen. [BiFo97b]

1.3 Gliederung

In Kapitel 2 werden Sicherheitsprobleme erörtert. Dabei wird auf verschiedene Sichtweisen des Begriffs Sicherheit eingegangen. Es erfolgt eine Eingrenzung der Bedeutungsvarianten auf die in dieser Arbeit zu untersuchenden. Schließlich wird ein Katalog von Sicherheitsanforderungen vorgestellt, anhand dessen in Kapitel 4 der Standard GSM untersucht wird.

In Kapitel 3 wird der Standard GSM vorgestellt. Es erfolgt eine kurze Einführung in die Systemarchitektur unter besonderer Berücksichtigung der dort behandelten Sicherheitsaspekte.

Kapitel 4 enthält eine Bewertung von GSM im Hinblick auf die in Kapitel 2 genannten Sicherheitsanforderungen.

2 Sicherheitsaspekte

Der Begriff *Sicherheitsaspekte* umfaßt unterschiedlichste Fragestellungen. Eine erste Unterteilung ergibt die Übertragung ins Englische, wo zwei Vorstellungen des Begriffs *Sicherheit* existieren: *safety* und *security*. *Safety* bedeutet, nicht in Gefahr zu sein (*freedom from danger*). *Security* bedeutet Schutz vor Gefährdung (*something that provides safety*, [Horn74]). Diese Begriffe werden nun auf den Bereich der Mobilkommunikation angewandt.

2.1 Safety

Schaden für Personen und Sachen kann auf verschiedene Weise verursacht werden. Hier betrachtet wird der Einfluß der elektromagnetischen Wellen auf Fremdelektronik, auf den menschlichen Organismus, sowie die Beeinträchtigung der Aufmerksamkeit eines Mobilteilnehmers durch die Mobilkommunikation.

2.1.1 Beeinflussung sensibler, kritischer Technologie

Es wurden einige Vorfälle berichtet, bei denen Mobiltelefone im Verdacht standen, die Elektronik von modernen Flugzeugen wie z. B. Airbus A300 gestört zu haben. Ein Zwischenfall soll dadurch verursacht worden sein, daß die Funkaktivität eines Handys den Autopiloten eines Verkehrsflugzeugs ausgeschaltet hat. [SPIE96b]

Daimler-Benz weist angeblich seine Kunden darauf hin, daß Handys ohne Außenantenne die Fahrzeugelektronik stören—der Airbag kann losgehen. [SPIE96b]

Feuerleitelektronik in militärischem Gerät stellt ebenfalls ein Beispiel für sensible Elektronik dar, die möglicherweise durch die Benutzung von Mobilstationen beeinträchtigt werden kann. Ein entsprechender Vorfall, bei dem zwei deutsche Soldaten der SFOR-Friedenstruppe getötet wurden, hat sich im Mai 1997 bei Sarajevo ereignet. [SPIE97b]

2.1.2 Tätigkeiten in kritischen Bereichen

Die gesundheitliche Unversehrtheit eines Teilnehmers der Mobilkommunikation oder Dritter kann gefährdet sein, wenn die Aufmerksamkeit des Teilnehmers einer kritischen Aufgabe gewidmet ist, und durch die Kommunikation oder die Handhabung seiner Mobilstation beeinträchtigt wird. Als Beispiel sei das Telefonieren im Auto genannt. Untersuchungen zeigen, daß die Aufmerksamkeit eines Fahrers erheblich abnimmt, die Fahrfehler zunehmen, wenn er während der Fahrt telefoniert. Gleichzeitig ist er sich dieses Umstandes nicht bewußt, d. h. subjektiv beeinträchtigt sich die Konzentrationsfähigkeit nicht. [Forg97], [SPIE96a]

Ein Beispiel für ein prominentes Opfer eines mutmaßlich während Handy-Benutzung verursachten Verkehrsunfalls findet sich in [SPIE97c].

2.1.3 Gesundheitliche Auswirkungen

Desweiteren sind Zweifel an der Ungefährlichkeit der verwendeten Technologie im Hinblick auf gesundheitliche Schädigungen noch nicht mit letzter Gewißheit ausgeräumt. Die für die Funkübertragung verwendete elektromagnetische Strahlung steht in der Öffentlichkeit im Verdacht, im menschlichen Gewebe Tumorbildung verursachen zu können. Hierzu ist festzustellen:

Der Mensch war während seiner ganzen Evolutionsgeschichte immer schon elektromagnetischen Wellen aus natürlichen Strahlungsquellen ausgesetzt.

Die künstlichen elektromagnetischen Wellen unterscheiden sich nicht von den natürlichen elektromagnetischen Wellen. Die Wissenschaft unterteilt sie in die sogenannten nichtionisierenden und die ionisierenden elektromagnetischen Wellen.

Die nichtionisierenden elektromagnetischen Wellen umfassen den Bereich der Niederfrequenz über die Radio- und Mikrowellen bis hin zum sichtbaren Licht. Die thermischen Wirkungen von hochfrequenten elektromagnetischen Wellen im nichtionisierenden Bereich sind seit langem bekannt.

Ionisierende elektromagnetischen Wellen liegen im Frequenzspektrum oberhalb des sichtbaren Lichts. Sie sind extrem kurzwellig und sehr energiereich. Von ihnen ist bekannt, daß sie bei entsprechender Intensität und entsprechend langer Exposition Veränderungen des Zellgewebes, aber auch genetische Veränderungen hervorrufen können (Röntgenstrahlen, Gammastrahlen).

Die Experten sagen heute schon, daß Gefahr nicht von den elektromagnetischen Wellen, sondern von unsachlichen Darstellungen droht. [Lobe94]

2.2 Security

Sicherheit in der Datenübertragung beinhaltet zwei Aspekte: *Datensicherheit* und *Ausfallsicherheit* (Verfügbarkeit). Datensicherheit umfaßt die Sicherheitsdienste *Zugangskontrolle, Abhörschutz, Datenintegrität* und *Urheberschaftsnachweis*. Der Mechanismus, der den meisten Sicherheitsdiensten zugrunde liegt, ist die Verschlüsselung. [Stol95]

Ausfallsicherheit umfaßt auch den Schutz vor physikalischen Störungen auf der Funkstrecke. Der GSM-Standard wurde auf die Sprachübertragung hin optimiert. Ein Sprachkodierer entfernt zunächst Redundanz und Irrelevanz, ein Kanalkodierer fügt durch FEC (*Forward Error Correction*) gezielt Redundanz hinzu, um besonders wichtige Informationen gegen Störungen auf dem Übertragungsweg zu schützen. Das Ziel ist, auf der Empfangsseite die subjektiv bestmögliche Reproduktion des Sprachsignals zu erreichen. Es ist nicht vordringlich, eine möglichst bitgenaue Reproduktion des digitalisierten Mikrophonsignals zu erreichen. [Detk96]

Datensicherheit (*computer security*) bedeutet zunächst „keeping anyone from doing things you don't want them to do to, from, on or with your computer or any peripheral devices“ ([ChBe94]). Diese Definition ist zwar sicherlich umfassend, aber zu weit gefaßt, um praktisch angewandt zu werden. Daher wird hier ein etwas engerer Begriff gewählt, der möglicherweise einige Aspekte außer Acht läßt, dafür aber an realen Systemen überprüfbar ist.

Ab hier wird der Schutz vor vorsätzlichen Angriffen menschlicher Angreifer behandelt. Einzelne Arten von Angriffen werden unterschieden. Eine mögliche Untergliederung ergibt sich durch die Betrachtung des Geschädigten. Hier steht auf der einen Seite der Teilnehmer an der Mobilkommunikation, auf der anderen Seite die Betreiber der Mobilnetze.

2.2.1 Angriffe gegen individuelle Mobilteilnehmer

Ein individueller Mobilteilnehmer kann in mehreren Punkten angegriffen werden:

Ein Abhören der Kommunikation erfüllt den eher abstrakten (im Sinne von „finanziell nicht bezifferbaren“) Tatbestand der Verletzung der Privatsphäre, die durch das Fernmeldegeheimnis geschützt ist. Daneben kann auch erheblicher wirtschaftlicher Schaden für den Kommunikationsteilnehmer entstehen, z. B. durch Industriespionage, Geschäfte mit Insiderinformation, bei der Vergabe öffentlicher Aufträge.

Ein Offenbaren der Identität kann für den Mobilteilnehmer Nachteile mit sich bringen, z. B. für anonyme Hinweisgeber und beim Schutz von Informanten, für Nutzer von Aids-Beratungsstellen und der Telefonseelsorge. Das Wissen um die Preisgabe der Identität wird zur Folge haben, daß diese Dienste nicht mehr genutzt werden.

Die Ermittlung des Aufenthaltsortes oder die Erstellung von Bewegungsprofilen stellt ebenfalls eine Beeinträchtigung des Kommunikationsteilnehmers dar. Wichtige Personen des öffentlichen Lebens setzen sich der Gefährdung durch terroristische Anschläge aus. Prominente werden Opfer der Belästigung durch aggressive Fotojournalisten, wie ein Vorfall im Jahr 1997 eindrücklich gezeigt hat.

Diebstahl oder Raub der Mobilstation kann Ziel eines Angriffs sein. Neben eventueller körperlicher Beeinträchtigung des Mobilteilnehmers steht hier der finanzielle Verlust durch Neubeschaffung des Geräts im Vordergrund. Außerdem können die vom Dieb beanspruchten Dienste zunächst dem Mobilteilnehmer in Rechnung gestellt werden, falls seine Identität mit der Mobilstation verknüpft ist.

Diebstahl von Gebühreneinheiten kann auch ohne Diebstahl einer Mobilstation erfolgen, falls es möglich ist, dem Netz die Identität eines anderen Mobilteilnehmers vorzuspiegeln, auf dessen Kosten dann Kommunikationsdienste genutzt werden.

2.2.2 Angriffe gegen den Netzbetreiber

Der Netzbetreiber kann geschädigt werden, falls ein Mobilteilnehmer den Netzzugang mißbräuchlich verwendet, z. B. durch ständiges Ein- und Ausbuchen. Hierdurch werden Ressourcen verschwendet, die der Betreiber zugunsten anderer Teilnehmer einsetzen könnte.

Der Betreiber muß einen Weg finden, die Teilnehmer am Zugang zum Netz zu hindern, die dazu keine Berechtigung haben. Dazu gehören Teilnehmer eines fremden Betreibers oder Teilnehmer mit abgelaufener Zugangsberechtigung.

Eine Möglichkeit eines Angriffs auf einen Betreiber besteht desweiteren darin, daß Teilnehmer Dienste in Anspruch nehmen und die anfallenden Gebühren unterschlagen. Dies kann eintreten, falls die Bonität von Teilnehmern fremder Netzbetreiber, die mit dem hiesigen Betreiber *roaming*-Abkommen geschlossen haben, fälschlich festgestellt wird.

Ein weiteres Szenario ist die Vorspiegelung der Identität eines berechtigten Nutzers zu einer Zeit, während dieser nachweislich keine Dienste genutzt hat und der Betreiber somit von ihm keine Gebühren einfordern kann.

Ein *denial-of-service-attack* kann darin bestehen, die Dienste eines Betreibers für die legalen Teilnehmer nicht nutzbar zu machen. Denkbar wäre die permanente Belegung sämtlicher Funkkanäle in einer Zelle eines zellularen Netzes durch entsprechende funktechnische Einrichtungen. Dies kann natürlich ebenso ein Angriff auf einen individuellen Kommunikationsteilnehmer sein.

In [Wong96] wird von einem Schaden für die Betreiber von Mobilfunknetzen in den USA in Höhe von 1 Million Dollar pro Tag gesprochen. Allerdings handelt es sich hierbei um analoge Netze mit unzureichender Teilnehmerauthentifikation.

2.3 Sicherheitsanforderungen

Der technischen Datenschutz erhebt eine Reihe von Forderungen an ein System der Mobilkommunikation. [Fede95]

2.3.1 Vertraulichkeit

Nachrichteninhalte sollen vor allen Instanzen außer dem Kommunikationspartner vertraulich bleiben.

Sender und/oder Empfänger von Nachrichten sollen voreinander anonym bleiben können, und Unbeteiligte (inkl. Netzbetreiber) sollen nicht in der Lage sein, sie zu beobachten.

Weder potentielle Kommunikationspartner noch Unbeteiligte (inkl. Netzbetreiber) sollen ohne Einwilligung den momentanen Ort einer mobilen Teilnehmerstation bzw. des sie benutzenden Teilnehmers ermitteln können. [Riha94b]

2.3.2 Integrität

Fälschungen von Nachrichteninhalten (inkl. des Absenders) sollen erkannt werden.

Gegenüber einem Dritten soll der Empfänger nachweisen können, daß Teilnehmer X die Nachricht Y gesendet hat.

Der Absender soll das Absenden einer Nachricht mit korrektem Inhalt beweisen können, möblichst sogar den Empfang der Nachricht.

Niemand kann dem Netzbetreiber Entgelte für erbrachte Dienstleistungen vorenthalten. Umgekehrt kann der Netzbetreiber nur für korrekt erbrachte Dienstleistungen Entgelte fordern.

2.3.3 Verfügbarkeit

Das Netz ermöglicht Kommunikation zwischen allen Partnern, die dies wünschen (und denen es nicht verboten ist).

3 GSM

Die CEPT (Conférence Européenne des Administrations des Postes et Télécommunications) hat 1982 eine Arbeitsgruppe mit dem Namen *Groupe Spéciale Mobile* (GSM) gegründet, die untersuchen sollte, welche technischen und rechtlichen Voraussetzungen geschaffen werden müssen, um ein Mobilfunksystem entwickeln zu können, das europaweit und grenzüberschreitend verfügbar ist. Es war dies die Geburtsstunde des Systems, welches heute nicht nur in Europa, sondern bereits in weiten Teilen der Welt erfolgreich eingesetzt wird und deshalb zurecht den Namen *Global System for Mobile communications* (GSM) trägt. [Lobe94]

3.1 Anforderungen

Den Anstoß zur Entwicklung von GSM gab der Wunsch, die bestehenden Systeme zur Mobilkommunikation um ein weiteres zu ergänzen, das geeignet ist, einen Massenmarkt zu bedienen.

Zu Beginn der Entwicklung von GSM wurden folgende Anforderungen identifiziert ([Walk92]):

- Sprachübertragung mit hoher Qualität ist der wichtigste Dienst.

- Leistungsmerkmale, wie sie in modernen öffentlichen digitalen Fernsprechnetzen üblich sind, müssen verfügbar sein.
- Den Teilnehmern soll europaweite, grenzüberschreitende Kommunikation ermöglicht werden. Das bedeutet, daß länderübergreifend Kompatibilität gewahrt werden muß. Außerdem muß es Abkommen zwischen den nationalen Betreibern der Telekommunikationsnetze geben (*international roaming*).
- Den Teilnehmern soll die ortsunabhängige Kommunikation ermöglicht werden. Dies bedeutet, daß ein Teilnehmer unter einer einheitlichen Rufnummer erreichbar ist, ohne daß der Rufende vorher den Aufenthaltsort ermitteln muß. Die Teilnehmernummer soll also unabhängig vom gegenwärtigen Standort des Teilnehmers sein.
- Die Teilnehmer sollen permanent erreichbar sein. Umgekehrt soll ihnen jederzeit der Zugang zum Netz möglich sein.
- Die Mobilität des Teilnehmers soll nicht eingeschränkt werden, d. h. die mobilen Geräte können bei bestehender Verbindung durch das Versorgungsgebiet bewegt werden, ohne daß der Kontakt abreißt. Das Netz hat die Fähigkeit, bestehende Verbindungen entsprechend der Bewegung der Mobilstation unterbrechungsfrei nachzuführen. Diese Fähigkeit ist unter der Bezeichnung *seamless handover* geläufig.
- Da die Luftschnittstelle neue Angriffsmöglichkeiten bietet, z. B. Peilbarkeit und erleichterte Abhörbarkeit, sollen besondere Sicherheitsdienste eingesetzt werden. Das Netz muß vor unerlaubtem Zugang geschützt werden. Die Privatsphäre des Teilnehmers muß besonders auf der Luftschnittstelle geschützt werden.

3.2 Technische Lösungsversuche

- Um Kompatibilität zum ISDN zu wahren und eine hohe Qualität der Sprachübertragung zu sichern werden digitale Verfahren eingesetzt.
- Anforderungen an die Teilnehmerkapazität und die Eigenschaften der Mobilstation (tragbar) legten die Ausführung des Mobilnetzes als zellulares Netz nahe. Dieses Konzept ist schon von analogen Netzen (C-Netz) bekannt.
- Die permanente Erreichbarkeit eines Teilnehmers wird gewährleistet, indem sein Aufenthaltsort in einer zentralen Datenbank mitgeführt wird (s. Kapitel 3.3.2).
- Das kontinuierliche Mitführen bestehender Verbindungen bei beweglicher Mobilstation wird durch den *handover*-Mechanismus unterstützt (s. Kapitel 3.5.2).
- Um den erhöhten Sicherheitsanforderungen Rechnung zu tragen, wurde Verschlüsselung auf der Luftschnittstelle eingeführt. Desweiteren werden Pseudonyme vergeben, um die Anonymität der Teilnehmer zu wahren (s. Kapitel 3.6).

GSM-Netze haben eine verteilte Netzstruktur. Das Versorgungsgebiet wird in Lokalisierungsgebiete (*Location Area*, LA) aufgeteilt. Diese Gebiete werden durch Vermittlungsstellen (*Mobile Service Switching Centre*, MSC) mit zugehörigen lokalen Datenbanken

(*Visitor Location Register*, VLR) verwaltet. Der Verbindungsaufbau zur Mobilstation eines Teilnehmers erfolgt mit vorheriger Anfrage an eine zentrale Datenbank (*Home Location Register*, HLR). Eine MSC kann für mehrere LAs zuständig sein. Ein LA wird durch einen *Base Station Controller* (BSC) gesteuert. Ein BSC wiederum verwaltet eine oder mehrere Funkfeststationen (*Base Transceiver Station*, BTS), die die funktechnische Einrichtung für jeweils eine Zelle darstellen. Die konkrete Anzahl der Zellen innerhalb eines LA läßt sich optimieren aus der Betrachtung der Teilnehmermobilität, der Anzahl der mobilen Teilnehmer im LA und dem Nachrichtenverkehr im LA. BSC und zugehörige BTSs werden als ein *Base Station System* (BSS) bezeichnet. [Bial95]

Aufgrund der Mobilität der Teilnehmer hat die Verwaltung von Aufenthaltsinformationen, das sogenannte *Location Management*, eine besondere Bedeutung. Zu den Verwaltungsfunktionen gehören das Einbuchen einer Mobilstation, der Verbindungsaufbau von einer MS (*Mobile Originated Call*, MOC) und zu einer MS (*Mobile Terminated Call*, MTC) sowie das Aktualisieren der Aufenthaltsinformationen (*Location Update*, LUP). Um die Erreichbarkeit eines Teilnehmers im zellularen Funknetz zu gewährleisten, sind Aufenthaltsinformationen über den Teilnehmer bzw. seine MS notwendig. In existierenden Zellnetzen erfolgt die Speicherung dieser Informationen (z. B. des augenblicklichen Aufenthaltsgebietes) in zentral organisierten Datenbanken (HLR), die zumindest durch den Netzbetreiber mißbräuchlich verwendet werden können, um Bewegungsprofile zu erstellen. [FeJP+96]

3.3 Systemarchitektur

Ein Mobilfunknetz, das nach dem Standard GSM arbeitet, besteht aus drei Teilsystemen (s. Abbildung 1): dem funktechnischen Teilsystem (*Radio Subsystem*, RSS), dem vermittlungstechnischen Teilsystem (*Network and Switching System, NSS*) sowie dem Betriebs- und Wartungssystem (*Operation and Maintenance Subsystem*, OMS).

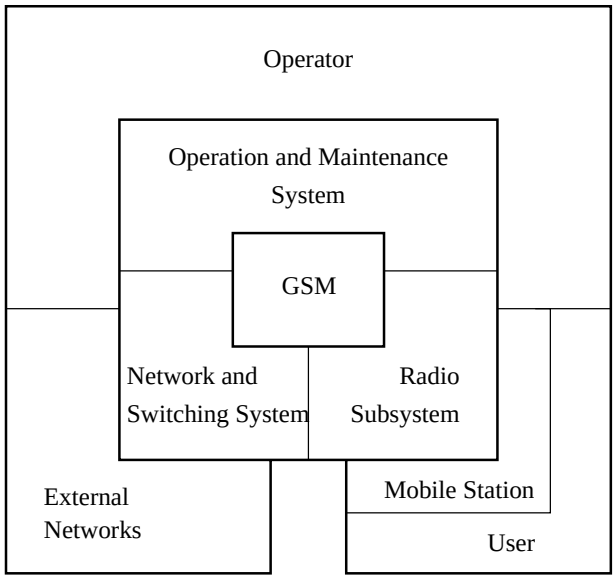

Abbildung 1: Architektur eines GSM-Systems

Die Komponente auf Teilnehmerseite ist das Endgerät (*Mobile Station*, MS), bestehend aus der funktechnischen Hard- und Software (*Mobile Equipment*, ME) und einer Chip-

karte (*Subscriber Identification Module*, SIM) mit den teilnehmerspezifischen Daten und Sicherheitsprozeduren für den Netzzugang. Die MS zählt nicht zu den Netzelementen, ist aber im Zusammenhang mit der Chipkarte das Systemelement, welches für den Mobilteilnehmer von größtem Interesse ist. Der Sicherheitsstandard, den GSM den Teilnehmern bietet, ist aufs engste mit dem SIM verknüpft. Sicherheitsrelevante Prozeduren werden nur in Interaktion zwischen dem GSM-Netz und dem SIM durchgeführt.

Ein schematischer Aufbau eines GSM-Netzes mit den wichtigsten Komponenten und ihren Zusammenhängen ist in Abbildung 2 dargestellt.

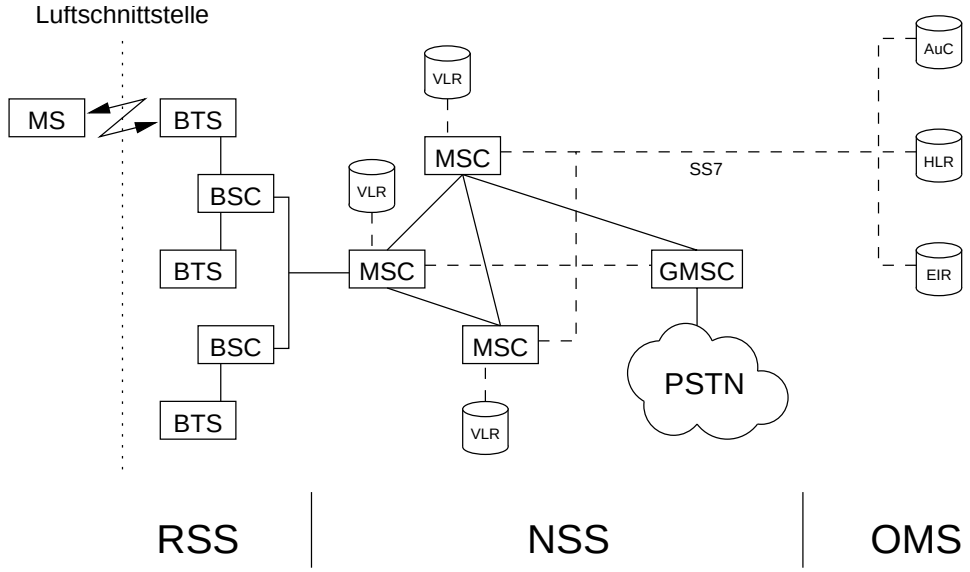

Abbildung 2: Schematischer Aufbau eines GSM-Netzes

3.3.1 OMS

Das OMS nimmt eine zentrale Stellung in einem GSM-Netz ein. Es beinhaltet die zum Betrieb notwendigen zentralen Datenbanken HLR (*Home Location Register*) und AuC (*Authentication Centre*) sowie das EIR (*Equipment Identity Register*).

Das HLR enthält feste, teilnehmerspezifische Daten, wie die öffentliche Rufnummer MSISDN und die netzinterne Kennummer IMSI (*International Mobile Subscriber Identity*) des Teilnehmers. Außerdem wird hier als variable Information die Adresse des VLR vermerkt, in dem der Teilnehmer zuletzt eingebucht war.

Im AuC findet die Zuordnung der IMSI zu den geheimen Teilnehmerschlüsseln Ki statt. Diese Zuordnung existiert sonst nur noch auf dem SIM. Das AuC erzeugt auch die für die Authentifikation nötigen Sicherheitsparameter (*Authentication Triplet*, AT) (siehe Kapitel 3.6.1). Im AuC sind außerdem die geheimen Algorithmen für die Erzeugung von Sitzungsschlüsseln (A8) und Authentifikationsparametern (A3) implementiert. Das gleiche gilt für das SIM. AuC und SIM stellen also die „Geheimnisträger“ in einem GSM-Netz dar.

Das EIR enthält eine Liste mit den Kennziffern (*International Mobile Equipment Identity*, IMEI) gestohlener Geräte.

Das OMS ermöglicht es dem Betreiber des GSM-Netzes, Fehler im System festzustellen, zu beheben und neue Konfigurationen im Netz vorzunehmen, z. B. die Zuordnung von Frequenzen in Basisstationen. Im OMS wird auch das Verkehrsaufkommen überwacht. Es ist möglich, Änderungen in der Netzplanung, die durch ein geändertes Verkehrsaufkommen naheliegen, über das OMS vorzunehmen.

3.3.2 NSS

Das NSS besteht aus den Vermittlungsknoten MSC mit jeweils zugeordneter Datenbank VLR. Über das NSS werden Verbindungen zwischen Teilnehmern im öffentlichen Telefonnetz und mobilen Teilnehmern aufgebaut.

Jedes VLR enthält die Daten der im MSC/VLR-Bereich registrierten Teilnehmer. Diese Daten umfassen die Einträge des HLR zuzüglich des aktuellen Pseudonyms TMSI (*Temporary Mobile Subscriber Identity*), eines Vorrats (min. 2, max. 6) von ATs, des aktuellen Sitzungsschlüssels Kc und seiner Referenznummer CKSN (*Ciphering Key Sequence Number*) sowie der Adresse LAI (*Location Area Identity*) des aktuellen Aufenthaltsbereichs des Teilnehmers.

3.3.3 RSS

Die Komponenten des RSS sind die BSSs (*Base Station System*), weiter unterteilt in BSC (*Base Station Controller*) und BTS (*Base Transceiver Station*). In der BTS sind alle funktechnischen Einrichtungen vorhanden, die erforderlich sind, um die Verbindung von und zu einem mobilen Teilnehmer sicherzustellen.

GSM verwendet ein kombiniertes Verfahren aus FDMA und TDMA. Die Übertragung von der MS (*uplink*) und zur MS (*downlink*) erfolgt in getrennten Frequenzbändern. Uplink benutzt das Band von 890 - 915 MHz, Downlink das Band von 935 - 960 MHz. Die Kanalbreite beträgt 200 kHz, d. h. in jeder Richtung gibt es 124 Trägerkanäle. Es gilt die Beziehung Downlink = Uplink + 45 MHz (Duplexabstand). Jeder Träger wird mit TDMA in acht Zeitschlitze von je 4.615 ms Länge aufgeteilt.

Jedem physikalischen TDMA-Kanal wird eine logische Kanalstruktur aufgeprägt. Bei den logischen Kanälen wird zwischen Verkehrskanälen (*TCH, Traffic Channel*) und Signalisierungskanälen (*CCH, Control Channel*) unterschieden. Die Aufteilung mit den wichtigsten Kanälen ist in Bild 3 abgebildet. [GaWi96]

Hinsichtlich der Abbildung von logischen Kanälen auf einen physikalischen Kanal verweise ich den interessierten Leser auf [DaBe96] und [ReWe95].

3.4 Geographische Aufteilung des Netzes

Das Netz ist hierarchisch strukturiert (s. Bild 4).

An unterster Stelle steht die einzelne Zelle, die durch eine Sende-/Empfangsanlage (BTS) versorgt wird. Eine Zelle hat eine räumliche Ausdehnung von 500m bis 35km. Die D-Netze in Deutschland bestehen jeweils aus ca. 2500 Zellen. Bei einer Fahrt von Hamburg nach München kreuzt man ca. 60 Zellen ([Jörn94]).

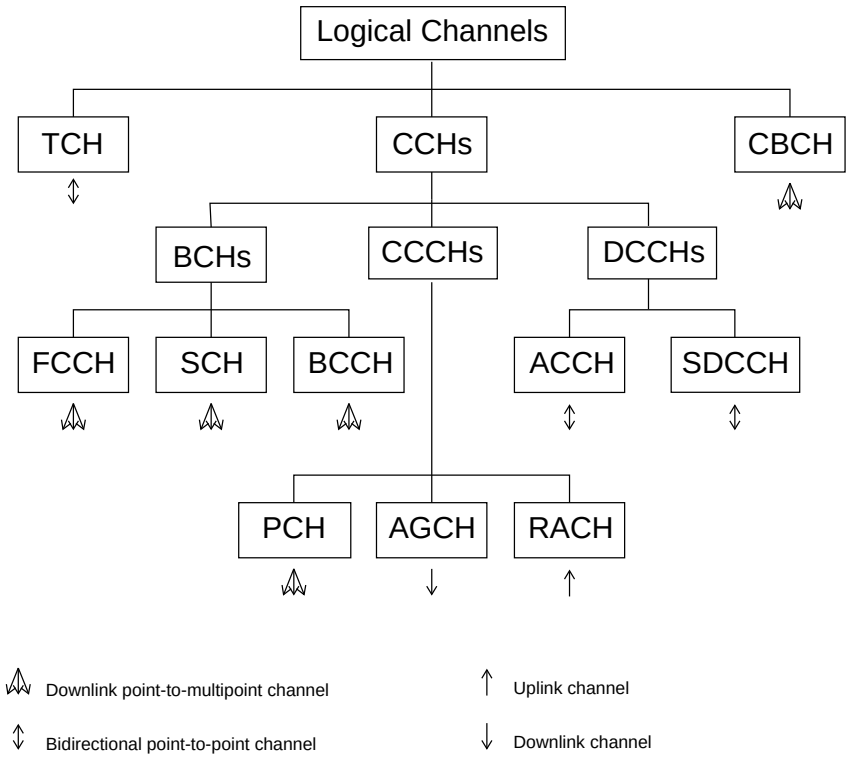

Abbildung 3: Logische Kanalstruktur

BCCH	*Broadcast Control CHannel*, allgemeine Daten über das PLMN, Feststations-ID
PCH	*Paging CHannel*, informiert die MS über eine Ruf vom Festnetz
RACH	*Random Access CHannel*, MS fordert hier Kanäle an
AGCH	*Access Grant CHannel*, BSC weist der MS einen SDCCH oder TCH zu
SDCCH	*Stand-alone Dedicated Control CHannel*, ohne TCH, für Signalisierung, location update, Authentifikation
SACCH	*Slow Associated Control CHannel*, TCH oder SDCCH beigeordnet, Messdaten
FACCH	*Fast Associated Control CHannel*, TCH beigeordnet, Handover, unregelmäßiger Steuerbedarf

Tabelle 1: Beschreibung der logischen Kanäle

Mehrere Zellen, deren BTSs durch ein gemeinsames BSC gesteuert werden, bilden ein Lokalisierungsgebiet (LA). Ein LA ist die kleinste geographisch adressierbare Einheit des Funknetzes. Durch die LAI, die über den BCCH ausgestrahlt wird, ist jedes LA eindeutig bestimmt.

Mehrere LAs werden einer Mobilvermittlungsstelle (MSC) mit VLR zugeordnet (*VLR-Area*). Die D-Netze umfassen je ca. 40 MSCs.

3.5 Prozeduren

Hier werden einige der Prozeduren beschrieben, die bei der Nutzung von GSM-Diensten ablaufen können.

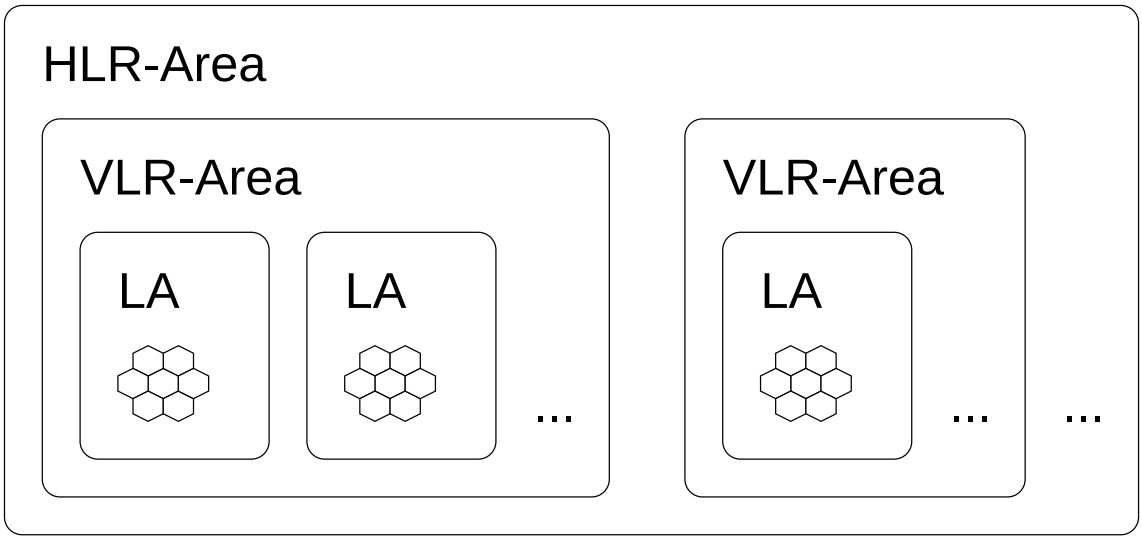

Abbildung 4: Hierarchische Gebietsaufteilung eines GSM-Netzes

3.5.1 Einbuchen

Wenn ein Teilnehmer den Zugang zum Netz herstellen will, was durch das Einschalten der MS angezeigt werden kann, wird ein Einbuchen ausgelöst (s. Abbildung 5). Die MS empfängt auf dem BCCH die LAI der Funkzelle, in der sie sich befindet. Mit dieser LAI wird eine *location registration* ausgelöst. Im Zuge dessen überträgt die MS die IMSI an das VLR. Dort wird eine neue TMSI generiert und nach erfolgreicher Authentikation (siehe 3.6.1) verschlüsselt an die MS übertragen. Außerdem werden die teilnehmerspezifischen Daten aus dem HLR abgefragt und die Aufenthaltsinformation dort aktualisiert.

Bild 5 zeigt ein erstmaliges Einbuchen, bei dem der MS keine TMSI bekannt ist. Hier muß die IMSI übertragen werden. Bei späteren Einbuchvorgängen wird immer die letzte bekannte TMSI benutzt, was die Anonymität des Teilnehmers besser schützt.

3.5.2 Handover (HOV)

Vom Netz ausgelöstes Weiterreichen einer Funkverbindung. Grundlage der Handover-Entscheidung sind Meßwerte über Signalstärke und Fehlerrate, die für uplink von den BTSs und downlink von der MS gemessen werden. Die MS überträgt die Meßwerte über den SACCH. Sorgfältig ausgewählte Algorithmen sorgen dafür, daß ein Wechsel der BTS nur selten nötig ist und ein „Flattern“ an der Grenze zweier Zellen vermieden wird.

Je nach Zugehörigkeit der neuen Zelle zu anderen Bereichen des Netzes ändert sich die Aufwendigkeit des *handover*.

Gehört die neue Zelle zum gleichen LA, wird das *handover* im zugehörigen BSC durchgeführt. Auch einfache Wechsel der Frequenz zur Vermeidung von frequenzabhängigen Störungen werden so behandelt.

Gerät die MS durch den HOV in ein anderes LA, was durch den LAI erkennbar ist, der über den BCCH gesendet wird, leitet die MS einen *location update* ein, d. h. MSC und VLR sind mitbetroffen.

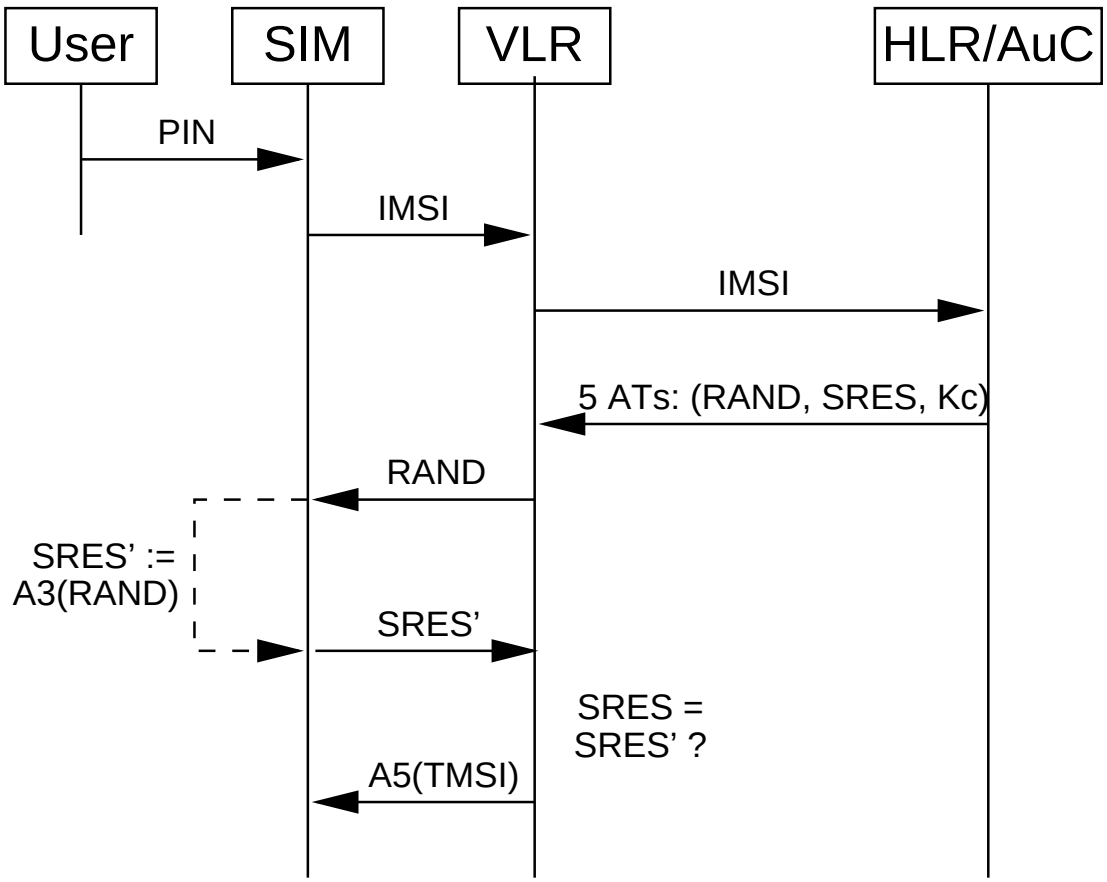

Abbildung 5: Ablauf beim erstmaligen Einbuchen einer MS

Am aufwendigsten ist der Wechsel zu einer Zelle eines Fremdnetzes. Hier wird vermutlich die Anforderung, das *handover* in 90 Prozent aller Fälle in weniger als 150ms durchzuführen, nicht erfüllt.

3.5.3 Location Update LUP

LUP bezeichnet eine von der MS ausgelöste Aktualisierung der Aufenthaltsinformation im VLR. Die MS sendet dem neuen BSC die alten Werte LAI und TMSI. Mit dem alten LAI wird das alte VLR bestimmt. Dort wird mit der alten TMSI die tatsächliche Identität IMSI des Teilnehmers bestimmt. Weiterhin werden alle noch vorhandenen ATs an das neue VLR übertragen. Mit der IMSI kann das neue VLR die Lokalisierungsinformation im HLR auf Stand bringen. Nach der Bestätigung des neuen Aufenthaltsortes löscht das alte VLR auf Anforderung des HLR die alte Aufenthaltsinformation. Das neue VLR vergibt eine neue TMSI, die nach erneuter Authentifikation verschlüsselt zur MS übertragen wird. [Schw97]

3.5.4 Mobile Terminated Call (MTC)

MTC bezeichnet den Rufaufbau zur MS. Damit sind die Signalisierungsvorgänge gemeint, die zur Lokalisierung des Teilnehmers führen. Davon ist der Verbindungsaufbau zu unterscheiden, der über andere Wege verlaufen kann und hier nicht betrachtet wird. GSM verwendet zur Zeichengabe das *signaling system No. 7* (SS7).

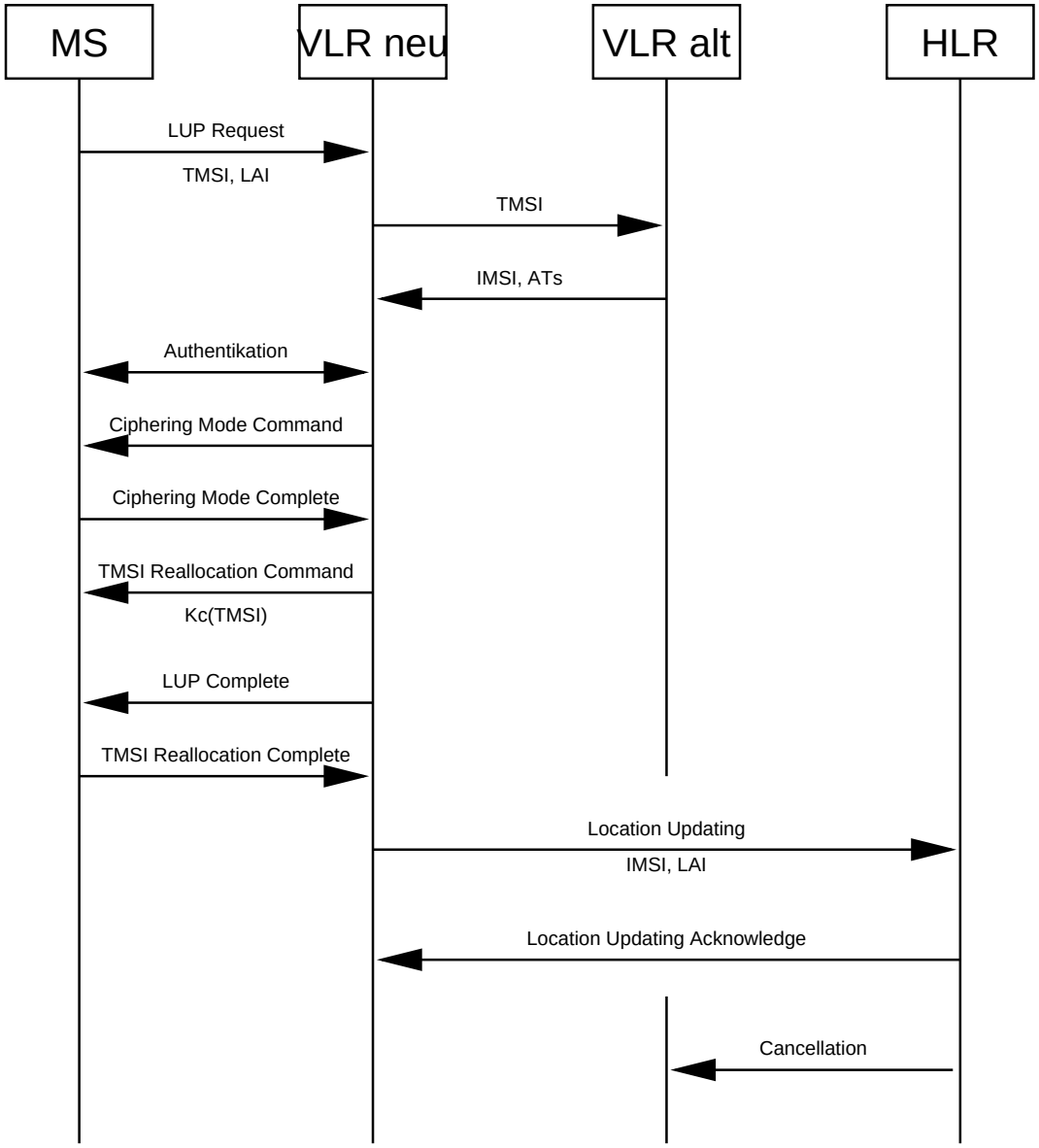

Abbildung 6: Aktualisierung der Aufenthaltsinformation (LUP)

Ein Ruf aus dem Festnetz erreicht zunächst eine dafür speziell vorgesehene MSC (*Gateway MSC*, GMSC). Ihr steht nur die Teilnehmernummer MSISDN des öffentlichen Netzes zur Verfügung. Mit ihr wird im HLR die Berechtigung des gerufenen Teilnehmers geprüft. Im positiven Fall wird die Adresse des VLR bestimmt, in dessen Bereich der Teilnehmer zuletzt eingebucht war. Das HLR verfügt über eine Kennzahl zur Verkehrslenkung (*Mobile Station Routing Number*, MSRN), die das VLR vergeben hat, und reicht sie an die GMSC weiter. Mit Hilfe der MSRN baut die GMSC die Verbindung bis zu der MSC auf, in deren Bereich der gerufene Teilnehmer eingebucht ist. Im zuletzt registrierten LA wird der Teilnehmer durch einen Rundruf (*paging*) der letzten TMSI über den PCH gesucht. Die MS antwortet auf den Rundruf, der die ihr bekannte TMSI enthält, mit einer *paging response*. Daraufhin wird ihr ein Kanal zugewiesen, über den eine Authentifikation durchgeführt und bei Erfolg die weitere Signalisierung abgewickelt wird. [Kemm96], [GaWi96]

3.6 Sicherheitsdienste

Durch den offenen Netzzugang über die Funkschnittstelle besteht erhöhte Gefahr, daß die Kommunikation mobiler Teilnehmer durch Dritte abgehört wird, bzw. die Netzressourcen unbefugt auf Kosten eingetragener Teilnehmer genutzt werden. Um dies zu verhindern, wurde der Zugang bzw. die Nutzung des Mobilkommunikationsnetzes durch Sicherheitsprozeduren geschützt, die sich im wesentlichen auf den Nachweis der Identität von Mobilteilnehmern durch Authentifikation, Verschlüsselung der Nachrichtenübermittlung (inkl. der Signalisierungsdaten), sowie Anonymisierung der Identität von Mobilteilnehmern durch zeitlichen Wechsel der Funkkennung stützen. Die digitale Nachrichtenübertragung beim GSM-Netz gestattet den Einsatz kryptographischer Verfahren.

3.6.1 Zugangskontrolle / Authentifikation

Gegenüber dem SIM authentisiert sich der Teilnehmer mit einer Geheimnummer PIN (*Personal Identification Number*), die auf dem SIM gespeichert ist und geändert werden kann. Nach mehrmaliger falscher Eingabe der PIN ist das SIM gesperrt und kann nur mit dem PUK (*Personal Unblocking Key*, ugs. Super-PIN) wieder funktionsfähig gemacht werden.

Gegenüber dem Netz authentisiert sich der Teilnehmer bzw. sein SIM mit einem *Challenge-Response*-Verfahren. Dabei leitet das VLR, dem ein AT für die IMSI oder TMSI bekannt sein muß, den Vorgang ein, indem es eine Zufallszahl RAND, die *challenge*, an das SIM sendet. Das SIM berechnet mit Hilfe des geheimen individuellen Teilnehmerschlüssels Ki die *response* SRES' durch den Algorithmus A3. Das SIM sendet SRES' an das VLR, wo mit der erwarteten Antwort SRES verglichen wird, die im AuC mit dem dort bekannten Ki berechnet wurde. Bei Gleichheit ist der Teilnehmer authentisiert, sonst werden alle bestehenden Transaktionen abgebrochen. Der Vorgang ist in Abbildung 5 dargestellt.

Nach einer fehlgeschlagenen Authentifikation kann das Netz von der MS die Übermittlung ihrer IMEI fordern. Im EIR existieren Listen mit gestohlenen oder in ihrer Funktionalität beeinträchtigten Geräten.

3.6.2 Vertraulichkeit

Sprache und Signalisierungsdaten werden nach erfolgreicher Authentifikation und einem entsprechenden Kommando der MSC (*ciphering mode command*) verschlüsselt übertragen. Es kommt ein symmetrisches Verschlüsselungsverfahren zum Einsatz. Bei jeder Authentifikation wird ein neuer Sitzungsschlüssel Kc bestimmt, der als Parameter für den geheimen Verschlüsselungsalgorithmus A5 dient. [Gerl97]

Der Sitzungsschlüssel Kc (64bit) wird im Zuge der Authentifikation jeweils neu aus der Zufallszahl RAND (128bit) und dem individuellen Teilnehmerschlüssel Ki (128bit) berechnet. Dies geschieht in der MS und im AuC. Der Schlüssel ist somit auf beiden Seiten der Luftschnittstelle bekannt, ohne jedoch darüber gesendet worden zu sein. Beide Seiten benutzen die CKSN um sicherzustellen, daß noch der gleiche Schlüssel benutzt wird. [Behe94] Die Schlüsselberechnung erfolgt mit dem vom Betreiber gewählten Algorithmus A8.

3.6.3 Anonymität

Mindestens bei jedem LUP wird eine neue temporäre Teilnehmerkennung TMSI vergeben und im VLR gespeichert. Sie wird ebenfalls verschlüsselt übertragen. Dieses Pseudonym sichert die Anonymität des Teilnehmers und verwehrt Außenstehenden die Möglichkeit, Aufenthaltsinformationen zu einem Bewegungsprofil zu verketten.

4 Bewertung der Sicherheit in GSM

Die derzeit im GSM verwirklichten Methoden zum Datenschutz beschränken sich auf die notwendigsten Bereiche. Es wurde das gleiche Sicherheitsniveau wie im Festnetz angestrebt. Die Möglichkeiten von GSM als für Weiterentwicklungen offenes System lassen erwarten, daß die Lücken zwischen technischer Realisierung und den Datenschutzanforderungen nach und nach geschlossen werden ([MoPa95]). Aber es gibt auch Bestrebungen, die bestehenden Sicherheitsstandards unverändert in ein zukünftiges Mobilkommunikationssystem UMTS zu übernehmen. [Pütz97].

4.1 Verschlüsselung

Da man Abhörversuche auf der Luftschnittstelle für am wahrscheinlichsten hielt, wurde hier Verschlüsselung der Nutz- und Verkehrsdaten vorgesehen. Es gibt jedoch keine Ende-zu-Ende-Verschlüsselung. Eine Verbindung zum PSTN kann dort ohne weiteres mitgehört werden.

4.2 Preisgabe des Aufenthaltsortes

Das an der Luftschnittstelle verwendete Übertragungsverfahren erlaubt eine einfache Ortung von sendenden Mobilstationen. Es existieren Verfahren, die diese Schwächen nicht haben, z. B. *direct sequence spread spectrum*. Dennoch werden sie in GSM nicht eingesetzt. [Thee94]

4.3 Speicherung von Teilnehmerinformation

Ein weiterer Kritikpunkt am GSM ist die Speicherung der teilnehmerspezifischen Erreichbarkeitsinformation in Datenbanken wie HLR und VLR. Der Zugriff auf diese Dateien ermöglicht auf einfache Weise die Erstellung eines Bewegungsprofils [Thee94]. Es existieren Ansätze, bei denen die Anonymität eines Teilnehmers auch vor dem Netzbetreiber gewahrt werden kann. Sie beruhen darauf, daß die Aufenthaltsinformation nicht mehr zentral, sondern verteilt in privaten Datenbanken der einzelnen Teilnehmer gespeichert werden. [FeJP+96]. Diese Ansätze sind jedoch im GSM nicht verwirklicht.

Die Bedrohung durch Angriffe über das Netz auf die Datenbanken wurde im GSM nicht berücksichtigt.

4.4 Geheimhaltung der Verschlüsselungsalgorithmen

Ein weiterer Schwachpunkt ist die Geheimhaltung der verwendeten Algorithmen. Dadurch ist es nicht möglich, deren kryptographische Stärke öffentlich zu diskutieren. Es bleibt der Verdacht, es könnten absichtlich Schwachstellen im Entwurf der Algorithmen vorgesehen worden sein. Im Übrigen ist es nicht möglich, bei der Vielzahl von Herstellern die Geheimhaltung zu garantieren. In einschlägigen *newsgroups* des Internet kursieren bereits C-Implementationen des A5-Algorithmus ([Ande94]).

4.5 Einseitige Authentifikation

Der größte Schwachpunkt ist jedoch die fehlende mehrseitige Authentifikation. Durch die fehlende Authentifikation des Netzes gegenüber dem Teilnehmer werden vielfältige Möglichkeiten für Angriffe geboten. Grundlage solcher Angriffe ist jeweils die Maskerade eines Mobilfunknetzes.

Die IMSI eines Teilnehmers soll nur im Fehlerfall unverschlüsselt über die Luftschnittstelle übertragen werden. Ein solcher Fehlerfall kann jedoch auch während eines Maskeradeangriffs durch einen vermeintlichen Netzbetreiber inszeniert werden. Geräte, die entsprechendes leisten, sind bereits verfügbar, z. B. von der Firma Rohde & Schwarz (München). Sie sind unter der Bezeichnung *IMSI-Catcher* geläufig. [Fox97a]. Diese Geräte fordern mittels eines *identity request* die Mobilstation dazu auf, die IMSI zu übermitteln ([Jobm]). Auf diese Weise wird die Pseudonymisierung von Teilnehmeridentitäten unterlaufen.

Die Anonymität der Teilnehmer ist also vor Außenstehenden nicht gewahrt. *Insider*, d. h. Personen, die mit dem Netzbetreiber zusammenarbeiten, haben darüberhinaus auch Zugriff auf den geheimen Teilnehmerschlüssel Kc und können damit Gespräche des Teilnehmers mithören [SPIE97a]).

Es sind weitere Varianten von Angriffen denkbar. Ein IMSI-Catcher könnte einer MS eine Vielzahl von *authentication requests* mit selbst gewählten Werten RAND vorspiegeln. Die MS antwortet mit dem von ihr berechneten Wert SRES (siehe 3.6.1). Auf diese Weise können Angreifer Paare (RAND, SRES) sammeln und für einen kryptographischen Angriff (*known plaintext attack*) auf den Algorithmus A3 verwenden.

Mitarbeiter der Cambridge University haben ein anderes Szenario entwickelt ([Luck97]). Sie wollen ausnutzen, daß die Netzbetreiber die ATs (RAND, SRES, Kc) bei der Übertragung zum BSC nicht verschlüsseln. Daher könne man durch Abhören der Richtfunkstrecken mit einem Mikrowellenempfänger ATs sammeln und mit einem modifizierten Handy zum betrügerischen Authentisieren und somit Telefonieren auf Kosten eines fremden Teilnehmers benutzen. Voraussetzung ist, daß eine fremde IMSI bekannt ist, was mit dem oben erwähnten IMSI-Catcher erreicht werden kann. Allerdings ist es nicht ganz so einfach. Die ATs werden nicht bis zu den BSCs übermittelt, sondern nur zum MSC/VLR. Bis dorthin sind Richtfunkstrecken unüblich, das Abhören praktisch nicht möglich. Das teilen jedenfalls die Betreiber mit.

5 Schluß

Zusammenfassend läßt sich sagen, daß GSM dem Teilnehmer einen höheren Sicherheitsstandard bietet, als es bisher in Festnetzen üblich war. Ein GSM-Netz ist zwar ein funktionierendes mobiles Netz mit Sicherheitsfeatures, erfüllt jedoch nicht alle Technischen Datenschutzanforderungen. Insbesondere der Mehrseitigkeitsaspekt von Sicherheit wurde nicht berücksichtigt.

Literatur

[Ande94] Ross Anderson. a5 (was: HACKING DIGITAL PHONES). *USENET newsgroups: sci.crypt, alt.security, uk.telecom*, 1994.

[Behe94] Johannes Beheim. Sicherheit und Vertraulichkeit bei europaweiter Mobilkommunikation. *Datenschutz und Datensicherheit* 18(6), 1994, S. 327–331.

[Bial95] Jacek Biala. *Mobilfunk und Intelligente Netze*. vieweg, Braunschweig, Wiesbaden. 2. Auflage, 1995.

[BiFo97a] Johann Bizer und Dirk Fox. Freiheit und Regulierung. *Datenschutz und Datensicherheit* 21(4), 1997, S. 182.

[BiFo97b] Johann Bizer und Dirk Fox. Spurenlos mobil? *Datenschutz und Datensicherheit* 21(6), 1997, S. 314.

[ChBe94] William R. Cheswick und Steven M. Bellovin. *Firewalls and Internet Security*. Addison-Wesley. 1994.

[Crui96] H S Cruickshank. A Security System For Satellite Networks. In *Satellite Systems for Mobile Communications and Navigation, 13-15th May 1996, Conference Publication No. 424*. IEE, 1996, S. 187–190.

[DaBe96] Klaus David und Thorsten Benkner. *Digitale Mobilfunksysteme*. Informationstechnik. B. G. Teubner, Stuttgart. 1996.

[Detk96] Kai-Oliver Detken. Schutzmechanismen. *Gateway* -(3), 1996, S. 96–105.

[DHSS96] Elmar Dorner, Markus Hofmann, Claudia Schmidt und Jochen Seitz (Hrsg.). *Seminarbericht Netzwerkmanagement und Hochgeschwindigkeitskommunikation, Teil XIII, Wintersemester 1995/96. Interner Bericht Nr. 14*. Institut für Telematik, Fakultät für Informatik, Universität Karlsruhe, Karlsruhe. 1996.

[Fede95] Hannes Federrath. Technischer Datenschutz in der Mobilkommunikation – Analyse von Sicherheitsproblemen in existierenden Netzen. *http://www.inf.tu-dresden.de/ hf2/DBSA4GforW3/DBSA4GforW3.html*, 1995.

[FeJP+96] Hannes Federrath, Anja Jerichow, Andreas Pfitzmann, Dogan Kesdogan und Otto Spaniol. Mobilkommunikation ohne Bewegungsprofile. *it+ti* -(4), 1996, S. 24–29.

[Forg97] Helmut Forgber. Stop&Call. *ADAC motorwelt* -(11), 1997, S. 6–8.

[Fox97a] Dirk Fox. IMSI-Catcher. *Datenschutz und Datensicherheit* 21(9), 1997, S. 539.

[Fox97b] Dirk Fox. Key Recovery. *Datenschutz und Datensicherheit* 21(4), 1997, S. 227.

[GaWi96] Vijay K. Garg und Joseph E. Wilkes. *Wireless and personal communications systems*. Prentice Hall Digital and Wireless Communications Series. Prentice Hall, Upper Saddle River, New Jersey. 1996.

[Gerl97] Rainer W. Gerling. Verschlüsselungsverfahren. *Datenschutz und Datensicherheit* 21(4), 1997, S. 197–201.

[GG10] GG10. Grundgesetz. Artikel 10 [Brief-, Post- und Fernmeldegeheimnis].

[Hamm97] Rainer Hamm. Kryptokontroverse. *Datenschutz und Datensicherheit* 21(4), 1997, S. 186–191.

[Horn74] A S Hornby. *Oxford Advanced Learner's Dictionary Of Current English*. Cornelsen & Oxford University Press, Berlin. 1974.

[HuPf96] Michaela Huhn und Andreas Pfitzmann. Technische Randbedingungen jeder Kryptoregulierung. *Datenschutz und Datensicherheit*, 1996, S. 23–26.

[Jobm] Klaus Jobmann. Sicherheit im GSM. Uni Hannover, Institut für Allgemeine Nachrichtentechnik.

[Jörn94] Fritz Jörn. *Mobiltelefonieren–leicht gemacht*. Sybex, Düsseldorf. 1994.

[Kemm96] Martin Kemmler. Mobile Datenkommunikation in GSM-Netzen. In Dorner und andere [DHSS96], S. 73–91.

[Krus92] Jörn Kruse (Hrsg.). *Zellularer Mobilfunk*. net-Buch Telekommunikation. R. v. Decker's Verlag, G. Schenk GmbH, Heidelberg. 1992.

[Lobe94] Hans Lobensommer. *Die Technik der modernen Mobilkommunikation*. Franzis, München. 1994.

[Luck97] Norbert Luckhardt. Wenn das Handy fremdgeht. *c't magazin für computertechnik* -(16), 1997, S. 31.

[Luka97] Jörg Lukat. Teilnehmerbezogene Daten im D1-Netz. *Datenschutz und Datensicherheit* 21(6), 1997, S. 317–320.

[MoPa95] Michel Mouley und Bernadette Pautet. Evolution of the GSM Systems. *IEEE Personal Communications* -(11), 1995, S. 9–19.

[Pütz97] Stefan Pütz. Zur Sicherheit digitaler Mobilfunksysteme. *Datenschutz und Datensicherheit* 21(6), 1997, S. 321–327.

[ReWe95] Siegmund Redl und Matthias Weber. *GSM-Technik und Meßpraxis*. Funkschau: Technik. Franzis-Verlag, Poing. 2. Auflage, 1995.

[Riha94a] Karl Rihaczek. Steganographie. *Datenschutz und Datensicherheit* 18(6), 1994, S. 340.

[Riha94b] Karl Rihaczek. Vertraulichkeit. *Datenschutz und Datensicherheit* 18(9), 1994, S. 512.

[Rubn97] Jeanne Rubner. Handy legt Spur zu seinem Benutzer. *Süddeutsche Zeitung, Mittwoch, 31. Dezember*, 1997, S. 1.

[Schw97] Silke Schwaß. *Vergleich und Bewertung von Sicherheitsaspekten in GSM und Mobile IP. Diplomarbeit.* Institut für Telematik, Universität Karlsruhe. 1997.

[SPIE96a] DER SPIEGEL. Gefährliche Handys. *DER SPIEGEL* 47(23), 1996, S. 17.

[SPIE96b] DER SPIEGEL. Wie Gewehre wegschließen. *DER SPIEGEL* 47(2), 1996, S. 21.

[SPIE97a] DER SPIEGEL. Abhören leicht gemacht. *DER SPIEGEL* 48(35), 1997, S. 16.

[SPIE97b] DER SPIEGEL. Bei Anruf Tod? *DER SPIEGEL* 48(39), 1997, S. 16.

[SPIE97c] DER SPIEGEL. Dreßler Unfall: Handy am Steuer? *DER SPIEGEL* 48(47), 1997, S. 18.

[Stol95] Peter Stollenmayer. ISDN und seine Bedeutung in der Datenkommunikation. *HMD—Theorie und Praxis der Wirtschaftsinformatik* 32(184), 1995, S. 122–125.

[Thee94] Jürgen Thees. *Konkretisierung der Methoden zum Schutz von Verkehrsdaten in Funknetzen. Diplomarbeit.* Technische Universität Dresden, Fakultät für Informatik. 1994.

[TKG] TKG. Telekommunikationsgesetz (TKG), 1. August 1996. Elfter Abschnitt: Fernmeldegeheimnis, Datenschutz, Sicherung. Paragraph 85.

[Walk92] Bernhard Walke. Technik des Mobilfunks. Dienste und Protokolle digitaler Netze. In Kruse [Krus92], Kapitel 2, S. 17–63.

[Wong96] Ken Wong. Mobile Phone Fraud. *Datenschutz und Datensicherheit* 20(12), 1996, S. 736–741.

Möglichkeiten der Satellitenkommunikation

Florian Krako

Kurzfassung

Die terrestrischen Mobilfunknetzte werden bald alle größeren Ballungsräume der Erde flächendeckend versorgen können. Wo diese allerdings versagen, oder nicht rentabel erscheinen, entfaltet die Satellitenkommunikation ihre volle Stärke. Mittels ihrer Broadcast-Fähigkeit und relativ geringem Aufwand werden riesige Gebiete mit einigen wenigen Satelliten lückenlos versorgt. Durch die Integration in vorhandene Infrastrukturen entsteht so das Kommunikationsnetz der Zukunft. Welche Möglichkeiten die Satellitenkommunikation dabei bietetet, welcher Komponenten es dabei bedarf, und wie die physikalischen Nachteile von Funkübertragungen durch neue Protokolle minimiert werden, wird hier diskutiert. Außerdem wird auf die Möglichkeiten eingegangen, Satellitenfunkstrecken zur Anbindung an ATM zu nutzen.

1 Einleitung

1.1 Historie

- 1957 SPUTNIK - erster Satellit im All
- 1963 SYNCOM - erster geostationärer Satellit
- 1965 INTELSAT I - erster kommerzieller geostat. Satellit, Lebensdauer 1,5 Jahre
- 1982 INMARSAT A - erstes mobiles Satelliten-Telefonsystem (maritim)
- 1993 INMARSAT-M - erstes digitales Satelliten-Telefonsystem
- ca.1998 - globale Satelliten-Telefonsysteme für Handies

1.2 Grundlagen

Mobilfunk ist heute schon für viele zu einem selbstverständlichen Bestandteil eines umfassenden Kommunikationsangebotes geworden. Die Kommunikationslandschaft von morgen sieht dagegen noch viel flexiblere Konzepte zur Integration der verschiedensten Dienste in einem weltumspannenden Mobilfunknetz vor. Das Schlagwort heisst hier

PCN (Personal Communications Network) beziehungsweise *PCS (Personal Communications System)*. Im Mittelpukt des Kommuikationsszenarios der Jahrtausendwende steht der persönliche, standortunabhängige Datenaustausch.

Der Trend geht weg von netzwerkbezogener- hin zu teilnehmerbezogener Übertragung; sowie sich ein Übergang von physikalischen- zu logischen, d.h. *virtuellen* Kanälen abzeichnet. Das zukünftige diensteintegrierende Netz stellt dem Nutzer transparent mehrere Übertragungsmedien gleichzeitig zur Verfügung (z.B. DECT /GSM / Satelliten), unter denen dann mittels einer Prioritätenliste automatisch das jeweils günstigste gewählt wird. Für den Endbenutzer jedoch stellt dies keine Einschränkung dar, da er stets unter seiner weltweit eindeutigen, persönlichen Telefonnummer erreichbar sein wird, wobei er die Möglichkeit hat sich der verschiedensten Dienste, wie Sprach- Daten- oder Faxübertragung, zu bedienen. Dazu setzen die Satellitensysteme, die nicht als Konkurrenz, sondern als Erweiterung der terrestrischen Netze verstanden werden wollen, vor allem auf sogenannte *multimode handies*.

Um diese Integration sinnvoll zu gestalten müssen diese nicht-terrestrischen Systeme allerdings die gleiche hohe Dienstgüte anbieten können, wie die vorhandenen leitungsgebundenen Netzwerke. Dazu bedarf es besonderer Mechanismen, in Hinblick auf die physikalischen Nachteile die die Funkverbindungen gegenüber festverdrahteten Netzwerken darstellen. Wir werden darauf später noch genauer eingehen.

Die bisher schon allgegenwärtigen Satellitenschüsseln deuten an, daß dieser Technologie eine Schlüsselrolle bei der Datenverteilung zukommt. Bisher benutzte analoge Broadcast Satelliten verfügen über eine hohe Sendeleistung und sind für wenige uplinks, aber viele kostengünstige Empfangsterminals (Sat-Receiver) ausgeleget. Die Entwicklung gibt jedoch ganz eindeutig den volldigitalisierten Systemen den Vorzug, da diese die breitesten Multiplex- sowie Fehlerkorrekturmöglichkeiten bieten. Sie sind natürlich dementsprechend kostenintensiv in Entwicklung und Wartung, wobei wir schon beim ersten Nachteil von Satelliten wären. Projekte wie IRIDIUM oder GLOBALSTAR [A.Bö95] verschlucken beispielsweise zweistellige Milliardenbeträge und müssen sich in kürzester Zeit amortisieren, da Satelliten im allgemeinen nur recht kurze Lebenserwartungen (um die 10 Jahre) haben. Des weiteren kämpfen Satellitenfunkstrecken mit enormen Signalverzögerungszeiten und -Dämpfungen, sodaß trickreiche Verfahren gefragt sind, um diese so weit wie möglich auszugleichen. Wie bei allen Funkmedien hat man auch hier nur eine begrenzte Anzahl an Frequenzen zur Verfügung, was erstens die Bandbreite der einzelnen Kanäle einschränkt, sowie neue Multiplexverfahren nötig macht, um den steigenden Anstrom von Daten zu bewältigen. Meteorologische Einflüsse wie Wolken und Nebel stellen, bei den lichtähnlichen Frequenzen welche im Satellitenfunk benutzt werden, ein weiteres Hindernis dar.

Die Vorteile, die ein Satellitensystem bietet liegen allerdings klar auf der Hand. Zum einen können durch geschickte Kombination verschiedener Multiplextechniken sehr hohe Durchsätze erreicht werden (bis zu 1 Gbit/sec und mehr). Desweiteren ist es bei dieser Übertragungsmethoden nicht nötig ein standortabhängiges routing zu implementieren, was sehr zugunsten der Systemleistung geht. Neue, sowie mobile Benutzer können ohne großen Aufwand in die bestehende Infrastruktur eingegliedert werden, und sofort über große Distanzen hinweg miteinander kommunizieren. Dies bedeutet also eine sehr schnelle Verfügbarkeit der Dienste nach der Installation der Satelliten im Orbit. Dem Benutzer entstehen dabei keine zusätzlichen, entfernungsabhängigen Kosten oder Kosten fuer die *last mile* (Teilnehmeranschluß), die ja in Zukunft be-

kanntlich eine nicht zu vernachlässigende Rolle spielen werden. Der wichtigste aller Vorteile der orbitralen Kommunikation ist jedoch ihre geradezu hervorragende Eignung zur Broadcast-Sendung. Dieser Schlüsselvorteil macht die Satellitentechnik, als Erweiterung terrestrischer Netzwerke, hin zu einem multicastfähigen Gesamtnetz so unverzichtbar.

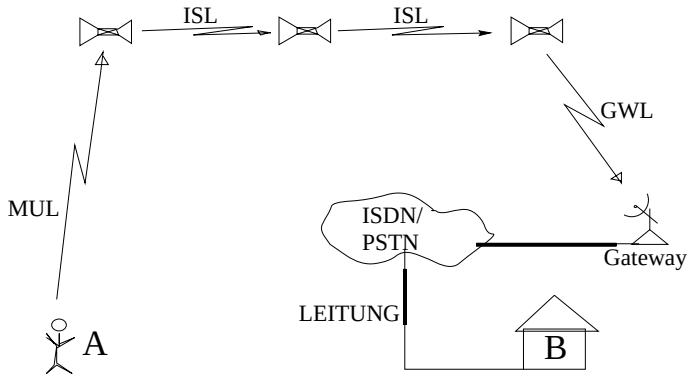

Abbildung 1: Verbindungszenario

Ein mögliches Verbindungszenario zwischen einem mobilen Teilnehmer A und einem stationären B stellt Abbildung 1 dar. Im Gegensatz zu terrestrischen Funksystemen, wie zum Beispiel GSM ist bei der Satellitenkommunikation eine Verbindung nur bei Sichtkontakt möglich. Bevor der *mobile-user-link* des Teilnehmers A durch die kurze Sichtbarkeitsdauer des Satelliten zusammenzubrechen droht, wird dieser von einem nachfolgenden Satelliten übernommen, ohne daß A davon etwas bemerkt (*transparentes Handover*).

1.3 Lage der Orbits

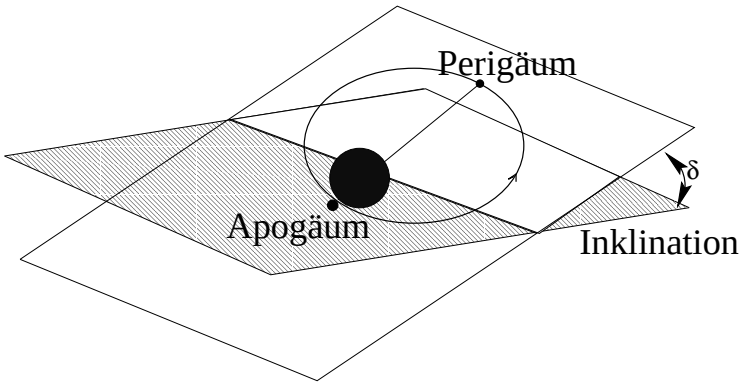

Abbildung 2: Raumgeometrie

Die jeweilige Umlaufbahn des Satelliten um die Erde macht den wichtigsten Parameter für das Systemdesign aus. Man wählt hier meist eine Kreisbahn, wobei die Umlaufdauer dem dritten Kepplerschen Gesetz zu Folge, proportional zu $t^{3/2}$ ist. Auch Ellipsen sind als Umlaufbahnen sinnvoll, wobei hier ein Brennpunkt im Massenmittelpunkt der Erde liegt und die Bahngeschwindigkeit abhängig vom jeweiligen Abstand der Erde, also nicht konstant ist. In jedem Punkt der Bahn ist die Gravitationskraft der Erde gleich der Zentrifugalkraft durch die Eigenbewegung des Satelliten, was die Stabilität

der Bahn garantiert. Der Winkel zwischen der Äquatorialebene und der Ebene der Satellitenbahn wird Inklination genannt (siehe Abbildung 2). Der momentane Erhebungswinkel unter dem ein Satellit sichtbar ist (gemessen bezüglich der Horizontalen) wird als Elevationwinkel bezeichnet.

1.4 Typen von Satelliten

Satelliten können nicht in beliebigen Höhen stationiert werden. Zwischen 1500 und 5000 km , sowie zwischen 13000 und 20000 km liegen die sogennanten *Van Allen Strahlungsgürtel*, in denen durch das Magnetfeld der Erde stark ionisierte Partikel festgehalten werden. Diese zerstören die empfindlichen elektronischen Geräte der Trabanten. Deshalb haben sich die folgenden Orbits herauskristalisiert:

1.4.1 geostationäre Satelliten (GEO)

Sie befinden sich in einer Synchronentfernung von exakt 35.786km, auf einer kreisförmigen Bahn, wo die Umlaufzeit konstant 24 Stunden beträgt. Die Flugrichtung entspricht der Richtung der Erdrotation. Diese Installation hat den Vorteil, daß sie von der Erde aus immer am selben Punkt zu finden ist, und sich ihr Versorgungsgebiet nie ändert. Da für eine Übertragung der Satellit immer von allen Stationen aus sichtbar ist, kann man auf diese Weise ein relativ einfaches Relais für oft benötigte bzw. ununterbrochene Verbindungen der gleichen Stationen verwirklichen. Allerdings kann man auf diese Weise nur sehr kleine Inklinationen verwirklichen, da sich sonst ja die Position zur Erde ändern würde. Die bedeutet aber, eine Konzentration über dem Äquator und somit eine relativ schlechte Versorgung von Gegenden in nördlicheren bzw. südlicheren Gebieten (also auch unseren Breiten). Des weiteren sind keine exakt konstant bleibenden Umlaufbahnen möglich, was bedeutet daß hin und wieder Kurskorrekturen der Satelliten nötig sind, was natürlich Energie kostet. Gerade dieser Faktor macht einen großen Teil der Lebendauer von Satellitensystemen aus, die ja nur eine begrenzte Menge an Antriebsbrennstoff mitführen. Die tolerierbaren Abweichnungen vom theoretischen Standort betragen hier 0,1 Grad oder 73,6 km. Die Hauptvorteile von geostationären Satelliten sind die relativ einfache Installation und Modularität, die in Bezug zu anderen Systemen weiten Ausleuchtungsgebiete, sowie ihre zeitinvariante Topologie und Signalverzögerung. Die Hauptnachteile bestehen aus der kleinen Elevation, die sie für uns nur begrenzt nutzbar macht, sowie die extrem hohe Signalverzögerung von 0,25 sec pro *roundtrip* (das ist die Strecke vom Sender zum Empfänger und zurück), welche für eine effektive Telefon- oder Datenverbindung nicht tolerierbar ist.

1.4.2 *big* low earth orbiter (LEO)

Diese liegen unterhalb der 2 Van Allen Strahlungsgürtel, aber immer noch hoch genug um keine atmosphärische Drift zu erfahren oder unnötig viele Handovers zu erzwingen. Ihre Umlaufbahnen sind wie die der GEOs kreisförmig, allerdings können hier sehr viel höhere Inklinationen realisiert werden. In diesem Bereich wurden bisher vor allem militärische Aufklärungssatelliten zur Beobachtung der Erde betrieben. Die Mobilkommunikation verspricht sich hier vor allem eine kürzere Signalverzögerung (ca.

0,025 sec *roundtrip*) und einfachere Installation von Satelliten. Die Sendeleistung der Bodenstationen kann ebenfalls gesenkt werden, sodaß eine Unterstützung von *handhelds* möglich wird. Beim Ausfall eines einzelnen Satelliten verliert man bei den hier installierten Systemen nicht mehr ganze Versorgungsgebiete wie es bei GEOs zwangsläufig der Fall ist, sondern kann eine *graceful degradation* (gleichmäßige Abnahme der Gesamtleistung des Systems) erreichen, indem die übrigen Komponenten die Aufgabe der Ausgefallenen mit übernehmen. Ergebnisse verschiedener Berechnungen zeigten außerdem, daß LEOs kosteneffektiver arbeiten als GEOs. Nachteilig macht sich an erdnäheren Orbits bemerkbar, daß man für eine globale Bedeckung, aufgrund der kleineren Ausleuchtungsgebiete, erheblich mehr Satelliten benötigt. Diese brauchen des weiteren noch komplexere Steuerungssysteme, da ihre Geschwindigkeiten erheblich höher sind, was sich nicht zuletzt auch in größeren Doppler-Effekt-Einflüssen als bei GSM bemerkbar macht. Durch die kürzere Sichtbarkeitsdauer wird eine recht aufwendige Technik der *handovers* zwischen den Satelliten nötig. LEOs arbeiten im Frequenzband um 2 GHz und sind im allgmeinen schmalbandiger ausgelegt als GEOs, welche meist wegen ihrer analogen Arbeitsweise noch mehr Bandbreite benötigen.

1.4.3 *little* LEOs

Sie sitzen in der gleichen orbitralen Zone, wie ihre *big-brothers*, sind aber kleiner, leichter, billiger, und noch schmalbandiger. Einige von ihnen finden Anwendung bei VHF/UHF-Übertragungen von Amateurfunkern.

1.4.4 *mega* LEOs

Sind die 'Giganten' unter den LEOs, arbeiten im 20/30 GHz-Band, und sind komplexer, breitbandiger, und umfangreicher in der Ausstattung. Zu Ihnen zählen beispielsweise die militärischen Aufklärungssatelliten, welche die niedrige Flughöhe für hochwertige Aufnahmen der Erde ausnutzen, jedoch durch Ihre enorm empfindlichen Geräte oft sehr groß ausfallen.

1.4.5 medium earth orbiter (MEO)

Diese liegen zwischen den zwei Strahlungsgürteln in ca. 10000 km Höhe, wo sie ebenfalls in kreisförmigen Bahnen untergebracht sind. Sie arbeiten im Bereich um 2 GHz und darüber und stellen einen Kompromiß zwischen LEOs und GEOs dar. Ihre Signalverzögerung beträgt ca. 0,15 sec pro *roundtrip*. Sie sind hoch genug um große Gebiete zu versorgen, aber noch nah genug an der Erde um mit relativ geringer Sendeleistung auszukommen.

1.4.6 highly elliptical orbiter (HEO)

Wie ihr Name schon sagt, wandern diese in elliptischen Bahnen um die Erde, was einen sehr großen Unterschied für das Systemdesign bedeutet. Man könnte auch hier von einem Kompromiß zwischen GEO und LEO sprechen, der aber auf eine sehr unterschiedliche Weise realisiert ist. Die Bahngeschwindigkeit ist am erdfernsten Punkt,

genannt *Apogäum*, am kleinsten, was in diesem Zielbereich eine erweiterte Sichtbarkeit bedeutet. Im sogenannten *Perigäum*, dem erdnähsten Punkt nimmt ihre Bahngeschwindigkeit enorm zu, so daß sie schnell wieder im Zielgebiet auftauchen. So differiert ihre Sichtbarkeitsdauer zwischen vier und acht Stunden. Jedoch sind die Signalverzögerungen und -verluste ähnlich hoch wie bei den geostationären Systemen, und es werden ähnlich große Antennen benötigt. Durch die unterschiedlichen Bahngeschwindigkeiten beeinflussen die verschiedensten Doppler-Effekte die Trägerfrequenzen. Die Haltbarkeit solcher Satelliten ist im allgemeinen kürzer als bei anderen, da diese bei jeder Umkreisung der Erde zweimal die gefährlichen Strahlungsgürtel durchqueren müssen und somit einer erhöhten Belastung ausgesetzt sind. Jedoch sind hier sehr hohe Elevationen (55-60 Grad) und damit eine optimierte Versorgung in polarnahen Gebieten möglich. Das System ist flexibler einsetzbar als die herkömmlichen, und kann diese auch sinnvoll erweitern, wie etwa beim geplanten europäischen System ARCHIMEDES [A.Bö95], welches für die Ausstrahlung des digitalen Radios DAB entwickelt wird.

2 Aufbau von Satellitensystemen

Dieses Kapitel widmet sich einerseits den in den Satelliten verwendeten Komponenten, sowie den Grundbausteinen einer Bodenstation. Ein besonderes Augenmerk liegt dabei auf den Antennen, welche für die Überbrückung der extrem langen Wege eine entscheidende Rolle spielen.

2.1 Komponenten eines Nachrichtensatelliten

- Antriebssystem

  Es wird benötigt um vom Aussetzungspunkt auf die jeweilige Bahn zu kommen und diese einzuhalten. Letzteres geschieht mit Hilfe von Kurskorrekturen, die von der Erde aus gesteuert werden.

- Stromversorgung

  Sie wird meist über Solarzellen die mit Pufferbatterien gekoppelt sind gewährleistet. Die Panels werden sofort nach dem Aussetzen des Satelliten ausgefahren und aktiviert.

- Lagestabilisierungssystem

  Es dient mit Hilfe des Antriebssystems zur exakten Ausrichtung des Satelliten beziehungsweise der Antennen in das vorgesehene Gebiet.

- Antenne

  Da man es bei Frequenzen ab 1 GHz mit lichtähnlicher Strahlung zu tun hat, benutzt man quasioptische Richtstrahler, sogenannte *Cassegrain-Antennen* (siehe Abbildung 3). Für die Überbrückung der bei der Satellitenübertragung auftretenden großen Entfernungen werden Antennen mit Doppelreflektoren benutzt, welche eine gute Bündelung des Signals und somit einen hohen Antennengewinn erzielen.

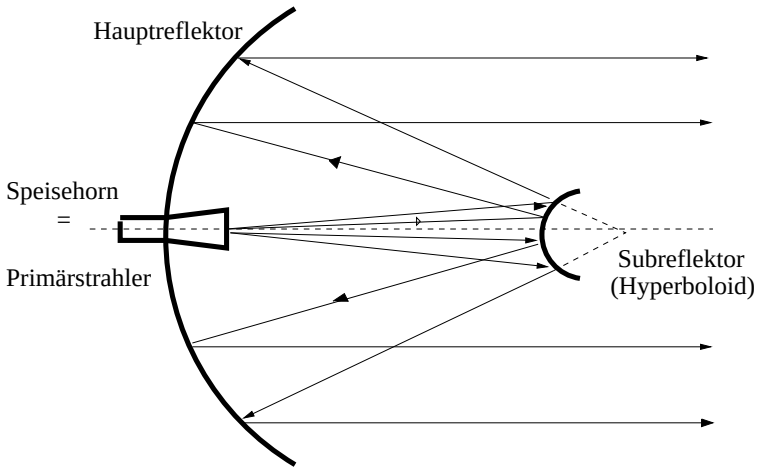

Abbildung 3: Strahlengeometrie der Cassegrain-Antenne

Sie lassen auch eine exakte Ausleuchtung bestimmter Zielgebiete zu. Bei den neueren LEOs denkt man darüber nach, schwenkbare SAT-Antennen zu verwenden, um eine längere Sichtbarkeit zu erreichen. Das Gebiet, welches von einer Satellitenantenne auf der Erde versorgt wird, wird *footprint* genannt.

- nachrichtentechnische Geräte

  Meist werden sogennante Transponder benutzt, welche die Umsetzung von der uplink-Frequenz in die downlink-Frequenz vornehmen. Zur Kopplung vom Sendeweg zum Empfangsweg werden statische oder dynamische Schaltfelder verwendet. Man spricht hier von *satellite swiched TDMA (SS-TDMA)*. Vor der Abstrahlung wird das Signal noch durch Leistungsverstärker auf die nötige Sendeleistung verstärkt. Es wird eine hohe Betriebssicherheit der Geräte benötigt, welche einen langen ununterbrochenen Einsatz von ca. 10 Jahren gewährleisten soll.

- Fernüberwachung / Fernbedienung

  Diese Komponenten sind für die Steuerung und Überwachung von der Erde aus zuständig, und lassen manuelle Kurskorrekturen, sowie eine Umkonfigurierung der jeweiligen Systemressourcen zu.

- Thermalsystem

  Bei Satelliten treten sehr hohe Temperaturschwankungen auf, je nach Exponierung in Richtung Sonne oder im Erdschatten. Deshalb sind besondere thermische Schilde zum Schutz der technischen Geräte nötig. Gegen eine Überhitzung sind dies beispielsweise Reflektoren an der Außenhülle.

- Außenhülle

  Diese muß insbesondere die hohe Belastung beim Start aufnehmen und einen Schutz gegen die ionisierten Teilchen in den Strahlungsgürteln bieten.

2.2 Betrieb von Bodenstationen

Die eingesetzten Geräte in den Bodenstationen sind im wesentlichen zum ersten die Zwischensysteme, welche die Daten von oder zu entfernten Netzwerken liefern. An-

schließend durchläuft das Signal den *Modulator/Demodulator (MOD/DEM)* welcher das zu übertragende Basisband in die Zwischenfrequenz bringt. Auf dem Weg zum Satelliten wird das Signal dann vom Sender auf die Sendefrequenz gebracht und über einen Leistungsverstärker mittels verlustarmen Hohlleitern auf das Erregersystem der Antenne gelegt.

Die hier eingesetzten Verstärker müssen mit den unterschiedlichsten Signalpegeln in einem sehr grossen Frequenzbereich arbeiten. Bis zu einer Sendeleistung von ca. 20 Watt können übliche Halbleiterverstärker eingesetzt werden. Sind jedoch höhere Leistungen gefragt ist es aber nötig, das Signal möglichst verzerrungsarm zu verstärken. Dies gelingt mit sogenannten *Wanderfeldröhrenverstärkern*, welche eine sehr lineare Kennlinie über einem großen Frequenzbereich aufweisen. Diese modular aufgebauten Geräte können auch parallel zur Verstärkung eines Signals betrieben werden, müssen jedoch wegen starker Wärmeentwicklung gut gekühlt werden. Sie arbeiten mit Spannungen um die 12 kV und liefern Sendeleistungen bis 10 Kilowatt. Jedoch sind sie sehr aufwendig und besitzen deshalb auch nur Lebensdauern von einigen tausend Stunden. Deshalb werden in den Bodenstationen zur Sicherheit gleich mehrere Konstellationen von Sendern und Vertärkern redundant betrieben.

Die Antennen sind drehbar gelagert, sodaß sich ein variabler Azimut, von 0 Grad im Süden bis 180 Grad im Norden, sowie eine variable Elevation, von 0 Grad horizontaler Ausrichtung bis 90 Grad senkrecht nach oben, ergibt. Sie erreichen Dimensionen von 30 Metern im Durchmesser, wobei eine maximale Abweichung vom theoretischen Paraboloid in der Konstruktion nicht mehr als +/- 0,5 mm betragen darf. Ihr Gewicht beträgt dann etwa 310 Tonnen, wobei diese Kolosse in jede Richtung bis 0,01 Grad genau einstellbar sind. Die ist nötig, um ein *Schielen* der Richtantennen zwischen Satellit und Bodenstation zu vermeiden. Dazu werden Antennennachführungssysteme benutzt, welche eine Windlast von bis zu 120 km/h aushalten, bei höheren Windgeschwindigkeiten von bis zu 200 km/h müssen die Antennen dann verriegelt werden. Die Nachführung (z.B. *step-track*) geschieht mit Hilfe von vom Satelliten ausgestrahlten Referenzsignalen, welche in den Bodenstationen möglichst naturgetreu empfangen werden sollen. Solche Nachführungssysteme benötigen eine sehr aufwendige Mechanik, um keine Eigenschwingung der Antenne zu erzeugen. Der Gewinn einer solchen Antenne, wie sie schematisch in Abbildung 3 zu sehen ist, berechnet sich nach [TELE96] zu:

$$G = 10 \cdot log \frac{\pi^2 D^2 \eta}{\lambda^2}$$

wobei D den Reflektordurchmesser, η den Antennenwirkungsgrad und λ die Wellenlänge darstellen. Um die Frequenzen doppelt ausnutzen zu können wird in verschiedenen Polarisationen abgestrahlt. Beispielsweise wird zirkulare oder lineare Polarisation verwendet.

Nachdem das Signal im Satelliten von der Uplink- auf die Downlinkfrequenz umgesetzt, und von der entsprechenden Erdfunkstelle aufgenommen wurde, wir es von einer Weiche, dem sogennanten Quadruplexer nach Frequenz und Polarisation getrennt. Danach durchläuft es einen rauscharmen Vorverstärker, welcher das sehr schwache Signal um 40-50 dB anheben muß. Die Eigenrauschleistung dieses Vorverstärkers macht zusammen mit dem Antennengewinn die Empfindlichkeit des Empfangssystems aus. Deshalb setzt man hier für niedrige Frequenzen parametrische Verstärker ein, die auf -273

Grad Celsius gekühlt werden. Für höhere Frequenzen bedient man sich sogenannter Galliumarsenid-Feldeffekttransistor-Verstärker (GaAs-FET).

Die nächste Stufe stellt der Empfangsumsetzer dar. Er setzt das Signal von der Sendefrequenz (GHz-Bereich) auf die Zwischenfrequenz (70/140 MHz) um. Auch diese Umsetzter sind modular aufgebaut, und mit redundanten Geräten gesichert. Abschließend wird das Signal im Demodulator wieder in das Basisband gebracht und an das entsprechende Zwischensystem weitergeleitet.

2.3 Transport zur Umlaufbahn

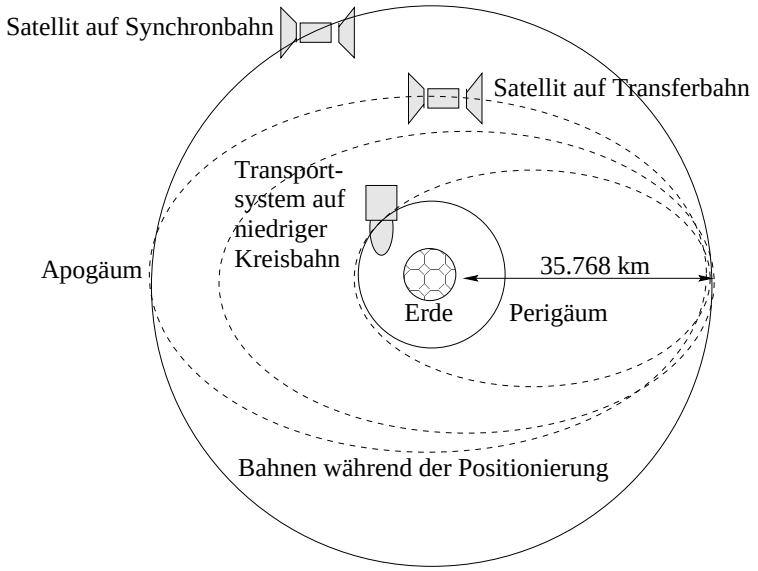

Abbildung 4: Umlaufbahnen des Satelliten während der Positionierung auf die Synchronbahn.

Üblicherweise werden Satellitensysteme mittels eines Trägersystems, welches entweder aus einer Rakete oder einem Space-Shuttle bestehen kann in eine für die Einrichtung des Systems optimale Transferbahn gebracht. Diese Trägersysteme können ein- oder mehrstufig konzipiert sein, je nach der Entfernung, die sie zurücklegen müssen. Beispiele sind das europäische Projekt ARIANE IV bzw. V, welches ein Transportgewicht von bis zu 4500 kg aufnehmen kann. Das amerikanische pendant dazu ist ATLAS I und TITAN III, welches ähnliche Lasten ins All befördert. Führend jedoch sind die Russen mit ihrer ENERGIA, welche bis zu 140.000 kg Nutzlast aufnimmt. Das chinesische Projekt LANGEN-MARSCH hat immerhin eine Kapazität von bis zu 9000 kg. Das Space-Shuttle der Amerikaner transportiert bis zu 23.000 kg in eine Operationshöhe zwischen 200 und 1000 km Höhe. Es bietet den großen Vorteil, daß mit seiner Hilfe sowohl Satelliten ausgesetzt, eingefangen, oder auch nur repariert werden können. Gestartet wird meist von Stationen in der Nähe des Äquators aus, wie zum Beispiel Kourou oder französich Guayana, und zwar möglichst senkrecht zur Erdachse, um die bremsende Atmosphäre so bald wie möglich zu verlassen. Das Aussetzen des Systems erfolgt somit im Perigäum der elliptischen Transferbahn. Kurz nach dem Aussetzen werden die für den Betrieb nötigen Solarzellen ausgefahren und in Betrieb genommen. Dann wird in einem Zyklus von 3 bis 5 Erdumläufen, nach der exakten Berechnung der aktuellen Bahn, durch zünden des bordeigenen Antriebs die vorgesehene Umlaufbahn

immer mehr angenähert. Vor der ersten Inbetriebnahme werden zahlreiche In-Orbit-Tests durchgeführt.

2.4 Funktion von Satellitensystemen

Hier wird die eigentliche Satellitentelematik verdeutlicht, welche Auf- und Abbau von Verbindungen, deren korrekte Weiterleitung, sowie die Aufrechterhaltung von Funkverbindungen bei ungünstigen Verhältnissen umfaßt. Außerdem kommen verschiedene Zugriffsverfahren zur Sprache.

2.4.1 Aufgabenbereiche

In den Bereich *TT&C (Telemetry, Tracking and Command)* fallen Dienste wie Fernmessen beziehungsweise das Senden von Kommandosignalen zur Konfiguration des Satelliten. Ebenso fällt hier das Feststellen der exakten Position des Satelliten im Orbit hinein, welches wichtig ist, um zum Beispiel Antennen nachzuführen oder die genaue Signalverzögerung der Links zu bestimmen. Dagegen ist der Bereich *CSM (Communication System Monitoring)* verantwortlich für die bestimmungsgemäße Funktion der Transponder, sowie für die Durchführung von etwaigen Überprüfungen oder Wartungen der Leitungen oder Bauelementen um eine Fehlereingrenzung oder eine Umkonfigurierung des Systems zu erreichen. Weiterhin fällt die Aktivierung von Ersatzschaltungen in dieses Resort. Bei Bedarf werden Meldungen gesendet, Alarme ausgelöst, oder Meßwerte archiviert.

2.4.2 Benutzte Frequenzbänder

Bei den im Satellitenfunk benutzten Frequenzen handelt es sich wie gesagt um eine lichtähnliche Strahlung, auf deren Eigenschaften bei der Nutzung eingegangen werden muß. Die vom Kurzwellenfunk bekannten Ionosphärischen Störungen und das kosmische Rauschen fallen bei Frequenzen ab 1GHz kaum noch ins Gewicht. Was sich aber ab 20 GHz bemerkbar macht, ist das thermische Rauschen in der Troposphäre. Deshalb sind Systeme mit kleinen Elevationen, deren Signale zwangsweise einen weiteren Weg durch die Troposphäre zurücklegen müssen, in dieser Hinsicht benachteiligt. Oberhalb Frequenzen von 5 GHz wird zusätzlich die Dämpfung von meteorologischen Einflüssen wie Nebel oder Regen zum Problem. Das nutzbare Frequenzspektrum ist daher sehr begrenzt, was eines der größten Probleme der Funkübertragung darstellt. Die Vergabe der Frequenzen ist Angelegenheit der verschiedenen staatlichen Hoheitsbehörden, welche allerdings an die Vorgaben der *World Administrative World Conference (WARC)* gebunden sind. Die Konkurrenz um die Frequenzen besteht nicht nur zwischen den verschiedenen Satellitensystemen, sondern auch zwischen Satelliten- und terrestrischen Systemen, wie beispielsweise dem Richtfunk. So brachte zum Beispiel eine Auktion für ein 120 MHz breites Frequenzspektrum der amerikanischen Bundesbehörde FCC 20 Milliarden Dollar ein. Die benutzten Frequenzbänder unterteilen sich wie folgt:

- L - Band (1.5 GHz downlink / 1.6 GHz uplink)
- S - Band (2/2 GHz)

- C - Band (4/6 GHz)
- Ku- Band (11/14 GHz)
- K - Band (12/14 GHz)
- Ka- Band (20/30 GHz)

Wie schon erwähnt wird der Frequenzknappheit mit Maßnahmen wie Mehrfachnutzung durch verschiedenartige Polarisierung, Wiederverwendung in verschiedenen Ausleuchtzonen oder Antennendiversität begegnet.

2.4.3 Qualität einer Satellitenverbindung

Die globale Bedeckung von Satelliten-Footprints stellt jedoch noch nicht sicher, daß die Funkfeldstärke in allen Bereichen immer gleich hoch ist. Wie schon erwähnt ist eine Verbindung vom Teilnehmer zum orbitalen Repeater durch die sehr schwachen Signale nur bei Sichtkontakt möglich. Die Freiraumdämpfung über die extrem langen Verbindungswege ist für diese schwachen Signale verantwortlich. Um die Versorgungsqualität bestimmter Systeme messen zu können, wird die reale Umgebung in verschiedene typische Bebauungsformen, wie zum Beispiel offenes Gelände, Wald, Stadt usw. unterteilt. Jedes dieser Gebiete hat einen anderen mittleren Abschattungsgrad, welcher für die lokalen Signalschwankungen (Fading) verantwortlich ist. INMARSAT (International Maritime Telecomunication Satellite Organisation) sieht für Problemzonen wie Gebäudeinnenräume eine kurzfristige Erhöhung der Sendeleistung des Satelliten INMARSAT-P um das tausendfache vor (*high penetration service*), um die wichtigsten Daten noch übermitteln zu können. Eine andere Möglichkeit, die Verfügbarkeit zu erhöhen ist die Ausnutzung von Satellitendiversität (Mehrfachsichtbarkeit). Das bedeutet, ein *transparentes handover* auf einen anderen sichtbaren Satelliten einzuleiten, falls die Verbindung zum momentan Benutzten abreißt. Ein weiteres Problem sind Überlagerungen des direkt empfangenen Signals mit einem großen, um beliebige Zeitabschnitte verschobenen Spektrum von reflektierten Kopien. Diese müssen vom Satelliten erkannt und herausgefiltert werden, was eine nichttriviale Aufgabe darstellt.

2.4.4 Zugangsverfahren

Bei analogen Basisbandsignalübertragungen wird zur Modulation fast ausschließlich die Frequenzmodulation (FM) verwendet, da sie relativ unempfindlich gegenüber hochfrequenten Störungen ist. Bei Übertrgungen von digitalen Basisbändern bedient man sich jedoch lieber der Phasensprungmodulation (4/8 PSK) da hier bei gleicher Bandbreite höhere Bitraten möglich sind. Jedoch ist diese Art der Modulation störanfälliger. Bei der Verwendung eines speziellen Zugansgsverfahren spielt die verwendete Modulationsart eine wichtige Rolle. Nicht alle Modulationsarten lassen sich mit allen Zugangsverfahren koppeln. Die Natur der Satellitenkommunikation verhindert, daß die Bodenstationen vor der Sendung den Kanal mithören können. Daher versagen bekannte Zugangsverfahren wie *carrier sense multiple access (CSMA)*. Ein Verahren, welches eine leichte Filterung der Signale zuläßt und mit relativ kleinen Antennen auskommt ist Frequenzmultiplexing (*frequncy division multiple access, FDMA*). Hierbei

wird aus einem vorgegebenen Kanalraster eine Trägerfrequenz ausgesucht und mittels FM moduliert. Jedoch ist die Ausnutzung der Bandbreite hier nicht sehr hoch, und es müssen zwischen den Kanälen zusätzlich Schutzbänder eingehalten werden um die Signale sauber trennen zu können. Eine andere Variante des Mehrfachzugriffs bietet das Zeitmultiplexing (*time division multiple access, TDMA*). Hier wird die Sendezeit in sogenannte Zeitschlitze aufgeteilt und entweder statisch oder dynamisch an die sendewilligen Stationen verteilt, welche im jeweiligen Zeitschlitz die gesamte Bandbreite des Kanals zum Senden zur Verfügung haben. Dabei ist innerhalb eines TDMA-Rahmens eine exakte Synchronisation der verschiedenen Stationen erforderlich. Referenzstationen überwachen dabei die Einhaltung des Zeitrahmens und steuern den Erstzugriff. Dies bedeutet einen erheblichen technischen Aufwand wegen der verschiedenen Signallaufzeiten. Bei neueren Satelliten kommt ein Zugriff zum Einsatz, der FDMA und TDMA verbindet und *multifrequency TDMA (MF-TDMA)* genannt wird.

Bei herkömmlichen PCM Sprachkanälen wird jedem RF-Träger ein Kanal aufmoduliert *(single channel per carrier SCPC)*. Zusätzlich kann dabei das sogenannte *digital speech interpolation (DSI)* zur Anwendung kommen, welches die Ausnutzung der Bandbreite um biszu 60 Prozent steigert, indem es in Sprechpausen den RF-Träger entweder abschaltet oder anderen aktiven Verbindungen zur Verfügung stellt.

Beim Codemultiplexing *(code division multiple access, CDMA)* senden alle Stationen zur selben Zeit auf derselben Frequenz und nutzen so gemeinsam die gesamte Bandbreite des Kanals, welche weit über der eines einzelnen Nutzkanals liegen muß. Der Sender verknüpft das Nutzsignal mittels XOR mit einer Pseudozufallszahlenfolge und spreizt es über die gesamte zur Verfügung stehenden Bandbreite auf(*spread-spektrum technology*). Der Empfänger kann mithilfe der bekannten Pseuodzufallszahlenfolge und einer Korrelationsfunktion das Originalsignal eines einzelnen Senders rekonstruieren. Der Hauptvorteil dieses Verfahrens liegt in der Größe des zur Verfügung stehenden Coderaumes 2^{32} im Vergleich zum Frequenzraum (ca.2^7 bei GSM). Der Nachteil liegt in den sehr komplexen Berechnungen, die für diese Methode nötig sind. Eingesetzt wird es zum Beispiel bei GLOBALSTAR und ODYSSEY [A.Bö95].

Eine weitere Möglichkeit des Zugriffs besteht in der dynamischen Reservierung von Bandbreite, wie es beim nachfragegesteuerten Multiplexen *(demand asignment multiple access, DAMA)* üblich ist. Diese Methode eignet sich vor allem für Hochgeschwindigkeitskanäle mit sehr burstartigem Verkehr. Dabei findet entweder eine explizite oder eine implizite Reservierung von Zeitschlitzen *(timeslots)*, in sogenannten *minislots*, zu Beginn der Übertragung statt. DAMA wird häufig zur Effizienzerhöhung zusammen mit MF-TDMA/SCPC verwendet.

2.4.5 Routing von Verbindungen

Der *mobile user link (MUL)* stellt die Funkverbindung zwischen dem mobilen Teilnehmer und dem Satelliten dar (siehe Abbildung 1). Zur Kommunikation zwischen den Satelliten, um zum Beispiel eine Übergabe eines Gespräches *(handover)* zu ermöglichen, werden die sogenannten *inter-satellite-links (ISL)* benutzt. Da diese jedoch eine sehr komplexe Antennenausrichtung und Lagestabilisierung benötigen, welche dann auch mehr Treibstoff verbraucht, wird an eine Implementierung von ISLs momentan nur bei IRIDIUM nachgedacht. Die Übertragung vom Satelliten zu terrestrischen Bodenstationen erfolgt über *gateway links (GWL)*. Gespräche aus dem terrestrischen Netz werden

über Ausscheidungsziffern an ein Gateway herangeführt. Diese Gateways führen auch das Routing durch, welches sich ähnlich wie bei GSM gestaltet. Jeder Teilnehmer wird in einem *home location register* (HLR) geführt. Daneben existiert ein Register, wo der zuletzt registrierte Standort des Benutzers eingetragen wird, das sogenannte *visited location register* (VLR). Darüberhinaus ist ein Register mitzuführen, welches die momentan vom Benutzer aus sichtbaren Satelliten enthält (SUMR) und gegebenenfalls ein Handover einleitet.

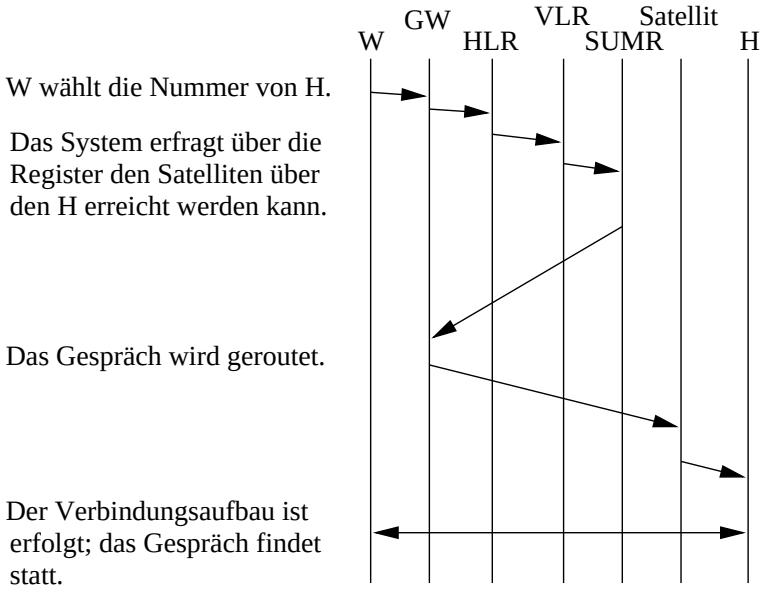

Abbildung 5: Signalisierinformation beim Aufbau einer Verbindung.

Bei einem Anruf an den mobilen Teilnehmer, wird vom Gateway seine momentane Position über die Register HLR, VLR und SUMR angefordert und das Gespräch zum entsprechenden Footprint eines Satelliten weitergeroutet. Um die Kosten des Gesprächs niedrig zu halten, wird es erst möglichst spät in ein Fremdnetz eingespeist, man spricht hier von zielnaher Übergabe von Verbindungen. Jedoch müssen in einem Satellitensystem ohne ISLs genügend Gateways zur Verfügung stehen, damit jeder Satellit stets Kontakt zu mindestens einem dieser Übergabepunkte hat. Wenn das Gespräch mittels ISLs erst einmal rund um die Erde geroutet wird, ergeben sich untolerierbare Verzögerungszeiten, was auch bei Systemen mit ISLs bedeutet, daß mehrere Gateways zum Einsatz kommen müssen. Die sich ständig ändernde Topologie und die damit verbundenen Handover sind fast ausschließlich durch die Bewegung der Satelliten bestimmt, sodaß die etwaige Mobilität der Benutzer kaum ins Gewicht fällt.

3 ATM Satellitennetze

3.1 Architektur

ATM bietet die Grundlage für Breitband ISDN (B-ISDN) und wird weltweit als die Netzwerktechnologie der Zukunft gehandelt. Mittels seiner flexiblen Bandbreitenzuteilung integriert es die verschiedensten Dienste wie Sprache, Video und Datendienste mit den verschiedensten Anforderungen an die Übertragung in ein Netz. Außerdem kommt

es bei der LAN/MAN Anbindung zum Einsatz. Durch die zusätzliche Flexibilität in der Bandbreitennutzung und der Broadcastfähigkeit von Satelliten wird es in seiner Funktion durch deren Einsatz optimal erweitert.

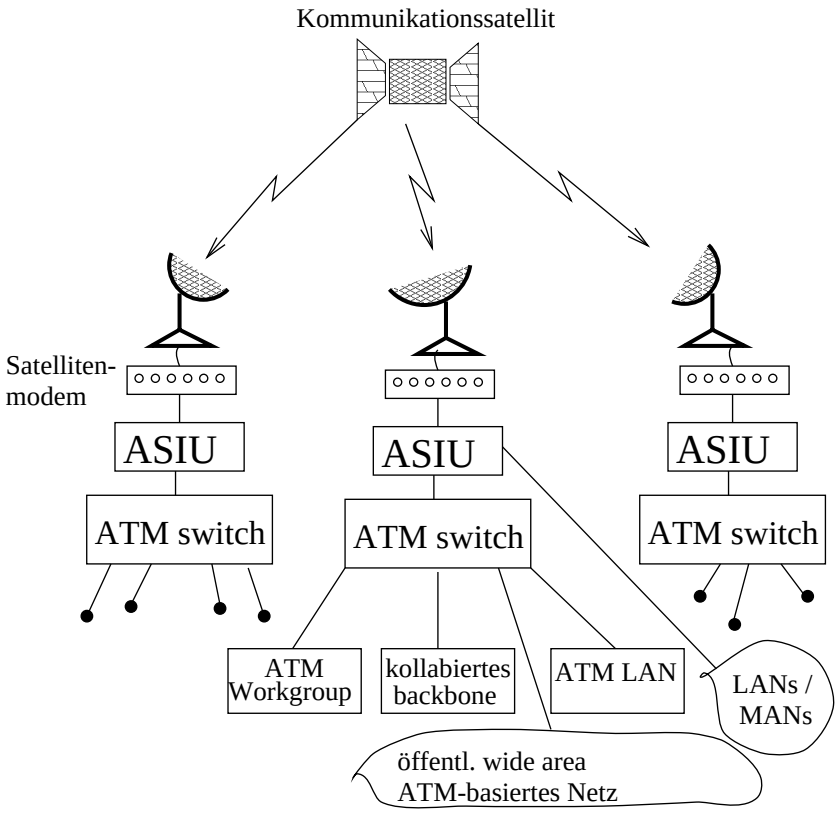

Abbildung 6: Netwerkarchitektur eines ATM-Satellitennetzwerkes

3.2 ASIU

Bei der Anbindung von ATM an Satellitennetzwerke stellt nach [Akyi97] die *ATM satelite interworking unit (ASIU)* die zentrale Komponente dar. Diese ist verantwortlich für die Verwaltung der Systemressourcen und die Administrierung des Gesamtsystems. Initiale Aktionen, wie Bandbreitenreservierung, Zugangskontrolle, Synchronisationen und Anrufüberwachung fallen ebenso in ihr Metier, wie die Fehlerkorrektur und Flußkontrolle auf den unteren Schichten. Um eine Integration in die vorhandene ATM-Welt zu ermöglichen, muß die ASIU eine Kompatibilität zu den Hierarchien SONET/SDH, PDH und PLCP gewährleisten. Die ATM-Zellströme werden nach Verkehrsklassen und Prioritäten klassifiziert und vor dem Senden in verschiedene Puffer eingereiht. Eine effektive Fehlerkorrektur, wie sie weiter unten beschrieben wird, ist dabei nötig, da die Bitfehlerrate (BER) vom Satellitennetzwerk an das optische Netzwerk angeglichen werden muß. Um ein faires Bandbreitenmanagement zu gewährleisten wird DAMA zur aktuellen Reservierung eingesetzt. Weiterhin bestehen Möglichkeiten zum Bridging und Routing um verschiedenartige Netzwerke zu koppeln. Dazu enthält die ASIU sogenannte *interface units* für die Anbindung an das jeweilige Netzwerksystem, die da wären: FDDI, Token Ring, Ethernet usw.

3.3 Zelltransport

Beim eigentlichen Zellentransport wird SDH gegenüber PDH bevorzugt, da hier ein einfacheres Multiplexen und ein exakterer Takttransport möglich sind. Hierbei muß der Zeiger auf Beginn der Daten im SDH-frame besonders gesichert werden, da sonst Folgefehler bei der Übertragung entstehen können. Dies ist Aufgabe der Fehlerkorrektur, welche sich in der Satellitentechnik mit völlig anderen Fehlerarten als bei Glasfasertechnik beschäftigen muß. Treten bei dieser nämlich hauptsächlich **zufällig verteilte** Fehler auf, hat man es bei der Funktechnik besonders mit **Bündelfehlern** zu tun. Der ATM Zellkopf sieht aber zum Beispiel nur die Korrektur von Einzelbitfehlern vor, und eine n-Bit CRC Checksumme, wie sie auf der AAL-Schicht realisiert ist, kann Burstfehler nur bis zu einer Länge von maximal n Bits erkennen. Deshalb kommt hier ein Konzept zur Anwendung, welches von der Audio-CD her bekannt ist, und mit *interleaving* bezeichnet wird. Dabei werden die Daten nicht seriell, sondern verschachtelt übertragen, sodaß sich Bündelfehler beim Empfänger nur als Einzelbitfehler bemerkbar machen, welche mit den herkömmlichen Korrekturmöglichkeiten zu behandeln sind. Dabei kann wahlweise nur der ATM-Header oder die gesamte Zelle *interleaved* werden. Auf Schicht 2 wird *automatic repeat request (ARQ)* mit *selektive repeat* und einer *go-back-n* Strategie zur Fehlerbehebung eingesetzt. Außerdem wird eine *forward error correction (FEC)* eingesetzt, welche a priori die Fehlerrate minimieren soll.

3.4 Staukontrolle

Da Satellitennetzwerke im allgemeinen mit geringeren Datenraten als Glasfasernetze arbeiten, ist eine Staukontrolle hier besonders wichtig, und muß besonders effektiv und schnell zum Einsatz kommen. Gerade im Hinblick darauf, daß sich die QoS zu denen terrestrischer Netze nicht ändern sollen, muß auf diesen Bereich sehr viel Wert gelegt werden. Beispielsweise werden beim Sicherungsprotokoll HDLC die Sende- und Empfangsfenster an die höheren Kapazitäten der Satellitenfunkstrecke angepaßt, sprich vergrößert. Ebenso müssen andere Parameter, wie *time-outs* oder *trip-delays* angepaßt werden. Sogar ganz typische Verkehrsgewohnheiten bestimmter Dienste werden mit aufwendigem *traffic-shaping* auf die jeweiligen Kanäle zugeschnitten. Als reaktives Verfahren auf eine Stausituation werden Zellen in Abhängigkeit von ihren Prioritäten verworfen, was wiederum ein Problem beim ARQ-Verfahren verursacht. Desweiteren können mit *source quenching* Nachrichten die Verursacher von Staus zum Senken der Datenrate veranlaßt werden. Mithilfe einer *connection admission control (CAC)* wird bei staugefährdeten Systemen a priori geprüft, ob den Anforderungen einer neuen Verbindung genügt werden kann, oder nicht. Somit können Pufferreservierungen versuchen eine lähmende Stausituation gar nicht erst aufkommen zu lassen. Hinzu kommen Bandbreitenregulierungen von den Kontrollzentren aus, die auf der physikalischen Schicht greifen. Als Transportprotokoll ist das meisteingesetzte TCP unbrauchbar, da es zu sehr auf *time-outs* basiert und zu kleine Sende- und Empfangsfenster bietet. Deshalb wurde das effizientere SSCOP Transportprotokoll für den Einsatz mit ATM entwickelt, welches ein 24bit breites Fenster und ein *selective reject* bietet.

ÜBERSICHT								
Syst.	TYP	# SATs	Höhe in km	Sicht-bark.	Freq. (GHz)	Zu-griff	Kanäle	Dienste
Inmar-sat-P	MEO	10	10.354	40	20-30	FDM/ TDM	5300	Tel.Fax.Dat. Pag.Pos
Iri-dium	LEO	66	780	10	1,6	FDM/ TDM	4000	Tel.Fax.Dat. Pag.Pos.
Global-star	LEO	48	1414		1,6	CDMA	2700	Tel.Echtz. Pag.Pos.
Odys-sey	MEO	12	10.354	40	1,6 2.4	CDMA	2800	Tel.Dat.Pag. RDSS
Tele-desic	LEO	840	700	10	20-30	FDM/ TDM	2500	Breitband, Sprache
Archi-medes	HEO	5-6	1000-26.800		20-30	FDM/ TDM		DAB

Tabelle 1: Einige geplante Systeme nach [Abri96]

4 Beispiele geplanter Satellitensysteme

Die Systeme GLOBALSTAR und IRIDIUM, welche hauptsächlich für die Sprachkommunikation entwickelt wurden, bieten zusätzlich auch schmalbandige Datenanwendungen mit einer Übertragungsrate von bis zu 9,6 kbit/s an. Darüberhinaus wird als Nebenprodukt die Positionsbestimmung möglich. INTELSAT (International Telecommunication Satellite Organisation) und EUTELSAT (European Telecommunication Satellite Organisation) bieten flexible Übertragungstechniken, wie zum Beispiel einen Zugang über Satellitenmodems mit Raten von 9,6 kbit/s bis zu 9,312 Mbit/s mittels verschiedener Modulationsverfahren an. TELEDESIC hat sich auf breitbandigere Multimediaanwendungen über ATM spezialisiert, und will unter anderem Sprachübertragung mit Festnetzqualität (16 kbit/s), Videokonferenzen mit 64 kbit/s und Client-Server Dienste mit bis zu 2,048 Mbit/s liefern. Als Endgeräte kommen hierbei etwa Dockingstations für Laptops oder Palmtops in Frage. Wie bei IRIDIUM soll mit Hilfe von ISLs eine gewisse Unabhängigkeit von Bodenstationen beziehungsweise dem Festnetz erreicht werden. Interressant ist die Beteiligung von deutschen Firmen an den verschiedenen Projekten. So unterstützen zum Beispiel Vebacom mit 140 Millionen Dollar, und Siemens mit 32 Millionen Dollar das IRIDIUM Projekt. Die DASA ist bei GLOBALSTAR beteiligt, und die TELEKOM setzt auf INMARSAT-P.

Ein Anwendungsbeispiel aus den USA zeigt heute schon, wie eine mögliche globale Realisierung von Datenübertragung via PC aussehen könnte. Das Projekt *direcPC* [Seid96] benötigt zum Empfang eine 24 Zoll Satellitenantenne, eine Steckkarte für den PC und entsprechende Software. Es bietet eine asymmetrische Simplex-Kommunikation, bei der der Downlink zum Beispiel aus einem 10Mbit/s schnellen Internet-Zugang besteht, während der uplink, also die Verbindung zum Provider auf dem herkömmlichen Weg (Modem, ISDN) erfolgt. Genutzt werden hier die Satelliten des Systems *Galaxy IV*. Es können Dienste in Anspruch genommen werden, wie *realtime audio/video* Übertragungen, *virtual private network* (zum Beispiel zur Softwareverbreitung) sowie eine geplante

Einbindung in das TV-Programm (*interaktive tv*). Dabei sind die anfallenden Kosten vergleichbar mit denen von Kabelmodems.

5 Fazit

Von den hier vorgestellten Systemen werden sich bis zur Jahrtausendwende ca. zwei bis drei in Betrieb befinden. Die besten Chancen werden dabei INMARSAT eingeräumt, da es in großer Höhe mit relativ wenig Eigendynamik und hohen Elevationswinkeln eine sehr hohe Dienstgüte und Verfügbarkeit erreichen wird. Jedoch wird IRIDIUM wahrscheinlich zuerst installiert sein, und den großen Vorteil haben, den Markt aufrollen zu können. Wie schon erwähnt, stellen die Satellitensysteme keine direkte Konkurrenz zu terrestrischen Systemen dar, sondern es wird vielmehr auch darauf ankommen, wie gut sich die einzelnen Systeme in vorhandene Strukturen wie GSM oder DECT einzugliedern vermögen. Wegen der zu erwartenden Leistungssteigerung und Massenproduktion, sowie fallenden Kosten wird der Einsatz von Satelliten in stetig wachsendem Umfang möglich sein. Dabei wird sich ATM allem Anschein nach als Basis für eine schnelle, effektive Satellitenkommunikation etablieren, obwohl hier noch sehr viel *engineering* gefragt ist. Für den Endkunden ist eines sicher: Die Satellitentechnik wird die heutige Mobilfunklandschaft und den Fernsehmarkt soweit verändern, daß es auf jedem Punkt der Erde jederzeit möglich sein wird, jede erdenkliche Information oder jeden erdenklichen Gesprächspartner zu erreichen, ohne auf die Installation von eventuellen terrestrischen Leitungen warten zu müssen. Wem diese ständige Erreichbarkeit zuviel wird, der kann natürlich wie bisher sein Endgerät abschalten.

Literatur

[A.Bö95] A.Böttcher. Mobile Satellitenkommunikation. *Serie in Gateway*, April-Juni,1995.

[Abri96] Farrokh Abrishamkar. PCS Global Mobile Satellites. *IEEE Communications Magazine, 132-136*, September 1996.

[Akyi97] Ian F. Akyildiz. Satellite ATM Networks: A Survey. *IEEE Communications Magazine, 30-43*, Juli 1997.

[Seid96] Lawrence P. Seidman. Satellites for Wideband Access. *IEEE Communications Magazine, 108-111*, Oktober 1996.

[TELE96] TELEKOM. Satellitenfunk - Eine Einführung in Technik und Betrieb bei der Deutschen Telekom. *Unterrichtsblätter Jg.49,460-479*, 9/1996.

Das Transis System

Pinghua Zhou

Kurzfassung

Dieser Artikel beschreibt Transis, ein Kommunikationssubsystem für hohe Verfügbarkeit. Transis ist eine Transportschicht, die zuverlässige Dienste unterstützt. Ihre Besonderheit liegt in der effizienten Implementierung von Broadcastanwendung. Die Basis von Transis ist die automatische Unterstützung dynamischer Mitgliedschaft. Der Mitgliedschaftalgorithmus ist symmetrisch, arbeitet innerhalb des regulären Datenflusses, überwindet Partitionen und Mischen. Die höhere Schicht bietet variable Multicast-Dienste für Prozeßmengen.

1 Einleitung

Transis ist eine Baueinheit der Transportschicht, die mehrere Arten von Diensten für zuverlässige Datenübertragung innerhalb eines Netzwerks beliebiger Topologie unterstützt. Nachrichten werden an mehrere Empfangstellen (Multicast) adressiert. Die Topologie ist in Broadcast-Domains eingeteilt. Den Mittelpunkt dieses Artikels bildet die Datenverteilung innerhalb einer Broadcast-Domain. Transis ist verantwortlich für die Datenübertragung zu allen bezeichneten Empfangsadressen. Transis empfängt Nachrichten vom Netzwerk, behandelt alle Fehlerbehebungen und Konsistenzanforderungen, und überträgt Nachrichten zur Bearbeitung in jedem Prozessor.

Transis basiert auf einem neuen Kommunikationsmodell, das allgemein ist und die charakteristischen Eigenschaften vorhandener Hardware ausnutzt. Das folgende Modell besteht aus einer Prozessoren-Gruppe, die zu einer Broadcast-Domain zusammengefaßt sind. Typischerweise wird eine Broadcast-Domain mit einem LAN korrespondieren. Das Modell nimmt Hardware- und Software-Unterstützung für nicht-zuverlässigen Broadcast (Multicast) in einer Broadcast-Domain an. Der *Lansis-Baustein* bietet den Transis Transportschichtdienst in einer Broadcast-Domain an. Die Broadcast-Domains sind durch unzuverlässige Punkt-zu-Punkt-Verbindungen zu *Kommunikation-Domains* verbunden.

Lansis bietet die folgenden Dienste:

1. **Membership**: unterstützt die Mitgliedschaft von Prozessoren.
2. **Basic Multicast**: garantiert Nachrichtenübertragung zu allen aktiven Stellen. Dieser Dienst überträgt die Nachricht gleich zu der höheren Ebene.

3. **Causal Multicast**: garantiert, daß Übertragungsordnung Kausalität erhält. (Definition siehe unten)

4. **Agreed Multicast**: überträgt Nachrichten in der gleichen Ordnung zu allen Stellen. Es gibt verschiedene Protokolle, um die Agreed-Ordnung zu erreichen. Manche enthalten zusätzliche Nachrichten [MSMA90, PeBS89] nicht; andere enthalten einen zentralen Koordinator [BiJo87, ChMa84]. Hier wurde ein symmetrischer Algorithmus entwickelt, der die Agreed-Ordnung erreicht [ADKM92].

5. **Safe Multicast**: überträgt eine Nachricht, nachdem alle aktiven Prozessoren ihren Empfang bestätigt haben.

Die Hauptvorteile der Transis-Methode sind, daß sie auf nicht-zuverlässigen Kommunikationkanälen arbeitet und eine effektive Anwendung der Netzwerk-Broadcastfähigkeit macht.

2 Anwendung von Broadcast

Verteilte Systems sind heute in den meisten Computing-Umgebungen verbreitet. Bild 1(a) zeigt eine normale verteilte Umgebung, die aus vielen durch Gateways und Punkt-zu-Punkt-Verbindungen untereinander verbundenen LANs besteht.

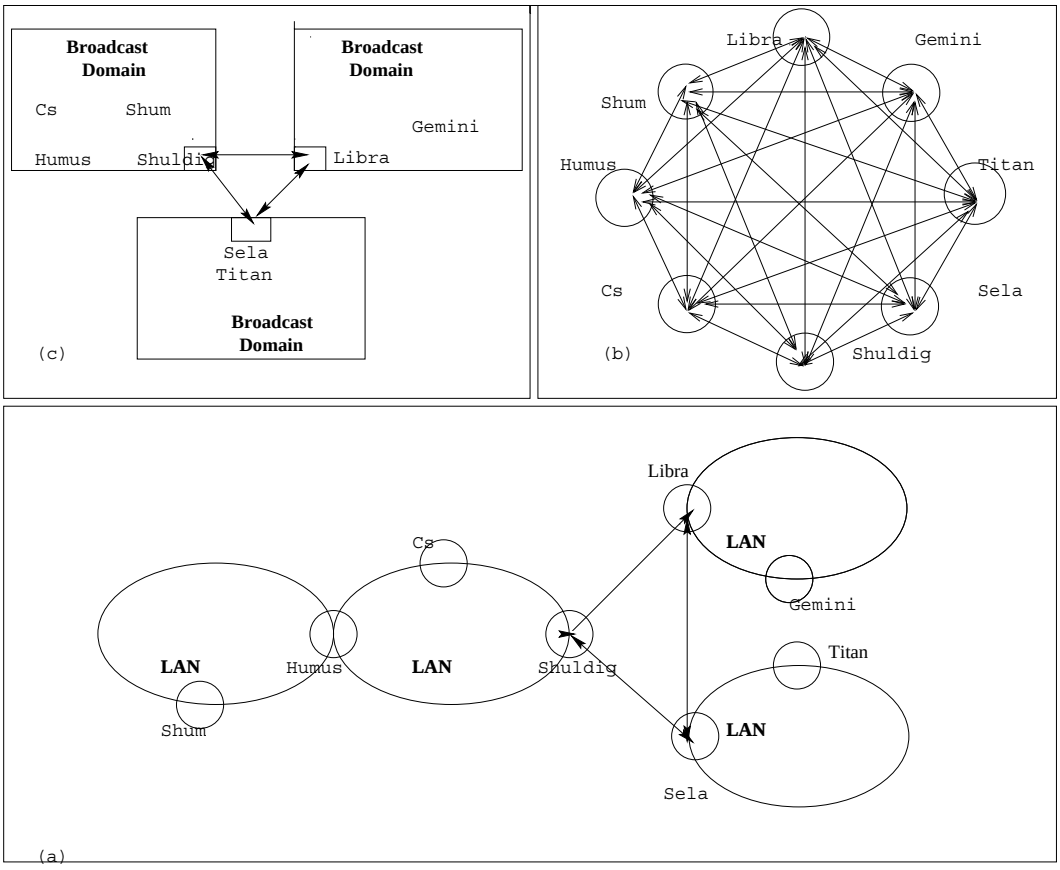

Abbildung 1: Kommunikationsstruktur: Realität und Modell

Die meisten existierenden Systems modellieren ihre Kommunikationsumgebung als eine Prozessorgruppe, die durch Punkt-zu-Punkt-Verbindungen miteinander verbunden sind

(siehe Bild 1(b)). Das Modell hat den allgemeinen Vorteil, die meisten existierenden Umgebungen auf dem Modell graphisch darstellen zu können. Alle lokale Kommunikation wird ausschließlich durch Broadcastmedien (Ethernet, FDDI, etw) erledigt. Die Anwendung von Punkt-zu-Punkt-Multicast verursacht eine starke Vergrößerung der Datenmenge, wenn das unterliegende Kommunikationsystem Broadcastfähigkeit hat.

Die lokalen Gruppen (wie LANs) sind die einzige angemessene Kommunikationstrukturhierarchie. Bild 1(c) zeigt, daß das Systemmodell eine Sammlung von Broadcast-Domains(BD) enthält, die durch (logische) Punkt-zu-Punkt-Verbindungen miteinander verbunden sind. Die BDs entsprechen den physikalischen LANs. Wie das Bild zeigt, kann eine BD mehrere LANs umfassen, die durch transparente Gateways verbunden sind, oder auch nur Teile von LANs.

Die Vorteile der Broadcastnutzung sind: alle erwünschten Eigenschaften von Punkt-zu-Punkt-Protokollen (Zuverlässigkeit, Flußkontrolle, Verbindungsdienste) und mehr werden angeboten in Multicastprotokollen mit besserer Leistung. Der "Lansis" -Abschnitt bietet Leistungsdetails. Nachteile sind die Störung nicht-interessierter Prozessoren und die Protokollkomplexität.

Transis ist eine Transportschicht, die zuverlässigen Multicast unterstützt. In der Transportschicht gibt es keine langfristige Garantie für Nachrichtenübertragungen. Es ist unmöglich, Informationen im sekundären Speicher zu speichern. Deshalb ist Transis verantwortlich für: (a) die Informationsübertragung zu allen derzeit aktiven und verbundenen Prozessoren, (b) pünktliche Benachrichtigung über den Systemzustand und den Erfolg der Übertragung an den Benutzer.

3 Lansis: Verteilte Dienste in einer Broadcast Domain

Die Basisbaustein der Transiseinheit ist Lansis. Lansis arbeitet in einer einzigen *Broadcast-Domain* (BD). Eine BD ist am besten geeignet für ein einziges LAN oder mehrere LANs, die transparent duch Brücken verbunden werden. Lansis kann jedoch unterschiedliche Topologien umfassen.

Die Lansiseinheit besteht aus drei logischen Schichten:

1. **Mitgliedschaft**.
2. **Multicast-Dienste**.
3. **Kommunikationssteuerung und Namensdienste**.

Definition: das CCS (*Current Configuration Set*), das sich dynamisch ändert, besteht aus den aktiven Prozessoren. Die Basisschicht von Lansis, *Mitgliedschaft* (siehe Abschnitt 4), unterstützt automatisch das CCS bei der Einigung zwischen allen CCS-Mitgliedern.

Die nächste Schicht der dynamischen Mitgliedschaft ist verantwortlich für zuverlässige Nachrichtenübertragung innerhalb des CCS. Sie überträgt sowohl reguläre Nachrichten

Dienste-Typ	Index der Synchronität
Safe	n
Agreed	$n/2+1$
Causal	1
Basic	1

Tabelle 1: Dienste Hierarchie

als auch spezielle *Konfiguration-Änderungen* zu der oberen Ebene. Lansis unterstützt verschiedene primitive Operationen, die die Übertragung von Multicast-Informationen zu unterschiedlichen Prozessoren koordinieren (siehe unter).

Konfigurations-Änderungsnachrichten werden innerhalb des regulären Datenflusses übertragen. Lansis garantiert Konfigurations-Änderungsereignisse in einer konsistenten Ordnung mit Übertragung von Nachrichten zu allen Stellen. Jeder Prozessor empfängt die gleiche Nachrichtengruppe zwischen jedem Paar von Konfigurations-Änderungsereignissen.

Die obere Schicht ist die Kommunikationssteuerung. Sie bietet die Schnittstelle für mehrfache Benutzerprozesse auf jedem Prozessor. Man beachte, daß nur ein einziger Lansis-Prozeß auf jedem Prozessor durchführt wird. Die Namensdienste in der Schicht erlaubt Nachrichtenadressierung zu Prozeßgruppen, die irgendeine Subgruppe der aktiven Prozesse sein kann. Prozeß-Gruppen werden dynamisch gebildet durch Prozesse, die sie miteinander „verbinden". Ein Prozeß, der sich mit einer Gruppe verbindet, empfängt alle weiteren gesendeten Nachrichten für die Gruppe.

Multicast-Dienste

Das folgende Kommunikationsystemmodell von Lansis ist vollständig asynchron und nimmt beliebige Kommunikationsverzögerungen und Verluste an. Deshalb kommen Nachrichten zu verschiedenen Zeiten und in verschiedenen Reihenfolgen zu verschiedenen Prozessoren an. Um die Nachrichtenübertragung zu verschiedenen Stellen zu koordinieren, bietet Lansis unterschiedliche Multicastdienste.

Die verschiedenen Dienste, die auf dem Protokoll verhängen, können durch die Verzögerung rangiert worden sein.
Definition: der *Index der Synchronität* ist die Anzahl von Prozessoren, die den Nachrichtenempfang bestätigen müssen, bevor das Protokoll die Nachricht zu der oberen Ebene übertragt. Tabelle 1 präsentiert die Lansisdienste und ihre Synchronitätsindizes. n ist eine Variable, die die sich ändernde Größe von der aktuellen Konfigurationsgruppe bezeichnet.

Basic Multicast

Der Basic-Multicast ist der elementare Dienst. Er garantiert die Nachrichtübertragung zu allen aktiven Stellen. Das bedeutet, daß die Stellen die Nachricht empfangen, die in einem Zeitbereich um die Nachrichtensendezeit aktiv sind.

Jeder Prozessor, der eine Basis-Nachricht empfängt, überträgt sie direkt zu der oberen Ebene. Deshalb ist der Synchronitätsindex des Basis-Dienstes 1 (einschließlich des sendenden Prozesses).

Causal Multicast

Der Causal-Multicast verbreitet Nachrichten zwischen allen Prozessoren, so daß die *Causal-Ordnung* der Übertragung gewahrt bleibt. Die Causal-Ordnung der Nachrichtenübertragung ist als die transitive Hülle definiert worden:

$$m \overset{\text{Kausalität}}{\rightarrow} m' \text{ wenn } \text{empfangen}_q(m) \rightarrow \text{send}_q(m') \tag{1}$$

$$m \overset{\text{Kausalität}}{\rightarrow} m' \text{ wenn } \text{send}_q(m) \rightarrow \text{send}_q(m') \tag{2}$$

Man bemerkt, daß $\rightarrow$ vorkommende Ereignisse an Prozessor q reihenfolgenen ordnet, und deshalb die Ordnung zwischen ihnen wohldefiniert ist. Das Causal-Multicast-Atom garantiert, daß für jeden Prozessor p, der beide von ihnen empfängt,

$$\text{übertragung}_p(m) \rightarrow \text{übertragung}_p(m') \tag{3}$$

wenn $m \overset{\text{Kausalität}}{\rightarrow} m'$. Der Synchronitätsindex ist auch hier 1.

Agreed Multicast

Der Agreed-Multicast überträgt Nachrichten in der gleichen Ordnung zu allen überlappenden Stellen. Der Ordnung ist konsistent mit der Causal-Ordnung. Der Unterschied zwischen dem Causal-Multicast und dem Agreed-Multicast ist: der Agreed-Multicast ordnet **alle** Nachrichten. Sie enthält Nachrichten, die gleichzeitig gesendet werden, d.h. es gibt keine Causal-Relation zwischen ihnen. Während die Causal-Ordnung eine Teil-Ordnung ist, muß sich deshalb der Agreed-Multicast auf eine einzige totale Ordnung von Nachrichten festlegen. Man bemerkt, daß eine Mehrheitsentscheidung die Agreed-Ordnung nicht erreichen kann, weil die Umgebung asynchron ist und Fehler vorkommen. Der Agreed-Multicast ist durch den ToTo-Algorithmus [ADKM92] implementiert. Der Synchronitätsindex in ToTo ist $n/2 + 1$.

Safe Multicast

Manchmal ist der Benutzer daran interessiert, daß eine Nachricht von allen Prozssoren empfangen worden ist, bevor sie eine Aktion starten. Der Safe-Multicast bietet die Information und überträgt die Nachricht zu der oberen Ebene nur, wenn alle Prozessoren in der aktuellen Konfigurationgruppe den Nachrichtenempfang bestätigt haben. Trotz Prozessorausfällen blockiert nicht der zuverlässigen Dienst wegen dessen dynamischen Mitgliedschaft. Der Synchronitätsindex ist hier n.

3.1 Lansis auf einem LAN

Die prinzipielle Idee von zuverlässiger Nachrichtenübertragung in Lansis ist nicht neu. Es gibt ein paar neue Aspekte in der Implementierung und der Flußkontrollenbehandlung. Die Besonderheit ist jedoch, wie die Ideen in einer verteilten Dienstegruppe in einer dynamischen Umgebung kombiniert werden.

Das Lansis-Protokoll nutzt die Fähigkeit des Netzwerkbroadcast für die Nachrichtenverbreitung zu mehreren Empfangsadressen durch eine einfache Übertragung aus (wenn eine Multicast-Fähigkeit unterstützt ist, wie in [Deer89], kann Lansis sie dazu ausnutzen, nur die beteiligten Maschinen zu adressieren.). Es gibt verschiedene Multicast-Protokolle, die Broadcast-Hardware ausnutzen, sowie UDP [Post80], und das

IP-erweiterte Multicast-Protokoll [Deer89]. Sie bieten jedoch nur eine best effort Übertragungsleistung, und sind nicht vollständig zuverlässig. Gründe für Nachrichtverluste sind:

1. Hardwarefehler, die durch das Netzwerk verursacht werden.
2. Der fehlgeschlagene Versuch, Nachrichten eines Netzwerks mit hoher Datenrate abzufangen, weil Interrupts verpaßt werden.
3. Das Protokollverhalten verursacht einen Pufferüberlauf in der Software.

Die erste Ursache ist abhängig von einer Technologieverbesserung. Die letzten zwei Gründe werden schwerwiegender, wenn ein neueres, schnelleres Netzwerk eingesetzt wird. Deshalb ist man auf Protokolle angewiesen, die Nachrichtverlust behandeln und den Fluß der Nachrichtverbreitung kontrollieren.

Das Lansis-Protokoll

Das Prinzip des Lansis-Protokolls lautet wie folgt: Nachrichten können durch alle beteiligten Prozessoren abgehört werden. Lansis benutzt ein kombiniertes System von *positiven ACKs und negativen ACKs*, um Nachrichtenübertragungen zu allen Prozessoren zu garantieren.

Jeder Prozessor überträgt Nachrichten mit steigenden Sequenznummern, die als Nachrichtenidentifikator (z.B. P_A sendet $A_1, A_2, \ldots$) dienen. Ein ACK besteht aus der letzten Sequnznummer der übertragenen Nachrichten eines Prozessors. ACKs werden auf Broadcastnachrichten mitgeschickt. Ein Grundprinzip des Protokolls ist, daß jedes ACK nur einmal gesendet werden muß. Die Nachrichten, die aus anderen Prozessoren folgen, bilden eine "Kette" von ACKs (bezeichnet durch Kleinbuchstaben), die vorherige Nachrichten in der Kette implizit anerkennen, wie die Folge:

$A_1, A_2, a_2B_1, B_2, B_3, b_3C_1, \ldots$.

Prozessoren auf der LAN können Nachrichtenverluste erleben. Sie können das erkennen, indem sie die empfangenen Nachrichtsketten analysieren. In der folgenden Kette kann ein empfangender Prozessor erkennen, daß er Nachricht B_3 verloren hat:

$$A_1, A_2, a_2B_1, B_2, b_3C_1, \ldots \tag{4}$$

Der empfangende Prozessor sendet hier ein negatives ACK auf Nachricht B_3 aus, um nochmal die Übertragung von B_3 anzufordern. Die übertragenen Nachrichten werden zur Sicherheit durch allen empfangenden Prozessoren gespeichert. Auf diesen Weise können Wiederholungsübertragungsanforderungen durch irgendeinen Teilnehmer ausgesendet werden. Offensichtlich können diese Nachrichten nicht ewig durch die Prozessoren behalten werden. Der folgende Abschnitt 'Implementierungsaspekte' erklärt, wie die Nachrichtenanzahl für Wiederholungsübertragungen konstant gehalten wird.

Wenn das LAN ohne Verlust läuft, dann legt es eine einfache totale Ordnung der Nachrichten fest. Sobald es Nachrichtenverluste gibt, empfangen Prozessoren nochmalig übertragene Nachrichten, und die ursprüngliche Totalordnung ist verloren. Die mitgelieferte Information dient dazu, daß Benutzer die ursprüngliche Teilordnungen von übertragenen Nachrichten rekonstruieren können.

In Lansis erhält eine neue Nachricht ACKs für alle kausal übertragbaren Nachrichten. Das ist ein wesentlicher Unterschied zwischen Trans und Lansis. Die ACKs in Lansis bestätigen eher die Übertragungsfähigkeit von Nachrichten als ihren Empfang. Deshalb reflektieren sie die Benutzer-orientierte Causal- und Wirkung-Relation *direkt*. In Trans entspricht die Teilordnung nicht der Benutzer-Ordnung von Ereignissen, und resultiert aus der Anwendung des sogenannten OPD-Prädikate auf der Quittungen [MSMA90] bekommen. Ferner wird das Übertragungskriterium in Lansis sinnvoll vereinfacht durch diese Modifikation.

Die Causal-Ordnug ist wie ein gerichteter azyklischer Graph (Englisch: DAG): Die Knoten sind die Nachrichten; die Kanten verbinden zwei Nachrichten, die direkt abhängig von der Causal-Ordnung sind. Der Causal-Graph enthält alle gesendeten Nachrichten im System. Die Prozessoren sehen den gleichen DAG, der ihre unterschiedlichen Ordnungen stufenweise zeigen kann, während er sich entwickelt.

Implementierung der Dienste

Die Multicast-Dienste werden durch übertragene Nachrichten, die im DAG liegen, angeboten. Sie unterschieden sich durch die Kriterien, die entscheiden, wann Nachrichten von dem DAG zu der oberen Ebene zu übertragen sind. Diese Kriterien arbeiten auf der DAG-Struktur, und sie betreffen äußere Einflüsse, wie Zeit, Verzögerung etc., nicht.

Die Kriterien sind wie folgt:

1. Basic: unmittelbare Übertragung.
2. Causal: wenn alle direkten Vorgänger im DAG übertragen worden sind.
3. Agreed: Ein neues Übertragungskriterium: ToTo wurde entwickelt, das im besten Fall eine Verzögerung von $n/2 + 1$ Nachrichten erzielt[ADKM92].
4. Safe: wenn die Pfade von der Nachricht zu den DAG-Blättern eine Nachricht von jedem Prozessor enthalten. Das zuverlässige Kriterum ändert sich automatisch, wenn die Mitgliedschaft sich ändert.

Der Mitgliedschaft-Algorithmus in Lansis wird in einem folgenden separaten Abschnitt beschrieben.

Implementierungsaspekte

Das Transport-Protokoll muß den Wiederholungsübertragungspuffer endlich halten, indem es Nachrichten, welche von alle Prozessoren gesehen wurden, verwirft. Ferner muß der Nachrichtenfluß geordnet sein und sich der Geschwindigkeit der langsamsten Prozessoren anpassen. Auf NACKs zu warten ist nicht optimal. Die Übertragung von verlorenen Nachrichten ist teuer, und das System muß auf verlorene Nachrichten wegen der verspäteten Antwort warten.

Lansis wendet eine neue Methode dafür an, daß Lansis den Nachrichtfluß kontrolliert. Die Methode versucht, Pufferüberläufe zu vermeiden, um mögliche Nachrichtenverluste zu verhindern, und vermindert dann die Geschwindigkeit, wenn Verluste vorkommen. Ein *Netzwerk-Sliding-Window* besteht aus allen erhaltenden Nachrichten, die durch alle Prozessoren immer noch nicht bestätigt wurden. Jeder Prozessor errechnet

das Netzwerk-Sliding-Window von seinem lokalen DAG. Man beachte, daß das Window Nachrichten von *allen* Prozessoren enthält, nicht wie synchrone Protokolle, wie TCP/IP, die nur ihre selbst gesendeten Daten aufbewahren. Die Netzwerk-Sliding-Window legt eine variable Verzögerung für die Übertragung durch die Windowgröße fest, ordnet die minimale Verzögerung bei kleinen Größen an, und verlangsamt sich bis zur unendlichen Verzögerung (blockiert neue zu sendende Nachrichten), wenn das Window eine maximale Größe überschreitet. Das System blockiert jedoch nicht unendlich. Wenn das Window für eine gewisse Zeitspanne blockiert ist, greift der Mitgliedschaft-Algorithmus ein und entfernt fehlerhafte Prozessoren aus der Konfiguration. Sie macht den Window-Block frei und setzt den Nachrichtfluß fort.

Leistung von Lansis

Ein nicht-optimales Transis über einen UDP-Broadcast-Socket über eine 10 Mbit/s-Ethernet zeigt vielversprechende Leistungsergebnisse. Zum Beispiel erreicht es einen Durchsatz von 160 1kB-Nachrichten pro Sekunde in einem Ethernet-Netzwerk von zehn Sun-4-Workstations. Der Durchsatz wird in den Fällen erreicht, wenn alle Teilnehmer Nachrichten gleichzeitig aussenden und alle Nachrichten empfangen. Zum Vergleich, die Übertragungsrate via TCP/IP in eine Richtung zwischen zwei Beteiligten im Netzwerk ist über 350kB/s. Deshalb ist die Leistung von Transis für die Kommunikation von drei oder mehr Maschinen besser als Punkt-zu-Punkt für das gleiche Ziel. Lansis zeigt nur einen langsamen Abfall der Leistung, wenn die Anzahl beteiligter Maschinen zunimmt.

Lansis ist ein nützliches Werkzeug, wenn es vorsichtig benutzt wird. Es ist wichtig anzumerken, daß es Kosten für zusätzliche Kommunikationen und Interrupts mit sich bringt, wenn Nachrichten zu Stationen gesendet werden, welche an diesen nicht interessierten sind. Außerdem gibt es einem Overhead in Komplexität und Codegröße der Transportschicht.

4 Die Mitgliedschaft in Lansis

Jeder Prozessor, der Verbindungen zu anderen Prozessor unterhält, besitzt eine private Sicht der aktuellen Konfiguration, die alle Prozessoren enthält. Die Sicht wird als CCS (*Current Configuration Set*) bezeichnet. Man beachte, daß das nicht eine benutzerdefinierte Prozessorgruppe ist, sondern die aktuelle Zustandinformation aktiver Prozessoren in dem System repräsentiert. Alle Prozessoren in der CCS müssen sich über ihre Mitgliedschaft einigen. Wenn ein Prozessor neu hinzukommt, bildet er eine einzige CCS. Die CCS macht Änderungen durch während ihrer Operation: Prozessoren kommen dynamisch hinzu und treten aus, und die CCS reflektiert diese Änderungen durch eine Serie vom *Konfigurations-Änderungen.*

Das Lansis-Mitgliedschaft-Protokoll erreicht die folgende Eigenschaften:

- behandelt korrekt Partitionen und Mischen.
- erlaubt regulären Nachrichtenfluß und Mitgliedschaft-Änderungen.
- garantiert, daß Mitglieder der gleichen Konfiguration die gleiche Nachrichtmenge zwischen jedem Paar von Konfiguration-Änderungen empfangen.

4.1 Fehler-Behandlung

Der Fehler-Algorithmus(FA) behandelt Austritte aus der aktiven Prozessormenge. Fehlerhafte Prozessoren zu erkennen ist wichtig aus zwei Gründen: erstens ist diese Erkennung wertvoll für die obere Anwendungsebene. Zweitens verhindert sie unendliches Warten auf Antworten von ausgefallenen Prozessoren. Ein spezieller Fall, die Übertragung der Lansis-*Agreed* und-*Safe*-Nachricht könnte blockieren, wenn die DAG Nachrichten aus fehlerhaften Prozessoren nicht enthalten. Um einen Stillstand zu verhindern, werden die Fehler entdeckt und betrachtet. Das Ziel des Algorithmus ist es, Übereinstimmung zu erreichen zwischen der aktiven und verbundenen Prozessormenge über die ausgefallenen Prozessoren.

Der Fehler-Algorithmus wird immer dann initiiert, wenn die Kommunikation mit einem beliebigem Prozessor unterbricht. Jeder Prozessor identifiziert Fehler separat. Ein Prozessor, der einen *Kommunikations-Abbruch* mit einem anderem Prozessor identifiziert, sendet eine spezielle Nachricht aus. Diese Nachricht heißt FA-Nachricht, die den fehlerhaften Prozessor deklariert. Die jeweilige Methode, die Kommunikation-Abbrüche entdeckt, ist abhängig von der Implementierung und irrelevant für den Fehler-Algorithmus. In der Lansis-Umgebung z.B. erwartet jeder Prozessor, von anderen Prozessoren in seiner Mitglied-Gruppe regelmäßig zu hören. Wenn das versagt, versucht dieser Algorithmus den verdächtigten Prozessor durch einen reservierten Kanal zu verbinden. Wenn auch dies versagt, stellt der Algorithmus fest, daß der Prozessor fehlerhaft ist.

Der Fehler-Algorithmus arbeitet innerhalb des regulären Flusses von Transis-Nachrichten. Der Algorithmus verläßt sich auf die Fähigkeit von Lansis, Nachrichten zuverlässig zu übertragen. Weil die FA-Nachrichten *Konfiguration-Änderungen* im System repräsentieren, müssen sie in einer konsistenten Ordnung zu allen Prozessoren übertragen werden. Die wesentliche Schwierigkeit ist es, eine Einigung über die Identität der letzten empfangenen Nachrichten von ausgefallenen Prozessoren zu erreichen.

Um die Einigung zu erreichen, wird die Übertragung von FA-Nachrichten verzögert, bis sich alle verbleibende Prozessoren über *allen* Fehler geeinigt haben.

4.2 Verbindung

Ein voller Mitgliedschaft-Algorithmus muß es erlauben, Prozessoren in die Mitgliedschaft-Gruppe dynamisch aufzunehmen. Typischerweise ist das Problem dann gelöst, wenn ein einzelner Prozessor mit einer existierenden Mitglied-Gruppe verbunden werden kann [MSMA91, MiPS91, Cris88, BiSS91]. Tatsächlich können Prozessoren vorläufig austreten. Wenn sie wieder verbunden sind, vernachlässigen sie die Abtrennung und fahren fort, reguläre Nachrichten zu senden. Eine andere Schwierigkeit entsteht aus Partitionierungen und Wiedervereinigungen des Netzwerks. In diesem Fall gibt es zwei Prozessorgruppen, die miteinander verbunden werden müssen. Schließlich, außer bei Pannen startet das System spontan. Jeder Prozessor kommt wie eine einelementige Gruppe hoch, und dann mischen sich zwei oder mehr Prozessoren in große Gruppen.

In einem praktischen Mitgliedschaft-Algorithmus gibt deshalb es keine Verbindung-Seite und Verbindungsannahme-Seite. Er muß die Verbindung von zwei oder mehr Gruppen mit beliebigen Größen behandeln. Der Algorithmus muß angesichts vorkommender Fehler korrekt arbeiten, während er Prozessoren miteinander verbindet.

Der Verbindungsalgorithmus wird gestartet, wenn ein Prozessor eine 'auswärtige' Nachricht in der Broadcast-Domain entdeckt. Jeder aktive Mitgliedschaft publiziert ihren aktuellen Zustand durch eine *Versuchsverbindung* (engl.: Attempt Join) -Nachricht.

Der Verbindungsalgorithmus arbeitet in zwei logischen Phasen:

Phase 1: Das Ziel der ersten Phase ist, auswärtige Attempt-Join-Nachrichten zu sammeln. Ein Zeitmesser wird eingesetzt, um die Ausführung der Phase zu begrenzen. Die Gruppe aller gesammelter Prozessoren am Ende der Phase 1 wird mit "J" bezeichnet.

Phase 2: In der zweiten Phase legt sich der Prozessor auf die Verbinden-Gruppe J fest. Er sendet eine spezielle JOIN -Nachricht aus, die J enthält. Dem Prozessor wird nur erlaubt, sich einer neuen Verbindung-Gruppe einzugliedern, wenn es 'zuverlässig' ist.

Während der Verbindungsprozedur werden vorkommende Fehler auf ähnliche Weise durch den Fehleralgorithmus behandelt. Da aber nicht alle verbindende Prozessoren zu der gleichen Mitglied-Gruppe gehören, muß die Verbindungsprozedur dynamisch feststellen, welche Fehler in der "vorherigen" Einstellung vorkommen, und welche tatsächlich "nach" der Verbindung vorkommen.

5 Transis

Die *Kommunikation-Domain* (CD) von Transis besteht aus einer Sammlung von Broadcast-Domainen und bietet einen Multicast-Nachrichtenübertragungsdienst durch BDs. Die Verbindung zwischen verschiedenen BDs geschieht durch den Xport-Mechanismus. Dieser Mechanismus bietet einen zuverlässigen, selektiven Anschluß für übertragene Nachrichten zwischen Lansis-Domainen. Der Xport-Mechanismus wird von Transis-Diensten auf beliebigen Topologien unterstützt. Das erlaubt die Partitionierung von einem einzigen LAN in Sub-Domainen, die durch ein Xport-Element miteinander verbunden sind.

Der Grund für die Unterteilung einer CD in BDs ist, daß das Lansis-Protokoll zu schwierig auf einen großen Umgebung sein konnte, weil es erfordert, daß jeder Prozessor alle Nachrichten beobachtet und eine gemeinsame Sicherheit erhält. Deswegen bietet Transis die Multicast-Dienste auf einem größeren System mit mehreren BDs.

Die Verwendung einer Sammlung von BDs statt einer größeren BD, ist günstig mit Blick auf die folgenden Punkte:

- Skalierbarkeit: In einer Sammlung von BDs wird der Nachrichten- und Raumoverhead innerhalb der kleinen Gruppen von BDs gehalten.
- Dienst-Zuschnitt in der System-Struktur: Z.B. könnte es am besten sein, jede BD zu erhalten innerhalb eines physischen LANs, in dem die BD meistens die innere Netzwerke begünstigt. Die äußere Kommunikation außerhalb den LAN wenden die Xport-Mechanismen an.

- Partitionsalternative nach Leistungsberücksichtigung: Es ist z.B. schwierig, das Lansis-Protokoll einzusetzen, wenn ein LAN Prozessoren mit verschiedenen Geschwindigkeiten enthält. Stattdessen können die langsamen Rechner in einer BD gekoppelt sein und die schnellen Rechner konstruieren eine separate BD. Das reduziert die Aufgabe der Flußkontrolle im System auf die logische Verbindung zwischen den zwei Domains.

- Die Anwendungsstruktur kann eine Kommunikationsgruppen-Unterteilung empfehlen. Diese Kommunikationsgruppen werden am besten durch verschiedene BDs bedient. Die Anwendung kann am besten durch eine hierarchische Kommunikation-Domain bedient sein.

  Jede BD ist durch einen logischen Xport-Knoten repräsentiert (der durch einen einzigen oder mehrere Prozessoren implementiert sein kann). Die Xport-Knoten können in beliebiger Topologie verbunden sein. Nachrichten zu Knoten außerhalb der lokalen BD müssen durch die Xport-Schnittstelle übertragen werden.

  Das Causal-Multicast-Protokoll innerhalb einer BD erbringt zwei Aktivitätsarten.

- *Export*: Die Xport-Knoten exportieren die Nachricht selektiv zu allen relevanten BDs. Die Nachricht wird nur einmalig auf jeder Xport-Verbindung gesendet.

- *Import*: Empfangene Nachrichten aus fernen BDs werden lokal stellvertretend für ihren Sender verbreitet.

Hier bleibt das Garantieproblem der Nachrichtenordnung zwischen BDs. Das Problem erfordert globale Information über die alle BDs repräsentierenden Xport-Knoten. Causal- Agreed- und Safe-Ordnungsprotokolle werden zwischen Xporten ausgeführt. Man beachte, daß die Causal-Ordnung in der vollständigen CD einfach durch Erhalt der Kausal zwischen der Xportsgruppen gewährleistet werden kann. Die Agreed- und Safe-Multicast innerhalb einer CD sind komplizierter. Das ist heute ein aktiver Forschungspunkt im Transisprojekt.

6 Zusammenfassung

Heute bieten die meisten Transport-Schichten Punkt-zu-Punkt-Kommunikation oder nicht-zuverlässigen Multicast. Die effektive Implementierung wird durch die Broadcastanwendung erleichtert. Die vorläufige Implementierung über ein heterogenes Netzwerk von Sun-4 und Sun-3 Maschinen zeigt gute Ergebnisse. Bei mehr als drei Maschinen ist seine Leistung schon besser als die von Standard-Punkt-zu-Punkt-Protokollen.

Es wurde gezeigt, daß ohne irgendeine Art von Synchronisation keine Einigung möglich ist. Dieser Algorithmus führt eine dynamische lokale Gruppe ein, auf welche die Einigung gegründet ist. In einigen extremen Fällen können Prozessoren falsch entscheiden, daß ein anderer Prozessor versagt hat; sobald das herausgefunden wurde, läuft das System aber wieder normal.

Der Mitgliedschaft-Algorithmus arbeitet symmetrisch und automatisch. Ein neuer Aspekt ist die Fähigkeit, Partitionen zu verbinden. Der beste existierende Mitgliedschaft-Algorithmus behandelt nur die Verbindung einzelner Prozessoren.

Literatur

[ADKM92] Y. Amir, D. Dolev, S. Kramer und D. Malki. Total ordering of messages in broadcast domains. *Technical Report* Band CS92-9, 1992.

[BiJo87] K. Birman und T. Joseph. Reliable communication in the presence of failures. *ACM Trans. Comput. Syst.*, Feb 1987, S. 5(1):47–76.

[BiSS91] K. Birman, A. Schiper und P. Stephenson. Lightweight causal and atomic group multicast. *TR 91-1192, Dept. of Comp. Sci., Cornell Univ.* Band 91-1192, 1991.

[ChMa84] J.M. Chang und Maxemchuck. Reliable broadcast protocols. *ACM Trans. Comput. Syste.*, August 1984, S. 2(3):251–273.

[Cris88] F. Cristian. Reaching agreement on processor group membership in synchronous distributed systems. *Research Report RJ 5964*, Mar 1988.

[Deer89] S.E. Deering. Host extensions for IP Multicasting. *RFC 1112, SRI Network Information Center*, August 1989.

[DoMa96] Danny Dolev und Dalia Malki. The Transis. *CACM*, 1996.

[MiPS91] S. Mishra, L.L. Peterson und R.D. Schlichting. A membership protocol based on partial order. *In Proc. of the Intl. Working Conf. on Computing for Critical Applications*, Feb 1991.

[MSMA90] P.M. Melliar-Smith, L.E. Moser und V. Agrawala. Broadcast protocols for distributed systems. *IEEE Trans. parallel and distributed Syst.* Band 1, Jan 1990.

[MSMA91] P.M. Melliar-Smith, L.E. Moser und V. Agrawala. Membership algorithmus for asynchronous distributed systems. *In Intl. Conf. Distributed Computing Systems*, May 1991.

[MSMV90] P.M. Melliar-Smith, L.E. Moser und V.Agrawala. *Broadcast protocols for distributed systems.* IEEE Trans. Parallel. 1990.

[PeBS89] L.L. Peterson, N.C. Buchholz und R.D. Schlichting. Preserving and using context information in interprocess communication. *ACM Trans. Comput. Syst.* Band 7(3), August 1989, S. 217–246.

[Post80] J.B. Postel. User datagram protocol. *RFC 768, SRI Network Information Center*, August 1980.

Hochgeschwindigkeitskommunikation im privaten Bereich

Andreas Schmeiler

Kurzfassung

In diesem Beitrag werden Möglichkeiten, Chancen, Einsatzgebiete und Techniken der Hochgeschwindigkeitskommunikation für Privatnutzer betrachtet. Ein Schwerpunkt sind die xDSL-Ansätze, die zusammen mit anderen Ansätzen wie FTTH, FTTC, und HFC untersucht und gegenübergestellt werden. Am Ende dieses Beitrags wird kurz auf einen Modellversuch der Deutschen Telekom AG eingegangen.

1 Einleitung

In den letzten Jahren hat das Internet mit einer nahezu rasanten Geschwindigkeit Einzug in den privaten Bereich genommen. Mit der zunehmenden Popularität des Mediums Internet entwickelten sich auch die im Internet verfügbaren Angebote weiter; auf den Privatnutzer abzielend, entstehen neben den klassischen Informationsangeboten auch Angebote bei denen der Unterhaltungswert eine immer größere Rolle spielt. Damit verbunden ist in immer größerem Umfang die Übertragung von Bildern, Musik, Sprache oder gar Videos über das Internet. Eine natürliche Grenze bei der Übertragung solcher großen Datenmengen stellt die limitierte Bandbreite dar; kritisch ist hier insbesondere die Übertragung zwischen Internet-Provider und privatem Haushalt, da hier in der Regel eher niedrige Übertragungsraten erzielt werden. Mit der anhaltenden Entwicklung des Internet zeichnen sich neue Einsatzgebiete des weltweiten Rechnernetzes ab; die Übertragung auf Abruf von Videofilmen in (mindestens) gewohnter Fernsehqualität, Bild- und Tonkommunikation, Videoshopping oder das gemeinsame Arbeiten an verschiedenen Orten mit dem Internet als Verbindungsmedium sind einige neue Einsatzgebiete; Grundlage hierfür ist aber wiederum eine leistungsfähige, schnelle Anbindung von privaten Haushalten an das Internet. Betrachtet man die bisherige Entwicklung des Internet und die sich damit erschließenden Möglichkeiten, läßt sich schon heute sagen, daß Rechnernetze in naher Zukunft im privaten und professionellen Bereich *das* zentrale Medium sein werden. Die klassischen Medien könnten zum Teil oder ganz verdrängt werden. Eine Schlüsselrolle bei dieser digitalen Revolution werden sicherlich die Technologien spielen, die dem privaten Nutzer einen leistungsfähigen Zugang zu diesen Rechnernetzen ermöglichen. Die folgenden Betrachtungen beschäftigen sich genau mit dieser Problematik; im folgenden werden verschiedene Ansätze und aktuelle Modellversuche betrachtet.

2 Video On Demand

Exemplarisch für die erwähnten neuen Einsatzgebiete des Internet (oder anderer vergleichbarer Netzwerke) soll hier ein spezieller und besonders für den privaten Bereich interessanter Dienst betrachtet werden: *Video on demand. Video on demand* wird oft mit einer elektronischen Videothek verglichen. Der Benutzer wählt aus einem großen Angebot von verschiedenen Videofilmen mit Hilfe seiner Fernbedienung ein beliebiges Angebot aus und der Film beginnt sofort. Der einzige Unterschied zu der Benutzung einer klassischen Videothek besteht darin, daß der Benutzer nicht mehr aus dem Haus zur Videothek gehen muß, um sich eine Videokassette auszuleihen; ein Videorekorder wird ebenfalls nicht benötigt, denn der gewählte Film wird ja direkt zum Benutzer übertragen. Genauso denkbar ist natürlich auch das Abrufen und Anhören von Audio-CD's nach dem gleichen Prinzip; für Video bzw. Audio on demand findet man auch gelegentlich die Ausdrücke *pay-per-view* und *pay-per-hear*.

Aber ist nun Video on demand tatsächlich als eine elektronische Videothek zu verstehen, oder ist das Ganze eher mit einer Situation vergleichbar, bei der ein Benutzer einen Film aus einem von 500 oder 5000 Kabelfernsehkanälen auswählt ? Die Antwort auf diese Frage hängt von der Implementierung des Systems ab. Wenn Video on demand wirklich vollkommen vergleichbar mit einer elektronischen Videothek sein soll, so muss der Benutzer in der Lage sein, den Film anzuhalten, wieder zu starten, vor- und zurückzuspulen, so wie es ihm gefällt. Das bedeutet, daß der Video Provider für jeden Benutzer eine eigene Kopie des Films bereitstellen muss. Sieht man andererseits Video on demand als eine Art verbessertes Fernsehen an, so ist es sicherlich effizienter, daß der Video Provider jedes Video im Abstand von z.B. 10 Minuten auf einem anderen Kanal neu startet. Hier kann der Benutzer nicht interaktiv in den Ablauf des Videos eingreifen, aber es wird ein ähnlicher Effekt erzielt. Ein Benutzer muss maximal 10 Minuten auf den Beginn eines Videos warten; wenn er das Anschauen des Films unterbricht, so kann er später auf einem anderen Kanal den Film weiter ansehen. Möglicherweise sieht der Benutzer einige Minuten doppelt, aber er verpasst nichts. Dieses Prinzip wird auch *near Video on demand* genannt; es bietet das gleiche Potential wie Video on demand zu entscheidend geringeren Kosten, da mehrere Benutzer die gleiche Kopie des Films nutzen. Der Unterschied zwischen Video on demand und near Video on demand ist vergleichbar mit dem Unterschied selbst Auto zu fahren, oder den Bus zu nehmen. Ein typisches Modell für Video on demand oder near Video on demand ist in Abbildung 1 dargestellt. Das Herz dieses Systems ist ein wide area backbone Netzwerk, mit hoher Bandbreite. Angeschlossen daran sind tausende lokale Verteilernetze, etwa Kabelfernsehen- oder Telefonnetze; dieses lokale Verteilernetz reicht direkt bis zu den Privathaushalten. Der angeschlossene Benutzer benötigt nun noch ein spezielles Gerät, eine sogenannte set-top box, die nichts anderes ist, als ein spezialisierter Rechner.

Direkt am backbone angeschlossen, sind einige tausend *information providers*, die den Benutzern außer pay-per-view und pay-per-hear noch andere spezielle Dienste anbieten können, wie etwa home-shopping, Nachrichten, Internetzugang und eine Vielzahl anderer denkbarer Dienste. Eine genauere technische Betrachtung solcher Systeme soll an dieser Stelle nicht erfolgen, vielmehr soll dieses Beispiel ein mögliches Einsatzgebiet von Hochgeschwindigkeitsnetzwerken verdeutlichen.

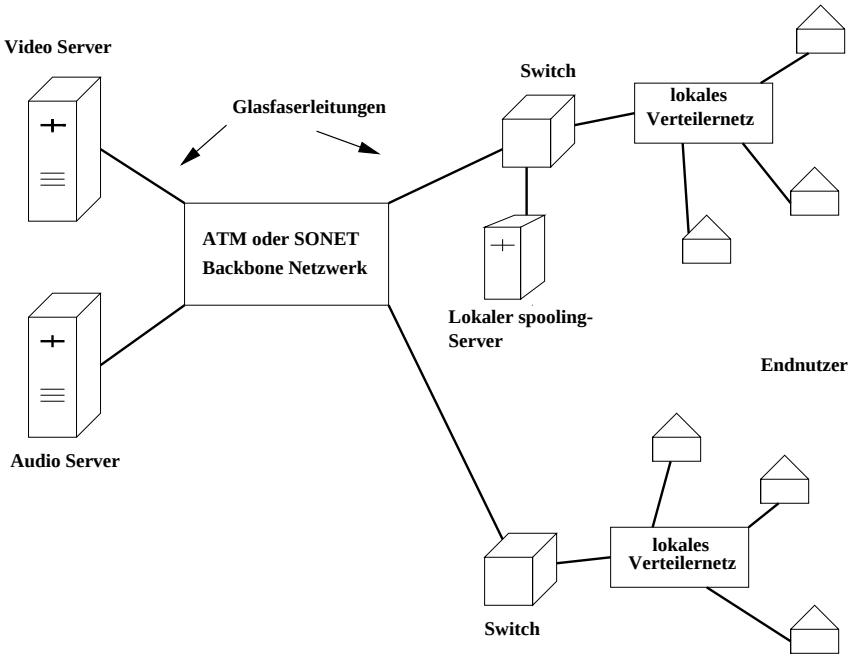

Abbildung 1: Video On Demand

3 Ansätze zur Realisierung von Hochgeschwindigkeitskommunikation bis in Privathaushalte

Schon lange suchen Telefongesellschaften und Kabelfernsehnetzbetreiber nach Möglichkeiten, die Bandbreite ihrer Netze zu erhöhen. Seit Ende der 80er Jahre wurden viele mögliche Varianten diskutiert, aber kaum ein Betreiber solcher Netze hat nennenswerte Investitionen in dieser Richtung vorgenommen. In den frühen 90er Jahren wurde zum erstenmal die xDSL-Technologie als eine mögliche Lösung in Betracht gezogen. Zum damaligen Zeitpunkt räumte man dieser Technologie aufgrund ihrer recht geringen Bandbreite und der hohen Kosten wenig Chancen ein; man konzentrierte sich daher auf Glasfaserlösungen, so wie FTTC (Fiber to the Curb), FTTH (Fiber to the Home) und HFC (Hybrid Fiber/Coax). Mit der rasanten Entwicklung des Internet, die seit 1995 absehbar war, rückten nun wieder die xDSL-Technologien in das Zentrum einiger Überlegungen; diese Technologien haben sich in den Punkten Leistung und Kosten stark verbessert, so daß man diesen Technologien heute eine sehr große Bedeutung beimißt. Im folgenden werden nun FTTC, FTTH, HFC und insbesondere xDSL genauer betrachtet. Weiterhin soll später im Rahmen eines Modellversuches noch kurz eine kabellose HF-Variante betrachtet werden, HFR: Hybrid Fiber Radio.

3.1 Auf Glasfasernetzen basierende Ansätze

Die Ansätze FTTC und FTTH sind Lösungen für Telefonnetze, und arbeiten daher mit Punkt-zu-Punkt Verbindungen. HFC hingegen ist eine Technologie, die auf Kabelfernsehnetzen aufsetzt und diese bestehenden Netze erweitern und für Kommunikationsdienste tauglich machen will. Alle drei Varianten setzen ganz oder zum Teil auf Glasfasernetzwerken auf.

3.1.1 FTTH

FTTH steht für *Fiber to the Home*, gemeint ist eine Technologie, bei der Glasfaserleitungen bis direkt in jeden Haushalt gelegt werden. Eine grobe schematische Darstellung zeigt Abbildung 2.

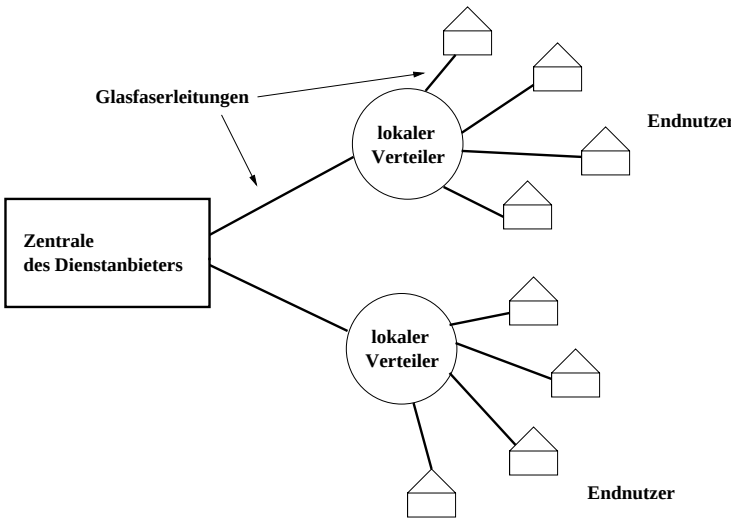

Abbildung 2: FTTH (Fiber to the Home)

Die Vorteile einer solchen Lösung springen sofort ins Auge: Die immense Bandbreite, die dem Anwender hier zur Verfügung gestellt wird, würde fantastische Nutzungsmöglichkeiten mit sich bringen. Der Grund, warum FTTH zum heutigen Zeitpunkt eine eher untergeordnete Rolle spielt, springt ebenso deutlich ins Auge: die Kosten. Mit FTTH verbunden wären erhebliche Investitionen seitens der Netzwerk-Trägergesellschaften, der zeitliche Aufwand bis zum Erreichen einer akzeptablen Vernetzungsdichte wäre ebenfalls sehr hoch, so daß man heute sagen kann, daß FTTH mittelfristig vermutlich keine große Rolle spielen wird.

3.1.2 FTTC

FTTC, wörtlich *Fiber to the Curb* stellt eine Kombination von Glasfaser- und Kupferschleifentechnologie dar. Hierbei werden Glasfaserkabel von einer Zentrale des Netzbetreibers bis sehr nahe an die anzuschliessenden Haushalte gelegt. Die Glasfaserleitungen enden jeweils in einem Gerät, daß als Optical Network Unit, kurz ONU, bezeichnet wird. Ebenfalls angeschlossen an diese ONU werden bis zu 16 herkömmliche Kupferkabelschleifen, also normale Telefonleitungen, wie sie in den meisten Haushalten bereits vorhanden sind. Die ONU stellt damit die Schnittstelle zwischen der existierenden Kupferverkabelung und den Glasfaserleitungen dar. FTTC ist symmetrisch, d.h. dem angeschlossenen Benutzer steht in beiden Richtungen, nämlich eingehend (*downstream*) und abgehend (*upstream*), die gleiche Bandbreite zur Verfügung. Abbildung 3 zeigt eine schematische Darstellung dieser Technik.

Welche Vorteile bringt nun aber eine solche Technik gegenüber der existierenden, klassischen Übertragung über Kupferkabel ? Auf den ersten Blick ändert sich nichts für den Endbenutzer, denn er wird weiterhin über die klassische Kupferverkabelung angeschlossen sein. Ein entscheidender Vorteil wird jedoch dadurch erzielt, daß die für das

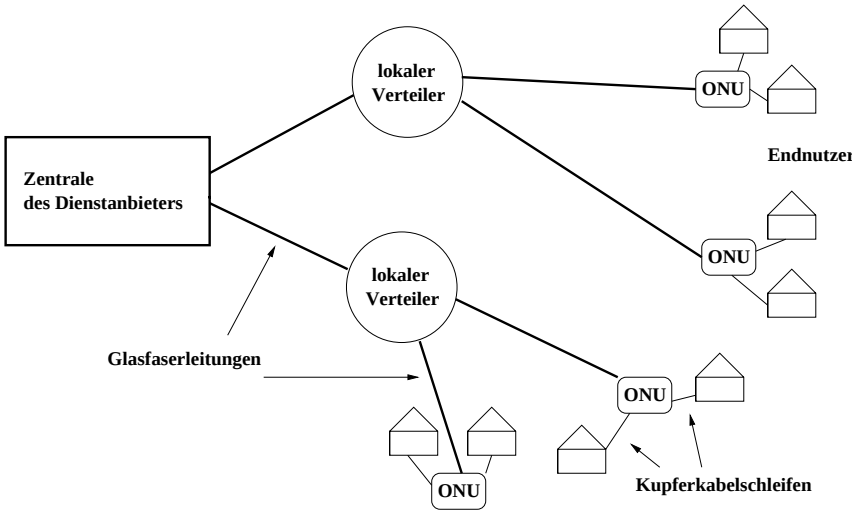

Abbildung 3: FTTC (Fiber to the Curb)

Signal zurückzulegende Wegstrecke über das Kupfermedium erheblich verkürzt wird. Dadurch daß die ONU sehr nah an den Haushalten liegen, werden die Kupferleitungen extrem kurz. Dadurch wird eine wesentlich höhere Übertragungsqualität erzielt; dies führt dazu, daß sich die nutzbare Bandbreite des Kupfermediums wesentlich erhöht; eine naheliegende Idee ist es, die Kupferleitungen mittels (symmetrischem) VDSL zu nutzen (siehe 3.2.3). Mit FTTC lassen sich in beide Richtungen im Vollduplexbetrieb MPEG-1 und sogar MPEG-2 Videostreams übertragen, Video on demand und Videokonferenzen sind somit möglich. Die Kosten für die Umsetzung dieser Technologie würden niedriger sein, als die für die FTTH-Variante, denn der nicht unerhebliche Aufwand jeden Privathaushalt neu zu verkabeln wird eingespart. Trotz allem ist auch diese Technologie noch mit immensen Kosten verbunden, denn die Telefonnetzbetreiber benötigen eine neue Infrastruktur.

3.1.3 HFC

Mit HFC wird eine Lösung für eine schnelle Datenübertragung in Koaxialkabelbasierten Netzen (Kabelfernsehen, abgekürzt CATV) bereitgestellt. HFC steht für *Hybrid Fiber/Coax*, hier werden also Glasfaser- und Koaxialkabelnetze kombiniert. Das Prinzip, das in Abbildung 4 dargestellt ist, sieht folgendermaßen aus: Das existierende Koaxialkabelnetz mit einer Bandbreite zwischen 300 und 450 MHz wird so modifiziert, daß nun eine Bandbreite von 750 MHz zur Verfügung steht. Hierzu werden die existierenden Zwischenverstärker, die die Bandbegrenzung verursachen, umgangen; damit wird nun die volle Bandbreite des Koaxialkabels zwischen dem Hausanschluß und dem ersten Verstärker ausgenutzt. Man erzielt auf dieser sogenannten *last mile* eine Erhöhung der Kanalanzahl; existieren momentan zwischen 50 und 75 Kanäle mit einer Bandbreite von 6 MHz, so erhält man mit der neuen Technik 125 Kanäle gleicher Bandbreite. 75 Kanäle hiervon werden wie bisher für das Kabelfernsehen genutzt, die 50 neu entstandenen Kanäle werden nun mittels QAM-256 moduliert. Jeder dieser Kanäle erzielt Übertragungsgeschwindigkeiten von 40 MB/s, die gesamte neugewonnene Bandbreite beläuft sich damit auf 2 GB/s. Lokale Verteilerstationen werden nun soweit ausgeweitet, daß maximal 500 Anschlüsse an einer solchen Leitung liegen. Eine einfache Division zeigt, daß somit jeder Haushalt eine Leitung mit 4 MB/s zur Verfügung gestellt

bekommt. Dieses Potential kann für MPEG-1 und MPEG-2 Übertragungen, digitale Telefonie und beliebige Datenübertragungen (up- und downstream) genutzt werden.

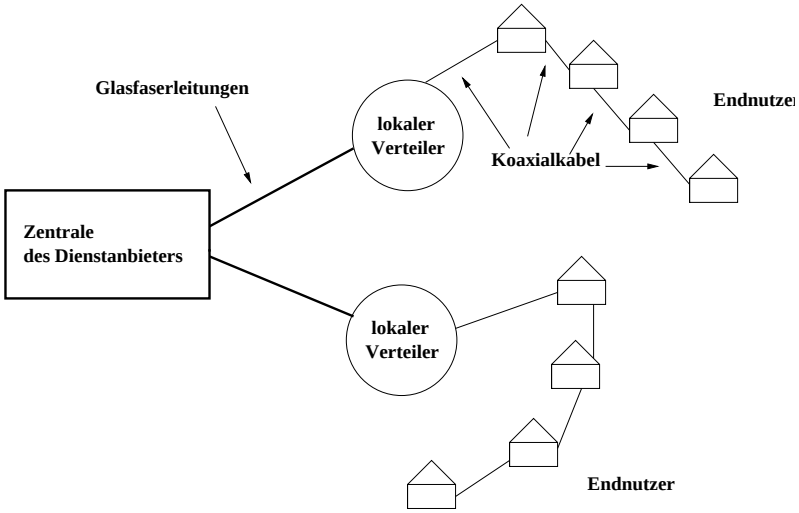

Abbildung 4: HFC (Hybrid Fiber / Coax)

Auffallend sind die Ähnlichkeiten zu FTTC, und tatsächlich lassen sich einige Parall elen zwischen HFC und FTTC ziehen. Der technische Aufwand für die Installation einer für HFC geeigneten Infrastruktur ist absolut vergleichbar mit dem für FTTC zu betreibenden Aufwand. In beiden Fällen werden Glasfaserleitungen bis in relativ kleine Bereiche gelegt, von dort erreicht man den Endnutzer über Kupferkabelpaare oder Koaxialkabel. Der entscheidende Unterschied zwischen beiden Systemen ist die Art der Verwendung des jeweiligen Mediums. HFC verwendet ein geteiltes Medium, ohne switching und routing. Alle Informationen, die auf dem Medium liegen, können von jedem beliebigen Benutzer abgenommen werden. FTTC hingegen kennt dieses Problem naturgemäß nicht, da jedem Benutzer eine eigene Leitung exklusiv zur Verfügung steht. Ergebnis dieser Überlegung ist, daß beispielsweise über HFC betriebene Videoserver verschlüsselte Videoströme senden werden, was auf Kosten der Leistung des Systems gehen wird, über FTTC hingegen ist eine Verschlüsselung in diesem Fall nicht notwendig. Insbesondere bei der Realisierung von Internetanbindungen mittels HFC sieht man sich hier mit erheblichen Sicherheitsproblemen konfrontiert.

3.2 xDSL

Unter dem Oberbegriff xDSL sind mehrere Technologien zusammengefaßt, unter anderem die vier Ausprägungen, die hier näher betrachtet werden sollen: Asymmetric Digital Subscriber Line (ADSL), High-bit-rate Digital Subscriber Line (HDSL), Symmetric Digital Subscriber Line (SDSL) und Very-high-bit-rate Digital Subscriber Line (VDSL). Die grundlegende Idee ist es, die Bandbreite der existierenden Kupferverkabelung voll auszuschöpfen; die Technologien unterscheiden sich aber in Einsatzgebieten und Datenübertragungsraten recht stark; so hängt die einsetzbare Technik unter anderem von der Streckelänge zwischen xDSL-Modem und Server und dem Durchmesser der vorhandenen Verkabelung ab. Da zur Spracherkennung im normalen Telefonbetrieb eine Bandbreite von 4KHz ausreichend ist, werden technisch zur Zeit nur eben genau

diese 4KHz ausgenutzt. Ein Kupferkabelpaar deckt aber grundsätzlich einen Frequenzbereich bis zu 1.1 MHz ab, also das 250-fache der momentan genutzten Bandbreite. Man teilt nun die gesamte Bandbreite in 3 Kanäle ein. Ein Kanal steht wie bisher dem klassischen Telefondienst, auch als Plain Old Telephone Service (POTS) bezeichnet, zur Verfügung. Ein weiterer Kanal, der Upstream-Kanal, dient der Datenübertragung vom Anwender zum Serviceanbieter, der dritte Kanal der Übertragung in umgekehrter Richtung, vom Serviceanbieter zum Nutzer (Downstream-Kanal). Um eine gleichzeitige Nutzung des POTS und der Datenübertragungskanäle zu erzielen, werden die Kanäle vom xDSL Modem voneinander getrennt. Dies ist eine der Aufgaben, die unter dem Schlagwort Advanced Digital Signal Processing zusammengefasst werden, realisiert wird diese Kanaltrennung häufig durch das sogenannte Frequency Division Multiplexing (FDM). Alle xDSL-Techniken benutzen single-carrier oder multi-carrier Modulationsschemata, die untereinander allerdings inkompatibel sind. Standards gibt es noch keine, allerdings sollen hier kurz drei mögliche Lösungen aufgezeigt werden.

- carrrierless Amplitude/Phase Modulation (CAP)
  Dies ist eine spezielle Version der bekannten Quadrative Amplitude Modulation (QAM). Hier wird ein eingehendes Signal durch ein einziges Trägersignal moduliert, das allerdings keine Informationen enthält und daher auch nicht mitüber tragen wird.

- Discrete Multi-Tone (DMT)
  Bei diesem Verfahren werden mehrere Trägersignale für die Übertragung eingesetzt. Die zu übermittelnden Daten werden somit auf etliche verschiedene Trägersignale verteilt, die alle eine Form der QAM einsetzen. Das Discrete Multi-Tone Verfahren basiert auf der bekannten Discrete Fast Fourier Transformation (DFFT).

- Discrete Wavelet Multitone (DWMT)
  Hier wird Multicarrier-Modulation eingesetzt, bei der das Trägersignal durch Wavelet-Transformation aufgebaut wird.

Betrachten wir nun die einzelnen DSL Varianten etwas genauer.

3.2.1 ADSL

ADSL ist die für den Heimanwender interessanteste DSL Technologie, da ADSL noch mit Kupferkabeln arbeitet, die bis zu 5.5 km lang sein dürfen, ohne zwischengeschaltete Repeater. ADSL benötigt eine Kupferdoppelader, so wie sie in fast jedem Haushalt vorhanden ist, und erlaubt sehr hohe Datenraten.

Abbildung 5 zeigt ein grobes Modell einer ADSL Realisierung.

ADSL arbeitet, wie der Name sagt, asymmetrisch; gemeint sind die Bandbreiten des upstream- und des downstream-Kanals. Neben dem Telefonkanal erreicht ADSL Über tragungsraten zwischen 1.5 Mbit/s und 8 Mbit/s auf dem downstream-Kanal und zwischen 64 und 640 Kbit/s auf dem upstream-Kanal. Grafik 6 veranschaulicht den Sachverhalt.

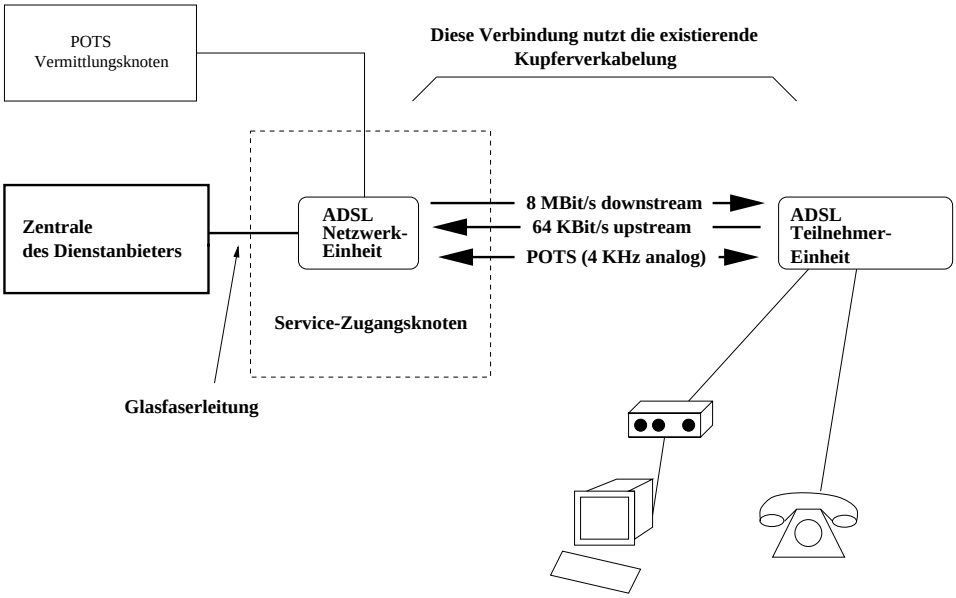

Abbildung 5: ADSL

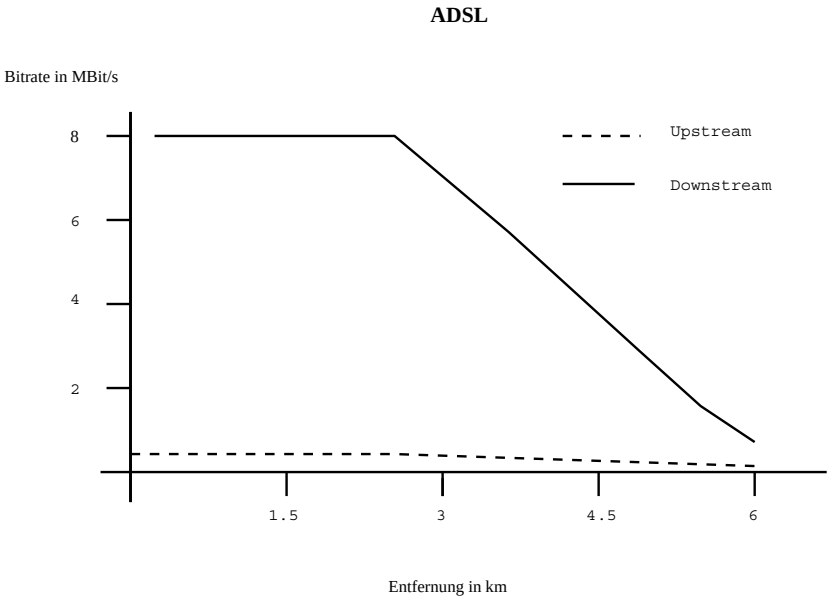

Abbildung 6: Übertragungsraten bei ADSL

Die Übertragungsraten variieren je nach Kabeldurchmesser; die obige Darstellung geht von einem Kabeldurchmesser von 0.5 mm aus, bei einem Durchmesser von nur 0.4 mm liegen die erzielbaren Übertragungsraten etwas niedriger. ADSL eignet sich besonders für Dienste, bei denen große Datenmengen zum Benutzer hin Übertragen werden sollen, etwa Video On Demand oder Homeshopping, aber auch für den Internetzugang ist ADSL äußerst interessant. Nicht geeignet ist ADSL für Anwendungen mit hohen Datenraten auf dem upstream-Kanal, die etwa bei Videokonferenzen oder ähnlichen symmetrischen Anwendungen anfallen.

3.2.2 HDSL, SDSL

HDSL und SDSL sind symmetrische Varianten. HDSL benötigt zwei Kupferdoppeladern, SDSL nur eine. Beide Techniken werden seit 1980 im professionellen Bereich innerhalb von Firmennetzen eingesetzt. Vorteile gegenüber den asymmetrischen DSL Technologien liegen in Bereichen, bei denen in beide Richtungen sehr hohe Datenraten zu übertragen sind. Zwei gravierende Schwachpunkte gegenüber ADSL liegen in den maximalen Übertragungsraten und der höchstmöglichen Kabellänge. HDSL überträgt maximal 1.5 MBits/s in beide Richtungen, VDSL nur die Hälfte: 768 KBit/s. Beide Techniken lassen sich nur in einem Bereich < 2.7 km einsetzen. Für den Anschluß privater Haushalte spielen diese beiden Varianten wohl eine eher untergeordnete Rolle.

3.2.3 VDSL

Eine weitere asymmetrische xDSL Variante stellt VDSL dar. Traumhafte Übertragungsraten von bis zu 51 MBit/s downstream und 6MBit/s upstream würden dem privaten Haushalt eine ganze Reihe neuer Kommunikationsmöglichkeiten erschliessen. VDSL benötigt wie ADSL nur eine Kupferdoppelader, arbeitet aber nur mit Kupferleitungen von maximal 1.5 km Länge. Besonders interessant dürfte VDSL in Kombination mit FTTC sein, denn hier finden sich entsprechend kurze Verbindungsstrecken. Eine Darstellung der Übertragungsraten findet sich in Abbildung 7.

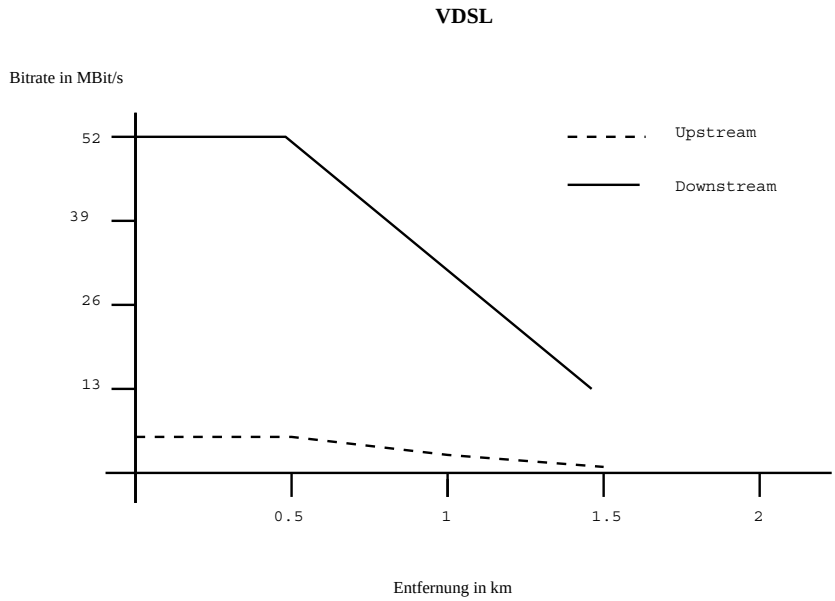

Abbildung 7: Übertragungsraten bei VDSL

Im Gegensatz zu ADSL ist VDSL weit weniger ausgereift, insbesondere die Zuverlässigkeit in Abhängigkeit von den Kabellängen ist noch nicht ausreichend geklärt. Neben der klassischen VDSL Technologie gibt es auch bereits Ansätze symmetrische Übertragungskanäle zu unterstützen (Symmetric-Mode-Transmission), diese Ansätze befinden sich zur Zeit im Experimentierstadium.

4 Modellversuch: Projekt IVES

IVES steht für Interactive Video Experimental System und ist ein Forschungsprojekt der deutschen Telekom AG, das sich mit interaktiver videobasierter Breitband-Datenübertragung beschäftigt. Interaktiv bedeutet hier zunächst einmal nur die Interaktion zwischen Benutzer und Maschine, also einem Videoserver. In erster Linie zielen diese Untersuchungen auf oben genanntes Video on demand ab. Im Rahmen des Projektes wurde exemplarisch ein entsprechendes Netzwerk zu Testzwecken aufgebaut, dieser Aufbau wird im folgenden "Demonstrator" genannt. Im Vorfeld des Projektes wurden folgende Zielsetzungen festgelegt:

- Der Demonstrator soll unbedingt auf erweiterten existierenden Netzwerken aufsetzen, dies hat in erster Linie natürlich ökonomische Gründe.
- IVES soll, soweit möglich, die gleiche Plattform für den upstream und downstream verwenden, um das System für Benutzer und Entwickler möglichst homogen zu halten.
- Das Abrechnungssystem für angebotene Serviceleistungen soll genau wie bei T-Online funktionieren (Abbuchung über die Telefonrechnung), T-Online ist daher in IVES integriert.

Abbildung 8 zeigt den prinzipiellen Aufbau des Demonstrators. Zum Einsatz kommen drei Plattformen, die im Rahmen von IVES beschrieben und verglichen werden sollen, nämlich HFC, HFR und ADSL über gewöhnliche Telefonleitungen. HFR steht für Hybrid Fiber/Radio, also eine drahtlose Übertragungsvariante.

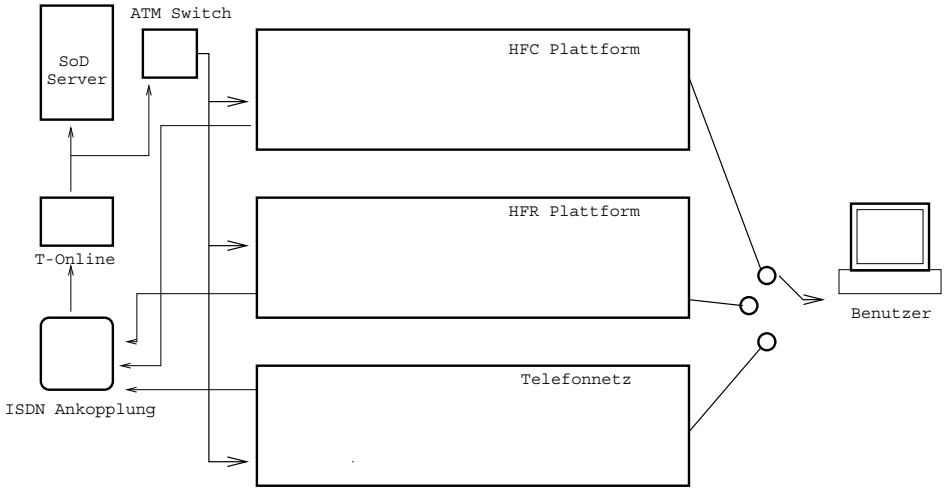

Abbildung 8: Aufbau des Demonstrators

4.1 ADSL Plattform

Abbildung 9 zeigt beispielhaft ein Konzept, das ADSL und glasfaserbasiert einen Hochgeschwindigkeitszugang realisiert. Der Server produziert hierbei ein gemultiplextes downstream Signal, das über lange Strecken (20 - 50 km) durch Glasfaserleitungen läuft.

Am Ende der Glasfaserleitung wird das Signal passiv aufgespalten. Jedes dieser abgespaltenen Signale wird jeweils über eine einzelne Glasfaser zu einer anderen Abteilung, dem sogenannten *Cabinet*, geleitet. In jedem dieser Cabinets teilt ein Demultiplexer den ankommenden Multimedia-Datenstrom in einzelne Kanäle für jeden Endbenutzer auf und führt sie auf die jeweilige ADSL-Einheit. In der ADSL-Einheit werden nun der Multimedia-Kanal und der vom externen Vermittlungsknoten zugeführte Telefon-Kanal gemeinsam auf die Teilnehmeranschlußleitung geführt. In umgekehrter Richtung (upstream) funktioniert dies ganz ähnlich; ein Multiplexer leitet die einkommenden Signale der ADSL-Einheiten zurück zum Aufteilungspunkt, der wiederum über einen Multiplexer die verschiedenen Signale aus den einzelnen Abteilungen zusammenfaßt und über eine Glasfaserleitung an den Server in der Kopfstation weiterleitet.

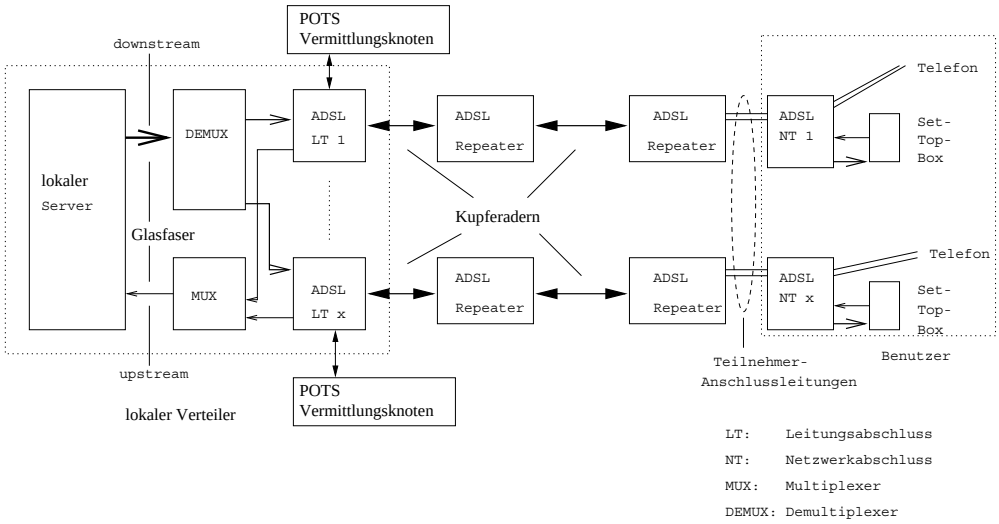

Abbildung 9: IVES ADSL Plattform

4.2 HFC Plattform

Da in Deutschland heute etwa zwei Drittel aller Haushalte an das Kabelfernsehnetz angeschlossen sind, ist eine HFC-Realisierung nicht ganz uninteressant. Das Kabelfernsehnetz ist hierarchisch gegliedert, auf oberster Ebene werden verschiedene Verteilerstationen miteinander verbunden, ganz unten steht die Anbindung der einzelnen Haushalte. Wichtig ist nun die Strecke vom lokalen Verteiler zum Haushalt, die sogenannte *last-mile*. Die Glasfaserkabel laufen vom Netzbetreiberzentrum auf irgendeinem Weg bis zum Ort der sogenannten C-Verstärker, das ist der in der Verstärkerkette letzte Verstärker, und von dort aus via Koaxialkabel in die einzelnen Haushalte. Abbildung 10 zeigt die grobe Strukur des IVES-Demonstrators auf HFC Plattform. Zur Übertragung wird hier das MPEG-2-Format verwendet.

4.3 HFR Plattform

Als letzte Plattform wurde im Rahmen von IVES HFR betrachtet. Diese kabellose Variante hat den Vorteil, daß sie recht einfach und kostengünstig zu installieren ist, denn es ist hierbei nicht notwendig Anschlußleitungen in jeden Haushalt zu legen. Ein

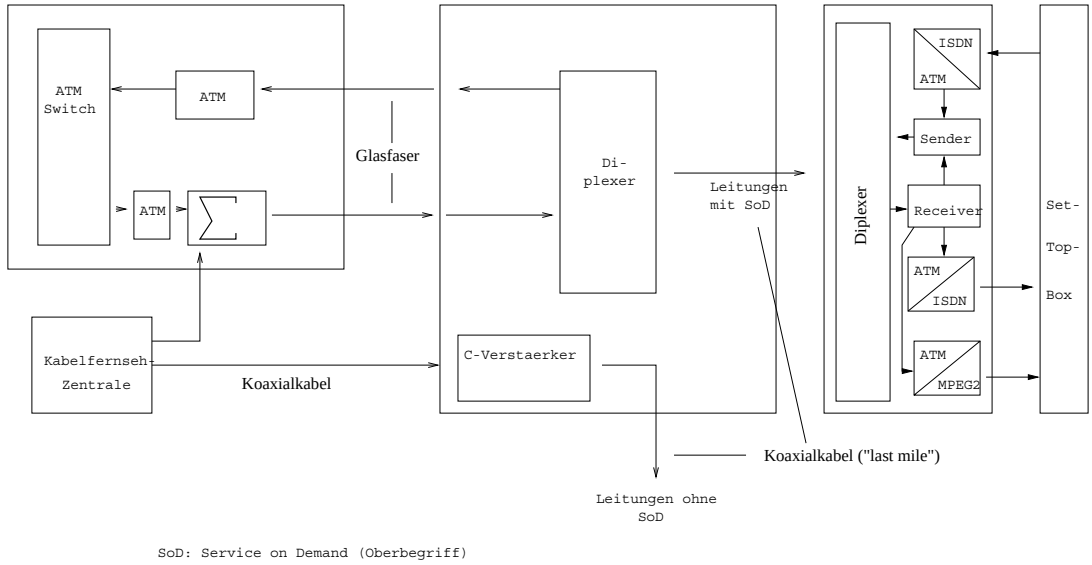

Abbildung 10: IVES HFC Plattform

Nachteil von HFR sind jedoch die derzeit recht schlechten Übertragungsleistungen. Im Gegensatz zu den HFC- und den ADSL-Ansätzen baut HFR nicht auf einem bestehenden System auf. HFR setzt digitales MMDS (Microwave Multipoint Distribution System) ein, eine Technik die dem Digitalfernsehstandard DVB recht ähnlich ist, und es daher erlaubt, Bauteile aus diesem Bereich wiederzuverwenden. Den Aufbau eines HFR basierten Übertragungsnetzwerkes zeigt Abbildung 11. Die Basisstation benutzt zwei Antennen, jeweils eine fuer den Upstream- und den Downstreamkanal, die Station des Benutzers dagegen nur eine.

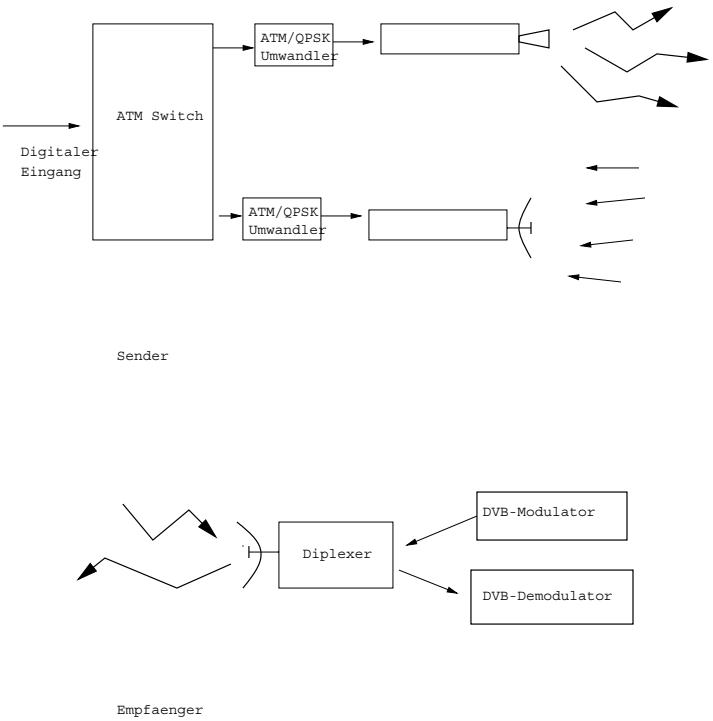

Abbildung 11: IVES HFR Plattform

4.4 Ergebnisse des Projektes IVES

Aus dem Projekt können abschließend einige Folgerungen gezogen werden. Um den Anschluß privater Haushalte an Hochgeschwindigkeitsnetze zu ermöglichen, müssen existierende Netzwerke mit Glasfaserleitungen ausgebaut oder zusätzliche Glasfaserleitungen gelegt werden. Beim Vergleichen der drei Plattformen stellt man folgendes fest: Das Telefonnetz stellt jedem Nutzer einen eigenen Zugang zur Verfügung mit exklusiv zur Verfügung stehender Bandbreite und erreicht eine sehr große Anzahl an potentiell en Nutzern. Nachteilig sind die eingeschränkte Bandbreite und die kurzen Reichweiten der auf dem Telefonnetz aufbauenden xDSL-Techniken. Das Kabelfernsehnetz dagegen hat eine höhere Bandbreite, die allerdings nicht exklusiv genutzt werden kann, da es ein geteiltes Medium ist. Eine Alternative könnten die bisher noch wenig untersuchten kabellosen Übertragungstechniken sein, die vermutlich recht billig und einfach zu installieren sind und eine große Bandbreite bieten. Allerdings gibt es hier noch erhebliche technische Probleme, unter anderem die hohe Störanfälligkeit solcher Systeme.

5 Zusammenfassung und Ausblick

Betrachtet man die derzeitige Situation, so liegt die Vermutung nahe, daß xDSL-Techniken, HFC und FTTC in der Zukunft eine bedeutende Rolle spielen können. FTTH ist sicherlich aus heutiger Sicht eine Technik, die mit unwirtschaftlich hohen Kosten verbunden ist, so daß im privaten Bereich mittelfristig FTTH sicherlich keine Rolle spielen wird. Ob mehrere der übrigen Technologien in Zukunft zum Einsatz kommen werden oder ob sich eine Technologie gegenüber den anderen durchsetzten kann, und welche dies sein könnte, ist aus heutiger Sicht kaum zu beantworten. Interessant ist es in diesem Zusammenhang einmal nach Großbritannien zu schauen, denn dort bieten schon seit 1990 Kabelfernsehnetz-Betreiber Telefondienste über ihr Netz an, mit sehr großem Erfolg: bereits 1997 hatten die Kabelnetz-Betreiber in Großbritannien mehr Telefonkunden als TV-Kunden. Der nächste Schritt, leistungsfähige Internetzugänge auf diese Weise anzubieten, liegt in unmittelbarer Nähe. In Deutschland ist die Situation auf dem Telekommunikationsmarkt sehr unklar, denn einige Rahmenbedingungen sind noch nicht geklärt. Ein nicht unwesentlicher Punkt hierbei ist die Frage, ob die Deutsche Telekom AG in Zukunft Nutzungsgebühren für Telefonleitungen zahlen muss, denn das Verlegen dieser Leitungen wurde zum größten Teil durch Steuergelder finanziert. Eines ist aber mit Sicherheit zu sagen: Der Telekommunikationsmarkt und besonders der Bereich der Hochgeschwindigkeitskommunikation im privaten Bereich, ist ein vielversprechender Markt und schon heute heiß umkämpft. Auch wenn es noch ungewiß ist, welche Technologien sich in Zukunft auf dem Markt behaupten werden, seien es die hier diskutierten oder heute noch unbekannte Technologien, so kann man auf jeden Fall davon ausgehen, daß sich für den Privatnutzer in naher Zukunft vieles ändern wird, und alledem was da kommen mag, darf man gespannt entgegensehen.

Literatur

[Acht97] K. Achtermann. An ATM-based Demonstration Model for Multimedia Services Using Different Access Networks. *European Conference on Multimedia Applications Services and Techniques - ECMAST 97*, Mai 1997.

[Free97] J. Freeman. An overview of xDSL technology and its Implications for High-Speed Networking. *Global Networking 97 Joint Conference*, Mai 1997.

[Schw97a] Guido Schwartzfeld. TV-Kabelnetze für Telefoniedienste. *c't*, Dezember 1997.

[Schw97b] Dr. R. Schwen. Übertragungstechniken auf xDSL-Basis: Sprinter auf kurzen Strecken. *Gateway*, Mai 1997.

Hochleistungsbusse für die Hochgeschwindigkeitskommunikation

Alexander Schulz

Kurzfassung

Die Bussysteme, die heute in PCs verwendet werden, stammen aus einer Zeit, in der noch niemand daran dachte, beispielsweise Videos von der Festplatte abzuspielen. Daher sind diese Busse auch nicht für die Anforderungen an die Bandbreite und an die Dienstgüte ausgelegt, die solche Anwendungen erfordern — sie werden zum Flaschenhals. Um diese Probleme zu lösen, wurden mit der Zeit bessere Buskonzepte entwickelt: zu Anfang EISA, dann verschiedene Local-Bus-Systeme, von denen aber nur der VESA-Local-Bus und der PCI-Bus größere Verbreitung fanden. Inzwischen kommen neue Bussysteme wie Firewire auf den Markt, die nicht nur im Rechner selbst zum Einsatz kommen sollen, sondern auch externe Verbindungen aufbauen und somit ein — wenn auch noch eng begrenztes — LAN bilden können.

1 Einleitung

Im Verlauf der Entwicklung von PCs sind vor allem die Prozessoren immer schneller geworden. Auch die Peripheriegeräte, insbesondere Festplattenlaufwerke und Graphikkarten, aber auch Netzwerkkarten vervielfachten ihre Arbeitsgeschwindigkeit. Zwischen diesen Komponenten wirken als Verbindung immer noch Busse, die in einer Zeit entstanden sind, als man sich unter „Multimedia" noch nichts vorstellen konnte. Insbesondere weisen diese Busse eine geringe Bandbreite auf, was sie zum Flaschenhals werden läßt. Aber auch im Bezug auf die Dienstgüte, also z.B. die Auslieferung der Daten zum richtigen Zeitpunkt und in der richtigen Reihenfolge, sind diese Busse nicht auf heutige Anwendungen vorbereitet. Im folgenden Beitrag soll daher nach einer kurzen Einführung der klassischen Bussysteme der serielle Bus IEEE 1394 — besser bekannt unter dem Namen *Firewire* — vorgestellt werden, der speziell für diese neuen Anforderungen entworfen worden ist.

2 Klassische Bussysteme im PC

Einer der wichtigsten Gründe für die Verbreitung des PC ist sein Slotkonzept. Damit ist es möglich, den PC als Grundsystem zu kaufen, und ihn mit Einsteckkarten zu erweitern. Damit kann man ihn dann für den eigenen speziellen Einsatzzweck optimieren, ohne teure, nicht benötigte Zusatzgeräte mitkaufen zu müssen. Daher ist bei PCs das Bussystem besonders wichtig.

2.1 ISA

Der ISA-Bus, oder Industry Standard Architecture, kam mit der Einführung des PC Anfang der 80er Jahre auf. Wirklich genormt wurde er erst 1990 mit der IEEE Empfehlung P996, so daß bis dahin verschiedene Implementierungen existierten. Diese unterschieden sich insbesondere im Timing, was dazu führte, daß Erweiterungskarten verschiedener Hersteller nicht unbedingt zusammenarbeiteten. Auch heute ist er noch der am weitesten verbreitete PC-Bus und wird zusätzlich zum schnelleren PCI-Bus für Erweiterungskarten mit geringen Anforderungen an die Datenrate — wie etwa Soundkarten — verwendet.

Der ISA-Bus wird prinzipiell mit 8 MHz getaktet und baut auf einem einfachen TTL-Interface auf. Ein Signal schaltet dabei zwischen Zugriffen auf Speicher und Zugriffen auf den I/O-Bereich um. Die unteren 256 Adressen des I/O-Bereiches sind für Komponenten auf dem Motherboard reserviert, die höheren Adressen können von Steckkarten benutzt werden. Dabei werden meist nicht alle Adressleitungen ausdekodiert, sodaß hier nicht der gesamte nutzbare Bereich zur Verfügung steht.

Der Anschluß an den ursprünglich in den ersten PCs mit 8088 oder 8086 Prozessor verwendeten ISA-Bus geschieht über einen 62-poligen Federkontaktstecker, in den Erweiterungsplatinen direkt eingesetzt werden. Der Bus kann mit 20 Adreßleitungen 1 MB Speicher adressieren und erreicht mit acht Datenleitungen eine maximale Datenrate von 4 MB/s, was zur damaligen Zeit als völlig ausreichend angesehen wurde.

Mit dem 80286 Prozessor wurden dem Bus 36 Leitungen hinzugefügt, wodurch ein weiterer Federkontaktstecker erforderlich wurde, der direkt hinter dem alten Stecker angeordnet ist. Der so entstandene AT-Bus kann mit 24 Adreßleitungen 16 MB Speicher verwalten und erreicht mit 16 Datenleitungen maximal eine Datenrate von 8 MB/s. Es können aber auch weiterhin 8-Bit-Karten in einem 16-Bit-Slot verwendet werden. Einige moderne 16-Bit-Karten erkennen sogar, wenn sie nur in einem 8-Bit-Slot stecken und können sich entsprechend einstellen. Neben zusätzlichen Daten- und Adreßleitungen wurden weitere Interrupt- und DMA-Request-Leitungen hinzugefügt. Um die Kompatibilität mit den 8-Bit-Karten zu wahren, kamen noch zwei Signale hinzu, mit denen eine Karte anzeigen kann, daß sie einen 16-Bit Transfer initiieren will.

Zu Anfang wurden die Adressen der Steckkarten mit Hilfe von Jumpern oder Dipschaltern eingestellt, aber mit der Zeit gingen die meisten Hersteller dazu über, die Karten per Software zu konfigurieren. Inzwischen ist die Schnittstelle zu den Konfigurationsdaten weitgehend genormt, so daß die Einstellungen vom Betriebssystem vorgenommen werden können („Plug & Play").

2.2 EISA

Mit der Einführung des 80386 Prozessors und deutlichen Geschwindigkeitszuwächsen bei den Peripheriegeräten, insbesondere bei Festplatten und Grafikkarten, ergab sich die Notwendigkeit, ein 32 Bit breites Bussystem zur Verfügung zu stellen. Zu diesem Zweck entwickelte eine Gruppe von PC-Herstellern den EISA-Bus oder Extended Industry Standard Architecture. Dieser hat gegenüber dem kurz vorher von IBM eingeführten Microchannel den Vorteil, daß ISA-Karten in einem EISA-Rechner weiterhin verwendet werden können. Diese Kompatibilität wird erreicht durch einen Takt von

8,33 MHz und einen speziellen Stecker, der die ISA-Signale im oberen Teil enthält und die zusätzlich für EISA benötigten Kontakte dazwischen im unteren Teil. Durch Sperren in dem so entstandenen 188-poligen Stecker kann eine ISA-Karte nur bis zur Hälfte der Steckerhöhe eingesteckt werden, sodaß sie keinen Kontakt zu den EISA-Signalen bekommt (Abbildung 1).

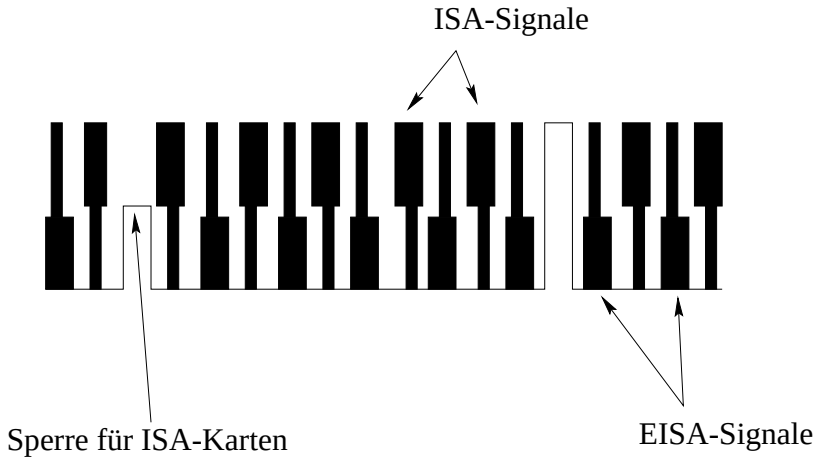

Abbildung 1: Steckverbindung einer EISA-Karte

Die Adressen der Karten werden grundsätzlich per Software eingestellt. Deshalb gehört zu jedem EISA-Motherborad ein Konfigurationsprogramm (EISA Configuration Utility ECU) und zu jeder Erweiterungskarte eine Datei mit Konfigurationsdaten. Dabei muß genau beschrieben werden, welche Karte sich in welchem Slot befindet.

Die Datenrate auf dem EISA-Bus beträgt maximal 33 MB/s. Dies wird im wesentlichen durch den breiteren Datenbus und durch größere Blöcke bei DMA-Zugriffen erreicht. Der Bus erlaubt mehrere gleichberechtigte Busmaster, die Daten auch ohne Beteiligung der CPU austauschen können. Diese wird dadurch von vielen Aufgaben des Peripheriezugriffs entlastet und erreicht somit eine bessere Verarbeitungsleistung. Allerdings müssen Betriebssysteme und Anwendungsprogramme die Vorteile des EISA-Busses auch ausnutzen können. Deshalb und aufgrund der hohen Kosten für EISA-Systeme wurde auch beim 386er meist weiterhin der ISA-Bus verwendet.

2.3 Local-Bus-Systeme

Um den Leistungsengpaß der ISA-Architektur und die hohen Kosten der EISA-Rechner zu umgehen, entwickelten verschiedene Firmen in der Folgezeit Local-Bus-Systeme, die hauptsächlich schnelle Grafikkarten anschließen sollten. Diese bestanden meist aus einem zusätzlich hinter dem AT-Bus-Stecker angeordneten Stecker, der 32-Bit-Signale mit CPU-Takt führte. Das waren proprietäre Systeme, die keiner Normung unterlagen und physikalisch wie elektrisch zueinander inkompatibel waren.

Um Ordnung in die verschiedenen Systeme zu bringen, entwickelte das VESA-Konsortium den VESA-Local-Bus oder VL-Bus. Dieser Standard legt einen 116-poligen Stecker fest, der auch bei IBMs Microchannel verwendet wird, und begrenzt den Bustakt auf 40 MHz sowie die Anzahl der anschließbaren Geräte auf drei. Bei einem Takt von 33 MHz erreicht der VL-Bus eine maximale Datenrate von 132 MB/s, bei einem

Takt von 40 MHz muß er Wartezyklen einfügen, sodaß der Durchsatz auf 80 MB/s absinkt. Die Einstellung der Adressen erfolgt im wesentlichen wie beim ISA-Bus, von dem die VL-Bus-Karten noch einige Steuersignale und Interrupt-Request-Leitungen benutzen. Die Spezifikation sieht zwar mehrere Busmaster vor, Karten mit Masterfunktion sind jedoch selten.

Nur der VL-Bus und der ebenfalls zu den Local-Bus-Systemen gehörende PCI-Bus (siehe Abschnitt 2.4) haben größere Verbreitung gefunden.

2.4 PCI

Nicht als Erweiterung bestehender Busse, sondern als neues Konzept stellte Intel 1992 den PCI-Bus vor, der nicht nur einen Bus, sondern eine ganze CPU-Umgebung definiert. Ursprünglich war der PCI-Bus nur zur Verbindung von Komponenten auf dem Motherboard selbst gedacht, weshalb Schnittstellen zur Emulation eines ISA- oder EISA-Bus vorgesehen wurden. Damit ist auch die Kompatibilität mit vorhandenen und preisgünstigen ISA-Karten gewahrt. Der Bus erlaubt bis zu zehn Geräte und mehrere Busmaster, die den Bus jederzeit anfordern können, ohne bestimmte Zeitscheiben beachten zu müssen. Die Signale sind CMOS-kompatibel und haben 5 Volt oder 3,3 Volt, was die Elektronik auf den Karten vereinfacht.

Später wurden der Spezifikation noch Stecker für Erweiterungskarten hinzugefügt. Es wird — wie schon beim VL-Bus — ein Microchannelstecker verwendet. Da die Adreß- und Datenleitungen im Zeitmultiplex übertragen werden, reichen 47 Kontakte aus. Ein festgelegtes Boardlayout (PCI-Speedway) und auf einer Seite der Chips konzentrierte Datenanschlüsse sollen für kurze Signalwege und somit für ein definiertes Signalverhalten sorgen.

Die heute in PCs am häufigsten verwendete Variante des PCI-Bus überträgt bei 32 Bit Datenbusbreite und 33 MHz Takt normalerweise 66 MB/s und bis zu 132 MB/s im Burstmode, bei dem bis zu 100 Datenblöcke nach nur einmaliger Adressierung übertragen werden. Die Spezifikation sieht aber auch eine Erhöhung der Taktrate auf 66 MHz und/oder der Datenbusbreite auf 64 Bit vor. Die Erweiterung auf 66 MHz wird beispielsweise von Sun benutzt.

40 Firmen haben sich zur PCI Special Interest Group zusammengeschlossen, um PCI-Chipsätze und Erweiterungskarten zu entwickeln, so daß inzwischen eine große Zahl von Karten zu Verfügung steht und der PCI-Bus sich im PC durchgesetzt hat. Inzwischen wird er aber auch in anderen Rechnersystemen wie etwa von Apple oder Sun verwendet.

2.5 Zusammenfassung

Die Tabelle 1 faßt nochmals die wichtigsten Daten der angesprochenen Bussysteme zusammen.

Die Tabellen 2 und 3 geben einen Überblick über die Bandbreiten weiterer heute üblicher und älterer Bussysteme, um eine Einordnung der hier genannten Systeme zu ermöglichen. Insbesondere im Bereich der Hochleistungsserver werden natürlich deutlich höhere Datenraten benötigt, die dann aber auch entsprechend teuer sind. Aber auch hier ist oft keine isochrone Datenübertragung vorgesehen, so daß die Qualität von Videofilmen von der Auslastung der Busse abhängt.

	Firewire	ISA 16 Bit	EISA	VL-Bus	PCI
Breite	seriell	16 bit	32 bit	32 bit	32 bit
Takt	200 MHz	8 MHz	8,33 MHz	33/40 MHz	33 MHz
Master	Mehrere	Einer	Mehrere	theo. Mehrere	Mehrere
Datenrate	25 MB/s	8 MB/s	33 MB/s	132/80 MB/s	66-132 MB/s

Tabelle 1: Klassische Bussysteme des PC

	S Bus	MicroChannel	IPI	SCSI2	UW-SCSI
Breite	32 bit	32 bit	16 bit	8-16 bit	16 bit
Takt	16-25 MHz	Asynchron	Asynchron	10 MHz	20 MHz
Master	Mehrere	Mehrere	Einer	Mehrere	Mehrere
Datenrate					
(32 bit read)	33 MB/s	20 MB/s	25 MB/s	20 MB/s	40 MB/s
Peak	89 MB/s	75 MB/s	25 MB/s	20 MB/s	40 MB/s

Tabelle 2: Verschiedene Bussysteme ([HePa96])

	HP Summit	SGI Challenge	Sun XD Bus
Breite	128 bit	256 bit	144 bit
Takt	60 MHz	48 MHz	66 MHz
Master	Mehrere	Mehrere	Mehrere
Bandbreite	960 MB/s	1200 MB/s	1056 MB/s

Tabelle 3: Serverbussysteme ([HePa96])

3 Firewire

3.1 Einführung

Der Name *Firewire* wurde 1987 von der Firma Apple für ihre Implementierung des seriellen IEEE 1394 Busses gewählt. Der Bus wurde zuerst nur von Apple und NeXT benutzt und verschwand danach wieder vom Markt. Jetzt, da das Interesse an seriellen Bussen allgemein wächst, wie auch an der Entwicklung des Universal Serial Bus (USB) zu sehen ist, werden Anstrengungen unternommen, ihn in PCs zu integrieren. Da Firewire zu Zeit die einzige erfolgversprechende Implementierung von IEEE 1394 ist, werden die Begriffe synonym verwendet.

Im Jahre 1994 wurde die *1394 Trade Association* gegründet, deren Mitgliederzahl inzwischen auf über 50 angewachsen ist. Darunter sind Computerfirmen wie Adaptec und Intel, aber auch Unternehmen aus der Unterhaltungselektronik wie Sony. Sie wollen mit Firewire ein Bussystem etablieren, das möglichst schnell, einfach zu benutzen und zudem preiswert sein soll. Es soll vor allem für Audio- und Videoanwendungen genutzt werden, aber auch in der Meßtechnik und für externe Massenspeicher. In der Meßtechnik erhofft man sich eine Ablösung des inzwischen 30 Jahre alten IEEE 488-Busses.

3.2 Hardware & Topologie

Firewire gibt es grundsätzlich in zwei Varianten: Eine Variante über Kabel zur Verbindung externer Geräte, und eine Backplanevariante, die in parallele Bussysteme eingebettet wird, wofür bei einigen Busspezifikationen zwei zusätzliche Leitungen vorgesehen wurden. Da die Backplanevariante mit 50 MBit/s die langsamere ist, die auch im PC kaum eine Rolle spielt, soll es hier hauptsächlich um die Kabelversion gehen.

Abbildung 2 zeigt den verwendeten Stecker und den Querschnitt des Kabels. Die Signale werden in zwei getrennt abgeschirmten Twisted-Pair-Leitungen übertragen, dazu eine Spannungsversorgung von 8 bis 40 Volt und maximal 1,5 Ampère. Das gesamte Kabel ist dann noch einmal geschirmt, um die Störfestigkeit zu erhöhen. Trotzdem wurde bei der Konstruktion darauf geachtet, daß das Kabel leicht und flexibel aber robust bleibt. Die zusätzliche Versorgungsspannung dient dazu, die Übertragungseinheiten der Geräte — und damit die integrierten Repeater — mit Strom zu versorgen, wenn die Geräte selbst abgeschaltet sind. So können auf dem Bus dahinterliegende Geräte weiterhin angesprochen werden. Einfache Geräte können auch vollständig von dieser Spannung gespeist werden.

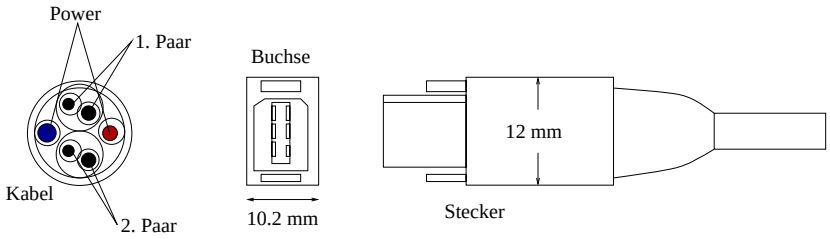

Abbildung 2: Kabel und Steckverbindung bei IEEE 1394

Jedes Kabel stellt eine Punkt-zu-Punkt Verbindung zwischen zwei Geräten her, die bis zu 4,5 Meter lang sein darf. Geräte mit mehreren Anschlüssen fungieren als Repeater, mit denen bis zu 16 Kabelsegmente hintereinandergeschaltet werden können, was eine Gesamtlänge eines Strangs von 72 Metern bedeutet. Kleinere Konfigurationen werden erkannt und die Zugriffe daraufhin optimiert. Ein Bus kann maximal 63 Geräte ansprechen. Über Bridges können 1023 Busse miteinander verbunden werden, sodaß man das Netz auf bis zu 64449 Knoten erweitern kann. Neben den Bridges sind auch Splitter und Repeater als dedizierte Geräte vorgesehen, die zur Verlängerung eines Busses bzw. zur Aufspaltung eines Strangs in mehrere dienen. Die Topologie des Busses muß einen zyklenfreien Graphen bilden. Der Bus sieht *Hotplugging* vor, das heißt, Geräte können im laufenden Betrieb an- und abgesteckt werden, ohne daß sie dabei beschädigt werden oder die Kommunikation der anderen gestört wird.

Zur Zeit gibt es drei Geschwindigkeiten auf dem Kabel: 100 und 200 MBit/s, die auch tatsächlich implementiert sind, und 400 MBit/s, für die es im Moment noch keine Sende- und Empfangschips gibt. Die 1394 Trade Association arbeitet an einer Erweiterung der Spezifikation bis 1 GBit/s. Geräte mit unterschiedlichen maximalen Datenraten können auf einem Bus zusamenarbeiten, dann bestimmt allerdings das langsamste Gerät die Geschwindigkeit, da alle Geräte die Signale weiterleiten können müssen.

Bei der Backplane-Variante hängt die erreichbare Datenrate von der Schaltgeschwindigkeit der verwendeten elektronischen Bauteile ab und reicht von 24 MBit/s bei TTL-

Technik bis 50 MBit/s. Das Interface wird dabei in der selben Chiptechnik aufgebaut, die auch das parallele Bussystem verwendet, in das der Firewire eingebettet wird.

3.3 Protokoll

Das Protokoll des Firewire wird in drei Schichten eingeteilt. Der Transaction Layer stellt mit Status- und Control-Registern das Interface zu einem beliebigen parallelen Rechnerbus zur Verfügung. Der Link Layer erledigt die Paketbildung und Steuerung, während der Physical Layer für den Anschluß an das Kabel zuständig ist. Er sorgt für die Serialisierung und Übertragung der Daten sowie die Arbitrierung und die Initialisierung nach dem Anschließen an den Bus.

Der Link Layer und der Transaction Layer werden in einem gemeinsamen Chip untergebracht, während der Physical Layer in einem eigenen Chip realisiert wird. Zur Zeit sind Physical Layer Chips für 100 und 200 MHz, Link Layer Controller für 400 MHz erhältlich. Um Masseschleifen zu vermeiden, werden die beiden Chips mit Optokopplern oder Transformatoren galvanisch voneinander getrennt.

Auf dem Kabel werden Differential-Signale verwendet, die eine sicherere Übertragung der Signale über längere Strecken ermöglichen. Die einzelnen Pakete werden mit einer CRC-Prüfsumme versehen und vom Empfänger mit einem Acknowledge-Paket bestätigt.

3.3.1 Data-Strobe-Codierung

Die Signale werden auf dem Kabel mit Hilfe der sogenannten *Data-Strobe-Codierung* übertragen. Dabei wird auf einer Leitung ein NRZ Signal geführt. Dieses hat den Vorteil, daß es relativ sparsam mit der Bandbreite umgeht, aber es kann bei längeren Ketten gleicher Bits zu Problemen mit der Synchronisierung kommen. Statt nun den Takt auf der zweiten Leitung zu übertragen, wird ein Signal angelegt, das seinen Wert immer dann ändert, wenn zwei gleiche Nutzbits nacheinander gesendet werden (Abbildung 3). Dadurch liegen die Flanken auf den beiden Leitungen immer etwa die Zeit für ein Bit auseinander, was die Störfestigkeit erhöht. Der Takt kann aus dieser Anordnung leicht zurückgewonnen werden, indem man die Signale der beiden Leitungen mit XOR verknüpft. Die Daten werden dann sowohl auf der steigenden wie auch auf der fallenden Flanke des so generierten Taktsignals gelesen.

3.3.2 Automatische Konfiguration

Eine der Eigenschaften, die den Umgang mit dem Firewire deutlich vereinfachen soll, ist die automatische Konfiguration des Busses. Der Anwender braucht die Geräte nur noch zu verbinden und einzuschalten, den Rest erledigt der Bus selbst. Es gibt keine Schalter oder Jumper für die Einstellung von Adressen, auch die notwendigen Terminatoren sind bereits in den Endgeräten integriert.

Um das zu ermöglichen, mußten die Entwickler ein Verfahren vorsehen, mit dem die Knoten die Adressen untereinander aushandeln können. Dies geschieht in drei Stufen.

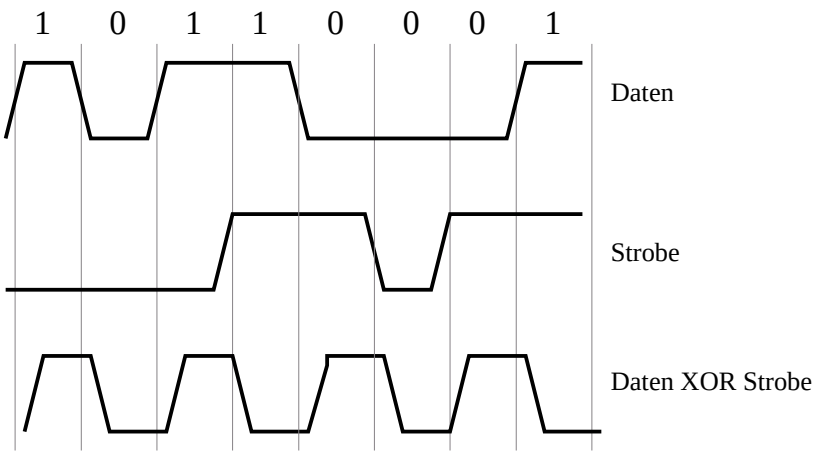

Abbildung 3: Pegel bei der Data-Strobe-Codierung

Initialisierung

Nach dem Einschalten weiß ein Busteilnehmer nur, wieviele Verbindungen er nach außen hat. Hat er nur eine, so ist er ein Blatt, hat er mehrere, so ist er ein Ast in einer Baumstruktur, die zu diesem Zeitpunkt allerdings noch keine Wurzel hat. Wenn später ein zusätzliches Gerät an den Bus angeschlossen oder eines vom Bus genommen wird, bekommen alle Knoten ein Signal, das sie wieder in diesen Anfangszustand zurückbringt. Abbildung 4 zeigt eine Beispielkonfiguration.

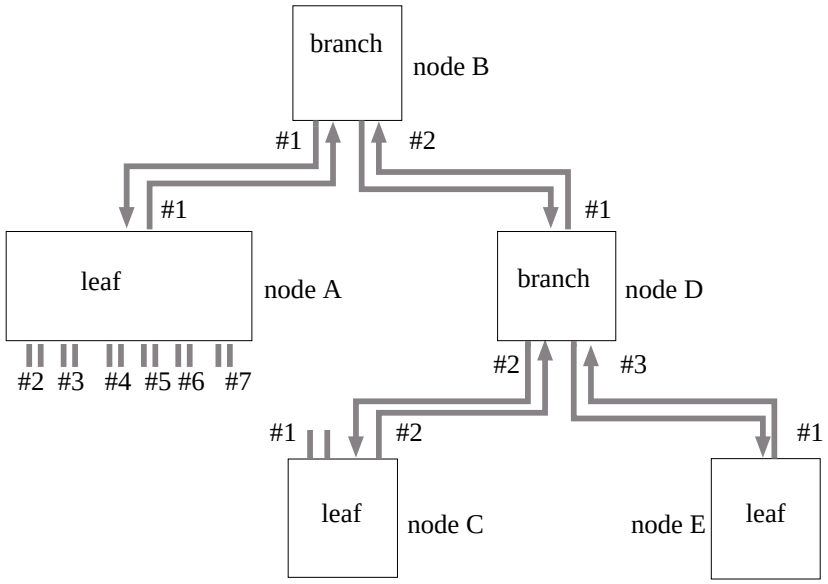

Abbildung 4: Zustand der Geräte nach dem Einschalten

Tree identify

Dann beginnt die zweite Phase, bei der die einzelnen Ports markiert werden. Ports, die keine Verbindung haben, werden als off markiert, Ports, die zu einem näher bei der Wurzel gelegenen Knoten führen, mit p für parent und die übrigen mit ch für child

(Abbildung 5). Die Blätter senden ein „Parent-Notify"-Paket über ihren einzigen Port auf den Bus und markieren ihren Port mit p. Empfängt ein Knoten ein solches Paket, markiert er den Port, auf dem es zu ihm gekommen ist, mit ch, da dort offensichtlich ein Child-Knoten sein muß. Hat er dann nur noch einen unmarkierten Port übrig, sendet er seinerseits ein Parent-Notify-Paket auf diesem Port aus. Danach steht folglich auch die Wurzel des Baumes fest. Wer Wurzel wird, hängt von dem Zeitpunkt ab, zu dem die Knoten ihre Pakete senden. Der Knoten, der am längsten gewartet hat, ohne selbst zu senden, hat am Ende nur Ports, die als Child markiert sind — er wird zur Wurzel des Baumes. Somit kann auch ein Blatt zur Wurzel werden, wenn es nur lange genug nicht sendet.

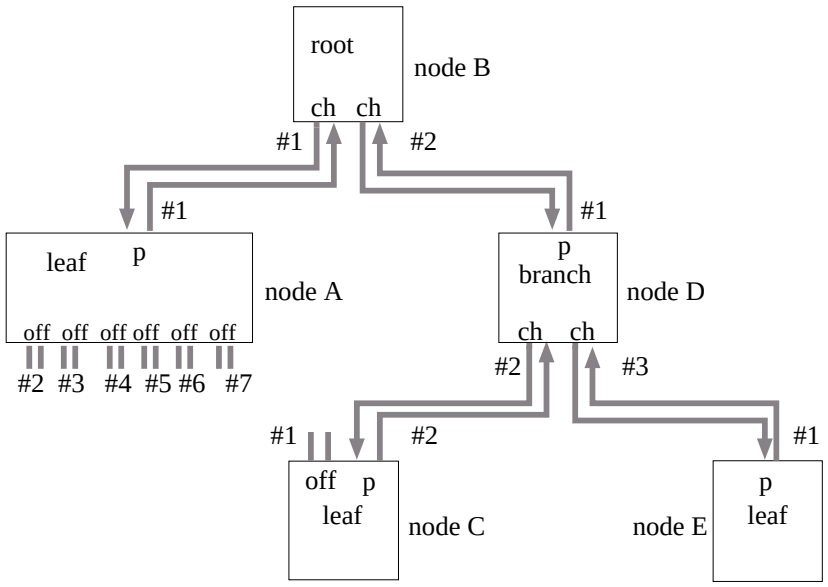

Abbildung 5: Zustand der Geräte nach dem „Tree identify"

Selbstidentifikation

In der dritten Stufe wählt jeder Knoten eine Physikalische Adresse, die ihn bei der späteren Datenübertragung eindeutig kennzeichnet. Der Wurzelknoten gibt zuerst die Kontrolle an das Gerät, welches an seinem ersten Port angeschlossen ist. Hat dieser selbst wieder Kinder, gibt er die Kontrolle rekursiv an seinen ersten Kindknoten weiter. Sobald ein Blatt erreicht ist, sendet dieses einige kurze Pakete, die seine Adresse und weitere Statusinformationen enthält, wie den Stromverbrauch, seine maximale Geschwindigkeit und die Markierungen aller seiner Ports. Die Anzahl der Pakete hängt von der Anzahl der Ports ab, die in dem jeweiligen Gerät vorhanden sind. Hat ein Knoten auf diese Weise alle seine Kinder abgearbeitet, sendet er selbst seine Identifizierungspakete und gibt die Kontrolle an seinen Parent-Knoten zurück. Dies setzt sich fort, bis als letztes der Wurzelknoten an der Reihe ist (Abbildung 6).

Jeder Knoten zählt dabei mit, wieviele Geräte ihre Pakete schon gesendet haben und wählt die nächsthöhere Nummer als seine Adresse. Das erste Blatt bekommt somit die Adresse 0, die Wurzel bekommt die höchste Adresse auf dem Bus.

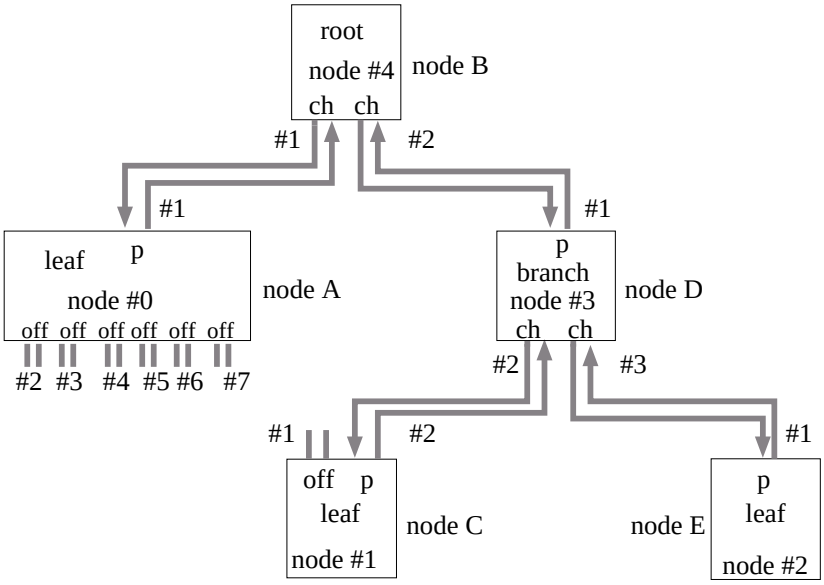

Abbildung 6: Zustand der Geräte nach der „Selbstidentifikation"

3.3.3 Arbitrierung

Nachdem die Konfiguration abgeschlossen ist, können die Knoten ihre Sendewünsche mitteilen. Die Zugriffssteuerung auf dem Bus funktioniert dabei über Idle-Zeiten oder Gaps. Ein Gerät, das senden möchte, wartet, bis der Bus für eine bestimmte Zeit frei ist (ein subaction gap) und sendet dann ein Request-Paket an seinen Parent-Knoten. Dieser sendet ein Data-Prefix-Paket an alle seine anderen Kinder, um etwaige weitere Requests zu unterbinden, und gibt das Paket seinerseits an seinen Parent-Knoten weiter. Sollte auf dem Weg zur Wurzel irgendwo schon ein früherer Request anliegen, so bekommt der Knoten, der die Anfrage ursprünglich gestellt hatte, über die selben Stufen ein Data-Prefix zurück, was ihm zeigt, daß sein Zugriff verweigert wurde. Erreicht das Paket jedoch die Wurzel und liegen keine weiteren Requests vor, so sendet die Wurzel ein Grant-Paket zurück und gewährt damit den Zugriff auf den Bus. Der Knoten sendet daraufhin seinerseits ein Data-Prefix, um seine Sendebereitschaft anzuzeigen. Damit sind alle Knoten im selben Zustand und kennen die Richtung, in der die Data-Prefix-Pakete gelaufen sind, was der Richtung vom Sender weg entspricht. Das zweite Adernpaar, das bisher die Rückrichtung für die Duplexverbindung auf dem Bus übernommen hat, ändert nun seine Richtung und kann als Strobe-Leitung zum Takten der Daten benutzt werden.

3.3.4 Fairness und Dringlichkeit

Soweit geschehen die Zugriffe auf den Bus noch nicht gleichberechtigt. Der Knoten, der am nächsten an der Wurzel liegt, wird die Arbitrierung immer gewinnen. Deshalb führt man zusätzlich ein sogenanntes Fairness-Interval ein. Innerhalb dieses Zeitabschnittes darf jedes Gerät nur einmal senden. Alle Knoten setzen zu Beginn des Intervals ein Flag in ihrem internen Speicher, das ihnen das Senden erlaubt und nach dem Senden wieder zurückgesetzt wird. Wenn alle Knoten, die senden wollen, ihr Senderecht „verbraucht" haben, tritt auf dem Bus eine längere Pause — ein arbitration reset gap — ein, die den Beginn eines neuer Fairness-Intervalls signalisiert.

In der Backplane-Variante des Firewire haben Knoten, die den Bus dringend benötigen, zusätzlich die Möglichkeit, sogenannte „urgent-Pakete" zu senden. Solche Geräte dürfen dann bis zu dreimal senden, bevor sie wieder einem „fairen" Knoten den Vortritt lassen müssen. Damit kann ein solches Gerät bis zu 3/4 der gesamten Übertragungszeit nutzen. Zu diesem Zweck hat jedes Gerät, das diese Übertragungsart nutzen möchte, einen Zähler, der auf drei gesetzt wird, wenn ein Fairness-Intervall beginnt, oder wenn ein Gerät, das die faire Arbitrierung benutzt, ein Paket sendet. Dieser Zähler wird dann bei jedem Senden eines urgent-Paketes um eins erniedrigt. Ist dieser Zähler bei einem Knoten bei Null angekommen, so darf er nicht mehr senden.

3.3.5 Isochroner Datentransfer

Da der IEEE 1394 Bus für Audio- und Videoanwendungen genutzt werden soll, muß er zusätzlich zu der bisher besprochenen asynchronen Datenübertragung auch die Möglichkeit eines isochronen Transfers vorsehen, mit dem Daten zu festgelegten Zeitpunkten und mit festen Abständen und Verzögerungszeiten gesendet werden können. Bei einer Dateiübertragung kommt es beispielsweise hauptsächlich auf hohen Durchsatz an, der genaue Ankunftszeitpunkt der einzelnen Pakete ist weniger wichtig. Auch eine falsche Ankunftsreihenfolge läßt sich leicht korrigieren. Bei Audio- oder Videodaten hingegen machen sich Pausen und andere Störungen im gleichmäßigen Datenstrom durch Knacken oder stockende Bilder unangenehm bemerkbar. Besonders bei Sprachübertragungen reagiert der Mensch empfindlich auf solche Störungen.

Um also Daten, die dies erfordern, unabhängig von anderen Sendern auf dem Bus in einem gleichmäßigen Strom zu übertragen, wird der Knoten mit der höchsten Priorität — also die Wurzel in der Kabelvariante — zum „Cycle Master". Er sendet alle 125 μs ein Cycle-Start-Paket auf den Bus. Ist zu diesem Zeitpunkt ein anderer Sender aktiv, wird das Paket verzögert, bis dieser fertig ist. Ein asynchrones Paket darf deshalb höchstens 62 μs lang sein. Um noch höhere Genauigkeit zu erreichen, wird die Zeit, um die das Paket verzögert wurde, im Cycle-Start-Paket an alle Knoten gesendet. Geräte, die den isochronen Datentransfer nutzen, bewerben sich daraufhin sofort um den Bus, ohne auf den subaction-gap zu warten, wie es asynchrone Geräte tun müßten. Sobald sie den Zugriff gewährt bekommen, senden sie ihre Daten. Dadurch kann eine längere Pause, wie sie für einen subaction-gap nötig ist, erst auftreten, wenn alle isochronen Geräte ihre Daten übertragen haben. Diese Pause erkennen dann die asynchronen Geräte und bewerben sich nun ihrerseits um den Bus.

Beim isochronen Datentransfer geht die Adressierung nicht mehr nach physikalischen Adressen vor sich, sondern nach Kanalnummern, die ein Gerät vor dem ersten isochronen Senden mit einem Kanalmanager mit Hilfe asynchroner Pakete aushandelt. Dieser Kanalmanager muß dabei auch dafür sorgen, daß die gesamte Sendezeit der isochronen Geräte die 125 μs in einem Zyklus nicht überschreitet.

Abbildung 7 zeigt, wie die Paketverteilung dann insgesamt aussieht, wenn sowohl isochroner wie auch asynchroner Datenverkehr stattfindet. Man sieht, daß die asynchron sendenden Geräte die Auslieferung der isochronen Daten nicht beeinflussen können.

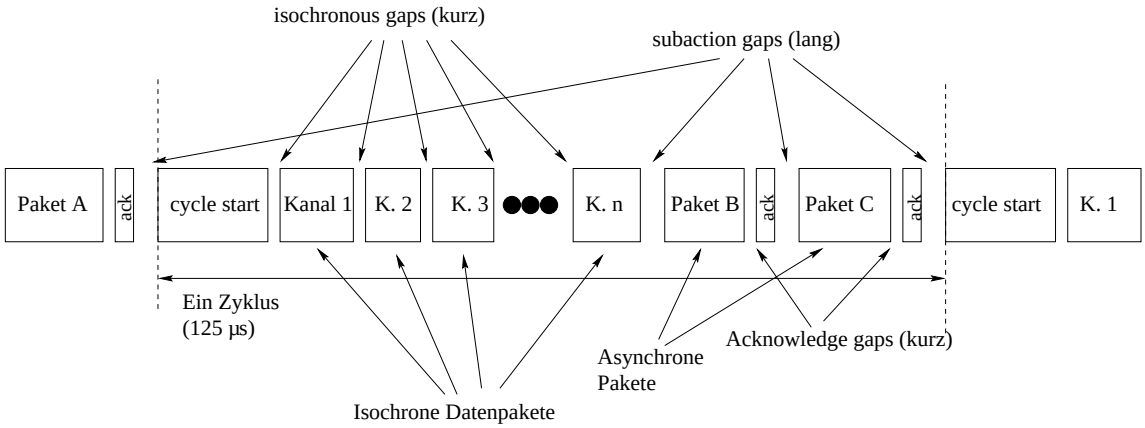

Abbildung 7: Pakete auf dem Firewire

4 Zusammenfassung und Ausblick

Der Firewire-Bus ist von der Theorie her eine sehr gute Sache. Insbesondere bei der Übetragung von Daten mit festen Anforderungen an die Auslieferungszeitpunkte ist er den Systemen, die keinen isochronen Datenverkehr unterstützen, überlegen, da er die zeitkritischen Daten von Multimediaanwendungen auch bei hoher Auslastung noch gleichmäßig, rechtzeitig und in der richtigen Reihenfolge ausliefert. In Silizium ist davon bisher leider noch nicht viel beim Endbenutzer angekommen. Die Spezifikationen sind inzwischen bei 1 GBit/s angekommen, das Interface zum Rechnerbus schafft immerhin 400 MBit/s, aber die Übertragung auf dem Firewire selbst bremst den Bus noch immer auf 200 MBit/s aus. Inzwischen sind einige wenige Firewire Hostadapter — zum Beispiel von Adaptec — vorhanden, aber einzeln kaufen kann man sie noch nicht. Es gibt einige Videokameras, von 2000 DM aufwärts. Captureprogramme dafür gibt es zwar auch, aber sie sind noch sehr lückenhaft. Bis das Ziel, einen preiswerten und überall verfügbaren High-Speed-Bus zur Verfügung zu stellen, verwirklicht ist, ist also noch einiges zu tun. Ich persönlich sehe zum jetzigen Zeitpunkt keine Gefahr für den Fortbestand von SCSI oder gar PCI, die auf einen sehr großen Bestand an vorhandenen Geräten und installierten Systemen aufbauen können und sich im praktischen Einsatz bewährt haben. Auf lange Sicht werden wir aber — insbesondere für die immer mehr zunehmenden Anwendungen im Bereich des Videoschnittes und ähnlichem — eine schnelle, digitale, isochrone Datenverbindung wie den Firewire-Bus brauchen.

Literatur

[Demb93] K. Dembowski. *Computerschnittstellen und Bussysteme*, S. 235–274. Markt&Technik. 1993.

[Demb97] K. Dembowski. Feuerdraht – Firewire und andere serielle Bussysteme. *c't*, Februar 1997, S. 284–290.

[DySm] T. Dyke und P. Smolen. Rallying around the IEE 1394.

[HePa96] J. L. Hennessy und D. A. Patterson. *Computer Architecture*, S. 501. Morgan Kaufmann. 1996.

[Hoff95] G. Hoffman. IEEE 1394: A ubiquituous bus. In *Proceedings of Compcon '95*, 1995.

[Klot93] A. Kloth. *Bussysteme des PC*, S. 272–283. Francis'. 1993.

[Teen93] M. Teener. New Technology in the IEEE p1394 Serial Bus - Making it Fast, Cheap and Easy to Use. *Apple Computer, Inc*, 1993.

Neue Betriebssystemkonzepte für die Kommunikation: EXO-Kernel des MIT

Hipolito Vasquez Lucas

Kurzfassung

Im Bereich der Arbeiten auf dem Gebiet der Hochleistungskommunikation wurde schon bald festgestellt, daß schnelle Netze und leistungsfähige Anwendungen nicht ausreichen, um einem Anwender eine hohe Leistungsfähigkeit zur Verfügung zu stellen. Nachgefragt sind unter anderem auch Betriebssysteme, die verschiedene Dienstqualitäten eines Netzes an eine Anwendung weiterreichen können. Unterstützende Mechanismen sind unter anderem echtzeitfähige Schedulingmechanismen, Mikrokernel, Protokollmechanismen ohne Kopiervorgänge, usw. Der Exokernel-Ansatz des MIT, USA, stellt einen relativ radikalen Ansatz dar, der beinahe alle Funktionen eines Betriebssystems außerhalb des Kerns ansiedelt,um sehr schnelle Kontextwechsel zu gewährleisten. In dieser Arbeit, wird die Struktur eines Exokernelsystems dargestellt und vor allem wird gezeigt, wie Kommunikationssysteme sie ausnutzen können, um Hochleistungskommunikation anzubieten.

1 Einleitung

Die *Abstraktion* und das *Umschalten* der physikalischen Ressourcen sind die zwei wesentlichen Aufgaben, auf die die Mehrheit der Betriebssysteme ausgelegt sind [EnKa95]. Die Systemarchitekten haben angenommen, daß Abstraktionen geeignet für alle Gebiete sein könnten und daß sie effiziente Leistung unter verschiedenen Bedingungen anbieten könnten. Auf der Seite der Anwendungen variiert die Zahl der Bedürfnisse. Manche Anwendungen benötigen mehr Ressourcen als andere; einige Bedürfnisse sind orthogonal zueinander. Daraus kann man den Schluß ziehen, daß es unmöglich ist, eine Abstraktion von Ressourcen zu finden, so daß sie brauchbar für alle Anwendungen sein kann. Es ist einfach unmöglich die Implementierung von effizienten Dienstleistungen für ungleichartige Dienstnehmer durchzuführen. Die Folge der oben genannten Punkte ist die niedrige Leistungsfähigkeit, Anpassungfähigkeit, Zuverlässigkeit und Flexibilität der Betriebssysteme in den letzten dreißig Jahren [EnKa95]. Um diese Dienste zu verbessern ist der *EXO-Kernel*-Ansatz entstanden.

Die Philosopie hinter einem Exokernelsystem ist die Unterscheidung zwischen dem *Umschalten* und dem *Management* der Ressourcen. Ein so konzipiertes System führt das gesicherte Umschalten durch den *Exokernel* durch, welcher versucht, so weit wie möglich, Dienstprimitive der Hardware den Anwendungen anzubieten (Beseitigung aller

Abstraktionen). Auf der anderen Seite wird das Management durch die sogenannten *libOS* Bibliotheken (Library Operating Systems) ermöglicht. Sie bieten Dienstleistungen je nach Bedürfnissen der Anwendungen an.

Diese Ausarbeitung ist folgenderweise gegliedert: Im Teil 1 ist die Struktur eines Exokernels dargestellt. Teil 2 stellt zwei spezifische EXO-Kernelsysteme dar und Teil 3 befaßt sich mit der Anwendung eines Exokernels im Bereich der Hochleistungskommunikation und schließlich stellt einen Vergleich der Leistungen eines konventionellen OSs (Operating System) und eines Exokernels dar. Die Schlußfolgerung findet man im Teil 4.

1.1 Warum ein neuer Ansatz für Betriebssysteme?

Traditionelle Betriebssysteme haben ein zentralisiertes Ressourcenmanagement. Dieses Management wird durch den Kernel oder privilegierte Software ausgeführt, deshalb können die Anwendungen es weder spezialisieren noch erweitern oder auswechseln; mit dieser Philosophie versuchen diese Systeme allen Anwendungen alle Features zu liefern. Lampson and Sproul [LaSp79] haben gezeigt, daß Mehrzweckimplementierungen von Abstraktionen einen Overhead verursachen.

Feste Abstraktionen an den oberen Ebenen eines Informationsverarbeitungssytems schaden der Leistung der Anwendungen. Es ist unmöglich, eine Abstraktion derart zu konzipieren, daß sie die beste für alle Benutzer ist. Bei der Implementierung von Abstraktionen ist ein Betriebssystem daher gezwungen, die verschiedenen Betriebsmittel abzuwägen, dabei werden bestimmte Anwendungen bestraft. Manchmal hat eine Datenbasis, zum Beispiel, eine bestimmte vorhersehbare Zugriffsart aber ihre Leistung wird verringert, wenn eine Mehrzweckstrategie für das Seitenauswechseln wie LRU (Least recently used) durch das Betriebssystem benutzt wird.

Bei vielen Abstraktionen sehen die Anwendungen nicht alle verfügbaren Informationen. Hardware-nahe Ausnahmen, Timer-Unterbrechung oder Direkt-E/A Geräte sind den oberen Ebenen nicht zugänglich. Das ist die Ursache dafür, daß eine Anwendung ihr eigenes Ressourcenmanagement nicht ausführen kann. Programmierer sind gezwungen, über gegeben Abstraktionen zu emulieren, statt mit den Primitiven der Hardware zu arbeiten. Dieses Problem sieht man zum Beispiel, bei der Implementierung von Leichtgewichtprozessen (Threads) über einen konventionellen Prozeß.

Gegebene Abstraktionen begrenzen die Funktionalität der Anwendungen. Abstraktionen liefern nur eine bestimmte Schnittstelle, die fast nie geändert werden kann, weil jede Anwendung sie als eine Einheit benutzt.

Unter Berücksichtigung der oben genannten Punkte haben die Entwickler des EXO-Kernelsystems das Ressourcenmanagement den jeweiligen Anwendungen zugeordnet.

1.2 Architektur eines Exokernelsystems

Der Unterschied zwischen der Architektur eines EXO-Kernelsystems und anderen Architekturen (Mikrokernels, monolithische Architekturen, usw.) liegt darin, daß in einem

EXO-Kernel das *Ressourcenmanagement* vom *Ressourcenschutz* getrennt wird. Um diese beiden Aufgaben zu erledigen, ist ein EXO-Kernelsystem in zwei wesentliche Teile zerlegt. Die sogenannten *libOS*s (Library Operating Systems) und ein kleiner *EXO-Kernel*. Der EXO-Kernel bietet den Schutz der Anwendungen durch folgende Dienste an:

- Das Umschalten der Zugriffe auf Hardware.
- Das Exportieren physikalischer Ressourcen durch einen Satz von Hardware-nahen Primitiven.

LibOS-Bibliotheken, die die Schnittstelle des EXO-Kernels benutzen, implementieren Abstraktionen je nach Bedarf der entsprechenden Anwendungen. In Abb. 1 sieht man ein EXO-Kernelsystem im Vergleich zu einer monolithischen Architektur.

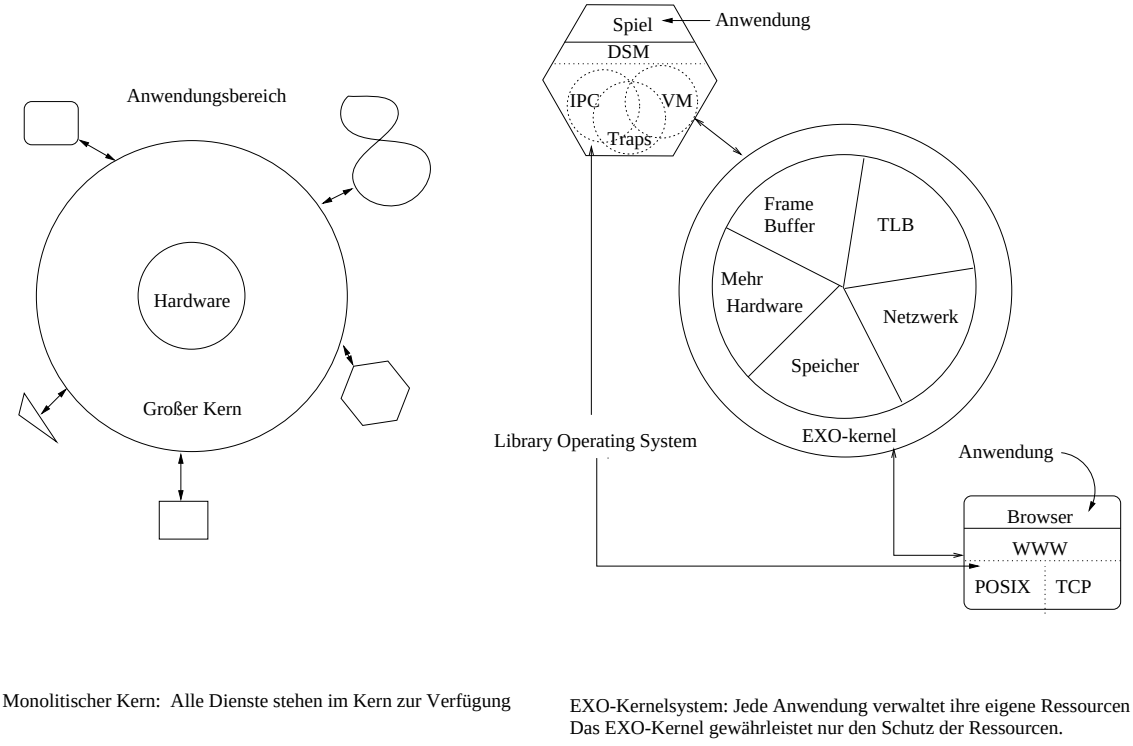

Abbildung 1: Das EXO-Kernelsystem und eine monolithische Architektur

Die Implementierung von Abstraktionen im Bereich der LibOS-Bibliotheken kann einfacher und spezialisierter sein als eine Implementierung im Kern, weil die Bibliotheken nur ihre bestimmte Anwendung bedienen. Diese Architektur kann soviel Portabilität und Kompatibilität wie gewünscht liefern. Die Anwendungen, die eine EXO-Kernel Schnittstelle benutzen, werden nicht portabel, weil die Schnittstelle Hardware-nahe spezifische Information liefert während die Anwendungen, die eine Standard-Schnittstelle haben (POSIX,usw.), portabel zu anderen Systeme mit der jeweiligen Schnittstelle sind. Die Portabilität und Kompatibilität hängen hier von Anwendungsprogrammierern ab und nicht von Systemprogrammierern wie bei den anderen Ansätzen. Anwendungsprogrammierer können jederzeit ohne Sonderrecht LibOS-Bibliotheken ändern, auswechseln und

sogar vernichten. Die Änderungen oder ein Fehler in einer hat keine Auswirkung auf die anderen. Wie in Abb. 1 grafisch dargestellt ist, sind die LibOS unabhängig voneinander.

1.3 Grundsätze der Architektur

Ein EXO-Kernel spezifiziert die Einzelheiten der Schnittstelle, die die LibOS brauchen, um die Betriebsmittel (Hardware) anzufordern, freizugeben und zu benutzen. Um diese Dienste zu gewährleisten, wurden folgende Punkte bei der Exokernel-Systementwicklung angewendet.

- Ein Kernel sollte Hardware-nahe Primitive anbieten, so daß der Zugriff auf die Ressourcen so direkt wie möglich gemacht werden kann. Diese Primitive betreffen den physikalischen Speicher, die Zentralprozesseinheit, den sekundären Speicher,TLB, DMA, privilegierte Befehle, Unterbrechungen, Ausnahmen, usw. Der EXO-Kernel muß den LibOS-Bibliotheken privilegierte Befehle liefern, um die Abstraktionen der traditionellen Betriebssysteme im Anwendungsbereich implementieren zu können. Solche Abstraktionen sind normalerweise im Rahmen der IPCs (Interprocess Communications) und des VMs (Virtual Memory) vorhanden.

- LibOs-Bibliotheken sollten spezifische physikalische Ressourcen anfordern, dafür sollte der EXO-Kernel ihnen die richtigen Dienste anbieten.

- Der EXO-Kernel sollte die physikalischen Namen der Ressourcen den LibOS-Bibliotheken liefern. Das eliminiert die Verzögerung der Umsetzung von physikalischen in logische Namen.

- Sichtbare Hardware-Freigabe sollte der EXO-Kernel auch ermöglichen, so daß die LibOSs bestimmte Betriebsmittel aufgeben können.

1.3.1 Gesichertes Binden

Der Kernel führt das Umschalten und das Schützen der Betriebsmittel durch. Wie schützen sich verschiedene Anwendungen voneinander? Der Kernel bietet den oberen Schichten ein gesichertes Binden (Secure Binding). Unter gesichertem Binden versteht man den Mechanismus, welcher die Trennung zwischen dem Benutzen und der Ermächtigung eines Betriebsmittel ermöglicht. Bei diesem Mechanismus sind zwei verschiedene Zeiten zu unterscheiden: die Bindezeit (bind time, Zeit der Ermächtigung) und die Zugriffszeit (Access time, Zeit des Benutzens). Es gibt drei wesentliche Techniken zur Implementierung eines gesicherten Binden [EnKJ95]: Hardware Mechanismus, Software Caching und Laden (Downloading) von Anwendungscode in den Kernel. Die erste bietet hardware-nahen Schutz, so daß spätere Zugriffe auf die Betriebsmittel ohne spezielle Erlaubnis möglich werden können. Gesichertes Binden kann auch in einem Kernel gecached sein. Das könnte als eine Emulierung der Hardware im Kernel angesehen werden. Bei dem dritten Mechanismus geht es darum, bestimmte Anwendungsoftware in den Kernel zu Laden; die bei dem Zugriff einer bestimmten Ressource aufgerufen wird. Dieser Mechanismus erlaubt das Ablaufen eines Programms nur mit den Ereignissen

des Kernels (Beseitigung von Kreuzungen, Kernel/Anwendungen) und kann ein Programm ablaufen lassen, sogar wenn es nicht geplant (scheduled) ist. Hier kann man die Situation eines Programms erwähnen, bei dem der Kontextwechsel unmöglich zu realisieren ist.

1.3.2 Das Umschalten vom Netzwerk

Das effiziente Umschalten vom Netzwerk ist eine Herausforderung. Es erfordert spezifisches protokoll-orientiertes Wissen zur Interpretation des Inhalts der einkommenden Pakete und zur Erkennung der richtigen Empfänger. Umschalten kann sowohl mit Hardware als auch mit Software durchgeführt werden. Ein Beispiel im Bereich der Hardware stellen die VCs (Virtual Circuits) in den ATM Zellen dar, die ein gesichertes Binden zwischen den Paketen und den Anwendungen ermöglichen, während Paketfilter (Packet Filter) ähnliche Dienste im Rahmen der Software anbieten. Paketfilter können angesehen werden, als eine Implementierung von gesichertem Binden, in der die Anwendugssoftwarekodierung (Filter) in den Kernel geladen wird. Hier betrifft das protokoll-orientierte Wissen die verschiedenen Anwendungen und der EXO-Kernel schickt die Pakete an die entsprechenden Anwendungen. Das Umschalten des Netzwerks zwischen ausgeheden Paketen wird durch das Kopieren vom Anwendungsbereich in den sendenden Puffer ausgeführt. Paketfilter können Nachrichten schon bearbeiten, sogar wenn die entsprechende Anwendung nicht vorbereitet ist.

Application-specific Safe Handlers (ASHs) entsprechen einer Art des dritten gesicherten Bindens. Ein ASH-Handler ist mit einem Paketfilter verbunden und läuft bei dem Empfang eines Pakets ab. Ein ASH kann das Senden von Nachrichten anfangen, deswegen ist die Antwortzeit des Netzwerks verringert, denn die Antwort muß nicht von einer Anwendung kommen.

1.3.3 Sichtbare Ressourcen

Traditionelle Betriebssysteme geben Betriebsmittel ohne Benachrichtigung den Anwendungen frei. In solch einem System ist das Wissen über die Verfügbarkeit der Ressourcen begrenzt. In einem EXO-Kernel ist die Freigabe sichtbar für jede Anwendung. Die Belegung und Freigabe wird hier zwischen den LibOS-Bibliotheken und dem Kernel unter der Mithilfe von physikalischen Namen gehandelt, dafür benutzt man ein AP (Abort Protocol) Protokoll [EnKJ95] .

2 Spezifische EXO-Kernelsysteme

In diesem Abschnitt sind zwei verschiedene EXO-Kernels und die entsprechende LibOS-Bibliotheken dargestellt. Beide Betriebssysteme wurden bei der PDOS (Parallel and Distribuited Operating Systems) Gruppe am MIT entwickelt.

2.1 Aegis/ExOS

2.1.1 Aegis

Der Aegis EXO-Kernel und die entsprechende ExOS LibOS Bibliotheken bauen ein Exokernelsystem auf, das auf MIPS-basierten DEC-Workstations implementiert worden ist. Aegis exportiert den Prozessor, physikalischen Speicher, TLB, Abbrechungen, Ausnahmen und die Schnittstelle des Netzwerks unter Mithilfe von einem Paketfilter, der mit dynamischer Code-Erzeugung funktioniert. Dynamische Code-Erzeugung ist später in diesem Abschnitt erklärt.

ExOS implementiert Prozesse, Virtueller Speicher, Benutzerebene Ausnahmen, IPC Abstraktionen und verschiedene Protokolle (ARP/RARP, IP, UDP und NFS). Tab. 1 stellt eine Untermenge der Aegis-Schnittstelle dar. Einige Primitive sind in Tab. 2 zu sehen.

Systemaufruf	**Beschreibung**
Alloc	Belegung von Ressourcen
Dealloc	Freigabe von Ressourcen

Tabelle 1: Untermenge der Aegis-Schnittstelle

Primitive	**Beschreibung**
TLBwr	Einfügen von Mapping in den TLB
FPUmod	Ein- Ausschalten der FPU
CIDswitch	Installation des Kontextidenfikator
TLBvadelete	Löschen von virtueller Adresse vom TLB

Tabelle 2: Muster der Aegis-Primitiven

Die Primitive sind Pseudobefehle ähnlich mit dem PALcode bei den ALPHAs [Digi96]. Aegis macht das Scheduling durch den Round Robin Algorithmus; dabei wir die Position eines Prozesses benutzt, um Termine einzuhalten und um das optimale Abwägen zwischen Wartezeit und Durchsatz auszuführen.

Protected Control Transfer:
Aegis bietet ein Protected Control Transfer (PCT) als Infrastruktur für eine effiziente Implementierung von IPC Abstraktionen im Anwendungsbereich an. Ein PCT läßt den Befehlszähler zu einem bestimmten Platz im angerufenen Prozeß springen, legt einen Zeitschlitz dem angerufenen Prozeß ab und installiert schließlich die erforderlichen Elemente für den Prozessorkontext. Aegis bietet Synchrone und Asynchrone PTCs an; asynchrone geben nur den Rest des Zeitschlitzes dem Prozeß, während synchrone PCTs den betreffenden Zeitschlitz und seine künftigen Instanzierungen zur Verfügung stellen.

Dynamischer Paketfilter:
Aegis beinhaltet ein Netzwerksubsystem (Dynamic Packet Filter), welches eine dynamische Code-Erzeugung benutzt, um effiziente Nachrichtenbearbeitung zu gewährleisten. DPFs bestehen aus konventionellen Packetfiltern, die ausführbare Code während der Laufzeit erzeugen. DPF nutzen dynamische Code-Erzeugung folgenderweise aus:

1. dynamische Code-Erzeugung beseitigt Overhead bei einer Interpretation; Paketfilter sind schon als ausführbarer Code im Kern.
2. die konstante Anzahl der Filter optimiert das Ausführen der Code.

Um Portabilität anzubieten, sind DPF in VCODE [Howe97] programmiert.

2.1.2 ExOS

IPC Abstraktionen Pipes wurden in ExOS unter Mithilfe von einem Shared Memory Circular Buffer realisiert. Hier sind zwei Arten von Pipes zu unterscheiden, die konventionelle Implementierung (Pipes) und die optimierten Pipes (Pipes') [WaEK96]. Pipes' nutzen die Tatsache aus, daß die ExOS-Bibliothek schon im Anwendungsbereich ist, um das Lesen mit dem jeweiligen Schreiben außerhalb des Kernels zu verbinden. *Shared Memory* und *lrpc* (Leight Weighted Remote Procedure Call) sind auch in ExOS programmiert.

Application Specific Safe Handlers. Normalerweise bieten ExOS-Bibliotheken eine gute Leistung im Anwendungsbereich an. Die Ursache dafür sind die geringeren Kosten der Kreuzung zwischen den Anwendungen und dem Kernel. Obwohl man alle Anwendungen über ExOS-Bibliotheken implementieren könnte, gibt es im Rahmen der Kommunikation zwei verschiedene Gründe um Code in den Kernel zu laden:

- Die Netzwerkpuffer können nicht so leicht im Anwendungsbereich implementiert und bearbeitet werden, aber im Kernel kann, zum Beispiel, die Checksumme während des Kopierens eines Pakets durchgeführt werden.
- Man versucht die Antwortzeit des Netzwerks von dem Scheduling der beteiligten Prozesse zu trennen.

Application Specific Handlers (ASHs) sind Nachrichten-Handler in ExOS, die im Anwendungsbereich programmiert und danach in den EXO-Kernel geladen werden. Ein ASH führt folgende Aufgabe durch:

1. Direct Dynamic Message Vectoring (DDMV): Ein ASH kontrolliert wo die Nachrichten im Speicher kopiert sind und kann zwischenliegende Kopien vermeiden.
2. Dynamic Integrated Layer Processing (DILP): ASH können integrierte Daten-Manipulationen wie Prüfsumme und Umwandlungen schon in der Daten übertragenden Maschine (Data Transfer Engine) durchführen. Solche Manipulationen können mit Pipes realisiert werden. Die Integration von Pipes ermöglicht eine horizontale Bearbeitung der Nachrichten, das bedeutet, daß zu einem bestimmten Zeitpunkt verschiedene Operationen an einem Paket ablaufen können.

3. Message Initiation (MI): ASH-Handler können das Schicken einer Nachricht anfangen, um die Antwortzeit in einem Netzwerk zu vermindert. Siehe Abb.2.

4. Control Initiation (CI): ASH-Handler führen allgemeine Berechnungen durch, deswegen können sie Steuerung-Operationen schon bei der Empfangszeit bearbeiten.

In Abb. 2 ist die Lokalisierung eines ASH-Handlers in Aegis dargestellt.

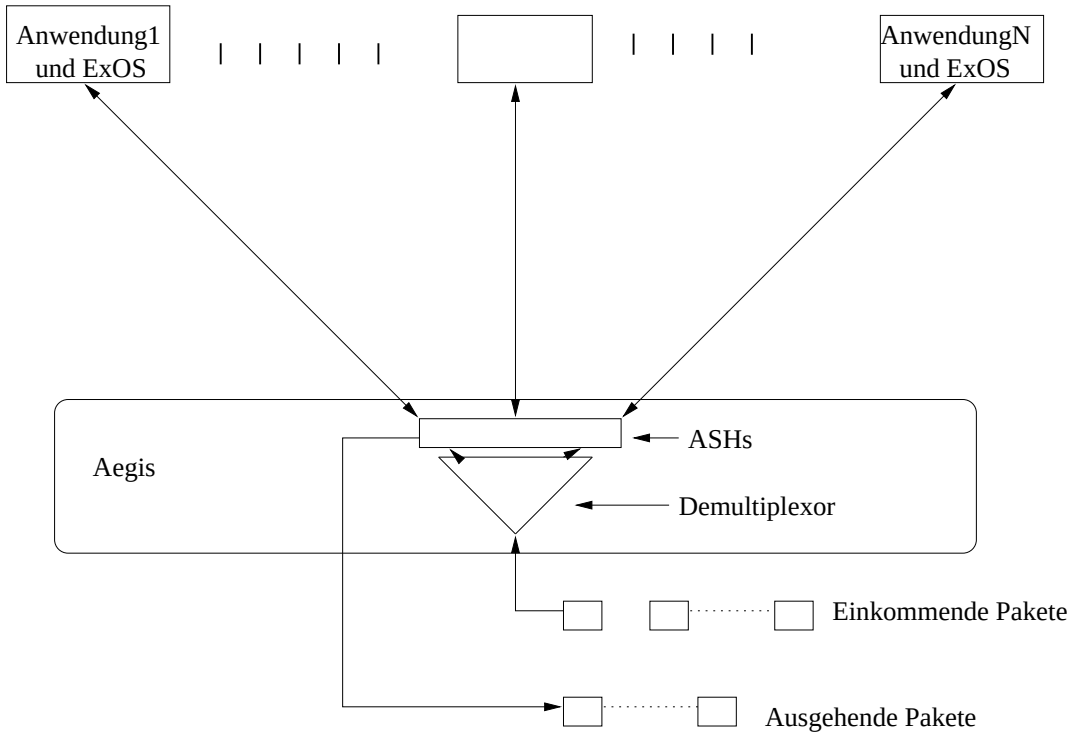

Abbildung 2: Stelle eines Application-Specific-Handlers in Aegis

Man unterscheidet drei verschiede Arten von ASH-Handlern:

- ASH mit ILP (Integrated Layer Processing)
- ASH mit DILP (DynamicIntegrated Layer Processing)
- ASH mit DCP (Dynamic Composed Protocols) [WaEK96]

ASH-Handler können in beliebigen Betriebssystemen implementiert werden, aber sie nutzen besser den Aegis-EXO-Kernel aus, weil in Aegis Kernel-Kreuzungen (Kernel Crossing) optimiert worden sind. Aegis-Kreuzungen sind fünfmal schneller als die beste in der Literatur genannte [EnKJ95].

2.2 Xok/ExOS

Xok ist ein EXO-Kernel für Intel x86-basierte Rechner und ExOS ist seine Default-Bibliothek. Auf diesem System laufen viele unmodifizierte UNIX Programme wie perl, gcc, telnet und die Mehrheit der Dateidienstprogramme. Das Xok/ExOS ähnelt dem Aegis/ExOS-Exokernelsystem [BrHM+97].

3 EXO-Kernels und Rechnernetzwerke

Heutige Computernetzwerke können ihre Leistung durch die Verwendung eines Exokernelsystems erhöhen [WEKG96]. In diesem Abschnitt wird ein Teil der experimentellen Arbeit (EXO-Kernel und Rechnernetzwerke) der PDOS-Gruppe dargestellt.

3.1 Der Cheetah HTTP/1.0 Server

Das Exokernelsystem ist geeignet für den Aufbau schneller Server wie NFS (Network File System) oder Web Server [EnKJ95]. Serverleistung ist entscheidend bei Client/Server-Anwendungen und die E/A-zentralisierte Natur eines Servers macht be triebssysteme-orientierte Optimierung sinnvoll. Die PDOS-Gruppe hat eine erweitbare E/A-Bibliothek (Extensible I/O Library, XIO) für schnelle Server mit einem Exokernelsystem entwickelt, dazu haben sie auch den Cheetah-HTTP-Server als Musteranwendung entwickelt. Diese Bibliothek ist gedacht für die Anwendungsprogrammierer, um domain-spezifisches Wissen auszunutzen und für die Vereinfachung des Aufbaus eines Hochleistungskommunikationsservers; hier bekommt man die richtigen Dienste ohne das OS (Operating System) zu betrügen (durch das Speichern von gecacheten Seiten in verschiedenen Verzeichnissen um einen Namen schnell zu suchen, zum Beispiel). Cheetah bietet u.a. folgende Dienste an:

- Merged File Cache and Retransmission Pool: Daten werden an den Client direkt aus der Cache geschickt. Die CPU (Central Proccesing Unit) macht keine Kopie. Dateien werden zusammen mit ihrer entsprechenden vorberechneten Checksumme gespeichert (bereit zum Senden), deshalb ist hier ein TCP-Cache für erneutes Übertragen nutzlos.
- Knowledge-based Packet Merging: Der Server nutzt sein Wissen über seine Zustände, um so wenig wie möglich E/A-Operationen auszuführen. Beispielsweise vermeidet er das redundante Senden von Steuerungspaketen durch die Verzögerung der ACKs während einer HTTP-Client Anforderung, weil er das Piggybacking bei der Antwort erreichen will. Diese Optimierung ist besonders wertvoll bei kleinen Dokumenten, weil die Reduzierung einen großen Bruchteil der gesammten Anzahl der Pakete ausmacht.
- HTML-based File Grouping: Der Server speichert Dateien zusammen, auf die in einem HTML-Dokument hingewiesen wird. Er speichert sie in physikalische Blöcke so nahe wie möglich vom HTML-Dokument. Wenn der Dateicache die Mehrheit der Client-Anforderungen nicht einfangen kann, verdoppelt dieser Mechanismus den Durchsatz [BrHM+97].

In Abb.3 ist der Durchsatz der Anforderungen als Funktion der Seitengröße für fünf Server dargestellt. Der Versuch wurde von der PDOS-Gruppe mit den folgenden Servern durgeführt: NCSA, Harvest-Cache, Socket-basierter Server alle auf OpenBSD 2, Socket-basierter Server und Cheetah auf Xok. Hier ist es zu sehen, daß die Sockets über die XIO-Bibliotheken eine höher Leistung als über das OpenBSD-Betriebssystem haben. Mit einer 1KByte Seite kann Cheetah den besten OpenBSD-basierten Server achtfach übertreffen. Laut der PDOS-Gruppe war in diesem Fall die Leistung

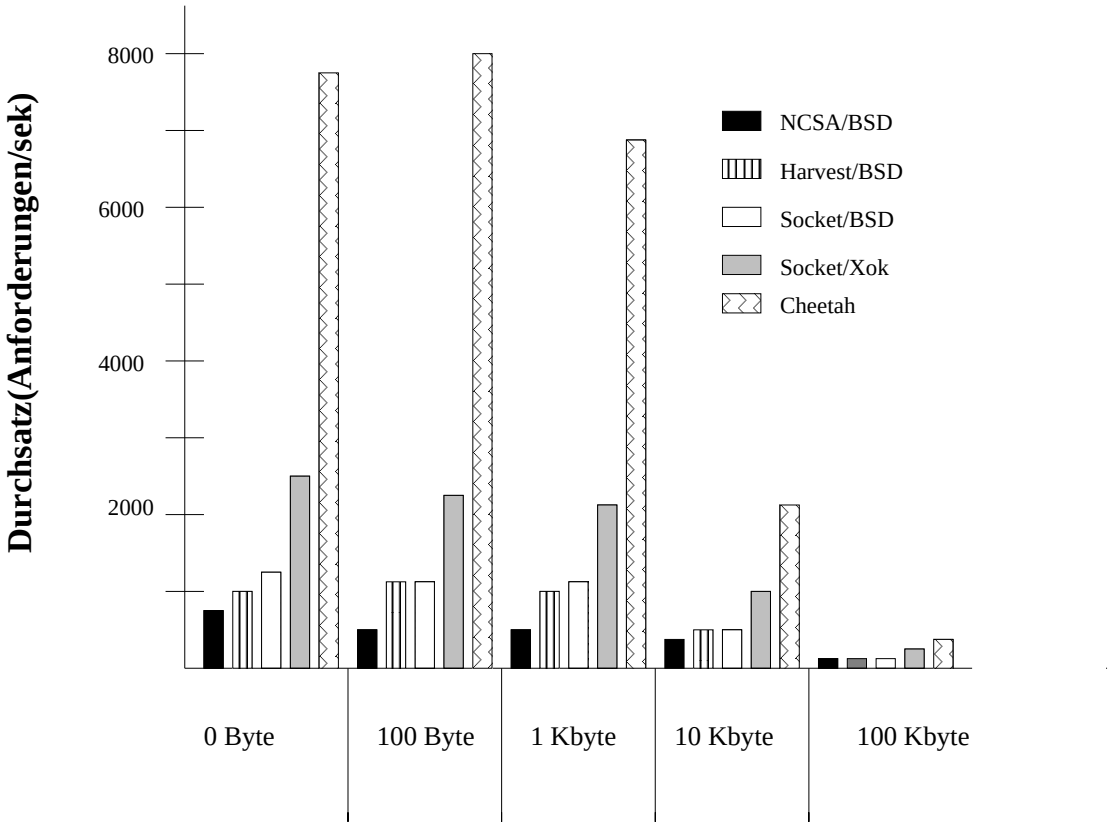

Abbildung 3: Durchsatz eines Exokernel-basierter Servers und andere Systeme als Funktion der Seitegrösse

Cheetahs durch die Bandbreite des Netzes begrenzt (drei 100Mbit/Sek Ethernets) und nicht durch die Serverhardware [BrHM+97]. Die Socket-basierte Implementierung war begrenzt zu 16.5MByte/Sek und hat die CPU 100% belegt, während Cheetah über 29.3MByte/Sek übertragen konnte und die CPU war über 30% nicht aktiv.

3.2 ASH-Basiertes Netzwerk

In Tabelle 3 ist die Antwortzeit (Round Trip Latency) über Ethernet eines ASH-basierten Betriebssystems (ExOS/ASH) im Vergleich zu ExOS (ohne ASH), Ultrix Betriebssysteme und FRPC (Der schnellste RPC in der Literatur auf vergleichbarer Hardware) [ThLe93] dargestellt. Diese Antwortzeit von Aegis und Ultrix wurde durch das ping-ponging eines Zählers in einem 60-Byte-UDP/IP-Paket 4096 mal gemessen. Die Pakete wurden zwischen zwei Prozessen im Anwendungsbereich ausgetauscht. In der Tabelle wird deutlich gezeigt, wie ExOS/ASH und ExOS die Laufzeit verringern können. In dem gleichen Versuch hat die PDOS-Gruppe auch festgestellt, daß die Antwortzeit bei einer ExOS-Bibliothek (ohne ASH) im Zusammenhang mit der Anzahl der Prozesse beim Empfänger steht. Je mehr Prozesse desto höher ist die Antwortzeit. In einer ExOS/ASH-Konfiguration ist die Antwortzeit orthogonal zu der Anzahl der Prozesse. Das liegt darin, daß bei einer ExOS/ASH-Konfiguration keine unbedingte

Abwicklung (Scheduling) gebraucht wird. Mehr Information über den Versuch findet man in [EnKJ95].

Maschine	**Betriebssystem**	**Antwortrzeit**
DEC5000/125	ExOS/ASH	259
DEC5000/125	ExOS	320
DEC5000/125	Ultrix	3400
DEC5000/200	Ultrix/FRPC	340

Tabelle 3: Antwortzeit eines 60-Byte-Pakets über Ethernet und verschiedene Betriebssysteme. Die Zeit ist in Mikrosekunden gegeben.

4 Schlußfolgerung

- Ein EXO-Kernel-basiertes Betriebssystem unterscheidet das Ressourcenmanagement von dem Ressourcenschutz.
- Exokernelsysteme bestehen aus einem kleinen EXO-Kernel (Aegis, Xok, usw.) und LibOS-Bibliotheken (ExOS, usw). Der EXO-Kernel schützt die Ressourcen, während die LibOS-Bibliotheken die Ressourcen verwalten.
- Application Specific Handlers (ASHs) verringern die Antwortzeit in einem Netzwerk, sogar wenn die beteiligten Prozesse nicht geplant (scheduled) sind.
- In einem Netzwerk mit ASH-basiertem Betriebssystem ist die Antwortzeit orthogonal zur Zahl der laufenden Prozesse.
- EXO-Kernel sind geeignet für den Aufbau schneller Server (NFS, Web-Server). Die E/A-zentralisierte Natur dieser Server macht Betriebssystem-orientierte Optimierung sinnvoll.
- Schnelle Netzwerke können Exokernelsysteme ausnutzen, weil ein Exokernel eine geringere Kreuzung (Kernel/Anwendungen) liefert.

Literatur

[BrHM+97] Héctor Briceño, Russell Hunt, David Maziéres, Thomas Pinckney, Robert Grimm, John Jannotti, Kenneth Mackenzie, Dawson R. Engler, Frans Kaashoek und Gregory R. Ganger. Server Operating Systems. *Proceedings of the 16th SOSP*, Oktober 1997.

[Digi96] Digital. *Unix Documentation Library*. Digital. 1996.

[EnKa95] Dawson R. Engler und Frans Kaashoek. Exterminate All Operating System Abstractions. *HotOS-V*, Mai 1995.

[EnKJ95] Dawson R. Engler, Frans Kaashoek und James O'Toole Jr. Exokernel: An Operating System Architecture for Application-Level Resource Management. *15th Symposium on Operating Systems Principles*, Dezember 1995.

[Hors93] Wettstein Horst. *Systemarchitektur*. Hanser. 1993.

[Howe97] Denis Howe. *Free On-Line Dictionary of Computing*. http://wombat.doc.ic.ac.uk/foldoc/index.html. 1997.

[LaSp79] B.W. Lampson und R.F. Sproull. An Open Operating System for a single-user Machine. *Proceedings of the Seventh ACM Symposium on Operating SystemsPrinciples*, Dezember 1979, S. 98–105.

[ThLe93] C.A. Thekkath und H. M. Levy. Limits to low-latency communication on high-speed Networks. *ACM Transactions on Computer Systems*, Mai 1993, S. 179–203.

[WaEK96] Deborah A. Wallach, Dawson R. Engler und Frans Kaashoek. ASHs: Application-Specific Handlers for High-Performance Messaging. *ACM SIGCOMM Computer Communication Review* 26(4), Oktober 1996, S. 1–13.

[WEKG96] Deborah A. Wallach, Dawson R. Engler, Frans Kaashoek und Gregory R. Ganger. Server Operating Systems. *SIGOPS European Wokshops*, September 1996, S. 9–11.

Neue Betriebssystemkonzepte für die Kommunikation: Nemesis

Martin Schneider

Kurzfassung

Im Rahmen der Arbeiten auf dem Gebiet der Hochleistungskommunikation wurde schon bald festgestellt, daß schnelle Netze und leistungsfähige Anwendungen alleine nicht ausreichen, um einem Anwender eine hohe Leistungsfähigkeit zur Verfügung zu stellen. Gefragt sind unter anderem auch Betriebssysteme, die verschiedene Dienstqualitäten an eine Anwendung weiterreichen können. Unterstützende Mechanismen sind unter anderem echtzeitfähige Schedulingmechanismen, Mikrokernel, Protokollmechanismen ohne Kopiervorgänge etc. Das Nemesis Betriebssystem der Universität Cambridge stellt Anwendungen vielerlei Mechanismen zum Multiplexen von Ressourcen zur Verfügung, wie jedoch diese Ressourcen von Anwendungen genutzt werden, bleibt letzteren selbst überlassen. Dieser Ansatz wird in diesem Seminar mit Betonung der kommunikationsrelevanten Komponenten vorgestellt.

1 Einleitung

1.1 Anforderungen an ein Betriebssystem für Echtzeitanwendungen

Zu den Aufgaben eines Betriebssystems gehört neben der Schnittstellenfunktion zwischen der Hardware und den Applikationen auch die Verteilung der Betriebsmittel wie Prozessor, Speicherplatz und I/O-Funktionalitäten auf die verschiedenen gleichzeitig ablaufenden Prozesse. Dieses sogenannte Scheduling wird meistens durch die Virtualisierung der verschiedenen Betriebsmittel realisiert, man denke nur an virtuellen Speicherplatz oder virtuelle Prozessoren. Oft geht durch diese Virtualisierung der Bezug zum Ablauf der realen Zeit verloren, z.B. Prozessorzeit wird anhand dimensionsloser Faktoren vergeben, wie dem nice-Befehl unter Unix. Man erreicht eine Virtualisierung der Zeit. Für zeitkritische Multimedia- oder Hochgeschwindikeitsanwendungen ist diese Art der möglichen Einflußnahme der Applikation bzw. des Benutzers auf das Scheduling nicht ausreichend. Ein Betriebssystem für Echtzeitanwendungen sollte also den Applikationen mehrere Quality of Service (QoS) Parameter für das Scheduling zur Verfügung stellen, anhand derer dann die einzelnen Applikationen die gewünschten Ressourcen zugeteilt bekommen. Wünschenswert wäre zusätzlich, daß die Applikationen einen Teil des Schedulings innerhalb ihres Ablaufs nach eigenen Gesichtspunkten

vornehmen könnte, der Kernel also nur Scheduling zwischen Applikationen durchführen müßte. Dies würde allerdings erfordern, daß die zu einer Applikation gehörenden Prozesse dieser eindeutig zugeordnet werden können. Diese Überlegungen wurden an der University of Cambridge mit dem Betriebssystemprojekt Nemesis in die Tat umgesetzt. Nemesis ist inzwischen für Pentium und verschiedene DEC-Plattformen in stabilen Versionen verfügbar. Es handelt sich um den Prototyp eines Betriebssystemkernels für Multimediaworkstations.

1.2 Entwurfsprinzipien für Nemesis

Nemesis ist ein Betriebssytemkernel, der den einzelnen Applikationen Quality of Service (QoS), soweit überhaupt möglich, garantiert. QoS bedeutet in diesem Zusammenhang Prozessorzeit, Speicher, Bandbreite einer Netzwerkverbindung und Echtzeitfähigkeit, falls gewünscht. Eine Applikation wird innerhalb einer sogenannten *Domain* organisiert. Eine *Domain* in Nemesis entspricht einem ausführbaren Programm, allerdings werden im Gegensatz zu z.B. Unix-Prozessen alle Funktionalitäten einer Applikation in einer *Domain* gekapselt. In Unix wird eine Applikation oft durch verschiedene Prozesse realisiert, die in Client-Server Beziehungen stehen. Einzelne Server-Prozesse erbringen Dienste für viele Applikationen und verbrauchen Resourcen, die dann dem Server-Prozeß und nicht der einzelnen Applikation zugeordnet werden. QoS-Management ist nicht möglich, die von einer Applikation verbrauchte Rechenzeit oder der Speicherplatz ist nicht dem Client zuzuordnen. Nemesis dagegen verlangt, daß der größte Teil der von einer Applikation benötigten Dienste auch von dieser erbracht wird, Kommunikation und Datenaustausch zwischen *Domains* sollten sich auf ein Minimum beschränken. Die Kapselung in Nemesis führt allerdings zu mehr Komplexität in den Applikationen. Diese wird noch erhöht, indem die einer *Domain* vom Kernel-Scheduler zugeteilten Resourcen von den *Domains* nach eigenen Strategien innerhalb ihrer zugeteilten Prozessorzeit verbraucht werden. Eine *Domain* kann mittels Threads parallele Abläufe erreichen. *Inter-Domain-Communication (IDC)* wird mittels verschiedener Mechanismen realisiert, unter anderem durch einen Ein-Adreß-Raum *(SAS - Single Address Space)*, *Remote Procedure Call (RPC)* und dem *RBUFS*-Mechanismus, wobei *RBUFS* hier für Remote Buffers steht. Auch das Memory-Management, die Abbildung virtuellen Speichers auf physikalischen Speicher, wird der einzelnen User-*Domain* eigenverantwortlich übertragen. Insgesamt ergibt sich eine bedeutend höhere Komplexität der Programmierung von Applikationen, welche aber durch standardmäßig mitgelieferte Shared-Libraries wieder abgemildert wird. Nemesis besitzt eine vertikale Struktur, die Applikationen ruhen wie Säulen auf dem Kernel. Dieser ist Dank der Verlagerung der Komplexität in die *Domains* recht klein ausgefallen, es werden gerade einmal ca. 250 Instruktionen benötigt.

1.3 Hochgeschwindigkeitskommunikation und Nemesis

Kommunikation in einem Netzwerk wird meistens mittels verschiedener miteinander kommunizierender Prozesse auf einem Rechner abgewickelt. Um eine SMTP-Verbindung aufzubauen, muß z.B. der Name-Service befragt werden, TCP-Pakete werden von einem anderen Prozeß generiert, ein Netzwerkkartentreiber erzeugt Ethernetframes etc. Diese Vielfalt an verschiedenen Prozessen, die ineinader verzahnt einen

Dienst erbringen, widerspricht dem vertikalen Prinzip von Nemesis. Hier ist eine *Domain* selbst zuständig für die komplette Implementierung der Schichtenarchitektur. Eine *Domain* muß letztendlich z.B. Ethernet-Rahmen erzeugen, die sie an den Treiber der Netzwerkkarte übergibt. Nemesis unterstützt vor allem verbindungsorientierte Kommunikation, Treiber für ATM-Karten sind schon für Nemesis realisiert. Paketorientierte Hochgeschwindigkeitskommunikation wird durch den *RBUFS*-Mechanismus beschleunigt, zu verarbeitende Pakete werden nicht zwischen den beteiligten *Domains* kopiert, es werden nur Zeiger auf einen Pufferspeicher ausgetauscht, der die Pakete enthält. Generische Gerätetreiber bzw. eine vorgegebene Treiber-Architektur ermöglichen schnelle verbindungsorientierte Kommunikation der Applikationen mit den Netzwerkkomponenten.

2 Struktur von Nemesis

2.1 Nemesis Kernel und Domains

Der Nemesis-Kernel besteht aus dem Scheduler zum Multiplexen der Resourcen auf die verschiedenen *Domains* und dem sogenannten *Nemesis Trusted Supervisor Code (NTSC)*. Es handelt sich dabei um Instruktionen für die Interprozeßkommunikation, Mechanismen zur Interrupt-Kontrolle, für Speicherzugriffsfehlerbehebung und TLB-Fehlzugriffsbehandlung, einem Timer und Initialisierungsmechanismen nach dem Booten des Systems. *NTSC* ist zusätzlich die für Benutzer-*Domains* systemunabhängige Schnittstelle zur Hardware, die Implementierung des *NTSC* dagegen ist plattformabhängig. Die einzelnen *Domains* besitzen für die *IDC* jeweils einen sogenannten *Domain Control Block (DCB)*, d.h. Shared Memory, auf den die anderen *Domains* zur Kommunikation Zugriff haben. Zusätzlich ist jeder *Domain* eine *Scheduling-Domain* zur Bestimmung der CPU-Zeitzuteilung und eine *Protection-Domain* zum Verwalten der Speicherzugriffsrechte dieser *Domain* zugeordnet. Hardware wird mit ihren Treiber-*Domains* über Stub-Register (Stub=Stummel, Kippe) angesprochen, deren Inhalt dann in den Adreßraum der Treiber-Domänen kopiert werden (Abb. 1). Alle Betriebssystemschnittstellen werden mit einer *Interface Definition Language (IDL)* namens *MIDDL* plattformunabhängig als abstrakte Datentypen spezifiziert. *MIDDL* unterstützt den Ein-Adreß Raum und die Migration von Betriebssystemcode in die Applikation. Man kann sich die Funktionsweise dieser Schnittstellen wie Prozeduraufrufe vorstellen. Es werden die in *MIDDL* definierten Paramater von der aufrufenden Instanz an die aufgerufene übergeben, diese liefert die in *MIDDL* definierten Ergebnisse. Auch Fehler bzw. Ausnahmeverfahren können so behandelt werden.

2.2 Virtual Processor und Nemesis Domain

Nemesis präsentiert einer *Domain* den oder die realen Prozessoren eines Rechners mit Hilfe des sogenannten *Virtuellen Prozessor Interfaces*, einer plattformunabhängigen Abstraktion für CPU-Kontextwechsel und *IDC*. Dieses ist wohlbemerkt nur eine systemunabhängige Schnittstelle und kein virtueller Prozessor in dem Sinne, daß jede

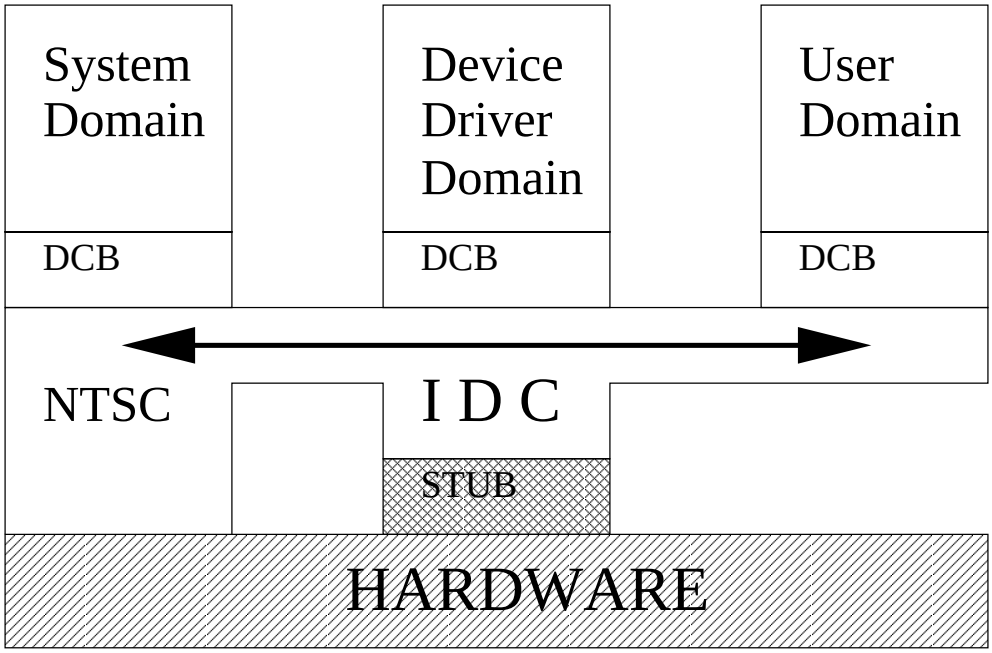

Abbildung 1: Vertikale Struktur von Nemesis

Domain ihren eigenen virtuellen Prozessor besitzt. Das *Virtuelle Prozessor Interface* und die Routinen für *IDC* sind in *NTSC* implementiert. Die Nemesis-*Domain* ist für elementare Dienste wie Zugriffskontrolle auf den Speicher mittels des *Stretch Allocators*, Unterstützung der *IDC* mit einem *Binder*, der Shared Memory zweier *Domains* verwaltet, Umleiten und Abfangen von Hardwareinterrupts auf die entsprechende Device Driver *Domain* etc. verantwortlich. Der *Stretch Allocator* teilt den von Nemesis umgestetzten Ein-Adreß-Raum in *Stretches*, Portionen des Speichers, die den verschiedenen *Domains* zugeordnet werden. Der *Binder*, eine System-*Domain*, baut *IDC*-Verbindungen zwischen zwei *Domains* auf und wieder ab.

2.3 Inter Domain Communication (IDC)

Nemesis liefert verschiedene Mechanismen zur Unterstützung der *IDC*. Grundsätzlich sollte sie zwischen User-*Domains* aber so wenig wie möglich auftreten, da sie eigentlich dem vertikalen Prinzip von Nemesis widerspricht. Natürlich läßt sich *IDC* nicht ganz vermeiden, z.B. zwischen einer Treiber-*Domain* und einer Applikation. Direkte *IDC* wird vom Nemesis-Kernel durch sogenannte *Events* ermöglicht. Jede *Domain* besitzt dazu *Channel-Endpoints*, zwei miteinaner verbundene Channel-Endpoints zweier *Domains* ergeben eine asynchrone Simplex-Verbindung. Auf dieser können 64-Bit-Werte direkt ausgetauscht werden, insbesonders zur Synchronisation der Zugriffe auf Shared Memory der beiden *Domains*, um Inkonsistenzen bzw. gleichzeitigen Zugriff auszuschließen. Dieser Mechanismus wird dann *Event* genannt.

IDC via Shared Memory wird durch die globale Gültigkeit von Pointern in Datenstrukturen erleichtert, natürlich kann eine *Domain* ihren Speicher auch vor Zugriff anderer *Domains* durch Setzen entprechender restriktiver Zugriffsrechte in der *Protection-Domain* schützen. Remote Procedure Call (RPC) zwischen auf einem Rechner laufenden *Domains* wird für Nemesis-Applikationen verwendet, die trotz der vertikalen

Struktur auf eine Client-Server-Beziehung angewiesen sind. Nemesis unterstützt die Implementierung von RPC durch Libraries, die Komplexität des Verbindungsaufbaus bleibt dem Programmierer der User-*Domains* verborgen.

Gerade für Hochgeschwindigkeitskommunikation interessant ist der *RBUFS*-Mechanismus. Zwischen *Domains* werden Pointer auf Datenstrukturen ausgetauscht, so muß z.B. ein eingehendes Ethernetpaket nicht zwischen dem Treiber der Netzwerkarte und dem für den auf der Schichtenarchitektur höherliegenden Applikation kopiert werden, es werden also nur Pointer auf Pakete weitergerecht. Auf *RBUFS* soll aber im Kapitel 3. Nemesis und Hochgeschwindigkeitkommunikation genauer eingegangen werden, da gerade dort *RBUFS* eine entscheidende Rolle spielen.

2.4 Scheduling und Quality of Service

Scheduling bezeichnet diejenige Betriebssystemfunktionalität, welche Hardwareressourcen auf gleichzeitig ablaufende Prozesse multiplext. Nemesis ermöglicht im Gegensatz zu anderen Betriebssystemen neben dem Scheduling von CPU auch das von Speicherplatz und der Bandbreite von Netzwerkkomponenten. Die Mechanismen hierzu sind sich recht ähnlich, beispielhaft wird auf Prozessor-Scheduling eingegangen.

Herkömmliche Betriebsysteme lassen den Applikationen meist nur geringen Einfluß auf das Scheduling, die Startegie ist vom Betriebssytem vorgegeben. Der Kernel hat die totale Kontrolle über die gleichzeitig ablaufenden Prozesse. Diese werden oft in nebenläufige Kontrollfäden (Threads) geteilt. Dieses Multithreading führt zwar zu im Vergleich zu Prozeß-Wechseln effizienteren Prozessor-Kontextwechseln, nachteilig wirkt sich allerdings aus, daß dieses von unkooperativen Prozessen ausgenutzt werden kann, indem ein Prozeß mehrere Threads benutzt und somit Resourcen an sich bindet. Nemesis unterscheidet zwischen Inter-Prozeß (*Inter-Domain*) und Intra-Prozeß (*Intra-Domain*) Scheduling. Der Kernel-Scheduler multiplext zwischen *Domains*, QoS wird nach Parametern erbracht, die die einzelnen *Domains* dem Kernel-Scheduler in ihren *Domain Control Blocks (DCB)* anzeigen (Abb. 1). Bei diesen Parametern handelt es sich um ein Tupel (p,s,x,l), das die von der *Domain* benötigte Prozessor-Bandbreite angibt:

- p Die Periode der *Domain* in ns.
- s Der CPU-Zeitschlitz, der einer *Domain* zugeteilt wird, in ns.
- x Ein Flag, das anzeigt, ob die *Domain* extra CPU-Zeit animmt.
- l Latenz in ns, d.h. ein Hinweis, wie zeitkritisch die Anwendung ist.

Diese Werte werden mittels des *Event-Mechanismus* zwischen dem Kernel-Scheduler und der entsprechenden *Domain* ausgetauscht. Er versucht die QoS-Anforderungen der einzelnen *Domains* unter der Bedingung $\sum_{i=1}^{n} \frac{s_i}{p_i} \leq 1$ umzusetzten. Der Ein-Adreß-Raum von Nemesis unterstützt eine hohe Rate von Kontextwechseln zusätzlich, da diese nicht zu der sonst zu erwartenden hohen Zahl von Cache-Misses bzw. TLB-Misses führt. Hat eine *Domain* CPU-Zeit zugeteilt bekommen, muß sie diese nach eigenen Kriterien auf ihre möglicherweise existenten Threads aufteilen. Dafür werden von Nemesis keine

Vorgaben gemacht, es unterstützt die Programmierung des *Intra-Domain*-Scheduling durch Thread-Libraries, um die steigende Komplexität der Applikationen abzumildern. Es sind sowohl preemptive als auch nicht preemtive Thread-Packages in Nemesis als Shared-Libraries vorhanden, eine *Domain* kann sogar auf internes Scheduling verzichten. Eine Applikation wählt ihre eigene interne Scheduling-Strategie ähnlich einem Benutzer, der mit seinem eigenen Window-Manager arbeitet. Im *DCB* einer *Domain* exisitiert ein Vektor namens *Activation-Handler*, der nach der in der *Domain* festgesetzten Scheduling-Strategie die der *Domain* zugeteilte CPU-Zeit auf die Threads multiplext. Eine *Domain* erzeugt Last immmer in Abhängigkeit davon, ob die ihr zugeteilte Zeit der ihr anhand des Tupels (p,s,x,l) garantierten Zeit oder im System übrigen Zeit entspricht, die der *Domain* jederzeit entzogen werden kann.

2.5 Memory-Management

Nemesis ist ein Ein-Adreß-Raum-Betriebssystem, d.h. die Übersetzung von virtuellen in physikalische Adressen ist in allen *Domains* gleich. Nemesis steht hier in Kontrast zu allen existierenden kommerziellen Betriebssytemen wie Unix und WindowsNT. Dort besitzt jeder Prozeß seinen eigenen virtuellen Adreßraum. Der Ein-Adreß Raum von Nemesis besitzt natürlich einige Vorteile:

- Programmcode wird mehreren *Domains* zugänglich gemacht, Shared Libraries.
- Kontextwechsel werden erleichtert und beschleunigt.
- Persistente Speicherobjekte sind möglich.
- *Inter-Domain-Communication* beschleunigt, siehe *RBUFS*.

Man benötigt aber eine Zugriffskontrolle der Form, daß eine *Domain* einen Teil ihres Speichers vor dem Zugriff anderer *Domains* schützen kann. Der virtuelle Speicher einer Maschine wird in *Pages* geteilt, die den einzelnen *Domains* zugeordnet werden. Dieses ist die kleinste Speichereinheit, die von einer *Domain* entweder gesperrt oder freigegeben werden kann. Der virtuelle Speicher wird auf den physikalischen Speicher, also RAM, Festplatten etc. auf sogenannte *Frames* physikalischen Speichers abgebildet. Eine virtuelle Page ist genauso groß (in Bytes) wie ein physikalischer *Frame*. Jede *Domain* besitzt ihre sogenannte *Protection-Domain*, eine Zusammenfassung ihrer Zugriffsrechte auf die einzelnen existenten Speicherplätze. Eine *Domain* bekommt sogenannte Stretches zugeteilt, es handelt sich um zusammenhängende Pages, also virtuellen Speicher, die sich nicht überlappen und deren Größe im späteren Verlauf auch nicht verändert werden kann. Innerhalb einer User-*Domain* regelt der sogennate *Stretch-Driver* die Zugriffe auf die der *Domain* zugeordneten *Stretches*, insbesonders Page-Zugriffsfehler und die Abbildung der virtuellen Adresse auf die physikalische Adresse. Er ist mit dem in System VR4 bekannten Segment-Driver vergleichbar. Die *Domain* hat selbst die volle Verantwortung, z.B. für die Auslagerung von Hauptspeicher auf eine Festplatte, sie kann eigene Strategien zur Optimierung der Speicherzugriffe verwenden und wird dabei auch wieder von Shared Libraries von Nemesis unterstützt. Der systemweite *Stretch-Allocator* teilt den *Domains* ihre Stretches zu, zieht sich dann aber aus deren Verwaltung zurück (Abb. 2).

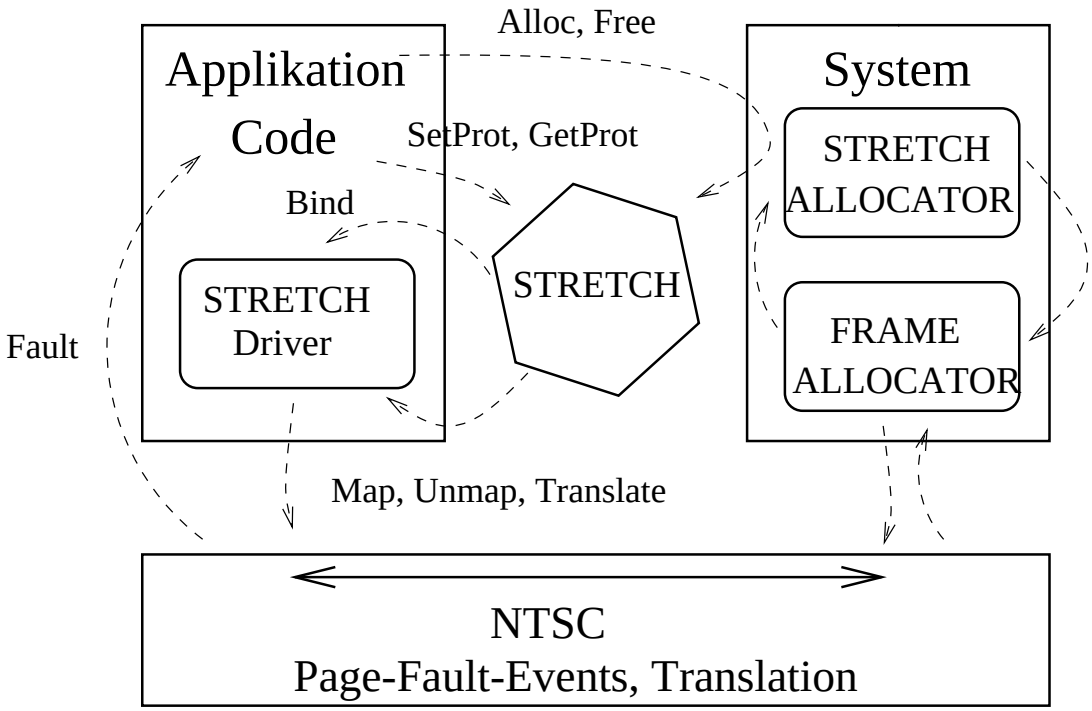

Abbildung 2: Virtueller Speicher und Memory Management

3 Nemesis und Hochgeschwindigkeitskommunikation

3.1 Nemesis-Prinzipien kontra Kommunikationsprotokollhierachie

Das vertikale Prinzip von Nemesis widerspricht eigentlich dem hierachischen Aufbau der ISO/OSI oder TCP/IP-Protokollstapel, welche in der Regel mit in Client-Server-Beziehungen stehenden Prozessen umgesetzt werden. Diese Prozesse erbringen für verschiedene Applikationen Dienste, ihre verbrauchten Resourcen werden aber den Server-Prozessen und nicht den die Dienste benutzenden Client-Prozessen zugeordnet. Das vertikale Prinzip für QoS von Nemesis verlangt dagegen, daß eine Applikation, die Netzwerkverbindungen erfordert, diese alleine betreibt. Das bedeutet im Endeffekt, daß sie Pakete der Schicht-2, z.B. Ethernetframes oder ATM-Pakete, erzeugt bzw. annehmen kann. Die Treiber-*Domains* der Netzwerkhardware multiplexen eingehende Pakete der existierenden Verbindungen auf die gleichzeichzeitig ablaufenden Applikationen. Nemesis unterstützt somit insbesonders verbindungsorientierte Kommunikation, paketorientierte Kommunikation wird auf verbindungsorientierte Kommunikation abgebildet. Nemesis regelt Netzwerkverbindungen mit Hilfe des *Flow-Managers*, eine System-*Domain*, die für eine User-*Domain* eine gewünschte Netzwerkverbindung über alle Schichten des unterstützten Protokollstapels aufbaut. Eine Applikation muß aus einem eingehenden Ethernetframe IP-Pakete, aus diesen dann TCP oder UDP-Pakete und den Port herausziehen und die Funktionalität der Anwendungschicht realisieren und beim Senden eines Ethernetframes in diesem ineinander verschachtelt die Pakete der höheren Schichten einpacken. Die Komplexität der Applikationen steigt beträchtlich, die aber auch hier durch Shared-Libraries für den Programmierer abgemildert

wird. Insbesonders ist für jede Schicht des TCP/IP Protokollstapels eine eigene Library vorhanden, die miteinander kombiniert werden können.

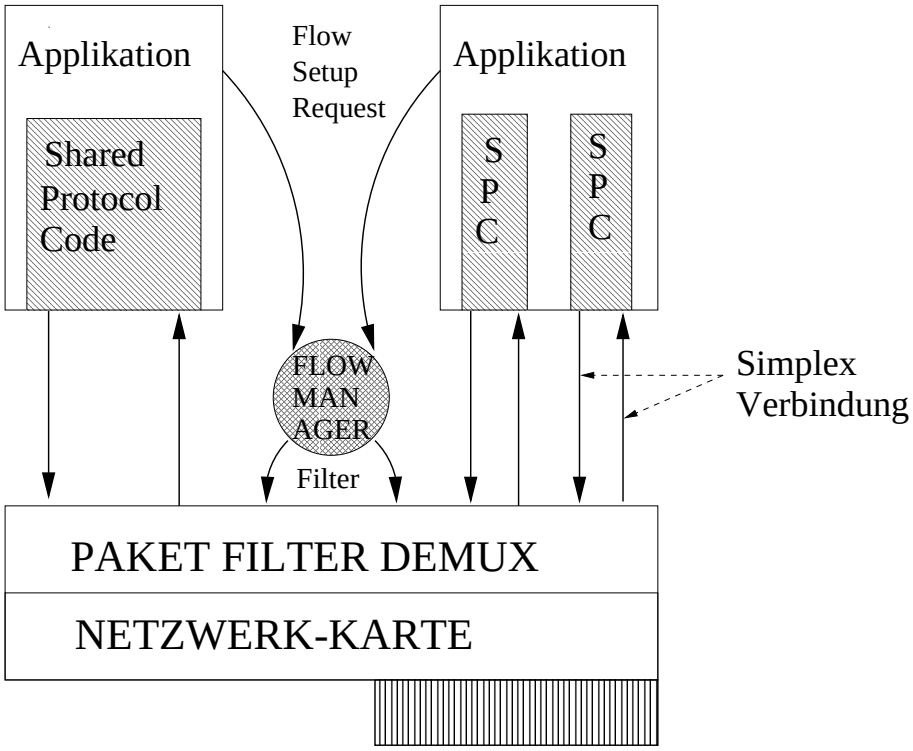

Abbildung 3: Flow-Manager und Verbindungsaufbau

3.2 Integrated Layer Processing ILP

Der *Flow-Manager* bindet unter Verwendung des *RBUFS*-Mechanismus' die User-*Domain* mit der Treiber-*Domain* der Netzwerkkarte. Die User-*Domain* übergibt als Argumente z.B. die Zieladresse der Anwendungschicht, die Portnummer, den gewünschten Transportdienst und die Adresse und Größe von benötigtem bzw. von der User-*Domain* bereitgestelltem Pufferspeicher an den Flow-Manager. Dieser beliefert die Applikation mit allen zum Verbindungsaufbau benötigten Daten, so z.B. mit der lokalen und der IP-Adresse des Zielhosts, der lokalen Ethernetadresse und der Ethernetadresse der nächsten Bridge oder des nächsten Routers und stellt ihr alle netzwerkspezifischen Funktionen zur Verfügung. Der *Flow-Manager* selbst muß z.B. mittels des ARP-Protokols die internen Routing-Tabellen ständig aktualisieren. Die User-*Domain* ist nun in der Lage, die vom vertikalen Prinzip von Nemesis geforderten Schicht-2-Pakete zu generieren bzw. von der Treiber-*Domain* zu empfangen, der *Flow-Manager* zieht sich aus der Kommunikation dieser *Domain* zurück, die *Domain* ist unabhängig von allen anderen Applikationen in der Lage, die Kommunikation abzuwickeln. Da Nemesis vor allem verbindungsorientierte Verbindungen unterstützt, wird z.B. auch UDP verbindungsorientiert realisiert. Nemesis verlangt eigentlich von den Netzwerkkarten und deren Treiber-*Domains* die Fähigkeit, bestehende Verbindungen selbst zu multiplexen. ATM-Karten haben diese Fähigkeiten, Ethernetkarten in der Regel nicht. Für diese muß zwischen Treiber-*Domain* und Applikation-*Domain* ein *Paket-Filter-Demux* eingeschaltet werden, welcher eingehende Pakete auf die einzelnen User-*Domains* multiplext. Das ganze Prinzip des Verbindungsaufbaus mittels des *Flow-Managers* und

die von anderen *Domains* unabhängige Kommunikation einer Applikation wird *Integrated Layer Processing (ILP)* genannt, da eine Applikation das ganze Schichtenmodell selbst realisieren und abwickeln muß (Abb. 3). QoS für Netzwerkkommunikation bedeuted Bereitstellung von Bandbreite. Nemesis benutzt einen dem Prozessorzeit-Scheduling ähnlichen Verfahren, um den konkurrierend auf die Netzwerkkomponente zugreifenden Applikationen einen Teil der Übertragungsbandbreite zuzuteilen, indem jede User-*Domain* ihren Bedarf an Bandbreite der entsprechenden Treiber-*Domain* mitteilt, welche diese dann auf die einzelnen *Domains* verteilt.

3.3 Geschwindigkeitsoptimierung: Der RBUFS-Mechanismus

Der *RBUFS*-Mechanismus dient zum schnellen Austausch von paket-basierenden Daten zwischen verschiedenen *Domains*, insbesonders zwischen der Treiber-*Domain* einer Netzwerkkomponente und einer User-*Domain*. Hintergedanke ist, daß bei der *IDC* nicht jedesmal ein ganzes Paket kopiert wird, sondern nur Zeiger auf Datenstrukturen ausgetauscht werden müssen. Dieses Prinzip wird mit je zwei *Event-Channels* zwischen zwei *Domains* realisiert und dank des Ein-Adreß-Raums von Nemesis effektiv unterstützt.

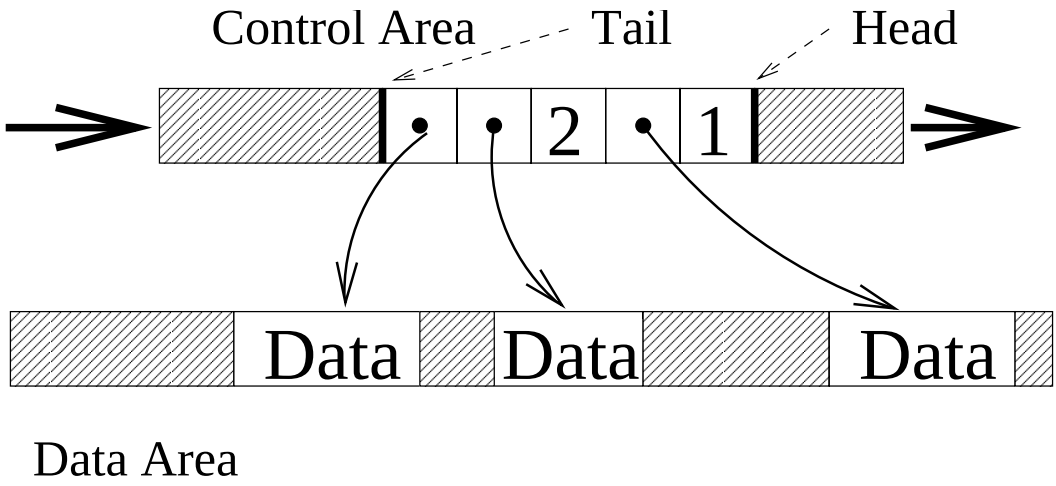

Abbildung 4: Event-Channel für *RBUFS*-Mechanismus

Die *Control-Area* entspricht einem *Event-Channel*, die Data-Area entspricht Shared-Memory der beiden *Domains* (Abb. 4). Die zwischen den *Domains* via *Event-Channels* ausgetauschten Paketinformationen werden *IOrecs* genannt. Sie bestehen im Grunde nur aus einem Zeiger auf die im Shared Memory gespeicherten Pakete. Will eine *Domain* Pakete empfangen, überträgt sie Zeiger auf leeren Pufferspeicher an die Treiber-*Domain*, diese füllt den Puffer und bringt die möglicherweise veränderten *IOrecs* auf den neusten Stand. Umgekehrt schickt eine sendewillige *Domain IOrecs* mit Zeigern auf die zu verschickenden Pakete an eine Empfängerdomain, in der Regel ein Treiber, welche die Daten weiterverarbeiten kann. *RBUFS* ermöglicht einfaches Entfernen und Hinzufügen von Headern an einzelne Pakete durch Anpassung der *IOrecs*, ohne daß die Pakete kopiert werden müssen. Dabei wird soviel Speicherplatz für ein Paket im Puffer reserviert, wie für das Paket mit allen Headern aller Schichten benötigt wird, um das Speichermanagement in Bezug auf die Geschwindigkeit zu optimieren. Wichtig zu Erwähnen ist auch, daß die von einer Treiber-*Domain* verbrauchten Resourcen derjenigen Applikation zugerechnet werden, welche den Dienst des Treibers in Anspruch genommen hat (Abb. 5).

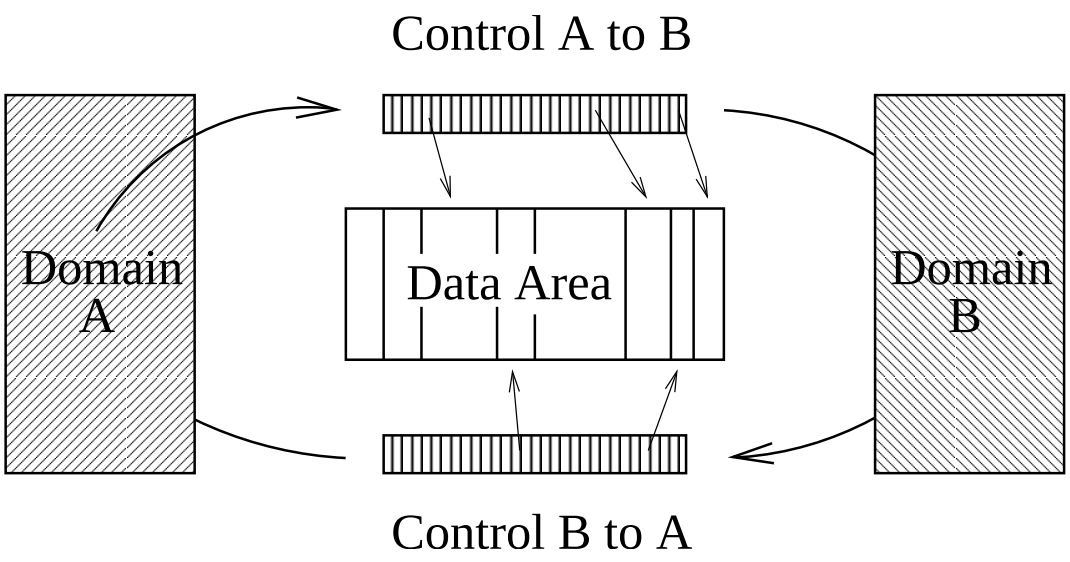

Abbildung 5: *RBUFS*-Mechanismus

3.4 Geräte-Treiber und I/O-Unterstützung in Nemesis

Hardware wurde bisher oft unter der Annahme konstruiert, daß ihre asynchron dem System gesendeten Interrupts im Betriebssytem sofort zu Kontextwechseln führen und der zugehörige Treiber der Hardwarekomponente sofort zur Ausführung kommt. Dieses Vorgehen widerspricht aber dem in Nemesis garantierten QoS einer gerade auf dem Prozessor ablaufenden *Domain*, die nicht unterbrochen werden darf. Diese Hardware kann dann aber oft nicht mit einer Verzögerung der durch den Interrupt angezeigten Aktion mangels Puffer-Register in der Hardware fertigwerden. Nemesis fängt diese Fälle mit Hilfe sogenannter *Stub-Register ab*, die diese Interrupts zwischenspeichern und den Treiber-*Domains* der Hardware signalisieren, sich vom Kernel-Scheduler schnell aufrufen zu lassen. Diese Taktik verhindert Timeouts bzw. sonstige Fehler in der Peripherie und unterbricht nicht unnötig zeitkritische Anwendungen. Moderne Multimedia-Geräte sind inzwischen nicht mehr so latenz-anfällig, sie besitzen interne Puffer und können auf Überlast-Situtionen entsprechend reagieren. QoS, z.B. Bandbreite für eine User-*Domain*, kann nur innerhalb einer bestehenden Verbindung kontrolliert und damit überhaupt realisiert werden, weswegen Nemesis ausschließlich verbindungsorientierten Datenaustausch zwischen Treibern und Applikationen kennt. Ein Treiber muß allerdings nur wenige I/O-Dienste erbringen, der Hauptteil wird innerhalb der User-*Domain* mit Hilfe von Shared-Libraries geleistet. Alle Treiber in Nemesis besitzen die gleiche Architektur (Abb. 6), es wird zwischen dem *Device-Data-Path-Modul (DDM)* und dem *Device Control-Path-Modul (DCM)* unterschieden. Diese können sowohl in einer als auch in zwei Treiber-*Domains* umgesetzt werden, welche mit den einzelnen Clienten mit dem *RBUFS*-Mechanismus kommunizieren. Man erhält also eine Out-of-Band-Steuerung der Kommuniktation.

Die vom *Data-Path-Modul (DDM)* erbrachten Dienste sind die folgenden:

- Umsetztung einer Datenstromadresse: Diese sollte unabängig sein von deren Ziel (Client), um Broadcasts und Multicasts realisieren zu können.
- Protection: Zugriffsregelung und Schutz einzeln für jede I/O-Verbindung (Client)

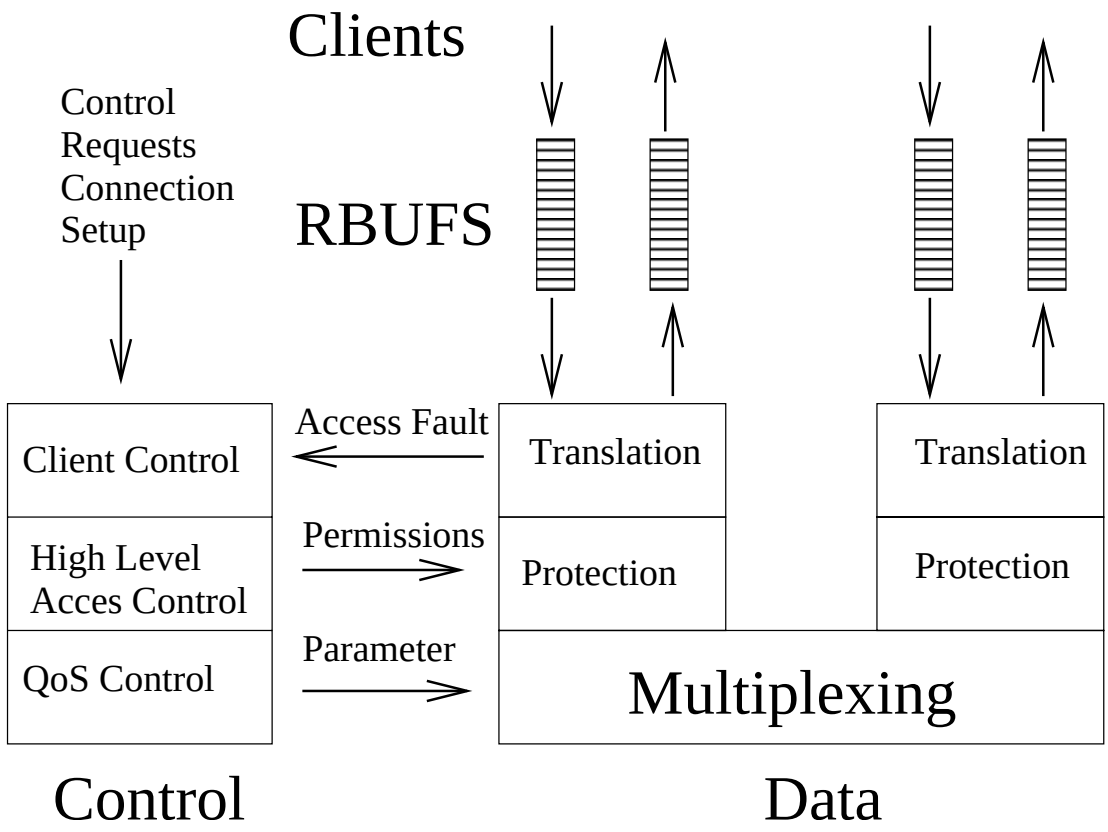

Abbildung 6: Nemesis Device Driver Architektur

- Multiplexing: Verteilung der physikalischen Ressourcen, Scheduler, QoS-Management

Der wesentliche Unterschied zu Treibern traditioneller Betriebssysteme ist in der Multiplexing-Funktionalität des *DDM* zu finden. Hier wird Scheduling von Netzwerkkomponenten nach QoS-Paramtern zusätzlich zum Prozessor-Scheduling betrieben. Die Scheduling-Algorithmen können komponentenspezifisch ausfallen, Nemesis macht hier den Herstellern keine Vorgaben. Das *Device-Control-Path-Modul* steuert das *Data-Path-Modul*, integriert neue Clienten und überwacht QoS der bestehenden Verbindungen, Zugriffsrechte und Zugriffspolitik der einzelnen Datenströme werden reguliert.

4 Anwendungen für Nemesis

4.1 Window-Manager

Nemesis bietet einen generischen Window-Manager *WSSvr*, der Tastatur und Maus-Ereignisse an die zugehörigen Anwendungen, die einzelnen Fenstern zugeordnet sind, weitergibt. Er bildet auch den geordneten Stapel der geöffneten Fenster auf den Bildschirm ab. Er ist die Schnittstelle zwischen dem Betriebssytem und dem vom Benutzer gewählten Window-Manager. Bisher wurde ein einfacher Windowmanager namens *SWM (Simple Window Manager)* implementiert.

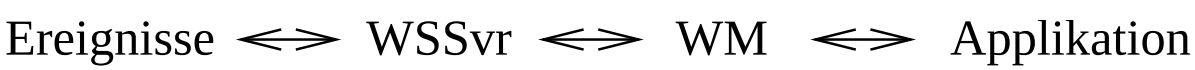

Abbildung 7: Generischer Windowmanager WSSvr

4.2 Multimedia

Nemesis ist als Prototyp eines Betriebssystems für Multimediaworkstations gedacht, entsprechend gibt es schon Video und Audio-Module. *RawVideoPlay* ist eine Applikation, die einen Videodatenstrom einer Fore-Systems AVA-200 und AVA-300 Kamera, welche Videodaten in ATM-Paketen erzeugt, darstellen kann. *NetworkAudio* ist ein Treiber für Sound Blaster Karten, der Audiodaten aus beliebigen Quellen übernehmen kann. Beide Anwendungen werden in der Videokonferenzdomain *Conference* vereint, man kann auch mit mehreren Partnern gleichzeitig konferieren.

4.3 Scriptsprache Clanger

Speziell für Nemesis entwickelt wurde die Scriptsprache *Clanger*, die neben der Konfiguration des Systems auch Start-of-the-Day Scripten zum Booten ausführt. *Clanger* wird zum Starten einzelner *Domains* mit in den Scripten definierten QoS-Paramteren ähnlich der rc-Files in Unix benutzt. Es bietet auch noch eine interaktive Schnittstelle, die mit den bekannten Unix-Shells vergleichbar ist.

5 Zusammenfassung

Bei Nemesis handelt es sich um den Prototyp eines Betriebssytemkernels für Multimediaworkstations, der am Computer Laboratory der University of Cambridge in der SRG (Systems Research Group) realisiert wurde. Das Projekt entstand aus der Einsicht, daß herkömmliche Betriebssysteme moderne Multimediaanwendungen mit deren Bedarf an Echtzeitfähigkeit nicht ausreichend unterstützen. Nemesis bietet die Möglichkeit, einer Anwendung die benötigten Ressourcen bedarfsgerecht zur Verfügung zu stellen, und diese mit einem Bezug zur realen Zeit zu quantifizieren. Eine Anwendung kann nicht nur ihren Bedarf an Prozessorzeit, sondern auch den anderer Betriebsmittel, wie z.B. Bandbreite einer Netzwerkverbindung dem Kernel mitteilen und wird dementsprechend bedacht. Die den einzelnen Anwendungen zugeteilten Ressourcen können von diesen nach eigenen Gesichtspunkten weiter verwendet werden, der Kernel gibt nur eine grobe Strategie vor.

Das Prinzip von Nemesis ist für multimediafähige Desktoprechner sehr vielversprechend, gerade weil der verbindungsorientierte Ansatz dem neuer Netzwerktechnologien wie ATM entspricht und Nemesis konzeptuell somit gut mit diesen harmoniert. Die veröffentlichten Beispielsitzungen mit mehreren parallel ablaufenden Anwendungen zeigen, daß das Quality-of-Service Management gut funktioniert. Zeitkritische Multimediaanwednungen bekommen die von ihnen benötigten Ressourcen zugeteilt, gleichzeitig ablaufende konventionelle Anwendungen arbeiten mit den restlichen Betriebsmittel, ohne die zeitkritischen Prozesse zu behindern. Die bisher schon erhältlichen Versionen für die Plattformen Pentium PC, DEC AXP-3000, RISC PC, DECchip EB164 und

EB64 Alpha lassen auf eine baldige Verbreitung hoffen. Für einen Server, der Dienste z.B. in einem Netzwerk erbringen muß, scheinen Betriebssyteme mit in Client-Server-Beziehung stehenden Prozessen besser geeignet. Die Konzepte, die hinter Nemesis stehen, werden sich bestimmt in Zukunft im Bereich der Einbenutzermultimediaworkstations durchsetzen. Ob dies auch für Nemesis selbst gilt, wird sich zeigen.

Literatur

[Auto93] Autor. Titel. *Journaltitel* Nummer des Jahrgangs(Nummer der Ausgabe), Dezember 1993, S. Seitenzahlen.

[Auto94] Autor. Titel. In *Buchtitel*. Verlag, 1994.

[BlSt96] Gerold Blakowski und Ralf Steinmetz. A Media Synchronization Survey: Reference Model, Specification, and Case Studies. *IEEE Journal on Selected Areas in Communication* 14(1), Januar 1996, S. 5–35.

[CNRS96] E. Crawley, R. Nair, B. Rajagopalan und H. Sandick. A Framework for QoS-based Routing in the Internet. *Internet Draft „draft-ietf-qosr-framework-00.txt“*, Marz 1996.

[Foru96] The ATM Forum. ATM Service Categories: The Benefits to the User. White Paper, The European Market Awareness Committee, Mai 1996.

[Göde57] Kurt Gödel. *Titel*. Verlag. 1957.

[HCCB94] David Hutchison, Geoff Coulson, Andrew Campbell und Gordon S. Blair. *Quality of Service Management in Distributed Systems*, Kapitel 11, S. 273–302. Addison Wesley. Editor: Morris Sloman, 1994.

[McSp95] David E. McDysan und Darren L. Spohn. *ATM: Theory and Application*. McGraw-Hill, New York. 1995.

[vBee12] Ludwig van Beethoven. *Titel*. Verlag. 1812.

Aktive Netzwerke

Götz Lichtwald

Kurzfassung

Ein aktives Netzwerk erlaubt es dem Anwendungsprogramm die Netzwerkknoten für seine Bedürfnisse zu konfigurieren. Dies geschieht durch Übermittlung von Programmen zum jeweiligen Netzwerkknoten, der diese speichert und bei Bedarf ausführt. Durch diese Technik wird es möglich neue innovative Protokolle schneller zum Einsatz zu bringen. Dies war bisher, bedingt durch die fehlende Flexiblität der Vermittlungssysteme, nicht möglich. Ein noch viel wichtigerer Aspekt ist jedoch, daß es diese Technik erlaubt Prokolle einzusetzen, die auf die individuellen Bedürfnisse der Anwender zugeschnitten sind.
Ziel dieser Technik ist es, die Netzlast zu senken und dennoch die Leistungsfähigkeit der (Netz-)Anwendung zu erhöhen. So sollen mit aktiven Netzwerken Videokonferenzen und andere zeitkritische Anwendungen möglich werden, die derzeit noch nicht befriedigend implementiert und angewendet werden können.

1 Einleitung

1.1 Wie funktionieren Aktive Netzwerke

1.1.1 Die Idee

Die traditionelle Aufgabe von Computernetzwerken ist es, Bitströme von einem Endsystem zu einem anderen zu übertragen. Hierbei spielen die Rechenoperationen, die innerhalb der Netzwerkes ausgeführt werden eine untergeordnete Rolle. Sie beschränken sich auf die im Datenkopf enthaltenen Informationen, die zum Routing oder Switching benötigt werden. Bei dieser Verfahrensweise ist es nicht möglich auf temporäre Anomalien des Netzwerkverkehrs angemessen zu reagieren. Ein Beispiel für eine solche Anomalie könnte ausgefallener Cache - Server sein.
Dieses Defizit soll mit Hilfe von aktiven Netzen behoben werden. Diese neue Art der Netze soll es dem Benutzer ermöglichen eigene Protokolle bzw. Programme in Netzknoten einzuspielen. Die Programme werden mit den jeweiligen Daten des Benutzers ausgeführt. Hierbei geht die Transparenz der Netzwerke verloren, da der Benutzer sich nun über den Weg, den seine Daten nehmen, im Klaren seien muß. Dafür bekommt er aber die Möglichkeit seinen Daten entsprechend die Anforderungen auf dem Weg zum Zielsystem zu verarbeiten, was die Leistungsfähigkeit der Anwendung u. U. steigert.
Der Austausch von Programmen zwischen den Netzwerkknoten hat gegenüber dem rein datenbasierten Austausch entscheidende Vorteile. So bietet er, wie schon erwähnt, die

Möglichkeit anforderungsspezifische Protokolle zu implementieren, die an strategischen Punkten im Netz plaziert werden könne. Dies steigert die Effizienz (zur Definition des Begriffs siehe 1.2.4) der Verarbeitung des Bitstroms erheblich.

1.1.2 Transport des Programms bzw. Protokolls

In der Literatur sind zwei Verfahren beschrieben, mit denen der Benutzer seine anwendungsspezifischen Protokolle bzw. Programme in die Netzwerkknoten einspielen kann.

- Die diskrete Übermittlung und
- die integrierte Übermittlung

Bei der diskreten Übermittlung trennt man die Übermittlung von Daten und Programm. Sendet der Benutzer ein Paket, so wird dessen Datenkopf analysiert. Handelt es sich um ein Programm, so wird dieses im Netzwerkknoten gespeichert und wartet dort auf seine Ausführung. Handelt es sich um Daten, so wird im Datenkopf angeben, von welchem Programm sie zu bearbeiten sind. Der Netzwerkknoten führt dieses Programm aus. Ist das angeforderte Programm dem Netzwerkknoten nicht bekannt, fordert er dieses bei dem vorherigen Netzwerkknoten an und bearbeitet die Daten damit. Der Nachteil dieses Übermittlungsverfahrens sind die hohen Speicheranforderungen an den Netzwerkknoten.
Ein Netzwerkknoten besteht im wesentlichen aus zwei Ebenen. In der Speicherebene

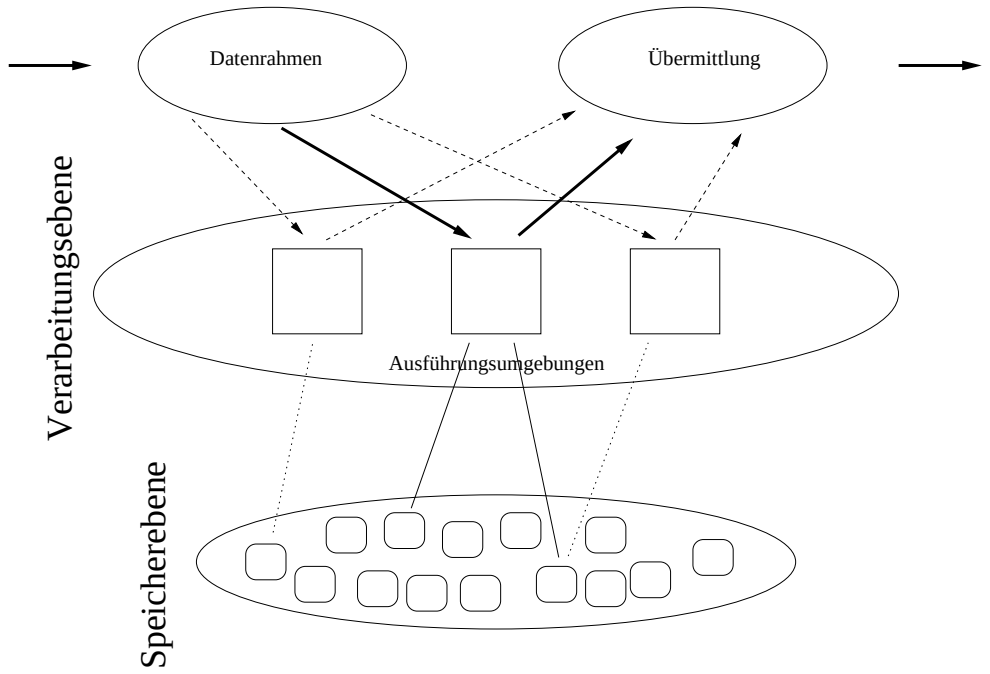

Abbildung 1: Organisation eines aktiven Netzwerkknotens

werden die Programme gespeichert. Die Verarbeitungsebene ist für die Analyse und das Ausführen der gespeicherten Programme verantworlich. Sie prüft ankommende Datenrahmen auf deren Inhalt. Enthält der Rahmen ein Programm, so wird dieses

an die Speicherebene weitergereicht und dort bis zur Ausführung zwischengespeichert. Sind im Rahmen Daten enthalten, so werden diese mit dem entsprechenden Programm ausgeführt.
Bei der integrierten Übermittlung sind Programm und Daten in einem Paket zusammengefaßt. Man kann sich das etwa so vorstellen, wie das Postscript Format. Hier werden Daten an den Drucker in Form eines Programms geschickt. Diese Technik hat den Nachteil, daß jede Kapsel [1] ein Programm enthalten muß und sei es nur eine einzige Anweisung. Mit diesem Verfahren umgeht man obigen Nachteil des Speicherbedarfs. Jedoch hat dieses Verfahren den Nachteil, daß es unnötig Last erzeugt.

1.1.3 Ausführung der Programme

Es soll erreicht werden, daß die Kapseln, die den Programmkode mit sich führen, auf allen Netzwerkknoten ausgeführt werden können. Um dieses Ziel zu erreichen gibt es folgende Möglichkeiten:

- Man bildet eine "virtuelle" Programmierschnittstelle auf die, in dem jeweiligen System, zugrundeliegende Hardware bzw. Software ab. Ein Beispiel hierfür wäre eine virtuelle Maschine. Die Programme würden dann in einem Bytekode übermittelt und von der virtuellen Maschine interpretiert.
- Die Programme werden in plattformabhängigen Binärkodierung übertragen, wobei jede Kapsel mehrere Binärkodierungen ihres Programms für die verschiedenen Plattformen mit sich führt.

Häufig benötigte Aufrufe werden in den Netzwerkknoten in Form von APIs zur Verfügung gestellt, was den Vorteil bietet, daß die Routinen für die jeweilige Plattform optimiert werden können.
Die Ausführungsumgebung muß dafür sorgen, daß die Programme kontrolliert ablaufen. Um dies zu gewährleisten, muß die Ausführungsumgebung folgende Anforderungen erfüllen:

- Die Rechenzeitzuteilung für die Programme muß durch einen preemptiven Zeitscheibenmechanismus geschehen, so das kein Programm die ganze Rechenzeit des Netzwerkknotens beanspruchen kann.
- Der Speicherbereich muß geschützt und die Speichernutzung limitiert sein.
- Der Ressourcenzugriff muß beschränkt sein, damit zum Beispiel Routingtabellen nicht beliebig geändert werden können.

Hierbei muß bedacht werden, daß, je mehr Flexibilität man dem Protokollprogrammierer zugesteht, desto größer ist die Gefahr des Mißbrauchs. Auf der anderen Seite beeinflußt jede Reglementierung auch die Effizienz negativ. Eine genauere Beschreibung der Ausführungsumgebung ist in der Seminararbeit von M. Schmidt (Thema 11) zu finden.

[1] Unter Kapsel versteht man ein Datenpaket, das ein Programm enthält

1.2 Vorteile

Die Vorteile von aktiven Netzwerken liegen in der Möglichkeit der anwendungsspezifischen Konfiguration. Sie bieten die Möglichkeit neue Protokolle einfach in einen Netzwerkknoten einzuspielen, wodurch sich in der Praxis testen läßt, ob das erdachte Protokoll das gewünschte Verhalten aufweist. Durch die aktiven Netzwerke bietet sich die Chance ein Protokoll schon vorab unter realen Bedingungen zu testen, was zu ausgereifteren Protokollen führt. Weitere Vorteile werden in 1.3.1 und folgende deutlich.

1.2.1 Vermittlungssysteme

Auch bei Vermittlungssystemen, die eine zentrale Rolle bei der Weiterleitung von Datenpaketen einnehmen, geht die technische Entwicklung schneller vonstatten, als die politische, also die Standardisierung. Um diesen Mißstand zu beheben, kann man auch bei Vermittlungssystemen die aktive Technologie einsetzen. Der Versuch, neue Funktionalität auf der Ebene des Internet Protokolls (IP) zu implementieren gestaltet, sich schwierig, da das IP nur eine minimale Funktionalität zur Verfügung stellt und alle Erweiterungen an Funktionalität durch sogenannte Overlays (z.B.: MBone oder 6Bone) realisiert werden müssen. Deutlich wird der Mißstand der mangelnden Funktionalität am Beispiel des Domain Name Service (DNS). Die Anforderungen an den DNS Server steigen ständig, da viele Anwendungen deren Namenauflösung benötigen. Nicht zuletzt ist hier das World Wide Web zu nennen, daß mit seinen namenbasierten Adressen eine erhebliche Last erzeugt. Es ist abzusehen, das die DNS Server nicht in dem Maße mitwachsen können, wie es nötig wäre. Zumindest nicht, in dieser statischen Netzwerkstruktur, wie sie heute vorherrscht.

1.2.2 Netzwerkmanagement

Das aktive Netz bringt aber noch andere Vorteile mit sich. So kann das Netzwerkmanagement erheblich verbessert werden. Das heutige Netzwerkmanagement beschränkt sich darauf Daten zu sammeln und Vergleiche vorzunehmen. Um ein hilfreicheres Netzwerkmanagement zu implementieren bedarf es mehr “Intelligenz”. Dies erreicht man durch Filterinstanzen, die nur die relevanten Ereignisse weiterleitet. Eine einfache Möglichkeit dies zu erreichen besteht darin ein Netzwerkmanagement zu implementieren, daß auf Modulen basiert, die als Kapseln verschickt werden.
Fehlermanagement ist eine sehr wichtige und schwierige Aufgabe, besonders bei großen Netzwerken und bei korrelierten Fehler. Korreliert Fehler können durch Umgebungsfaktoren, wie zufällige Unterbrechungen (räumliche Korrelation) oder durch mutwillige Störung (zeitliche Korrelation) hervorgerufen werden. Man glaubt, daß die aktiven Netzwerke in diesem Bereich eine erhebliche Verbesserung in Bezug auf die Fehlerentdeckung und die Managementmöglichkeiten bringen werden.
Die heutigen Netzwerkmonitore für die Fehlerdetektion beschränken sich auf eine Anzahl bekannter Maßnahmen zur Fehlerbehebung. Diese können also wenig dynamisch reagieren. Beim Fehlermanagement stellt sich noch ein weiteres Problem. Die Managementsysteme müssen mit verschiedenen Netzwerktypen interagieren, will man eine effiziente Fehlerbehandlung erreichen. Dies gestaltet sich jedoch schwierig, da die

heutigen Managementsysteme meist nur auf ihren Netzwerktyp konfigurierbar sind. Aktive Netzwerke bringen hier die gewünschte Flexibilität. Sie können weitaus einfacher neu konfiguriert werden (dynamische Programmierung), als das bisher möglich war. Ferner hat man hier nun auch die Möglichkeit dynamisch auf Fehler zu reagieren, indem man sie analysiert und entsprechende Lösungen generiert.

1.2.3 Selbstkontrolle

Eine weitere Möglichkeit das aktive Netzwerkkonzept gewinnbringend einzusetzen ist die Selbstkontrolle des aktiven Netzwerks. Die Idee geht auf ein Forschungsprojekt in den 70er Jahren zurück, in dem ein sogenanntes "Overseer" Konzept entwickelt wurde. Hierbei wird der Weg, den die Datenpakete nehmen, mit einem Protokollgraphen verglichen. Damit kann man feststellen, ob das Paket einen richtigen oder einen falschen Weg genommen hat, was einen Fehler induziert. Aktive Netze können nun basierend auf dieser Idee eine Netzinfrastruktur aufbauen, mit der sich das System selbst kontrollieren kann. Als erste Annäherung an dieses Ziel implementiert man ein Protokoll, dessen Datenfluß in einem Graphen repräsentiert wird, aus welchem der Kontrollfluß und die Datenemission ersichtlich werden. Verteilte Systeme verwenden ein solches Protokoll um im Netzwerk zu agieren. Nachfolgendes Beispiel soll das Prinzip erläutern. Betrachtet werden die Module A, B und C und deren Verhalten. Dieses wird im Protokoll dargestellt, was zu folgendem Entscheidungsbaum führt:

```
A:
 if ( conditionA ) then B else C;
B:
 if ( conditionB ) then C else STOP;
C: B;
```

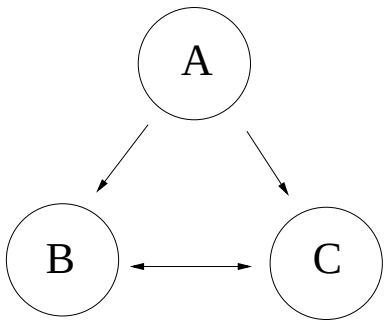

Abbildung 2: Graphische Flußkontrolle für die Module A, B und C

Wenn jede Transportstrecke eine Nachricht generiert, kann man die Reihenfolge der Nachrichten zur Korrektheitsüberprüfung heranziehen. So ist die Folge (A -> C, C -> B) richtig, wohingegen die Folge (A -> B, C -> B) auf einen Fehler hindeutet. Eine wichtige Eigenschaft dieses sog. "traverse<->message" Modells ist, daß die Anzahl der Pfade hierbei weitaus geringer sind, als bei einer umfassenden Analyse des Protokolls, bei der alle möglichen Pfade zwischen diesen Modulen überprüft werden müssen (die

Anzahl der Möglichkeiten wächst exponentiell mit der Länge des betrachteten Pfades). Um nur die Korrektheit des Protokolls nachzuweisen muß man nur eine geringe Anzahl von Pfaden untersuchen. Diese ist proportional zur Zahl der Nachrichten, die in diesem Protokoll versandt wurden. Die Überprüfung kann über das Netz verteilt werden, was die Effizienz erhöht und gleichzeitig den Vorteil der Skalierbarkeit mit sich bringt.

1.2.4 Effizienz

Die herkömmlichen Meßmethoden, wie der Durchsatz (Pakete pro Zeiteinheit) oder die Verzögerung innerhalb des Netzes sagen nicht unbedingt etwas über die Leistungsfähikeit einer Netzwerkanwendung aus. Also die Leistungsfähikeit des Netzwerkes ist nicht unbedingt positiv mit der Leistungsfähikeit der Anwendung korreliert.
In einem aktiven Netzwerk können wenige Pakete, die gleiche Leistung bedeuten, wie viele Pakete in herkömmlichen Netzwerken. Man kann sich bei der Leistungsbewertungen, also nicht auf den Durchsatz als einziges Leistungsmerkmal verlassen. Dies ist der Grund, warum man sich Gedanken über neue Leistungsbewertungen im Zusammenhang mit aktiven Netzwerken machen muß.
Als Kriterium für die Leistungsfähikeit bietet sich ein anwendungsbezogener Durchsatz an. In dem Beispiel des aktiven Cachings (siehe 1.3.1) reduziert das aktive Netzwerk den Durchsatz vonseiten des Servers, erhöht aber zugleich die Anzahl der Clients, die parallel eine Anfrage stellen können. Bei der Online Auktion (siehe 1.3.2) könnte die relevante Meßgröße die Anzahl der Gebote pro Zeiteinheit sein. Diese Verbesserungen wurden durch das Delegieren von Aufgaben an das Netzwerk und eine damit einhergehende Parallelisierung erzielt. Der Preis für die Leistungssteigerung ist der erhöhte Rechen- und Speicherbedarf innerhalb der Netzwerkknoten. Was eine Verlangsamung des anderen, also nicht aktiven Datenverkehrs mit sich bringt. Dennoch profitiert das Netz vom aktiven Verkehr, da er das Datenaufkommen senkt, was anderen Netzwerksegmenten zugute kommt.

1.3 Anwendungsbeispiele

Hier folgen nun einige Beispiele, an Hand derer die Vorteile von aktiven Netzwerken sichtbar werden.
Die Beispiele können auch mit herkömmlicher Technik realisiert werden. Jedoch ist mit aktiven Netzwerken eine bessere Ressourcenausnutzung möglich.

1.3.1 Caching

Die Aufgabe eines Caches ist es, häufig benötigte Informationen zwischenzuspeichern, so daß sich die Wartezeit auf die Informationen verkürzt und der Server nicht dauernd mit den gleichen Anfragen beschäftigt wird. Caches werden in lokalen Administrationsbereichen eingerichtet und arbeiten effizient, wenn sich viele Clients in diesem Administrationsbereichen befinden und die geographisch Ausdehnung gering ist (siehe Abbildung 3). Verteilen sich nun aber die Clients geographisch, so muß man sich ein neues Cachekonzept überlegen. Ein mögliche Strategie wäre, einige Server auszuwählen, die den Cache spiegeln. Die Clients werden dann automatisch an diese Server

verwiesen.

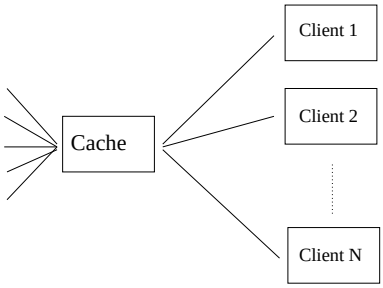

Abbildung 3: Herkömmliches Cacheprinzip

Bei aktiven Netzwerken speichern die Netzwerkknoten die übertragenen Objekte auf ihrem Weg zwischen Client und Server. Hier bedient also jeder Cache/Netzwerkknoten viele Clients und der Cache ist leicht zu finden, da er auf dem Weg zum Ziel (Server) liegt.
Der Cache bzw. Teile von ihm haben nun die Möglichkeit, abhängig von der Nachfrage und der Last, ihren (geographischen) Sitz zu verändern oder zu verteilen, womit auch der geographische Aspekt des Cachings gelöst ist (siehe Abbildung 4). Ein weiterer Vorteil des aktiven Caches ist, daß sie die Möglichkeit besitzen dynamische Objekte zu cachen, indem sie deren Generierungskode speichern und diesen bei Bedarf ausführen. Als Beispiel kann man sich dynamische WWW Seiten vorstellen.

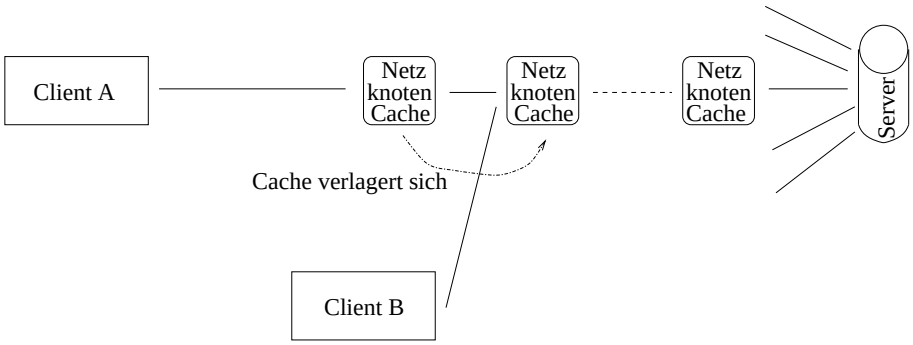

Abbildung 4: Cachen in aktiven Netzwerken

1.3.2 Auktionen

Ein weiteres Einsatzgebiet für aktive Netzwerke sind Online Auktionen. Hier hat der Server die Aufgabe, Gebote der Clients für das angebotene Objekt zu sammeln. Der Server muß auch auf Anfragen zum aktuellen Preis antworten. Aufgrund der Netzwerk- und der Serverlast kann der angegebene Preis beim Erreichen des Clients schon wieder veraltet sein, was zur Folge hätte, daß das nächste Gebot des Clients unter dem Marktwert läge und somit nicht mehr berücksichtigt würde. Das Abblocken von zu niedrigen Angeboten erzeugt jedoch zusätzliche Last auf dem Server, was wiederum den oben beschriebenen Effekt verstärkt.
Aktuelle Implementierungen solcher Server bearbeiten alle Angebote auf dem Auktionsserver. In einem aktiven Netzwerk bietet sich die Möglichkeit, ungültige Gebote

schon im Vorfeld auszusortieren. Dies wird dadurch erreicht, daß man die Filterfunktion an die dem Server vorgeschaltete Netzwerkknoten delegiert. Der Auktionsserver müßte dann nur noch die ihm vorgeschalteten Netzwerkknoten periodisch mit dem aktuellen Marktwert versorgen und könnte ansonsten seine Rechenzeit für die Berechnung des neuen Marktwertes nutzen. Hierdurch erhöhte sich die Anzahl der Gebote, die pro Zeiteinheit bearbeitet werden können.

1.3.3 Verteilen und Bündeln

Gegeben sei folgendes Problem: Mehrere geographisch verteilte Sensoren sammeln große Datenmengen und geben diese an mehrere Rechner via Netzwerk weiter. Dies tritt zum Beispiel bei Wetterstationen oder Umweltmessungen auf.
Eine einfache, aber nicht sehr effizient Lösung ist es, jedes Paket an jeden Rechner zu schicken. Geht man davon aus, daß nur gewisse Daten für den jeweiligen Rechner von Relevanz sind, so werden dabei auch Informationen übertragen, die der Empfänger verwirft, da er dafür keine Verwendung hat. Ein Beispiel hierfür sind Wetterdaten. Interessiert einen Rechner nur die Temperatur, so kann er mit der relativen Feuchte nichts anfangen und verwirft diesen Wert.
Die Übertragung von allem an alle ist nicht befriedigend und läßt sich mit Hilfe der aktiven Netzwerke besser lösen. Bei diesem Verfahren der Übertragung hat man die Möglichkeit Daten bereits innerhalb des Netzwerkes zu Bündeln und zu selektieren, da die Netzwerkknoten - bedingt durch ihre Programmierbarkeit - Filterfunktionen übernehmen können. Hierdurch erlangt man schon eine Reduktion des Datenaufkommens, da jetzt jede Station nur noch die Daten bekommt, die sie wirklich benötigt. Zusätzlich kann man ein Multicast Protokoll (IP) einsetzen, was wiederum eine Reduktion des Datenaufkommens mit sich bringt.

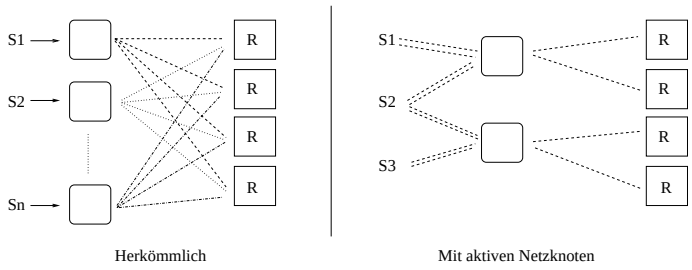

Abbildung 5: Möglichkeiten der Bündelung mit aktiven Netzen

1.3.4 "Spiegel"-Server (Mirror Server)

Der Mirror Server kann als eine Kombination von den, in den Abschnitten 1.3.1 und 1.3.3 vorgestellten, Techniken gesehen werden. Ein solcher Server hat die Aufgabe, andere Server lokal zu spiegeln. Derzeit muß der Benutzer den gespiegelten Server noch explizit ansprechen. In aktiven Netzen könnten die Netzwerkknoten entscheiden, an welchen Server der Benutzer verwiesen wird. Hierbei muß natürlich die Datenaktualität gewährleistet sein. Durch dieses Regulativ kann man die Netzwerkressourcen besser ausnutzen und Überlast besser vermeiden.

2 Integration von Aktiven Netzwerken in das Internet

Damit aktive Netze zum Einsatz kommen, muß eine Möglichkeit geschaffen werden, sie in existierende Strukturen zu integrieren. Sie müssen also die Fähigkeit besitzen auch in heterogenen Netzwerken (aktive und nicht aktive Netze) ihre Vorteile zum Einsatz zu bringen.

2.1 Einleitung

Will man die Integration vornehmen, so stellt sich die Frage nach dem Netzwerktyp. Hier bietet sich IP an, da es sehr verbreitet und im Internet Standard ist. Ein Ansatz für die Integration wird in [Murp97] gezeigt. In heutigen Netzwerkumgebungen ist es schwer experimentelle IP Implementationen im Testbetrieb einzusetzen. In lokalen Netzwerken mag dies noch machbar erscheinen, aber schon bei WANs ist dies wohl nicht mehr durchführbar (vergeliche MBone bzw. 6Bone).
Bei der Integration von aktiven Netzwerkknoten in IP Netze muß garantiert werden, daß der laufende Betrieb nicht gestört wird. Im wesentlichen müssen folgende Bedingungen erfüllt sein:

- Ein aktiver Netzwerkknoten muß in der Lage sein beide Arten von IP Paketen, also die aktiven und bisherigen IP Pakete (IPv4/v6), weiterzuleiten.
- Ferner darf ein aktives IP Paket keinen Verarbeitungsfehler in einem nicht aktiven Netzwerkknoten verursachen

Bei der Realisierung stellen sich im wesentlichen zwei Fragen. Wie kann man die Gemeinsamkeiten zwischen Paketen und Kapseln bei der Realisierung ausnutzen und wie sichert der Netzwerkknoten seiner Ressourcen?
Um die Gemeinsamkeit zu nutzen implementiert man im System, das mehr Funktionalität zu Verfügung stellt, also in diesem Fall dem aktiven Netzwerkknoten, die Fähigkeit IPv6 Pakete zu verarbeiten.
Um dem Sicherheitsaspekt gerecht zu werden, sollte ein aktiver Netzwerkknoten Pakete vor Mißbrauch schützen. Dieser Aspekt soll hier aber nicht weiter behandelt werden. Statt dessen sei auf Thema 11 (Markus Schmidt) verwiesen, wo diese Sicherheitsaspekte der Betriebssysteme und Ausführungsumgebungen behandelt werden.

2.2 IPv6

An dieser Stelle wird eine kurze Übersicht über IPv6 gegeben, die nicht den Anspruch hat vollständig zu sein. Vielmehr sollen die, für diesen Beitrag relevanten Aspekte, hervorgehoben werden.

2.2.1 Neuerungen

In den heutigen Netzwerken reicht die Größe des Adreßraums, die Leistungsfähikeit, die Flexibilität der Verbindung und die Sicherheit nicht mehr aus. Diese Mängel soll das IPv6 Protokoll beheben. Die einzelnen Aspekte kurz dargelegt.

- Der Adreßraum von 2^{32} ist wegen folgender Punkte nicht mehr ausreichend
  - Die Anzahl der LANs und der drahtlosen Netze steigt stetig, so daß eine Erschöpfung des Adreßraumes absehbar ist.
  - Der TCP/IP Standard wird in völlig neue Bereiche Einzug halten. So ist es zum Beispiel denkbar, daß jeder Fernseher mit einer eigenen IP-Adresse ausgestattet wird, was für das Tele Shopping von Vorteil wäre.
- Da auch die Anzahl der IP-Pakete, die pro Zeiteinheit geroutet werden müssen steigt, ist IPv6 bestrebt das bestehende Protokoll zu vereinfachen, um den Routern "unnötige" Auswertungen zu ersparen. Dies ist kein Widerspruch zu aktiven Netzen, da sich dort die Leistungsfähigkeit nicht nur als Durchsatz pro Zeiteinheit (siehe 1.2.4) bestimmt. An folgende Vereinfachungen ist dabei gedacht:
  - Die Anzahl der Datenfelder im Paketkopf wurde vermindert und alle Optionen wurden auf separate optionale Paketköpfe verlagert. Das entlastet den Router, da er nur noch den "Haupt" Datenkopf analysieren muß.
  - Der Paketkopf hat keine fest Länge mehr, wodurch unnötiges Stuffing vermieden wird.
  - Es gibt eine Prioritätenregelung, so daß zeitkritische Anwendungen bevorzugt behandelt werden können.
- Zur Flexibilisierung wäre anzumerken, daß IPv4 nur die Versendung eines Pakets an eine ganz bestimmte Adresse unterstützt (Unicast). Es besteht aber oft auch die Notwendigkeit Pakete an mehrere Rechner zu verschicken (Multicast). Das ist bei IPv4 nur mit Hilfe von Overlays möglich. Bei IPv6 ist diese Fähigkeit bereits integriert.
- Für die Sicherheit der Datenpakete soll eine standardisierte Verschlüsselung auf der Ebene des IP Protokolls stattfinden.

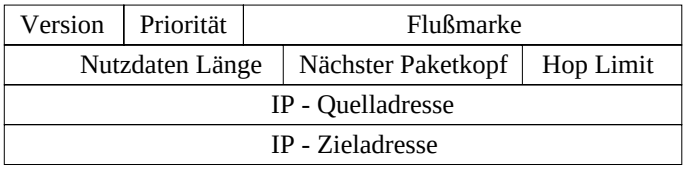

Abbildung 6: IPv6 PaketDatenkopf

2.2.2 Integration von aktive Netzwerke

Neben den schon erwähnten Argumenten (weite Verbreitung, hohe Akzeptanz im Internet), bietet sich die Möglichkeit, ein neues Protokoll innovativ zu erweitern. Zudem bietet es von der Struktur der Pakete gute Voraussetzungen für den Transport von Kapseln (siehe Abbildung 6 bzw 2.3.1).
Eine Möglichkeit Aktives IPv6 (AIPv6) in IPv6 Netzwerken zu übertragen ist das "Tunneln". Diese Technik wird angewandt, wenn sich zwei Netzwerkknoten nicht mit dem gleichen Protokoll verständigen können. Hierbei muß keine Veränderung an dem bestehenden Netzwerk vorgenehmen werden. Das "Tunneln" würde aktive Netzwerke

jedoch nicht in IPv6 integrieren, sondern lediglich IPv6 als Link Layer verwenden. Somit könnte eine Anwendung, die auf einem IPv6 Netzwerkknoten läuft nicht auf das aktive Protokoll zugreifen, da es den Bitstrom, wie gewöhnliches IPv6 behandelt. Eine andere Möglichkeit besteht darin die Pakete nur zwischen aktiven Netzwerkknoten zu "tunneln" und den "aktiven Payload" bei senden an herkömmliche IPv6 Netzwerkknoten wegzulassen. Dieses Verfahren würde eine Interaktion zwischen aktiven und nicht aktiven Netzwerkknoten garantieren. Ein Netzwerkknoten mit dieser Funktionalität nennt man Gateway. Diese Funktionalität müßte der aktive Netzwerkknoten aufbringen. Zudem müßte er noch Routingaufgaben übernehmen. Ein Problem dieses Verfahrens ist es, daß der Netzwerkknoten alle relativen Positionen aller IPv6 Netzwerkknoten und aller aktiven Netzwerkknoten im Internet kennen muß. Dieser Aufwand ist nicht vertretbar. Aufgrund der Tatsache, daß AIPv6 auf IPv6 Pakete abgebildet werden können (man muß nur das Programm herausnehmen), aber der umgekehrte Weg nicht möglich ist, also aus einem ehemals aktiven IPv6 Paket wieder ein solches zu machen, darf der aktive Netzwerkknoten den Programmteil nicht herausnehmen, solange nicht gewährleistet ist, daß das Paket keine weiteren aktiven Netzwerkknoten mehr passiert. Dies zu garantieren, wäre jedoch nur mit einem unverhältnismäßig hohem Aufwand möglich.
Eine Alternative zu den bisher vorgestellten Möglichkeiten ("Tunneln" und Gateway) Programme zu transportieren ist es, die aktive Eigenschaft des Pakets mit optionalen Parameter oder im Payload zu kodieren. Der Payload zum Beispiel ist nur durch die maximale Größe eines IPv6 Pakets beschränkt und eignet sich somit gut für den Transport von Programmen. Zudem könnte noch ein Eintrag in dem Hop-by-Hop Datenkopf, der mit Programmtext gefüllt ist, den AIPv6 Routern auf den Payloade hinweisen. Der Vorteil dieses Verfahrens wäre es, daß der Router nicht jeden erweiterten Paket-Datenkopf auf einen aktiven Datenkopf untersuchen muß. Ein ganz entscheidender Nachteil dieses Verfahrens ist es, daß diese neue Definition des Payloads nicht von allen IPv6 Netzwerkknoten unterstützt wird. Somit würde ein Endknoten das Paket verwerfen und einen ICMP [2] Fehler an den Absender schicken. Dieses Verfahren eignet sich also nicht zum Einsatz, da kein fehlerfreier Betrieb gewährleistet werden kann.
Es gibt jedoch eine Einstellung im IPv6 Paket, die signalisiert, daß dieses Paket nicht standardgemäß behandelt wird. Sie eignet sich für AIPv6 Pakete, um den Programmtext zu transportieren. Die Netzwerkknoten ignorieren den Programmtext, wenn sie ihn nicht verstehen bzw. der entsprechende Flag gesetzt ist. Anwendungen können durch dieses Verfahren, auch auf einem nicht aktiven Netzwerkknoten, auf das aktive Netzwerk zugreifen, sobald eine Netzwerk-Socket-Unterstützung für die Anwendung vorhanden ist. Bei diesem Verfahren ist die Transportmenge auf 256 Bytes beschränkt. Diese Tatsache schränkt die Tauglichkeit für den Programmtexttransport ein, der typischer Weise zwischen 750 und 1000 Bytes liegt. Da der Hop-by-Hop Datenkopf eine maximale Größe von 2056 Bytes hat und somit eine Vielzahl von Einstellungsmöglichkeiten bietet, könnte man eine dieser Einstellungsmöglichkeiten verwenden, die besagt, daß der Programmtext in Fragmente zerteilt ist. Weiter könnte man diese Fragmente in die Optionsfelder schreiben und so den Programmtext übertragen. Der Nachteil hierbei wäre, daß der aktive Netzwerkknoten fünf Flags eines jeden Pakets untersuchen müßte, um 1K Porgrammtext zu erhalten.

[2] Internet Control Message Protocol

2.3 Aktives IPv6

In [Murp97] wird folgende Lösung für die Integration gewählt. Der Transport gliedert sich in zwei Zustände, die Programmtextübermittlung und die Übermittlung der Statuswerte bzw. Daten.
In einem aktiven Netzwerk ist die Programmtextübermittlung ein Vorgang, der zwischen zwei aktiven Netzwerkknoten abläuft. Der aktive Netzwerkknoten A sendet eine Programmtextanfrage an den aktiven Netzwerkknoten B. Daraufhin übermittelt B den Programmtext an A. Aufgrund der "Ende-zu-Ende" (Tunneln) Kommunikation kann man in einem heterogenen Netzwerk den IPv6 Payload zur Programmtextübermittlung verwenden, da jeweils nur die aktiven Netzwerkknoten auf diesen zugreifen.
Schwieriger gestaltet es sich bei der Übermittlung von Statuswerten. Diese Übermittlung tangiert alle Netzwerkknoten eines aktiven Netzwerks. Das bedeutet, daß in einem heterogenen Netzwerk auch nicht aktive Netzwerkknoten diese Pakete "verarbeiten" müssen. Hierfür gibt es nun zwei Lösungen:

- Man speichert die Statuswerte im Optionsfeld. Hierbei hat man einen Größenbeschränkung von 256 Bytes
- Man speichert die Statuswerte im IPv6 Payload, dessen maximale Größe 2056 Bytes beträgt, muß hierbei aber sicherstellen, daß von der Quellstation nur Pakete an aktive Netzwerkknoten versandt werden.

Die robustere Methode um die Statuswerte zu übertragen ist es, sie im Optionsfeld zu speichern. Das AIPv6 Protokol in der Implementierung von [Murp97] benutzt die "marshaled option" und den Systempayload um den Transport von Programmtext und Statuswerten zu bewerkstelligen. Wobei die "marshaled option" die Statuswerte und der Systempayload den Programmtext überträgt.

2.3.1 Wie man IPv6 "aktiv" macht

Wie bereits in 2.3 erläutert wurde, hat man sich in [Murp97] dafür entschieden den Programmtext im Payload und die Statuswerte im Optionsfeld zu übertragen. Diese beiden Felder haben folgende Gemeinsamkeiten:

- Einen 128 Bit - Identifikator. In der Implementierung von [Murp97] ist dies eine Zufallszahl, jedoch könnte man diesen Identifikator als "Fingerabdruck" zur Authtisierung verwenden.
- Ein Spezifikationsfeld, daß es ermöglicht zwischen verschiedenen Formaten, die im "marshal" Optionsfeld und im Systempayload existieren, zu unterscheiden.

Das IPv6 Protokoll verlangt, daß jede Option eine Nummer besitzt. Dabei entscheiden die zwei höchstwertigen Bits, ob die IPv6 Netzwerkknoten das Optionsfeld auslassen oder bearbeiten. Das nächste Bit gibt an, ob sich das Optionsfeld während des Routings verändern darf oder nicht (bei Endsystemen ist diese Information bei der Berechnung der Checksumme zu berücksichtigen).
Jede Option besitzt ein Längenfeld. Der Wert dieses Feldes hängt von der Anzahl der

Statuswerte, die im Optionsfeld enthalten sind, ab.
Das "marshal" Optionsfeld enthält die Statuswerte für den Programmtext. Die Werte werden durch den Identifikator zugeordnet. Um den Verarbeitungsaufwand beim Demultiplexen zu verringern wurde in der Implementierung [Murp97] festgelegt, daß die erste Option im Hop-by-Hop Datenkopfoptionsfeld eine "marshal" Option seien muß, wenn es sich um ein aktives Datenpaket handelt.
Eine Gefahr beim Benutzen einer Option im Hop-by-Hop Datenkopf liegt darin, daß es IPv6 Knoten verboten ist, den Hop-by-Hop Datenkopf zu fragmentieren. Kommt dieser Datenkopf jedoch bereits fragmentiert an, so wird das entsprechende Programm für jedes Fragment ausgeführt. Das kann sowohl von Vorteil, als auch von Nachteil sein, je nachdem, was für eine Anwendung diese Pakete verschickt hat. Es scheint wohl am sinnvollsten, die Fragmentierung abzuschalten.
Sowohl die Anfrage, als auch die Antwort werden im gleichen Payload transportiert, im aktiven Payload. Das erste Byte des Payloads definiert das Protokoll, das im folgenden Teil des Payload kodiert ist. Das zweite Byte gibt die Größe des Payload in Einheiten von acht Oktetts an, jedoch ohne die ersten acht Oktetts.
Bei einer Anfrage überprüft der aktive Netzwerkknoten, ob er das entsprechende Programm besitzt, daß im "marshal" Optionsfeld eingetragen ist. Fehlt ihm dieses Programm, so fordert er dieses beim Absender des Paketes an. Bei der Antwort wird der Programmtext mit einem 128 Bit Identifikator spezifiziert. Die vier höchstwertigen Bits geben an, in wieviele Fragmente der Programmtext zerlegt wurde. Die vier niederwertigsten Bits im gleichen Byte geben die Fragment-Sequenznummer des Programmtextes in diesem Paket an. Ein Nullwert gibt an, daß es sich um das erste Fragment handelt. Ein Wert von 1 gibt an, daß es sich um das zweite Fragment handelt, usw..

2.4 Das Leistungspotential

Das Leistungspotential aktiver Netzwerke real zu beurteilen fällt schwer, da sie derzeit nur in Laborumgebungen laufen. Diese Umgebungen sind jedoch nicht sehr aussagekräftig, da viele Ansätz der Implementierung noch nicht berücksichtigt wurden. So gibt es den Bytecode-, den Hochsprachen- und den Binären-Ansatz. Abhängig von der gewählten Implementierung erzielt man bessere bzw. weniger gute Resultate.
Die im folgenden dargestellte Leistungsbewertung bezieht sich auf die in [Murp97] beschriebene Implementierung. Der Programmtext wird als Java-Bytecode übermittelt und von der virtuellen Maschine interpretiert. Java ist sicherlich nicht die schnellste Möglichkeit einen aktiven Netzwerkknoten zu realisieren, jedoch hat man eine große Anzahl von Plattformen zur Verfügung, auf denen der Bytecode interpretiert werden kann.

Leistung		
	Durchsatz mit Routing	Prozent vom Maximum
IPv6	979 Pakete / s	26.11
AIPv6	766 Kapseln / s	20.43

Tabelle 1: Leistung von aktiven Netzwerkknoten

Die Werte in Tabelle 1 wurden auf einem Pentium Pro mit 200MHz unter Linux gemessen. Als maximalen Durchsatz kann man auf diesem System 3750 Pakete / s erreichen.

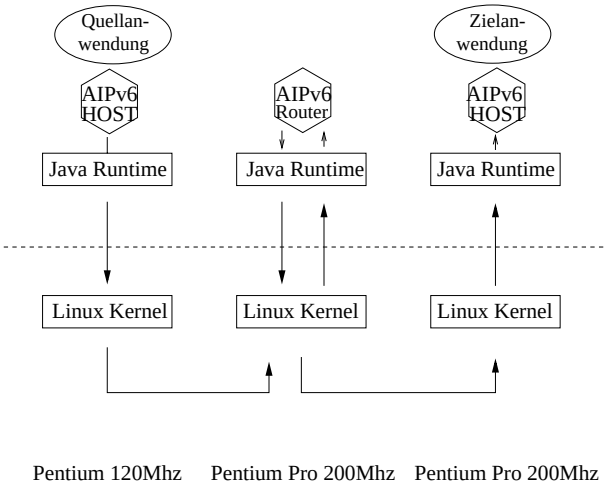

Abbildung 7: Experimentaufbau

Es gibt bei diesem Versuchsaufbau nur einen Weg zwischen Quelle und Senke. Da dieses Szenario jedoch wenig real ist, simuliert man zwei Wege, einen langsamen und einen schnellen Weg. Das zweite Paket im Datenstrom nimmt den schnellen Weg zur Senke. Beim schnellen Weg benutzt der Router seine Routingtabelle. Beim langsameren Weg wird das Programmtextnachladen simuliert.
Auch wenn der Druchsatz nicht so überragend erscheint, sollte man diesen Ansatz weiter verfolgen, da noch eine Menge von Optimierungsmöglichkeiten zur Leistungssteigerung zur Verfügung stehen. Eine Möglichkeit wäre den Bytecode in einen plattformabahängigen Binärcode zu übersetzen. Ferner darf man bei diesem Aufbau (Abbildung 7) nicht vergessen, daß man unnötige Kopieroperationen zwischen dem Kernel und der Java-Runtime-Umgebung ausführt. Hier könnte man zum Beispiel die Java-Runtime-Umgebung in den Linuxkernel integrieren oder eine Plattform verwenden, in der dies schon geschehen ist, wie zum Beispiel Java OS.

Zusammenfassend kann man feststellen, daß eine Integration durchaus machbar wäre. Der aktive Netzwerkknoten kann die Gemeinsamkeiten der beiden Architekturen vorteilhaft nutzen, indem er seine Verarbeitungvorgänge in konstante und variable Segmente aufteilt. Die vorgestellte Implementierung zeigt eine echte Integration. Es bleibt jedoch abzuwarten, ob sich dieses neue Konzept durchsetzt. Bei allen angeführten Vorteilen, die das System biete, darf nicht vergessen werden, daß sich die Skalierbarkeit der Netzwerkknoten recht teuer gestaltet. Schließlich werden nicht unerheblich wenige Programmtext zu speichern sein und sie jedesmal nachzuladen würde einen Teil des Leistungsgewinns wieder aufheben.

Literatur

[D.L.97] L. Shrira M.F. Kaashoek D.L. Tannenhouse S.J. Garland. From Internet to ActiveNet. *Internet*, 1997.

[Gutt97] U. Legedza D.J. Wetherall J. Guttag. Improving the Performance of Distributed Applications Using Active Networks. *Internet*, 1997.

[Mind97] D.L. Tannenhouse J.M. Smith W.D. Sincoskie D.J. Wetgerall G.J. Minden. A Survey of Active Network Research. *Internet*, 1997, S. 14.

[Muni97] Sohail Munir. Active Networks - A Survey. *Internet*, August 1997.

[Murp97] David M. Murphy. Building an Active Node on the Internet. *Department of Electrical Engineering and Computer Science*, May 1997, S. 65.

[Weth97] D.L. Tannenhouse D.J. Wetherall. Towards an Active Network Architecture. *Internet*, 1997.

Komponenten für Active Networks: Betriebssysteme, Sicherheit, Programmiersprachen

Markus Schmidt

Kurzfassung

Active Networks erlauben es dem Benutzer, eigene Programme in einen Netzwerkknoten einzubringen. Im extremsten Fall werden Pakete durch Programmfragmente ersetzt, die ausgeführt werden, sobald sie einen Knoten (Router, Switch) passieren.
Active Networks geben viele interessante neue Möglichkeiten; neue Anwendungen und Dienste können im Netz plaziert werden. Weiterhin kann die Anwendung von Active Networks Entwicklungen von Netzwerkkomponenten beschleunigen, da nur einmal eine Programmiersprache und die Laufzeitumgebung standardisiert werden muß.
Die Programmierbarkeit eines Knotens birgt jedoch große Gefahren in sich. Bei fehlerhafter Programmierung kann das gesamte Netzwerk lahmgelegt werden, Pakete können zum falschen Empfänger gelangen oder unterwegs verloren gehen.
Um einer Fehlprogrammierung oder Fehlleitung vorzubeugen, müssen bestimmte Sicherheitsmechanismen im System vorhanden sein. Diese können durch die Programmiersprache oder durch das Betriebssystem zur Verfügung gestellt werden.
In diesem Beitrag soll der Sicherheitsaspekt und die Unterstützung von Seiten des Betriebssystems sowie der Programmiersprachen näher beleuchtet werden.

1 Einleitung

1.1 Was sind Active Networks?

Ein *Active Network* besteht aus Elementen, die programmierbar sind, sei es knoten-, datenfluß- oder paketbezogen. Sie sind daher ein weiterer Schritt der Zusammenführung von Computing und Kommunikation, wobei die Kommunikationsinfrastruktur *virtuell* ist und mit Hilfe von Computern oder anderen Geräten aufgebaut ist. So ist beispielsweise das Internet ein gutes Beispiel einer virtuellen Infrastruktur, es bietet die Illusion eines riesigen adressierbaren Netzwerks durch Software, die auf heterogenen Subnetzen ausgeführt wird (konkret: TCP/IP-Protokollstack).
Dieser Grad an Programmierbarkeit stellt einen Hauptgedankengang in der neuartigen Denkweise über Netzwerke dar. Netzwerkelemente müssen mehr tun, als nur speichern/annehmen und weiterleiten (*store-and-forward*). Es wird immer wichtiger, das Prinzip (zumindest in vielen Fällen) zu *store-compute-and-forward* abzuändern. Dies kann in vielen verschiedenen Varianten geschehen: DNS-Proxies, aktive Mittel zur

Sicherheit, selbstüberprüfende Netzwerke und die Benutzung von ökonomischen Prioritätsmodellen.
Wenn ein Knoten in der Lage ist, selbst Programme auszuführen, steigt natürlich die Komplexität dieses Teilsystems an. Ein IP-Router führt das Multiplexing von Bandbreite und Puffern hauptsächlich mit statistischen Methoden durch, der Hauptberechnungsaufwand ist, die Route des Pakets mit Hilfe einer Tabelle auszulesen und das Paket zum nächsten Knoten zu leiten. Bei einem aktiven Router muß das Multiplexing zusätzlich noch andere Ressourcen (Speichergröße, Rechenzeitvergabe, ...) berücksichtigen. Weitergehend wird die Komplexität auch dadurch erhöht, daß das Multiplexing ebenfalls unter die Kontrolle eines oder mehrerer Programme fallen kann.
Wird die Kontrolle eines Netzwerkelements auch durch Programme möglich, dann stellt sich sofort die Frage nach Sicherheitsaspekten und Leistungsfähigkeit. Diese sollen im folgenden betrachtet werden.

1.2 Sicherheit, Flexibilität und Performance

Schwerpunkt bei aktiven Netzwerken liegt sicherlich in der Flexibilität. Programmierbare Netzwerkkomponenten bieten ein genügend großes Abstraktionsniveau, um erweiterte Dienste einfach dadurch einzuführen, indem sie auf die bestehende Infrastruktur aufzusetzen, anstatt sie auf jedem Host zu realisieren.
Der Umstand, daß die Knotenpunkte von vielen Benutzern genutzt werden müssen, führt dazu, daß verschiedene Benutzer unterschiedliche Absichten der Benutzung des Knotens haben können. Beispielsweise könnten alle Benutzer den Wunsch haben, 100% der verfügbaren Bandbreite für sich zu nutzen. Wenn sich zwei solcher Benutzer auf einem Knoten befinden, muß das System die Ressourcen zwischen beiden aufteilen. Gleiches passiert, wenn ein Benutzer programmierbare Ressourcen behandelt, als ob sie alleine für ihn vorgesehen wären. Rechner- und Speicherresourcen müssen nun ebenso aufgeteilt/gemultiplexed werden.
Diesen Problemen stehen alle Betriebssysteme gegenüber, typische Betriebssysteme erzeugen beim Benutzer daher die Illusion, daß sie den Rechner für sich alleine zur Verfügung zu haben. Dies wird möglich durch eine *virtuelle Maschine*, einer Abstraktion von der realen Maschine. Die virtuelle Maschine schränkt den Zugriff auf Systemresourcen und Komponenten ein (z. B. auf Adreßtabellen oder E/A-Geräte), um sicherzustellen, daß die Komponenten von allen Benutzer gleichermaßen sicher benutzt werden können. Der Nachteil hierbei ist jedoch ein Einbruch bei der Leistungsfähigkeit des Systems.
Wesentliche Frage ist somit auch bei den aktiven Netzwerken die geschickte Kontrolle über die Systemresourcen, so daß die Sicherheit nicht beeinträchtigt wird, mehrere Benutzer gleichzeitig Programme auf der Netzwerkkomponente ausführen dürfen und die Geschwindigkeit keine nennenswerte Einbrüche erleidet. Dies führt zur Frage, welche Einheiten der aktiven Netzkomponenten nun in Hardware auszuführen sind, welche in Software. Bei der Software ist weiterhin zu bedenken, welche Sicherheits- und Kontrollmechanismen im Betriebssystem und in der verwendeten Programmiersprache enthalten sein müssen.

1.3 Hardware, Compiler oder Betriebssystem?

Traditionell werden Hardwaresysteme erst einmal als Softwareprototypen realisiert, da die Komplexität nicht erlaubt, das System sofort in Hardware zu realisieren oder das System in einem experimentellen Stadium steht. Erst wenn das System von der Komplexität und dem Design ausreichend verstanden ist, wird man aus Geschwindigkeitsgründen zu einer Hardwareimplementierung überwechseln.
Sicherheits- und Kontrollmechanismen der untersten Ebene liegen an der Grenze zwischen Hard- und Softwarerealisierung: soll beispielsweise der Scheduler des Betriebssystems von Hardwarekomponenten unterbrochen werden dürfen? Wie können Schutzbereiche des Betriebssystems durch Hardwareadreßumrechnung trotzdem noch eingehalten werden? Wie wird mit Ausnahmebehandlungen/Interrups umgegangen? Muß die Hardware bei einer Änderung der Software angepaßt werden?
Spätestens hier wird deutlich, daß aktive Netzwerke ausführlich in Software getestet werden müssen. Weiterhin muß entschieden werden, welche Prüfungen von welchem Teilsystem gemacht werden müssen: statische Überprüfungen durch den Compiler, dynamische durch einen privilegierten Interpreter (Betriebssystem).
Ein Betriebssystem, das Sicherheitsaspekte in allen Ausprägungen unterstützt, mußnotwendigerweise auch durch fortgeschrittene Compilertechniken unterstützt werden. Deshalb soll im nachfolgenden auch auf die Anforderungen an ein Betriebssystem eingegangen werden, um daraus Schlußfolgerungen für geeignete Programmiersprachen ziehen zu können.

2 Betriebssysteme

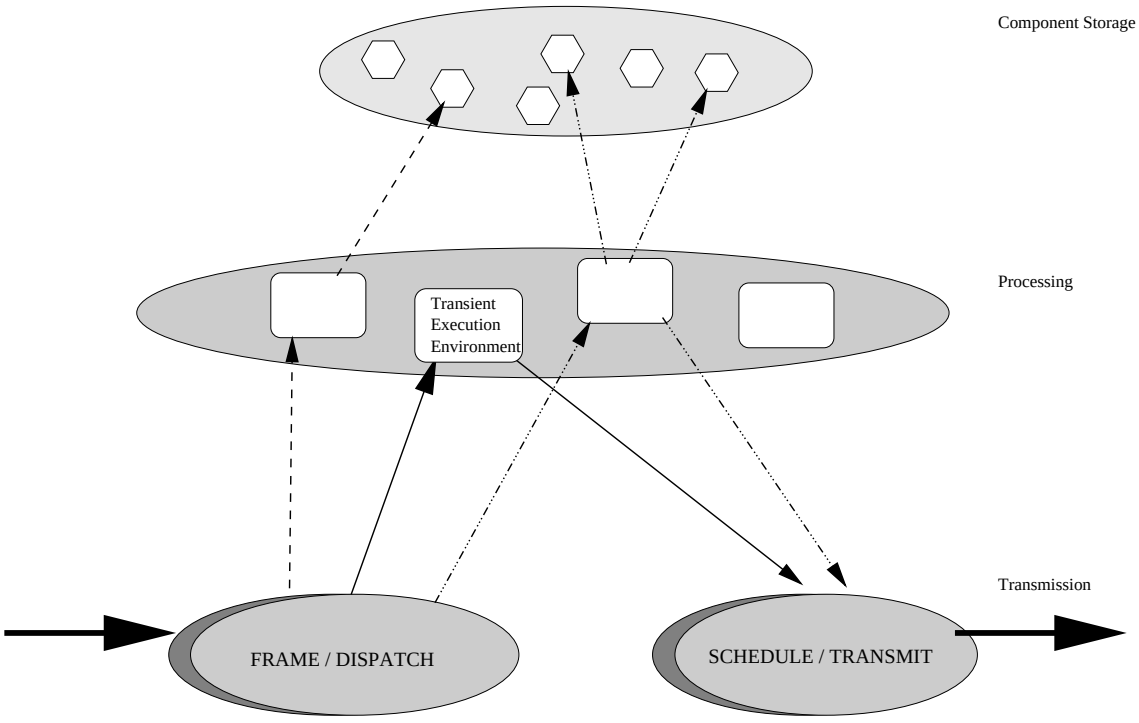

Abbildung 1: Organisation eines aktiven Knotens

Anforderung	Linux	RT-Mach	Nemesis	EROS
Echtzeitunterstützung	-	√	√	√
Reserve Delegation	-	-	-	√ über Capabilities
Netzwerkleistung	Mittel	Niedrig	Sehr hoch	Hoch
Sicherheit/Schutz	User ID Paged VM	User ID Segmente VM	SASOS VM IDL User ID	Capabilities
Protection Composable	-	-	-	√
IPC Leistung	Niedrig	Mittel	Hoch	Sehr hoch
Formal prüfbar	-	-	-	√

Tabelle 1: Zusammenfassung der Anforderungen an ein aktives Element (VM=Virtual Memory)

2.1 Anforderungen an ein Betriebssystem

Aktive Netzwerkelemente kombinieren die Anforderungen eines konventionellen Routers mit den Anforderungen eines sicheren allgemein verwendbaren Rechnerknotens. Router müssen Bandbreite, Durchsatz und Latenz optimieren. Sie müssen ausfallsicher, und skalierbar sein, sowie im Fehlerfall schnell wieder betriebsbereit sein. In kommerziellen Umgebungen sollen Router *Firewall*-Funktionen und Verkehrsfluß-Management durchführen. Der Anforderungen an den Router werden schon im voraus geplant, und Ressourcenanforderungen können dementsprechend angepasst werden. Falls notwendig, wird Persistenz für *Recovery* eingesetzt, statt für aktive Ausführung von Applikationen.
Aktive Elemente müssen sich dynamisch ändernder Arbeitslast anpassen. Zu diesen grundsätzlichen Anforderungen gesellt sich noch der Aspekt der Fehlerbehandlung, der allgemeinen Persistenz, des Managements und Accountings für persistente, von vielen Benutzern belegbare Ressourcen sowie des Schutzes der Applikationen gegeneinander. Fehlerbehandlung und Recovery sind hierarchisch angelegt; der gleiche Mechanismus, der für den Schutz der Applikationen zuständig ist, sollte auch erlauben, daß applikationsinterne Fehlerbehandlung durchgeführt werden kann. Sollten Applikationen intern aus mehreren Schutzbereichen Elemente beziehen (z.B. mehrere Prozesse), ist eine hochleistungsfähige Interprozeßkommunikation (IPC) notwendig, ebenso wie ein Ressourcenzuteilungsmechanismus.
Es sollte angemerkt werden, daß aktive Elemente in einer "feindlichen" Umgebung operieren. Für diese Umgebungen ist es unerlässlich, daß Sicherheitsprimitive einer formalen Prüfung unterzogen werden können. Die Verifikation durch ein formales Modell sollte kombiniert werden mit einer sorgfältigen Analyse der Implementation, um sicherzustellen, daß der Kern des aktiven Elements auch wirklich der verlangten Sicherheit entspricht. Diese Anforderungen sind zusammen mit dem Grad, den die Systeme erreichen, in Tabelle 1 kurz umrissen.

2.2 Zugriffskontrolle auf Ressourcen

Klassische Methoden der Zugriffskontrolle sind ACLs[1] und Capabilities. ACL benutzen Tupel (User, Actions), die jedem Objekt zugeordnet werden. Darin sind für jeden Benutzer die ihm zugeordneten Rechte festgehalten. Diese Methode wird oft in Betriebssystemen benutzt. Da ein aktives Netzwerkelement dem Authentifizierungssystem des Verbindungspunktes kein Vertrauen schenken kann, müssen Ressourcen und Zugriffskontrollen so ausgelegt sein, daß "anonyme" Benutzer dieses Element benutzen können. Mehr noch, die Ressourcen und Rechte, die für anonyme Verbindungen vergeben wurden, müssen voneinander abgeschottet und gesichert werden. ACLs sind für diesen Zweck jedoch nicht brauchbar.
ACLs sind unsicher. Wenn einem Programm zugestanden wird, Objekte für ein anderes beschreibbar zu machen, kann nicht garantiert werden, daß der Adreßraum der Objekte für ein Programm abgeschlossen bleibt. Dies erschwert natürlich die Fehlerisolation und macht Anpassungen schwer möglich.
Eine *Capability* ist ein nicht veränderbares Tupel (Name, Aktionen). Capability-Systeme beinhalten keine Kennung der Benutzeridentität. Das Belegen einer Capability ist notwendig für das Ausführen einer Aktion auf das Objekt. Der erreichbare Zustand eines Programms kann als reflexive transitive Hülle der anfänglichen Capabilities gesehen werden. Der Programmgraph ist geschlossen, was die notwendigen Grundlagen für die Sicherung der Information und die Eingrenzung von Programmfehlern liefert. Für aktive Netzwerke sind Capabilities wohl besser für das grundlegende Schutzmodell geeignet als ACL. Dies zeigen auch verschiedenartige Ansätze, die fast ausnahmslos auf ACLs verzichten und statt dessen Capabilities benutzen.

2.3 Scheduling und QoS in einem Netzwerkknoten

Aktive Netzwerkelemente müssen eine Unterstützung für nichtperiodische, isochrone und hart-periodische Übertragung bieten. Durch die dynamische Natur der Applikationen im Knoten wird auch verlangt, daß Kontroll- und Accountmechanismen für CPU-Auslastung, Bandbreitenauslastung, persistenten Speicherplatz, Arbeitsspeicher und den Overhead, der mit der Ressourcenvirtualisierung (Paging, ...) zusammen hängt, zur Verfügung stehen (Bsp: um Videobilder mit 60 Hz zu versenden (ein Bild pro 16.6 ms), darf die Applikation nicht durch Plattenzugriffe von der Dauer mehrerer Millisekunden unterbrochen werden). Während ein hoher Abstraktionsgrad eine praktische Anforderung für die Programmierung ist, ist die Fähigkeit, Ressourcenvirtualisierung auf einer möglichst niedrigen Ebene zu managen, essentiell. Eine klare Trennung dieser zwei Bereiche ist von großer Bedeutung.
Die Minimierung der Latenz-Abweichung ist kritisch für die Leistung der heutigen Internet-Protokolle wie TCP und allgemeiner für Protokolle mit *superlinearen Backoff-Strategien*. Speziell Ende-zu-Ende-Flußkontrollprotokolle (wie diejenigen, die bei TCP Verwendung finden) reagieren sensibel auf Bandbreitenvarianzen.

[1] Access Control List

2.4 Ausfallsicherheit und Fehlertoleranz

Die Ausfallsicherheit eines Netzwerkknotens hat lokale und globale Auswirkungen auf die Benutzbarkeit eines Netzwerks. Ausfälle eines Knotens haben die Tendenz zu kaskadieren, dies vor allem unter Hochlast. Recovery-Mechanismen könnten dazu beitragen, daß die Abhängigkeiten der Fehler ausgemerzt werden. Dies erfordert jedoch mindestens eine niedrige Fehlerrate, genaustens ausgearbeitete Recovery-Protokolle und vorreservierte Bandbreite von Recovery-Datenverkehr. Abhängig vom Fehlerintervall und dessen Dauer kann ein nicht-ausfallsicherer Knoten Datenverkehr im gesamten Netzwerk auslösen. Dies trifft vor allem auf die adaptiven Algorithmen zu (z.B. das *route flap*-Problem). Das Ausnutzen solcher Fehlerzustände ist ein möglicher Punkt für *denial-of-service*[2]-Attacken.
Dieses Problem wird in aktiven Netzwerken noch sehr viel schwerwiegender. Dadurch, daß individuelle Applikationen auch von Nicht-Experten implementiert werden können, besteht auch die Gefahr, daß eigene Recovery-Strategien eingesetzt werden. Im günstigsten Fall kann man damit rechnen, daß die Strategien nicht ausreichend getestet wurden. In vielen Fällen muß wohl davon ausgegangen werden, daß die Algorithmen fehlerhaftes Verhalten zeigen, weil sie nicht korrekt implementiert wurden und die Tragweite nicht erkannt wurde.

2.5 EROS

EROS[3] ist ein neuartiges Echtzeitsystem, das an der Universität von Pennsylvania entwickelt wird. Die Zugriffskontrolle ist bei EROS ausschließlich auf geschützten Capabilities aufgebaut. Capabilities sind in Capability-Pages (capages) gespeichert. Speicherschutz ist durch Partitionierung des Speichers sichergestellt. Capages können nicht in den Benutzeradreßraum abgebildet werden, daher besteht auch keine Möglichkeit, die Inhalte der Capages zu ändern. Jedes Objekt im System wird durch Capabilities benannt, alle Operationen auf diesen Objekten sind durch Capabilities authorisiert.
Alle Objekte in EROS sind persistent, auch Prozesse. Periodisch wird eine systemweite Überprüfung der Objekte durchgeführt und benutzte (veränderte) werden asynchron auf Platte gespeichert. Die Ausführungszeit für das überprüfen und Speichern beträgt augenblicklich unter 100 ms, sollte sich jedoch noch weiter in Richtung 10 ms verringern.
Weitergehende Beschreibungen finden sich vor allem in [Shap97], [ShFS96b] (Architektur) und [ShFS96a] (Implementierung).

2.5.1 Metadaten

Adreßräume und Prozesse in EROS werden aus Capages und Pages heraus gebildet. Ein Adreßraum ist als ein Baum aus Capages implementiert, dessen Blätter Pages sind (Abbildung 2). Nicht-abgebildete Seiten werden durch eine *Number Capability*, die den Wert 0 enthält, dargestellt. Adreßräume werden durch Capabilities benannt; Applikationen können, um Speicheraufteilung vorzunehmen, Adreßraum-Capabilities austauschen, die dann auf beide Adreßräume abgebildet werden.

[2] denial-of- service-Erklärung
[3] Extremely Reliable Operating System

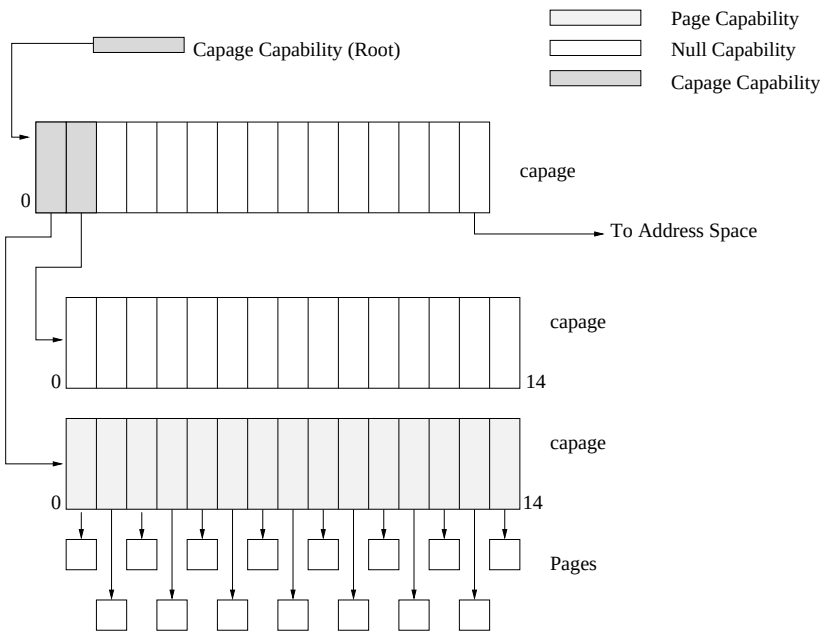

Abbildung 2: Ein Adreßraum von 15 Seiten

Prozesse werden als Arrangement von Capages konstruiert (Abbildung 3). EROS-Prozesse sind Single-Threads und besitzen zusätzlich zu den Architekturregister Capability-Register, die im Kernel implementiert sind.
Weil diese Metadaten (Mappings und Prozeßstruktur) aus persistenten Speicherobjekten zusammengesetzt werden, sind auch die daraus resultierenden Prozesse und Adreßmappings persistent. Dies bedeutet, daß Prozesse als ausfallsichere Objekte behandelt werden können. EROS braucht keine Dateien oder Verzeichnisse, die durch den Kernel unterstützt werden müssen, und bietet auch keine Hilfsmittel dafür an. Die von Unix bekannten Systemaufrufe `fork` und `exec` werden durch Benutzercode implementiert. Um die Effizienz zu steigern werden die Zustände der Mappings und Prozesse in maschinenabhängiger Form gespeichert [ShFS96b].

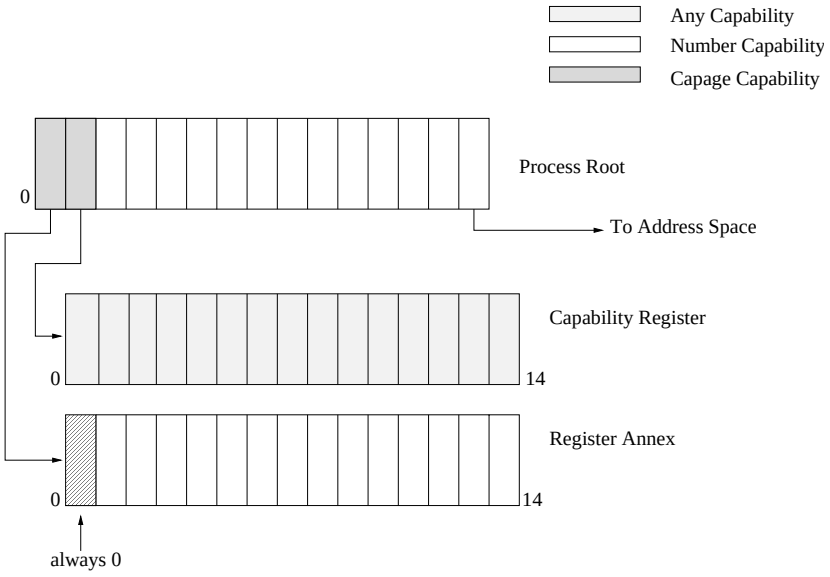

Abbildung 3: Ein EROS-Prozeß

2.5.2 Inter-Process Communikation

EROS-Prozesse kommunizieren mit Hilfe eines effizienten IPC-Mechanismus. Um einen Prozeß anzustoßen, muß der Auslösende eine *start capability* belegen und dem anzustoßenden Prozeß zuweisen. Jeder Capability-Anstoß transferiert 4 Capabilities und bis zu 64kB an zusammenhängenden Daten. Die grundlegenden Aufrufe sind `CALL`, `RETURN` und `SEND`. IPC-Operationen werden synchron ausgeführt und nicht gepuffert. Das Anstoßen eines Prozesses, üblicherweise um einen Dienst in Anspruch zu nehmen, blockiert bis zur Rückmeldung.
Die Bedingung, eine Start-Capability zu besitzen, stellt sicher, daß nur authorisierte Teilnehmer miteinander kommunizieren können. Diese Gegebenheit, kombiniert mit getrennten Adreßräumen, ist grundlegend für die Sicherheit des Systems.
Kerneldienste werden auch durch Einbinden von Capabilities ausgeführt. Alle diese Aufrufe sind atomar: die Ausführung geschieht entweder vollständig oder überhaupt nicht. Der Kernel ist durch Interrupts zu unterbrechen, geschachtelte Unterbrechungen werden in einer Bitmaske gespeichert, so daß alle Dienste entsprechend gehandhabt werden können, wenn der erste Trap endet.

2.5.3 Reserven: CPU-Kapazität und Working-Set-Reserven

Eine Erweiterung des Capability-Konzeptes für reale Ressourcen sind Capabilities für belegbare und virtuelle Ressourcen in Form von *CPU-Reserven* und *Working Set-Reserven*.

- CPU-Reserven
  Eine Prozessor-Kapazitäts-Reserve ist ein Tupel der Form (period, duration, quanta, start time, active priority, inactive priority).*Duration* ist hierbei die garantierte Mindestrechenzeit innerhalb einer *period*, beginnend von *start time*.*Quanta* spezifiziert, wie oft ein Prozeß von einem anderen verdrängt werden kann, der in der gleichen Reserve fährt. Der Eintrag *active priority* gibt die Priorität an, mit welcher der Prozeß innerhalb der Reserve ausgeführt wird. Die Priorität *inactive priority* wird benutzt, wenn die *period* des Prozesses beendet ist. Sollte diese geringer sein als die des Idle-Prozesses des Kernels, wird der Prozeß nur in der definierten Reserve ausgeführt.
  CPU-Reserven werden mit dem Reserve-Manager belegt. Dieser nimmt den Belegungswunsch entgegen und entscheidet, ob der Wunsch keine anderen Prozesse beeinträchtigt oder stört. Im Erfolgsfall wird eine Capability zurückgeliefert, die im Scheduler berücksichtigt werden kann.
  Im Gegensatz zu früheren Implementierungen können Prozessor-Reserven nicht über IPC-Operationen vererbt werden, dies würde nun eine Schutzverletzung nach sich ziehen. Würde dies zugelassen, so könnte ein Prozeß Prozessorzeit beanspruchen und nach dem Aufruf beenden, was zur Folge hätte, daß der aufgerufene Prozeß auf Rückmeldung warten würde und damit das System für andere Prozesse sperren würde. Dieses Konzept lehnt sich an *QoS cross-talk* an [LeMB+96].

- Working Set-Reserven
  Die Reservierung von Prozessorkapazität ist ohne großen Vorteil, wenn die Appli-

kation nicht im Speicher residiert. Um sicherzustellen, daß geschwindigkeitskritischer Code und zugehörige Daten auch im Speicher resident gehalten werden, bietet EROS sogenannte *Working set-Reserven*. Dieses Working Set beschreibt die Anzahl der Gesamtseiten und Capages, ebenso wie die Zahl der geänderten Seiten/Capages. Working Sets sind unterteilbar, eine Applikation kann freie Reserven belegen und Reservierungen von einem Working Set zu einem anderen transferieren. Weiterhin können Working Sets *non-persistent* sein, der die Daten aus der Gültigkeitsprüfung herausnimmt.
Typische Applikationen definieren ein einzelnes Working Set für ihren Prozeß. Applikationen können auch einen Adreßraum spezifizieren, der zu diesem Working Set zugeordnet ist. Dies ermöglicht Protokollmodulen, Working Sets als nicht-persistenten Eingang für Pakete zu deklarieren, was den Aufwand zur Überprüfung und beim Aufräumen beim Neustart deutlich reduziert.

2.5.4 Paketdemultiplexing

EROS implementiert seine Netzwerkprotokollunterstützung als eine Sammlung von Benutzerprozessen. Bei der Architektur des Systems wurde darauf geachtet, daß die Ausgeglichenheit zwischen Leistung, Sicherheit und Flexibilität gegeben ist. Die Unterstützung von *untrusted Software* hat jedoch ihren Preis: Speichersharing zwischen Prozessen erscheint nicht ratsam, das System erlaubt jedoch, Sicherheitsbarrieren bei *trusted Software* teilweise abzubauen (zur Erhöhung der Leistungsfähigkeit, z.B. bei IP) und bei anderen die Barrieren hochzusetzen.
Die Paket-Dispatcher und alle Applikationen teilen sich einen Speicherbereich, den *packet buffer*. Dieser Bereich ist in gleich große Paketzellen und Paket-Warteschlangen eingeteilt. Jede Netzwerkschnittstelle hat einen *packet reader/classifier* damit gekoppelt, dessen Aufgabe das Demultiplexen der Paketströme zu den entsprechenden Aplikationen ist (Abbildung 4).
Eingehende Pakete werden direkt in die nächste freie Speicherzelle eingelesen und ein Paketfilter entscheidet, welcher Applikation das Paket zugestellt werden soll. Das Paket wird in die entsprechende Warteschlange eingereiht und die Applikation mittels `SEND` benachrichtigt. Einigen Applikationen werden nur Leserechte auf die Pakete zugeteilt, diese müssen Kopien der Pakete anfertigen, um damit arbeiten zu können.
Jeder Client hat eine bestimmte Anzahl freier Plätze in der Wartschlange, darüberhinausgehende Pakete werden vom Dispatcher verworfen. Pakete können auf zwei Arten verworfen werden: *last-in* oder *first-in*, je nach Bedürfnissen des Clients. Der Dispatcher wirkt somit als erster Filter/Widerstand zwischen Eingabe und Netzwerk, dessen Inhalt unbekannt und unkontrolliert ist, und dem *protocol-processing client*, dessen Ressourcen durch den Knotenadministrator kontrolliert werden.

2.6 Scout: Ein Kommunikationsorientiertes Betriebssystem

Scout ist ein Betriebssystem, das für Systeme entwickelt wurde, die über die *National Information Infrastructure* (NII) verbunden sind. SCOUT bietet für die Erstellung von Betriebssystemcode eine kommunikationsorientierte Softwarearchitektur, speziell füer Systeme, die über NII erreichbar sein sollen.

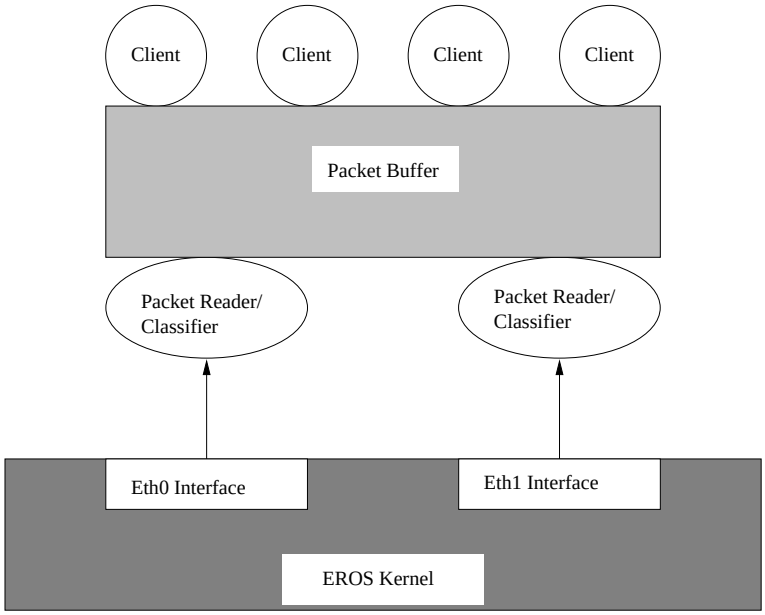

Abbildung 4: Paketklassifikation bei EROS

Scout bietet explizite Pfadabstraktion, die effektives Ressourcenmanagement erleichtert und Optimierungen des kritischen Pfades, dem die I/O-Daten folgen, ermöglicht. Diese Optimierungen, einhergehend mit fortgeschrittenen Compilertechniken, münden in ein System, das verlässliche und skalierbare Leistung zeigt.
Scout ist eine neue Betriebssystemarchitektur, die speziell zur Befriedigung der Bedürfnisse von kommunikationsorientierten Systemen entwickelt wurde. Scout besteht aus einer modularen Struktur, die durch eine neue Abstraktion, dem *Pfad*, ergänzt wird. Die modulare Struktur ermöglicht den effizienten Bau von Systemen, die genau auf die Erfordernisse bestimmter Komponenten[4] abgestimmt sind. Pfade adressieren Bereiche, die mit der Leistungsfähigkeit und Qualität des Kommunikationsdienstes in Verbindung stehen. Als Pfad kann man einen vertikalen Schnitt durch ein System mit mehreren Ebenen ansehen oder abstrakt einen bidirektionalen Datenfluß. Als solcher traversiert ein Pfad mehrere Module in einem Scout-System. Pfade bieten den Modulen, die die Daten durch das System senden, einen zusätzlichen Kontext. Dieser Kontext erlaubt es in manchen Fällen, die Datenberechnung effizienter zu gestalten oder die Qualität des Ressourcenmanagements, wie z.B. Scheduling oder Speicherverwaltung, zu erhöhen. Unter anderem benutzt Scout auch einen neuen Mechanismus, um Netzwerk-Datenpakete zu klassifizieren.
Mehrere Studien sollen belegen, daß die Pfadabstraktion von Scout die Latenzzeit zur Paketweiterleitung in einem Netzwerksubsystem verringert. Diese Studien bringen auch wichtige Einblicke in das Leistungsverhalten von Netzwerksubsystemen auf modernen RISC-Maschinen.
Für weitere Informationen sollten [MoMO+95], [Mosb97] herangezogen werden.

[4]Meist kleine spezialisierte, oft auch mobile Einheiten. Beispiele wären Fernbedienungen, PIMs, Platten, die über das Netz angesprochen werden, Kameras, Displays, ...

2.7 Weitere Betriebssytemansätze

Wie nicht anders zu erwarten, werden viele Betriebssystemansätze bei Active Networks verfolgt. Im folgenden sollen weitere Betriebssysteme oder -ansätze vorgestellt werden [SMSF97].

- Mach/RT-Mach, Mach 4
  Das Mach-Betriebssystem wird vielfach als Plattform für Forschungszwecke eingesetzt. Wie EROS führt es ein *capability-oriented scheme* ein, indem es Abstraktionen bereitstellt (z.B. Ports). Versuche, die Leistungsgrenze zu erhöhen, führten auf Beschränkungen hervorgerufen durch interne Entwurfsentscheidungen; bei EROS wurden diese Leistungsprobleme durch sorgfältige Auswahl der Abstraktionen umgangen, was auch in Benchmarktests gezeigt werden kann. Vor allem im Bereich der Interprozeßkommunikation wird dies deutlich.
  Für das Ressourcenmultiplexing weitaus interessanter sind die Spezialisierungen von Mach, wie z.B. RT-Mach. RT-Mach ist ein Ansatz, um Echtzeitverhalten von Anwendungen zu spezifizieren und zu erzwingen. Der wichtigste Mechanismus der sich aus dieser Forschungsarbeit herauskristallisiert hat, ist die Idee der *Processor Capacity Reserves*. Der Mechanismus bei EROS ist auf dieser früheren Arbeit aufgebaut, Reservierungen und prioritätsbasiertes Scheduling eingeschlossen. Als Ergebnis ist EROS fähig zur Reservierung von "weichen" Reserven als auch CPU-Zeiten.

- Nemesis
  Obwohl Nemesis (University of Cambridge) eher darauf abzielt, Multimedia auf Betriebssystemebene zu unterstützen, als zur Konstruktion von aktiven Netzwerken beizutragen, hat es viele gemeinsame Aspekte mit EROS. Speziell wurde eine relativ niedrige Schicht gewählt, auf der Anwendungsprogramme das Multiplexing steuern können, im Gegensatz zu anderen Ansätzen, bei denen dies über Prioritäten geregelt wird (z.B. UNIX scheduling). Dienstspezifikationen werden als Aspekte zur Sicherheit angesehen (z.B. *denial-of-service attacks*), das Nemesis-System bietet angemessenen Schutz gegen Ausspähen, Sperren des Rechners, ...
  Dennoch ist der Ansatz bezüglich Sicherheit und Schutz von EROS besser zum Aufbau eines aktiven Netzwerkknotens geeignet. Der Schutz des Systems bei Nemesis basiert auf einem einzigen Adreßraum, was ermöglicht, Code und Daten besser teilen zu können. Systeme werden durch Objekte aufgebaut, die über Schnittstellen miteinander kommunizieren, die mit MIDDL[5] definiert wurden. Es existiert zwar eine annehmbare Sprache zu Beschreibung der Schnittstellen, bei den Programmcodes wird aber darauf gebaut, daß es sich um vertrauenswürdigen Code handelt. EROS trifft diese Annahme nicht, Programmcode wird als nicht vertrauenswürdig angesehen.

- L4/Mungi
  L4 ist ein kleines System, das darauf beruht, daß Applikationen zur Verfügung stehen, die Betriebssystemdienste und Treiber bereitstellen. Die Hauptvorteile von L4 sind hauptsächlich seine Größe und seine schnelle IPC. Mungi ist ein *capability system*-Emulator, der auf den L4-Mikrokernel aufsetzt. Mungi basiert auf Paßwörtern und implementiert Persistenz auf Prozeßbasis.

[5] eine Schnittstellen-Definitionssprache, die es ermöglicht, Schnittstellen zu typisieren

1993 wurden bemerkenswerte IPC-Leistungsergebnisse für Pentiumrechner in Verbindung mit dem Redesign von L3 vorgestellt. Verschiedene Änderungen wurden eingebracht, unter anderem solche, die für eine schnelle IPC vonnöten sind. Eine sich daraus ergebende Folgerung war, daß Mikrokernel klein sein müssen, um die geforderte Leistung zu erreichen.
Während bei L4 alles daran gesetzt wird, den Kernel zu verkleinern, wird bei EROS ein vollständiger Supervisor eingesetzt. Bei EROS wurde versucht, durch gute Strukturierung eine bessere Leistung zu erzielen, obwohl der Supervisor alle notwendigen Funktionen und Sicherheitskontrollen beinhaltet. Weiterhin ist die Implementierung der Sicherheitsmechanismen bei L4 in verschiedenen Komponenten untergebracht, bei EROS jedoch in einem einzigen.
EROS ist noch in weiteren Aspekten L4 überlegen: EROS bietet CoProzeß-Aufrufe statt RPC als primitiven IPC-Mechanismus, da RPC in manchen Fällen neue (und nützliche) Formen von *Co-Tasking* und *Delegation* nicht erlaubt. Zweitens benutzt L4 operationsspezifische Systemaufrufe, EROS hingegen benutzt *capability invocations*.

- SCAP
  SCAP ist ein System, das auf *capabilities* und *access control lists* beruht. Systeme mit Capabilityeigenschaften können durch bestimmte Hardwaregegebenheiten verbessert werden. Vorgeschlagen wurden Speicherstrukturen, die ein effizientes Management erlauben, speziell die Autoritätenverwaltung und Zugriffsverwaltung. Doch wurde insgesamt nur das IPC-System von SCAP implementiert, das System als Ganzes wurde weder fertiggestellt noch mit realen Anwendungsprogrammen getestet.
  Die Arbeit an SCAP verfolgt vor allem die klare Definition mehrerer Sicherheits-Policies und die Bewertung der Effizienz, mit der Capabilities in diese Policies eingebunden werden können.
  Die Arbeit an SCAP betrachtet nicht den Aspekt der Persistenz, so daß auch keine Aussagen gemacht werden können, wie sich das System beim Speichern von Objekten verhält oder die Komplexität des Systems, wenn das Capability-Modell nicht auf die Speicherebene erweitert wird.
  Da nur Fragemente von SCAP implementiert wurden, ist es unklar, wie das Endresultat bewertet werden kann. SCAP bietet erste Beweise für Hardwareargumente und demonstriert, daß weitere Argumente noch nicht vollständig ausgeführt wurden.

- KeyKOS
  KeyKOS wurde als kommerzielles Capability-System konzipiert, aus dem später das EROS-Modell abgeleitet wurde. KeyKOS wurde für die System/370-Architektur entwickelt und war leidlich erfolgreich in Transaktionsanwendungen. Wie EROS bietet auch KeyKOS universelle Persistenz und eine kleine Menge von Architekturabstraktionen. Es wurde jedoch kein Versuch unternommen, das Betriebssystem der darunterliegenden Hardware und deren Kapazitäten voll anzupassen.
  KeyKOS bietet Ressourcenreservierung und *end-to-end scheduling*, grundlegende Kontrollen über Repräsentation und Scheduling wurden verborgen. Diese Bereiche wurden in EROS genauer beleuchtet und drastisch verbessert. Darüber hinaus hatte das KeyKOS-Design Schwächen in den Abstraktionen, die die Kon-

sistenzregeln für den Supervisor drastisch verkomplizierten.
Wo KeyKOS hauptsächlich die Frage betrachtet, was mit einem Capability-System machbar ist, geht EROS den Weg der Verfeinerung des Systems bis zu dem Punkt, bei dem die Notwendigkeit der Implementierung in Hardware klar wegfällt. Die EROS-Architektur baut auf die Erfahrungen mit KeyKOS auf, um ein neues, schnelleres und einfacher zu überwachendes System zu erstellen, das bessere Unterstützung für leistungskritische Anwendungen bietet.

- BiiN/OS
  Biin/OS ist ein Gemeinschaftsprojekt von Intel und Siemens aus dem Jahre 1983, das in einer komplett neuen Computerarchitektur mit Betriebssystem endete. Der Hardwareanteil des Systems wurde als Intel 80960XA-Mikroprozessor bekannt. Das BiiN-Projekt wurde durch den Rückzug von Siemens beendet. Trotz der Aufgabe ist es sicherlich das erfolgreichste hardwarebasierte Capabilitysystem, das bis heute implementiert wurde.
  Zur Hardwareebene ist zu erwähnen, daß dort die gesamte Prozeßwarteschlange abgearbeitet wird, ebenso werden Kontextwechsel direkt in Mikrocode ausgeführt. Die Hardware unterstützt direkt Aufrufe über verschiedene Rechner; die Aufrufe werden in 15 ms abgearbeitet, im Gegensatz zur lokalen Ausführungszeit von 1 ms. Prozesse werden in 32 Prioritätsklassen unterteilt, um es dem Betriebssystem zu erlauben, das Scheduling flexibler zu handhaben.
  BiiN war von Anfang an objektorientiert konzipiert, mit Adreßdeskriptoren (der Name von BiiN für "capability"), die als Objektreferenzen dienen. BiiN benutzt eine *tagged-memory*-Architektur, die es dem Prozessor erlaubt, zwischen Adreßdeskriptor und Daten zu unterscheiden. Die Adreßdeskriptoren sind ähnlich der Pentium-Segmentierung, Adreßdeskriptoren referenzieren eine globale Tabelle, die die Adressen der Objekte und Subsysteme ebenso wie die Rechte zum Zugriff auf diese enthält. Jedes Objekt hat seinen eigenen Adreßraum, der Zugriff wird durch einfache ACLs kontrolliert, basierend auf dem Hardware-Deksriptor-Mechanismus.
  Während BiiN/OS eine komplette Implementierung auf VLSI-Basis und Mikrocode ist, kann EROS auf üblicher Hardware ausgeführt werden und bietet darauf äquivalente Leistung. EROS benutzt, anders als BiiN/OS, Capabilities statt Objektreferenzen. Dies führt zu einem System, das sehr viel weniger komplex ist.

3 Programmiersprachen

3.1 Sicherheitsaspekte in Programmiersprachen

Ein Netzwerk als solches programmierbar zu machen, bietet große Vorteile und Flexibilität, erzeugt aber auch neue Sicherheitslücken. Die Herausforderung ist, die Flexibilität anzubieten, ohne die Sicherheit zu vernachlässigen. Eine weitere Frage ist, wie die Flexibilität mit einer akzeptablen Leistung geboten werden kann. Es erscheint nun sinnvoll zu untersuchen, wie sich ein programmiersprachlicher Ansatz auswirkt. Moderne Programmiersprachen wie Java, Modula-3 oder ML bieten signifikante Vorteile in Bezug auf die Sicherheit. Diese Vorteile ergeben sich aus der Benutzung von strenger Typisierung, automatischem Speichermanagement (Garbage

Collection) und Arraygrenzenüberprüfung. Sicherheit der Information bedeutet, daß die richtige Information den richtigen Empfänger zur richtigen Zeit am richtigen Ort erreicht. Sicherheitslücken können das unauthorisierte Ansehen von Informationen, denial-of-service und Einfügen falscher Informationen ermöglichen. Diese Fehlerquellen werden in Zukunft immer häufiger auftreten, wenn die Sicherheitsaspekte nicht im Entwurf eines Systems berücksichtigt werden.
Sicherheitsprobleme wie z.B. mit dem "Internet-Worm", den Web-Browsern oder auch mit Java, zeigen die Dringlichkeit der Sicherheitsfragen in verteilten Systemen, auch in aktiven Netzwerken. Obwohl alle diese Fälle aus Sicherheitslücken entstanden sind, sind einige von ihnen auf Schwachstellen in der Programmiersprache zurückzuführen, hauptsächlich von C. Sprachen wie SML oder Java vermeiden diese Probleme, indem "Zeigersicherheit" bereitgestellt wird oder keine Zeiger zur Verfügung stehen; in zeigersicheren Sprachen kann ein Zeiger nicht in Bereiche zeigen, die für das Programm nicht vorgesehen sind. Voraussetzung für diese Zeigersicherheit ist jedoch das Vorhandensein von strenger (nicht notwendigerweise statischer) Typisierung, Überprüfung von Arraygrenzen und einem automatischen Speichermanagement oder Garbage Collection.

Formale Semantik der Programmiersprachen. Für den Einsatz im Bereich der aktiven Netzwerke werden Sprachen bevorzugt, deren Semantik mit formalen Methoden überprüft werden kann. "Formale Methoden" sind Sammlungen von Techniken, die Ideen aus der Mathematik auch auf Probleme in Hardware oder Software übertragen. Solche Techniken sind nützlich und Bestandteil der Forschung der letzten 20 Jahre, speziell im Bereich der Programmspezifikation und dem Entwurf von Programmiersprachen.
Diese Techniken tragen wesentlich dazu bei, daß die Sicherheit von Programmiersprachen im voraus überprüft werden kann, ohne Tests durchführen zu müssen oder gar auf Fehlermeldungen von Benutzern warten zu müssen. SML oder Ada sind Beispiele für die Verwendung von formalen Methoden beim Sprachentwurf.

Authentizierte typgeprüfte Module. Viele Stimmen wurde laut, daß sicherer Speicher für Objekte in einem aktiven Netzwerk ebenso benötigt werde, wie neue kryptographische Technologien für digitale Unterschriften, die in dieser Umgebung benutzt werden können. Speziell sollten Module in einer maschinenunabhängigen Form gespeichert werden können, die dann direkt signiert oder durch einen sicheren Hash-Mechanismus unterstützt werden. Neue Technologien wie MD5 können das Typüberprüfungssystem im verteilten System wesentlich verbessern.

Nebenläufigkeit und Garbage Collection. Garbage Collection ist ein wesentlicher Punkt, der betrachtet werden muß, da viele Programmiersprachen, die in aktiven Netzwerken Verwendung finden sollen, eine automatische Speicherverwaltung haben, die natürlich ab und an eine Garbage Collection durchführen müssen. Garbage Collection ist wesentlich für Systeme wie aktive Brücken oder Router, da dem Benutzer nicht die Verantwortung für die Speicherverwaltung (speziell die Speicherfreigabe) überlassen werden kann, da durch falsche oder fehlende Anwendung das System leicht in einen katastrophalen Zustand übergehen kann. Andererseits legt eine Garbage Collection das System sehr leicht für einige Zeit lahm, was für interaktive Programme nicht allzu schlimm ist, für Echtzeitanwendungen jedoch dramatisch. Diese Überlegung hat zur Einführung von nebenläufigen Garbage Collections geführt, speziell für die Sprache ML ist eine sehr leistungsfähige Strategie entwickelt worden [Reppb]. Diese Strategie

(Concurrent GC replicating collection) ist eine sehr einfache und elegante Lösung, die die Vorteile der parallelen Ausführung ausnutzt und Pausen ausschaltet. Diese Technik könnte auch für Java implementiert werden, was auch dort einen Leistungssprung zur Folge haben dürfte.

3.2 PLAN

Die Programmiersprache PLAN[6] wurde in Zusammenhang mit dem *SwitchWare*-Projekt entwickelt [SmFa96]. Es dient als Basis für leichtgewichtigen mobilen Code, der in Paketen zwischen den Knoten transportiert wird und in den passierten Knoten ausgeführt werden soll. Wie bei Java für Browser oder Modula-3 für SPIN, ist ein Teil des Ansatzes die Benutzung von typsicherem Code und einer Garbage Collection, die garantieren sollen, daß Programme sicher vor Zeigerfehlern werden und deshalb nicht in unzulässigen Speicherbereichen arbeiten können. Die Integrität des Systems soll hiermit sichergestellt werden. PLAN geht jedoch darüber hinaus, indem aktive Sicherheitsmechanismen als Bestandteil der Sprache enthalten sind. Zentrales Ziel ist, reine PLAN-Programme überall im Netzwerk ausführen zu können, ohne daß der Code authentifiziert oder in spezieller Form gesichert werden muß, was PLAN-Programme äußerst kompakt und leichtgewichtig macht.
Schlüsselentscheidungen ermöglichen es, die Authentifizierung von PLAN-Programmen wegfallen zu lassen. So ist dies in erster Linie durch die formale Semantik von PLAN-Programmen zu begründen, die genaue Aussagen zulässt über die Funktion der Programme: was können diese tun und, viel wichtiger, was nicht. So kann durch den formalen Beweis der Typsicherheit von PLAN die Unterteilung in Adreßräume entfallen.
Ein neuer Aspekt der Semantik von PLAN ist die der *remote execution*. Weil einige Funktionen Authentifizierung verlangen (müssen), erlaubt PLAN den Aufruf von Diensten im entfernten System, diese Dienste können auch in anderen Sprachen als PLAN verwirklicht sein. Diese Routinen können die gewünschte Authentifizierung verwirklichen. Die Schnittstelle zwischen PLAN und diesen Diensten ist in einer RPC-ähnlichen Form formuliert. Dadurch können die Routinen in einer Vielzahl von Sprachen geschrieben sein, die Schnittstelle bleibt aber immer noch typisiert und formal greifbar.
Ein weiterer Mechanismus in PLAN ist ähnlich einer Paketfilterung. Die Ausdruckskraft der Sprache ist beschränkt, so daß unsichere Aktionen in vielen Fällen nicht möglich sind. So beschränkt PLAN vom Aufbau her bereits die Benutzung von Ressourcen (Prozessorzeit, Speicher, Bandbreite), so daß diese einfach zu kontrollieren sind.
PLAN ist eine Scriptsprache, die es erlaubt, zwei Arten von Funktionen im aktiven Netzwerk auszuführen:

- Ressourcenerkennung
- Diagnosefunktionen

Da Router Betriebsmittel sind, die von vielen Benutzern benutzt werden, muß die Benutzung vielfach beschnitten werden. Deshalb werden beim Ansatz mit PLAN zwei Funktionalitätsebenen unterschieden:

[6] Programming Language for Active Networks

- PLAN-level (untere Ebene)
  Bestehend aus PLAN-Programmen, die auf Standardfunktionen der Router basieren. Diese Funktionen können durch alle aktiven Pakete aufgerufen werden. Diese Ebene stellt die Funktionalität der Ressourcenerkennung und der Diagnose bereit, für die keine Authentifizierung notwendig ist. Die Benutzung dieser Ebene ist beschränkt durch die Prozessorzeit und den Arbeitsspeicher. Informationen können hier eingesehen und Dienste in Anspruch genommen werden.

- Service-level (höhere Ebene)
  Die Service-Ebene beinhaltet Dienste, die nur bestimmten Benutzern zugänglich sein sollen. Hierfür wird die Authentifizierung verlangt. Diese Ebene stellt eine reiche Auswahl von Protokollen und Funktionen des aktiven Netzwerks zur Verfügung. Allgemeine Programmierung, die auch verschiedene Prioritäten zulässt, wird hier geboten.

PLAN limitiert die Anzahl der Berechnungsschritte, die auf einem Router ausgeführt werden dürfen, und weiterhin die Anzahl der Router, auf denen das Programm ausgeführt werden darf. Die Ausführung eines Programms kann Pakete erzeugen, die über das Netz verschickt werden, jedoch wird die Anzahl der versendeten Pakete überwacht und an die Grenze des Quellpakets gekoppelt (*resource bound*): es darf nur eine bestimmte Anzahl von Paketen verschickt werden. Ein Programm auf PLAN-Ebene, das in einer bestimmten Rechenzeit nicht terminiert, wird angehalten ohne den Zustand des Routers verändert zu haben. Pakete können nicht direkt miteinander kommunizieren, außer indirekt durch Programme an den Endpunkten oder Dienstaufrufen, falls diese erlaubt sind. PLAN ist eine kleine funktionale Scriptsprache mit einer Syntax, die ähnlich ML ist und dessen grundlegendes Programmiermodell eine Spielart der entfernten Auswertung ist. Ein Beispielprogramm soll diesen Sachverhalt verdeutlichen:

```
fun ping (source:  host, destination:  host) :  unit =
    if (thisHost() = destination) then
       OnRemote(ack(), source, getRB(), defaultRoute)
    else
       OnRemote(ping(source, destination),
                destination, getRB(), defaultRoute)
```

`ack()` stellt hierbei eine einfache Acknowledge-Funktion dar. Das Programm wird mit den Parametern `source` und `destination` und aufgerufen, wobei `destination` den Rechner adressiert, der angepingt werden soll und `source` den Rechner angibt, an den das Acknowledge gesendet werden soll. Das Konzept des Programmes ist, ein Paket mit `OnRemote` weiterzuversenden. Diese Funktion benutzt vier Argumente:

1. die Funktion auf dem entfernten Rechner, die ausgeführt werden soll
2. die Adresse des Rechners, auf dem das Ergebnis vorliegen soll
3. die Grenze der Anzahl der Auswertungen (*RessourceBound*)
4. die Routing-Funktion

Jeder Aufruf der `ping`-Funktion mittels `OnRemote` erniedrigt den RessourceBound um 1. Da keine Funktion existiert, den Zähler zu erhöhen, wird das Programm spätestens nach n Schritten abgebrochen, d.h. es kann nicht passieren, daß das Programm nicht terminiert. Wesentlich brauchbarere Beispiele sind folgende Programme:

- Implementation von traceroute basierend auf dem bisherigen Vorgehen in nicht-aktiven Netzwerken:

```
fun traceroute(src: host, dest: host, count: int) : unit =
    OnRemote(ack(count, thisHost()), src, count, defaultRoute)
    if (thisHost() <> dest)
    then
       let val next: host = defaultRoute(dest) in
           OnNeighbor(traceroute(src, dest, count+1),
                      next, getRB())
       end else ()
```

  `OnNeigbor()` sucht den nächsten Host auf dem Weg zum Ziel. Diese Funktion ist der IP-Funktion nachempfunden, die eine Serie von Nachrichten verschickt, die sich erhöhende TTL[7]-Werte liefert. Dieses PLAN-Programm ist analog dazu, es liefert den "hop-count" und den zugehörigen Host.

- Neuer Ansatz:

```
fun collectroute(src: host, dest: host, l: host list) : unit =
    let val this: host = thisHost() in
        if (this <> dest)
        then
             let val next: host = defaultRoute(dest) in
                 OnNeighbor(collectroute(src, dest, this::l),
                            next, getRB())
             end
        else
             OnRemote(ack(this::l), src, getRB(), defaultRoute)
    end
```

  Diese Art, traceroute zu schreiben, ist sehr viel besser als die für IP-Netzwerke implementierte: es steht immer Information über eine konsistente Route zu Verfügung. Dieses Beispiel zeigt auch, daß durch die Benutzung von aktiven Netzwerken wesentlich weniger Bandbreite benutzt werden muß, als dies in herkömmlichen Netzwerken der Fall ist. Nach jedem "Hop" wird der aktuelle Hostname der Liste hinzugefügt und am Schluß auf dem aufrufenden Rechner zur weiteren Verarbeitung bereitgestellt.

Weitere Beispiele sollten [SMSF97], [GuHK+97b], [MoHi97], [pla97] sowie [GuHK+97a] entnommen werden. In diesen Veröffentlichungen werden auch noch weitere Details besprochen, die hier leider nicht aufgenommen werden können.

[7] Time to live

3.3 ML

Die Programmiersprache ML[8] existiert in vielen verschiedenen Versionen: ML/NJ (Standard ML of New Jersey), CML (Concurrent ML), Caml und einigen weiteren Derivaten.
Hier sollen einige Grundzüge der Programmiersprache betrachtet werden, die im wesentlichen allen Abkömmlingen gemeinsam sind.
ML ist eine von von vielen Programmiersprachen, die in den 80ern entwickelt wurden und die als geeignet für kritische Systeme und Anwendungen angesehen werden. ML bietet eine hervorragende Mischung aus Ausdrucksstärke und Komplexität. Wegen seines Typen- und Modulsystems kann ML Sicherheit und Robustheit mit der Flexibilität von Sprachen mit dynamischer Typisierung (z.B. Lisp) kombinieren. Im Gegensatz zu vielen anderen Sprachen sind die Grundlagen aus wissenschaftlichen Arbeiten hervorgegangen und deshalb gut untersucht. Die ML-Sprachen basieren auf dem λ-Kalkül, der für diesen Zweck jedoch etwas "benutzerfreundlicher" gestaltet wurde. Die Auswertung von Funktionen und Ausdrücken geschieht mittels *call-by-value*[9], Milners Typsystem wird verwendet.
Das LCF Beweissystem erschien 1972 an der Stanford University. LCF wurde dann die ML-Sprache hinzugefügt. Von 1981 bis 1986 wurde eine Version von ML in Zusammenarbeit von INRIA und der Cambridge University entwickelt. 1984 wurde ein erweitertes ML – Standard ML (SML) – aus der Taufe gehoben. 1987 erschien die erste Version von Concurrent ML und 1989 eine weitere Version von SML. Caml Light, eine kleinere, portablere Implementierung der Sprache Caml (ähnlich ML) wurde 1990 vorgestellt.
Die nachfolgende Zusammenfassung soll einen Einblick in die Fähigkeiten und Stärken von ML geben:

- Hochsprachliches Programmiermodell
  Eine Hochsprache macht das Programmieren einfacher und effizienter. Automatisierung der Erzeugung des Maschinencodes und die Abstraktion von der tatsächlichen Hardware wird ermöglicht und zuverlässiger gestaltet. Auch die Speicherverwaltung und die Darstellung verschiedener Datenformate kann dem Compiler überlassen werden. Das Standardmodell von SML basiert auf funktionalen Programmiersprachen, die hauptsächlich auf den λ-Kalkül aufbauen.

- Sicherheit
  Das Typisierungssystem, das Speichermanagement und die Ausnahmebehandlung garantieren, daß auch fehlerhafte ML-Programme beim Absturz nicht das komplette System mitreissen. Natürlich ist es denkbar, daß ein Programm nicht korrekt funktioniert, z.B. durch eine Ausnahmebehandlung, die zwar aufgerufen wird, aber nicht ausgeführt werden kann. Diese Programme werden jedoch vom System beendet und führen zu einer *Abnormal Program Termination.*

- Sicherheit und Robustheit
  Statische Typüberprüfung entdeckt viele Fehler bereits zur Compilierzeit. Die Fehlererkennung wurde um Mustererkennung erweitert. Die Mustererkennung

[8] Ursprünglich stand ML für Meta Language, als solche wurde ML in verschiedenen Forschungssystemen benutzt

[9] z.B. werden Argumente vor der Übergabe an Funktionen ausgewertet

ermöglicht, bei bedingten Anweisungen oder Fallunterscheidungen alle Fälle zu erfassen, da der Compiler schon erkennen kann, ob Ausdrücke unvollständig sind. Der Mechanismus dient auch zur Vervollständiggung von Ausnahmebehandlungsroutinen.

- Ausdrucksstärke
  Die Möglichkeiten, Funktionen als Variablen oder Werte zu benutzen, höherwertige Funktionen weitgehend als Kontrollstrukturen zu verwenden, Mustererkennung zur Analyse der Daten und die Verfügbarkeit von Befehlskonstrukten bieten eine große Vielfalt an Ausdrucksmöglichkeiten innerhalb eines einfachen und gleichbleibenden konzeptuellen Rahmens. Die Kombination von Zuständen und höherwertigen Funktionen ist ein sehr mächtiges Konzept, das jedoch wie bei objektorientierten Sprachen leicht zu Problemen in der Spezifikation führen kann.

- Flexible, aber strenge Typisierung
  Das Typsystem von ML ist einfach und flexibel. Jeder Typkonstruktor kann auf jeden Typ ohne Beschränkung angewandt werden, benutzerdefinierte Typen werden wie primitive Typen behandelt. Die Flexibilität wird durch parametrisierten Polymorphismus erreicht. Neue Typen können derart definiert werden, daß sie genau auf das Problem zugeschnitten sind, Datenabstraktion (Verbergen von Information) und Datenrepräsentation wird ebenso unterstützt. Automatische Ableitung der allgemeinsten Typschemata gestaltet die Programmierung einfach. Typüberprüfung bietet Sicherheit, indem eine groß Zahl häufig gemachter Fehler entdeckt wird. Der wichtigste Aspekt der Typüberprüfung ist vielleicht, daß ML einfache Konstrukte zur Definition von Typen bietet, jedoch an dieser Stelle einen Schritt weitergeht und auch automatisch die korrekte Verwendung der Typen sicherstellt.

- Modularer Aufbau
  Das Modulsystem von ML ist eine natürliche Weiterführung des darunterliegenden polymorphen Typensystems. Separation von Definition und Implementierung wird ebenso unterstützt wie Abstraktion und Parametrisierung. Diese Fähigkeiten von ML werden besonders effektiv bei der Strukturierung von großen Programmen und der Erzeugung von generischen, wiederverwendbaren Softwarekomponenten.

- Effizienz
  Für ML wurden bereits hochoptimierende Compiler entwickelt, deren Code mit dem anderer Sprachen durchaus mithalten kann (z.B. C). Andererseits ist das Speicherplatzaufkommen nicht allzu gut. Die Kosten für ein automatisches Speichermanagement und ein uniformes und einfaches Typensystem sind trotz fortschrittlicher Compilertechnik immer noch hoch.

- Verschiedene Anwendungsprofile
  Es gibt zwei Möglichkeiten, einen ML-Compiler zu verwenden: in der traditionellen Weise zur Erzeugung von Stand-alone-Anwendungen, die vom Laufzeitsystem aber zur Laufzeit nicht vom Compiler abhängen, oder als interaktive Kommandosprache in Form einer symbolischen Metasprache.

Generell kann gesagt werden, daß ML auf funktionalen Sprachen aufbaut, speziell der λ-Kalkül wurde herangezogen. Der Code wird meist in einen Zwischencode übersetzt, der formal geprüft werden kann. Weitere Details können der Literatur entnommen werden: [ApMa], [MacQ], [Reppa], [Maun] und [Reppb].
Caml bietet darüber hinaus noch ein kompaktes, maschinenunabhängiges Format (Byte Code) und dynamisches Binden an, was sich speziell für aktive Netzwerke und die Versendung von Programmpaketen als vorteilhaft erweist. Auch beherrscht Caml das Einfügen von Signaturen in den Bytecode.

Beispiel eines gepufferten Kanals. Dieses Beispiel zeigt einen gepufferten Kanal für asynchrone Kommunikation. Die Funktion `buffer()` erzeugt einen neuen Kanal, der aus einem Pufferthread, einem Eingabekanal und einem Ausgabekanal besteht.
`bufferSend()` ist die asynchrone Sendeoperation, `bufferReceive()` die asynchrone Empfangsfunktion.

```
abstype 'a buffer_can = BC of
{
        inch :  'a chan,
        outch :  'a chan
}

with
    fun buffer() = let
        val inCh = channel() and outCh = channel()
        fun loop([], []) = loop([accept inCh], [])
            | loop(front as (x::r), rear) = sync
                  ( choose [
                      wrap (receive inCh, fn y => loop (front, y::rear)),
                      wrap (transmit(outCh, x), fn () => loop(r, rear))
                           ])
            | loop ([], rear) = loop (rev rear, [])
        in
            spawn(fn () => loop ([], []));
            BCinch = inCh, outch = outCh
        end
    fun bufferSend (BCinch, ..., x) = send(inch, x)
    fun bufferReceive (BCoutch, ...)  = receive outch
end (* abstype *)
```

Gepufferte Kanäle sind nun eine neue Kommunikationsmöglichkeit, ein Thread kann nun diese Möglichkeit in jedem Kontext nutzen, in dem auch `receive` zugelassen ist.
Dies soll nur ein Beispiel dafür sein, wie einfach die Sprache ML erweitert werden kann, speziell im Bereich Active Networks ließen sich noch weitere interessante Möglichkeiten finden. Gerade im Bereich der Nebenläufigkeit von Programmen liegt noch sehr viel Potential, auch im Zusammenhang mit ML. Weitere Beispiele sind in [ApMa] enthalten.
Tabelle 2 zeigt eine kleine Übersicht über weitere Programmiersprachen, die in Verbindung mit Active Networks ebenfalls verwendet werden, jedoch eher eine untergeordnete Rolle spielen.

Projekt	M	S	E	Beschreibung
Safe-Tcl (Source)	X	X		Safe-Tcl (basierend auf Tcl) ist eine Skriptsprache, die Sicherheit durch Interpretation eines Quellprogramms und Abschluß des Namensraums bietet. Safe-Tcl setzt auf der eingeschränkten Umgebung und Korrektheit der Interpreter auf, um Programme davon abzuhalten, absichtlich oder unbeabsichtigt, die Laufzeitumgebung zu verlassen.
Java (Zwischencode)	X	X	Ø	Java benutzt einen Zwischencode (Bytecode), um Mobilität und Portierbarkeit zu erreichen. Dieser Zwischencode enthält Informationen, die die Authentizitätsprüfung des Codes zur Laufzeit möglich machen. Java verlangt eine korrekte Interpretation des Codes, jedoch werden die Voraussetzungen für ein korrektes Programm bereits durch den Compiler zur Verfügung gestellt.
Omniware (Objektcode)	Ø	X	X	Portbaler Omniware-Objektcode basiert auf SFI (Software-based Fault Isolation), um Sicherheit effizient zu erreichen. Omniware beschreibt eine Menge von Regeln, die eine Abfolge von Befehlen einhalten muß (z.B. Regeln, nach denen Speicherarithmetik vollzogen wird). Diese Regeln beschreiben nur den Rahmen, innerhalb dessen ein Programm agieren kann, den es allerdings nicht verlassen kann.
Proof-Carrying Code (Objektcode)	-	X	X	PCC verwendet einen neuen Ansatz: ein formaler Beweis der Eigenschaften eines binären Programms wird diesem angefügt. Der Empfänger kann testen, ob der Beweis gültig ist, was einen sehr viel weniger aufwendigen Prozeß darstellt, als den Beweis von Grund auf neu zu führen. Jedoch ist PCC bislang nur für kleine Programme praktikabel.

Tabelle 2: Programmcodiertechniken; M = Mobilität, S = Sicherheit, E = Effizienz; 'X'=gut, 'Ø'=mittel, '-'=nicht ausreichend

4 Aktueller Stand

Zu den aktuellen Forschungen kann man im WWW erhebliche Mengen an Veröffentlichungen finden. Jedoch besteht das Problem, daß sich viele der angebotenen Texte nur in Kleinigkeiten unterscheiden, die Grundaussage ist meist immer die gleiche. Auch der Inhalt der Texte ist in vielen Fällen dürftig, so wird oft darauf verwiesen, daß sich aktuelle Projekte noch in Planung befinden, und somit noch keine Ergebnisse veröffentlicht werden konnten. Obwohl viele Einrichtungen (in den USA) sich offiziell mit dem Thema Active Networks beschäftigen, sind es in vielen Fällen immer die gleichen Personen, die Artikel zum Thema veröffentlichen, daher sind Unterschiede in den gefundenen Artikeln nie sonderlich groß.
Weiterhin wird in einigen Artikeln (hauptsächlich bei den Programmiersprachen) ausgesagt, daß bestimmte Techniken oder Programmiersprachen benutzt werden, jedoch wird oft nicht darauf eingegangen warum dies so gemacht wird und warum dies sinnvoll erscheint und andere Techniken oder Sprachansätze nicht. Als Einstiegspunkt zu empfehlen:
http://www.cis.upenn.edu/jms/SoftSwitch.html
http://www.cis.upenn.edu/eros
http://www.cis.upenn.edu/jms/SCAN.html
Auch die Suche mit einer Suchmaschine bringt etliche Verweise an den Tag (die sich jedoch nicht immer als sehr hilfreich erweisen ...).
Im folgenden ein kurzer Überblick, über die Entwicklungen in den großen Forschungseinrichtungen:

- Massachusetts Institute of Technology
  Am MIT wird eine prototypische gekapselte Architektur entwickelt, die auf dem *capsule approach* [TeWe] beruht. Weitere Themen der Arbeit sind Komponentenspezifikation, *active storage*, *multicast negative acknowledgement fusion* sowie Filter für Netzverkehr. Ein Prototyp als Referenzplattform, der die gekapselte Architektur demonstriert, wird unter Linux mit Java entwickelt. Zusätzliche Technologien (weiterentwickelte Betriebssystemtechniken, *compilation on-the-fly*, sind ebenfalls in Entwicklung. *Capsules*[10] benutzen die Konstrukte von Programmiersprachen, um das Paket-Processing zu vollführen. Die Sprache wird erweitert um die Spezifikation von Komponenten (*foundation components*), die Primitive bereitstellen, um mit der lokalen Knotenumgebung zu kommnunizieren. Diese können um Spezialfunktionen für bestimmte Applikationen erweitert werden. Laden und Cachen von Programmen oder Programmteilen sind Strategien, um Programme kompakt zu halten und um den Overhead bei Transfer und der Überprüfung des Programmes gering zu halten. Caching vermindert außerdem, daß bereits benutzte Programme oder Programmteile neu geladen und aus Sicherheitsgründen neu gepüft werden müssen.
  Die Programmierung wird dadurch erleichtert, daß *soft-states* in einem Knoten zurückgelassen werden können. Beim Traversieren von Knoten, können Programme solche soft-states zurücklassen und weitere Pakete, die zur gleichen Verbindung gehören, können diese auslesen, bzw. der Code kann auf diese Zustände aufbauen oder die Zustände verändern. Verbindungen und Datenflüsse können in aktiven Netzwerken durch diesen Mechanismus mit größerer Leistungsfähigkeit

[10] ähnlich einer Message

und mehr Funktionalität ausgestattet werden, als dies mit heutigen Netzwerken der Fall ist. Eine persistentere Form des aktiven Speicherns ist der *workflow state*, der zur Unterstützung von losen synchronisierten Aktivitäten und Wegabhängigkeiten eingeführt wird.

- University of Pennsylvania
  Das *SwitchWare*-Projekt beschäftigt sich mit der Entwicklung eines programmierbaren Switches, der es erlauben soll, digital signierte, typgeprüfte Module in Knoten zu laden und auszuführen. Grundidee ist, das Abstraktionsniveau der Funktionen des Switches an die einer Turingmaschine anzulehnen. Sicherheitsaspekte führen zu Beschränkungen in den *trade-offs*, die auch in die Unterstützung anderer Ziele einfließen: die Ressourcenbelegung muß robust gegen *denial-of- service-Attacken* sein, die Erweiterbarkeit muß beschränkt werden, was jedoch auch für umfangreichere Anwendungen ausreichend sein sollte.
  Der Ansatz benutzt formale Methoden zur Überprüfung der Sicherheitseigenschaften von SwitchWare-Programmen. Der Hauptaugenmerk bei SwitchWare liegt in der Identifikation von Eigenschaften des darunterliegenden Systems, die formal ausgedrückt werden können. Die Beweisführung wird durch die verwendete Sprache *SML/NJ* unterstützt, die eine präzise Definitions- und Laufzeitunterstützung bei der Typenprüfung, bei Garbage Collection und Ressourcenbelegung bietet. Ein Vorteil der sich aus der Sicherheitsunterstützung auf Sprachebene ergibt, ist, daß vermieden wird, daß ein großer Overhead für die Sicherung von Einstiegspunkten beim Traversieren von Knoten benötigt wird. Diese Sicherung wird bereits zur Compilierzeit exklusiv bereitgestellt.
  Dieser Ansatz wird mit einem Prototypen getestet, der auf einem *shared-memory*-Multiprozessorsystem implemenetiert wurde. Frühe Anwendungsprogrammprototypen beinhalten softwaregesteuerte Bandbreitenbereitstellung, basierend auf einem allgemeinen Meschanismus für *inverses Multiplexing* (network striping), sowie Unterstützung von aktiven Paketen (*Switchlets*).

- Bell Communications Research
  Mehrere Aspekte des Designs von SwitchWare werden bei Bellcore intensiver betrachtet, jedoch aufbauend auf einer anderen Infrastruktur (*OPCV2*[11]), um die Erfahrungen mit unterschiedlichen Konfigurationen zu vergrößern. OPCV2 lehnt sich an den ATM [12]-Switch von Sunshine an, der für die AURORA-Gigabit-Testumgebung entwickelt wurde. Dieser Switch kann auch als stand-alone-Switch betrieben werden, um die Leitungsgeschwindigkeit von ATM-Datenströmen zu manipulieren. Dies erlaubt im wesentlichen, die Multiplexing-Algorithmen und die Systemfunktionalität des Laufzeitsystems von SwitchWare zu testen, die in einen Portcontroller eines skalierbaren Switches eingebettet wurde.
  Ein weiterer Punkt in der Arbeit von Bellcore ist die Spezifikation der Semantik eines aktiven Routers, sowie die Einbringung dieser Spezifikation in eine Weiterentwicklung des Prototypen der University of Pennsylvania. Der neue Prototyp benutzt ein Multiprozessorsystem als aktives Netzwerkelement, das ATM-Netze mit 10 Mb/s und 100 Mb/s-Ethernet-Netzwerken verbindet. Dieser aktive Router soll als Experimentierplattform zur Erprobung von Applikationen, die im SwitchWare-Projekt entwickelt wurden, dienen.

[11] Output Port Controller Version 2
[12] Asynchronous Transfer Mode

Bellcore beschäftigt sich außerdem mit der Frage, wie die neue Technologie eingesetzt werden kann, beispielsweise in Hinblick auf *self-paying information-transport*, womit Zahlungsinformationen in elektronischer Form übertragen werden und mit den aktiven Elementen ausgewertet werden können. Das Interesse von Bellcore in aktive Netzwerke ist hauptsächlich in Zusammenhang mit früheren Forschungen zu sehen: beispielsweise den Arbeiten an *protocol-boosters*, die automatisch Protokollkomponenten optimieren, sowie der Entwicklung des AIN[13], das die Implementierung von Diensten vom Switching trennt, indem die Dienstkontrolle in adjunkte Prozessoren verschoben wird.

- Columbia University
  Das *NetScript*-Projekt besteht aus einer Programmiersprache und einer Laufzeitumgebung. Die Sprache bietet scriptsprachliche Mittel für die Verarbeitung zur Bearbeitung von Paketströmen. NetScript ist speziell zur Implementierung von Routingabläufen, zur Paketanalyse, Signalisierungen und Managementfunktionen gedacht. NetScript-Agenten können zu entfernten Systemen (einschließlich Zwischenknoten, Routern und Switches) gesendet werden. Ziel ist, diese Knoten so einfach programmieren zu können wie Endsysteme.

- Carnegie Mellon University
  Entwicklung von Mechanismen zum Ressourcenmanagement von *application-aware networks*. Angestrebt sind drei Bereiche von Ressourcen, die programmiert werden können: physikalische Infrastruktur (mit Ausführung und Speicherung), Entscheidungsfindung in verschiedenen Zeitskalen (Applikationsstart, Paket- und Zellenscheduling), firmenweites Sharing der Infrastruktur. Die Mechanismen sollen alle drei Bereiche unterstützen.
  Weiterhin wird im Bereich der Unterstützung von Anwendungen (Videokonferecing, Data Mining), die mehrere Datenströme mit unterschiedlichen Charakteristika benutzen, geforscht. Diese Anwendungen sollen "network-aware" werden, so daß sie einfach auf einer Vielzahl von Netzwerken aufbauen und schnell an veränderte Netzwerkbedingungen angepaßt werden können.

- BBN
  Forschung auf dem Gebiet der Programmierbarkeit von *data dictionaries* und Authentifizierungsmechanismen im Kontext von IP. Ziel ist, Management und Diagnosefähigkeiten zu erweitern.

- Georgia Institute of Technology
  Anwendung von Konzepten in aktiven Komponenten, die es erlauben, in Überlastsituationen bestimmte Algorithmen zu benutzen: z. B. Kompression, selektives Fallenlassen von Paketen, Transcoding.

- University of Kansas
  Projekt zum Einsatz von aktiven Netzwerken in *deployable radio networks*.

- University of Arizona
  Entwicklung von *liquid software*, einer Sammlung von Techniken, die schnelles Compilieren von Zwischencode erlauben (beispielsweise während der Bearbeitung in einem Router).

[13] Advanced Intelligent Network

- University of Cincinnatti
  Forschungen im Bereich der formalen Spezifikation von Netzwerkelementen und deren Verhalten.

5 Schluß

Wie in diesem Text bereits an einigen Stellen deutlich wurde, sind die Konzepte der Active Networks noch nicht besonders weit ausgereift, vor allem im Bereich der Betriebssystemunterstützung und der Programmiersprachen sind noch viele Hürden zu überwinden. Viele der Konzepte sind noch nicht sehr weit gediehen, wie auch in den Veröffentlichungen der Forschungsabteilungen zu erkennen ist. Die meisten Ansätze sind noch nicht über das Konzeptstadium hinaus, so daß die Veröffentlichungen dementsprechend knapp ausfallen.
Betrachtet man das Themengebiet "Active Networks" intensiver, so wird man leicht feststellen, daß diese Ansätze einen Bereich der Telematik darstellen, der in den nächsten Jahren wohl sehr viel an Bedeutung gewinnt. Das Verfolgen der Forschung in diesem Bereich stellt somit sicherlich keine verlorene Zeit dar.

Literatur

[ApMa] A.W. Appel und D.B. MacQueen. Standard ML of New Jersey. Technischer Bericht, Princeton University, AT&T Bell Laboratories.

[ASNS97] D.S. Alexander, M. Shaw, S.M. Nettles und J.M. Smith. Active Bridging. *Communications of the ACM*, 1997.

[BhCZ97] S. Bhattacharjee, K.L. Calvert und E.W. Zegura. Implementation of an Active Networking Architecture. Technischer Bericht, Networking and Telecommmunications Group, College of Computing, Georgia Tech, Atlanta, 1997.

[GuHK+97a] C. Gunter, M. Hicks, P. Kakkar, J. Moore, S. Nettles und J. Smith. PLAN: A Programming Language for Active Networks. Technischer Bericht, Department of Computer and Information Science, University of Pennsylvania, 1997.

[GuHK+97b] C. Gunter, M. Hicks, P. Kakkar, J. Moore, S. Nettles und J. Smith. PLAN: Language-Based Safety and Security for Active Networks. Technischer Bericht, Department of Computer and Information Science, University of Pennsylvania, 1997.

[HMPP96] J. Hartmann, U. Manber, L. Peterson und T. Proebsting. Liquid Software: A New Paradigm for Networked Systems. Technischer Bericht, Department of Computer Science, University of Arizona, 1996.

[LeMB+96] I.M. Leslie, D. McAuley, R. Black, T. Roscoe, P. Barham, D. Evers, R. Fairbairns und E. Hyden. The Design and Implementation of an Operating System to Support Distributed Multimedia Applications. *IEEE Journal on Selected Areas in Communications* 14(7), September 1996, S. 1280–1297.

[LeWG98] U. Legedza, D.J. Wetherall und J. Guttag. Improving the Performance of Distributed Applications Using Active Networks. *IEEE INFOCOMM*, April 1998.

[MacQ] D.B. MacQueen. Reflections on Standard ML. Technischer Bericht, AT&T Bell Laboratories.

[Maun] M. Mauny. *Functional Programming using Caml Light.*

[MoHi97] J.T. Moore und M. Hicks. A Service Layer Routing Protocol for PLAN. Technischer Bericht, Department of Computer and Information Science, University of Pennsylvania, October 1997.

[MoMO+95] A.B. Montz, D. Mosberger, S.W. O'Malley, L.L. Peterson, T.A. Proebsting und J.H. Hartman. Scout: A Communications-Oriented Operating System. Proceedings of the Fifth HotOS Workshop, May 1995.

[Mosb97] D. Mosberger. *Scout: A Path-based Operating System.* Dissertation, Department of Computer Science, University of Arizona, 1997.

[pla97] *PLAN 2.0 Programmer's Guide*, October 1997.

[Reppa] J.H. Reppy. Concurrent ML: Design, Application and Semantics. Technischer Bericht, Department of Computer Science, Cornell University.

[Reppb] J.H. Reppy. A High-Performance Garbage Collector for Standard ML. Technischer Bericht, AT&T Bell Laboratories.

[Shap97] J.S. Shapiro. *EROS: A High-Performance Capability System*. Dissertation, Distributed Systems Laborytory, University of Pennsylvania, Philadelphia, 1997.

[ShFS96a] J.S Shapiro, D.J. Farber und J.M. Smith. The Measured Performance of a Fast Local IPC. Proceedings of the 5th International Workshop on Object Orientation in Operating Systems. IEEE, November 1996.

[ShFS96b] J.S. Shapiro, D.J. Farber und J.M. Smith. State Caching in the EROS Kernel - Implementing Efficient Orthogonal Persistence in a Pure Capability System. Proceedings of the 7th International Workshop on Persistent Object Systems, November 1996.

[ShWe97] J.S. Shapiro und S. Weber. Verifying Operating System Security. Technischer Bericht, Department of Computer and Information Science, University of Pennsylvania, 1997.

[SmFa96] J.M. Smith und D.J. Farbert. SwitchWare: Accelerating Network Evolution. *White Paper*, 1996, S. 17.

[SmFG+96] J.M. Smith, D.J. Farber, C.A. Gunter, S. Kannan, I. Lee und S.M. Nettles. Self-Checking Active Networks (SCAN). Technischer Bericht, Distributed Systems Laborytory, University of Pennsylvania, Philadelphia, 1996.

[SMSF97] J.S. Shapiro, S.J. Muir, J.M. Smith und D.J. Farber. Operating System Support for Active Networks. Technischer Bericht, Distributed Systems Laborytory, University of Pennsylvania, Philadelphia, 1997.

[TeWe] D.L. Tennenhouse und D.J. Wetherall. Towards an Active Network Architecture. Technischer Bericht, Telemedia, Networks and Systems Group, MIT.

[TGSK96] D.L. Tennenhouse, S.J. Garland, L. Shrira und M. F. Kaashoek. From Internet to ActiveNet, 1996.

Internet2: Die neue Generation des Internet

Nerantzoula Vassiliadis

Kurzfassung

In dieser Seminararbeit wird das Internet2 vorgestellt. Dabei handelt es sich um ein Projekt zur praktischen Erprobung neuer Technologien und neuer Kommunikationsprotokolle im Internet. Die zentrale Komponente des Internet2 ist der *Gigapop*, der die einzelnen Teilnetze des Internet2 zu einem Gesamtnetzwerk verbindet und dadurch einen flächendeckenden Datenaustausch ermöglicht. Die Verwaltung des Internet2 erfolgt dezentral durch die Betreiber der einzelnen Teilnetze. Mit dem Internet2 soll eine Kommunikationsinfrastruktur bereitgestellt werden, welche den gestiegenen Anforderungen insbesondere der wissenschaftlichen Anwender entspricht.

1 Motivation

Das stetige Anwachsen der Teilnehmerzahlen hat ebenso wie die zunehmende Kommerzialisierung des Internet zu einer Vervielfachung des Datenaufkommens innerhalb der letzten Jahre geführt. Infolgedessen beobachten die Benutzer des heutigen Internet immer wieder Überlastsituationen und Datenstaus, die eine schnelle und zuverlässige Kommunikation nicht mehr zulassen. Gerade im wissenschaftlichen Bereich besteht jedoch der Bedarf an einem Kommunikationsdienst für den schnellen Austausch großer Datenmengen. Aus dieser Situation heraus wurde die Initiative zur Realisierung des Internet2 gestartet.

Das Internet2 hat zum Ziel, neue Technologien zu erproben und schrittweise einzuführen. Es erfüllt also folgende Mission: Bereitstellung einer Entwicklungs- und Testplattform für neue Technologien und Protokolle, wie im Bild 1 veranschaulicht wird. Das Projekt wurde von verschiedenen Universitäten initiiert und wird vom Staat und interessierten Unternehmen unterstützt. Die am Projekt beteiligten Universitäten sollen zum einen das notwendige Know-How aus den Bereichen Technik und Wissenschaft einbringen. Zum anderen dienen sie als Anwender, um die neu geschaffene Kommunikationsinfrastruktur praktisch zu erproben und mögliche Schwachstellen zu identifizieren.

Neu am Kommunikationsdienst des Internet2 ist die Bereitstellung qualitätsorientierter Übertragungsdienste, die dem Benutzer eine zuvor vereinbarte Dienstqualität garantieren. Darüber hinaus integriert das Internet2 zahlreiche Sicherheitsmechanismen zum Schutz der Privatsphäre bei der Datenübertragung.

Die Kosten der Datenübertragung über das Internet2 hängen von der Qualität und dem Leistungsumfang des genutzten Übertragungsdienstes ab. Dafür sind entsprechende Tarifierungsmodelle zu entwerfen.

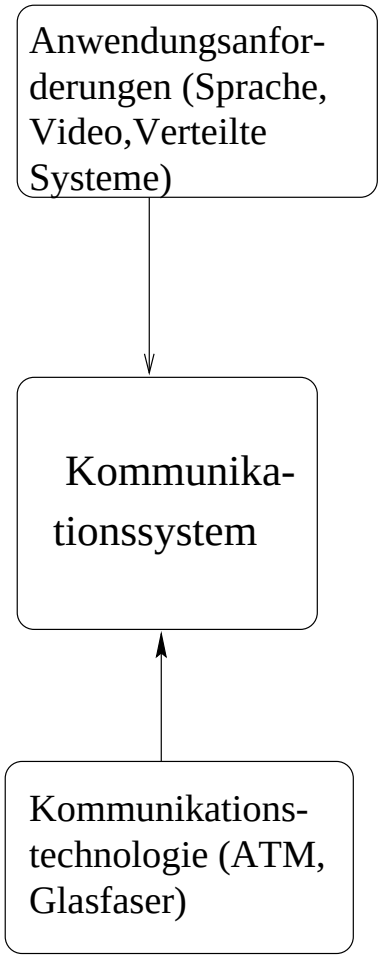

Abbildung 1: Einflußfaktoren

Das Internet2 wird das heutige Internet nicht kurzfristig ersetzen. Vielmehr wird das Internet2 als experimentelles Netzwerk zur Entwicklung und Erprobung neuer Technologien und Protokolle angesehen. So steht das Netzwerk auch nicht jedem offen. Nur ausgezeichnete Projektmitglieder erhalten Zugang zum Internet2.

Die wichtigsten Unterschiede des Internet2 zum heutigen Internet sind die folgenden:

1. *Schnellere Netze*: Durch den Einsatz der ATM-Technologie im Backbone des Internet2 wird die Übertragungskapazität stark erhöht.

2. *Fortschrittliche Kommunikationsprotokolle*: Neue Kommunikationsprotokolle (z.B. IPv6, RSVP) erfüllen die geänderten Anforderungen, die sich aus neuen Anwendungsfeldern (z.B. Videokonferenzen) ergeben.

3. *Unterstützung virtueller LANs*: Ein virtuelles LAN besteht aus mehreren lokalen Netzen, die über Weitverkehrsverbindungen miteinander gekoppelt sind. Für den Anwender ist die Kopplung über das Weitverkehrsnetz transparent. Im Bild 2 wird dieses Prinzip illustriert.

Das Internet2 (I2) wird zunächst keine Dienste wie WWW oder E-Mail anbieten. Der Zugang wird nicht durch öffentliche Internet Service Provider (ISP) bereitgestellt.

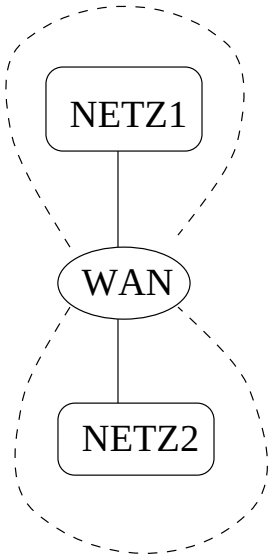

Abbildung 2: Virtuelles LAN

Nur offizielle Mitglieder der I2-Initiative (d.h. nicht profitorientierte Firmen oder Forschungseinrichtungen) können einen Zugang erhalten. Zukünftig sollen jedoch die Zugangsmöglichkeiten ausgedehnt und die Nutzung des Internet2 für die Allgemeinheit ermöglicht werden.

2 Das Internet2

2.1 Anforderungen an das Internet2

Fortschrittliche Anwendungen, wie beispielsweise Audio- und Videokonferenzen, stellen neue und höhere Anforderungen an die vom Internet2 bereitgestellten Kommunikationsdienste. Die wichtigtsten Anforderungen sind:

1. Geringe Verzögerung: Daten müssen schneller transferiert werden.
2. Geringere Verzögerungsschwankungen: Der sogenannte *Jitter* ist zu minimieren. Der Jitter ist der zeitliche Unterschied zwischen aufeinanderfolgenden Nachrichten, die mit konstantem Abstand gesendet wurden. Dies wird im Bild 3 veranschaulicht.

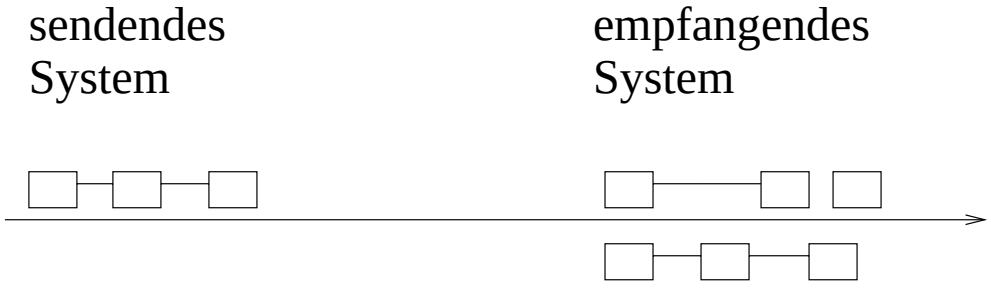

Abbildung 3: Verzögerungsschwankung

3. Realzeitanforderung: Daten müssen rechtzeitig bei den Empfängern ausgeliefert werden, damit ein Auseinanderlaufen der Datenströme vermieden wird.

4. Höhere Übertragungsraten.

5. Geringere Fehlerraten.

Dies waren die wichtigsten Leistungsmerkmale, die die Teilnetzwerke aufweisen sollen, damit sie überhaupt an das Internet2 angeschlossen werden. Nachdem diese Anforderungen von allen beteiligten Teilnetzen erfüllt sind, kann das prototypische I2-System entsprechend der vorgesehenen Netzwerkarchitektur in Betrieb genommen werden. Die Anordnung der Netzwerke und ihre Verbindung wurden so gewählt, daß die geforderten Leistungs- und Funktionsmerkmale erbracht werden können.

2.2 Architektur des Internet2

Die zentrale Komponente der Internet2-Architektur ist der *Gigapop*. Der Gigapop ist ein Vermittlungssystem zur Verbindung der I2-Teilnehmer bzw. der daran angeschlossenen Teilnetze. Beim Aufbau des Internet2 werden prinzipiell zwei Arten von Gigapops unterschieden. Zum einen kann ein Gigapop ausschließlich die direkt angeschlossenen Netzwerke miteinander verbinden (Abbildung 4) oder er koppelt die angeschlossenen Teilnehmer über ein Backbone-Netzwerk (Abbildung 5). Die Technik und der interne Aufbau des Gigapop ermöglichen die Bereitstellung von Kommunikationsdiensten mit garantierter Bandbreite und Qualität.

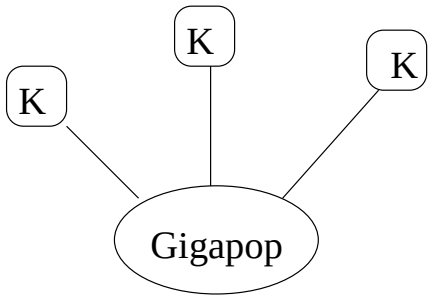

Abbildung 4: Verbundene Netzwerke

Im Internet2 wird eine Reihe fortschrittlicher Kommunikationsprotokolle eingesetzt. Im einzelnen sind dies:

- *IPv4*: Netzwerkprotokoll für unzuverlässige Datenübertragung.
- *IGMP - Internet group membership protocol*: Protokoll zur Verwaltung von Multicast-Gruppen. Das Protokoll wird zukünftig im ICMP (Internet Control Message Protocol) integriert.
- *RSVP - Ressource Reservation Protocol*: Mit Hilfe dieses Protokolls können Reservierungen von Netzwerkressourcen vorgenommen werden, so daß die für einen vorgemerkten Datentransfer benötigten Ressourcen von anderen Teilnehmern nicht verbraucht werden. Dieses Protokoll wird zusammen mit einer neuen Version des IP eingesetzt, bekannt als IPv6: Die hauptsächlichen Unterschiede des IPv6 von IPv4 sind folgende:

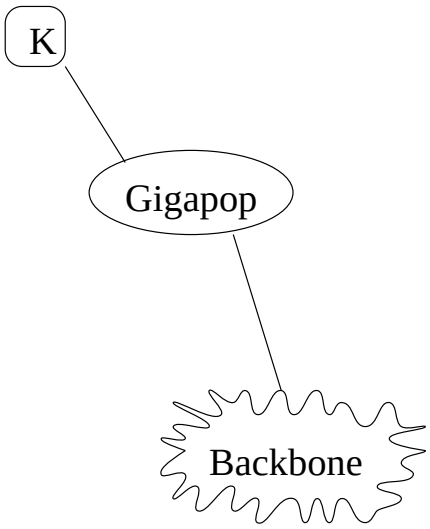

Abbildung 5: Nicht verbundene Netzwerke

1. In der Adressierung:
   - Statt 32-bit Adressen werden 128-bit Adressen benutzt.
   - Verwendung verschiedener Adreßtypen. Insgesamt kennt IPv6 die folgenden Adreßtypen:
     (a) Unicast-Adressen: sie identifizieren nur ein einziges Interface eines Zwischen- bzw. Endsystems. Das bedeutet, daß das Paket genau von einem Interface empfangen werden kann. Diese Adressen lassen sich in mehrere Typen unterscheiden, zum Beispiel: providerbasierte, geographisch basierte,...
     (b) Anycast-Adressen: sie adressieren ein Interface aus einer Gruppe von Interfaces.
     (c) Multicast-Adressen: damit kann eine vordefinierte Gruppe von Interfaces adressiert werden.

- In der Paketform: Die zu transferierenden Datenpakete werden nicht segmentiert.
- In der Sicherheit: Die Sicherheit ist einer der Hauptgründe für die Einführung von IPv6. Daten können durch das Attackieren (jemand möchte dem Netzwerk schaden), oder durch nicht autorisiertes (illegaler Zugang) und unkorrektes Benutzen (z.B Diebstahl) gefährdet werden. Dagegen werden in IPv6 Mechanismen zur Vertraulichkeit der Daten eingesetzt (z.B für Zulassungskontrolle und Fehlerbehebung zusätzlich zur Netzwerkkontrolle, die die Quelle eines Fehlers bestimmt). Der Zugang wird in manchen Fällen entsprechend abgelehnt. Grundsätzlich umfassen die IPv6-Sicherheitsfunktionen Erweiterungen zur Unterstützung von Authentifizierungen und vertraulicher Kommunikation (Verschlüsselung der Pakete).
  - Die Authentifizierung stellt sicher, daß die Daten wirklich vom angegebenen Sender gesendet wurden, und daß sie unverfälscht beim Empfänger angekommen sind.
  - Die Verschlüsselung der Pakete stellt sicher, daß nur der beabsichtigte Empfänger die Daten in eine lesbare Form umwandeln kann. Für die Verschlüsselung werden zwei verschiedene Modi definiert:

1. Tunnel-Modus: Das gesamte Paket wird verschlüsselt und ein neuer unverschlüsselter IP-Header wird vorangestellt.
2. Transport-Modus: Nur die Nutzlast und die Ende-zu-Ende-Erweiterungs-Header werden verschlüsselt. Der eigentliche IP-Header bleibt unverschlüsselt.

Solche Verschlüsselungen können auf folgenden Ebenen erfolgen:

1. Ende-zu-Ende (Transport-Modus)
2. Zwischen Endsystemen und Security-Gateways (Tunnel-Modus)
3. Zwischen zwei Security-Gateways (Tunnel-Modus)

- In der Unterstützung von Dienstgüten:
  - Angabe einer Priorität: Es gibt ein Prioritätsfeld, das einer Datenquelle erlaubt, die gewünschte Priorität der Pakete zu beschreiben.
  - Spezielle Kennzeichnung einzelner Datenströme durch Flußmarken: Flußmarken gehören zu den neuen interessanten Merkmalen dieses neuen Protokolls, und damit werden Pakete gekennzeichnet, die eine besondere Behandlung durch IPv6-Router benötigen.
- In der Unterstützung mobiler Endsysteme: Mobile IP-Knoten werden durch eine transiente Adresse, die sogenannte care-of-Adresse, und eine permanent gültige Adresse, die Home-Adresse, identifiziert.
- In der automatischen Systemkonfiguration: ICMP wird allgemein zum Kontrollnachrichtenaustausch zwischen IP-Instanzen verwendet. Während in ICMPv4 im wesentlichen Fehler- und Echonachrichten ausgetauscht wurden, wird ICMPv6 auch für die Gruppenverwaltung eingesetzt, die bislang vom IGMP ausgeführt wurde.

Durch ATM-Schalt-Elemente in den Gigapops wird eine vereinfachte, verbindungsorientierte Paketvermittlung realisiert, die sich für einheitliche Übertragung von Daten, Bildern, und Sprache im lokalen, und Weitverkehrsraum besonders eignet.

2.3 Darstellung des prototypischen Systems

Im vorigen Kapitel wurden die Charakteristika des neuen Internet und des Gigapops beschrieben, und jetzt wird im Bild 6 das ganze System verdeutlicht.

Bemerkung: K steht für Kampus, d.h für Universität (Mehrere Netzwerke können zusammen einen Kampus bilden); L steht für Laboratorium;

Daran kann man **zwei verschiedene Typen von Gigapops** erkennen:

1. Verbindung ausschließlich mit Internet2-Mitglieder: Es ist keine Protokollumsetzung durch Gigapop erforderlich.
2. Verbindung sowohl mit Mitgliedern des Internet2, als auch mit Teilnehmern des heutigen Internet: Der Aufbau und die Funktionsweise dieses Gigapops ist komplizierter, da eine Protokollumsetzung erforderlich ist.

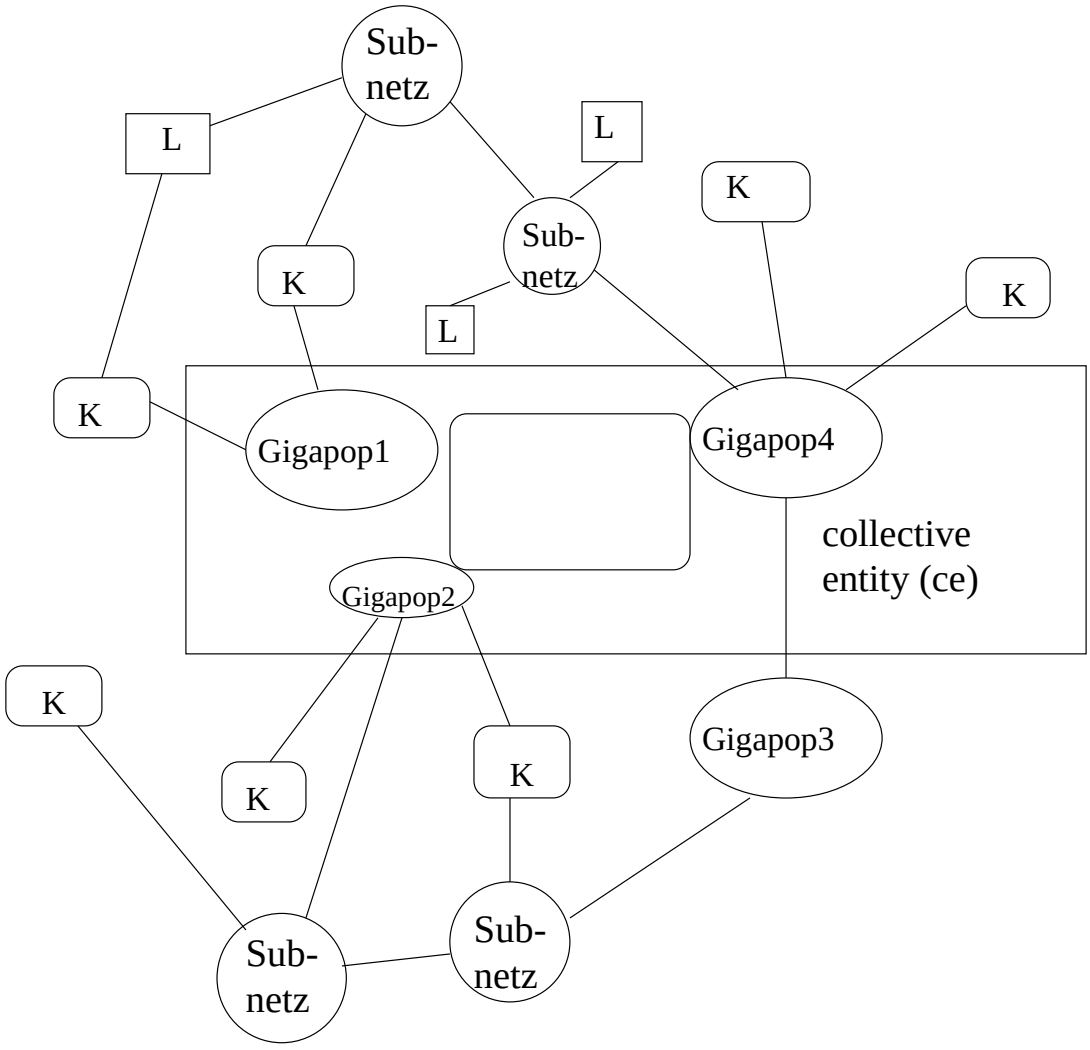

Abbildung 6: Gesamte Architektur

Es gibt zwei Arten von Verbindungen der Gigapops:

1. Gigapops, die ausschließlich mit lokalen Netzwerken verbunden sind (Mitglieder oder auch keine); jedes Netzwerk hat bis zu 622 Mbps Kapazität; bis zu 10 Netzwerke können daran angeschlossen werden.

2. Gigapops, die mit anderen Gigapops verbunden sind:

   - durch eine direkte Verbindung. Im Bild 6 wird so der Gigapop3 mit dem Gigapop4 verbunden.
   - durch ein komplexes Backbone-Netz. Im Bild 6 sind Gigapop1, 2, und 4, die miteinander auf diese Art agieren. Der Betrieb des Backbone-Netzes und der daran angeschlossenen Gigapops wird von einer organisatorischen Einheit, der sogenannten *Collective Entity (CE)* koordiniert. Diese Einheit präsentiert keine zentrale Organisation, sondern stellt vielmehr ein Gremium zur Koordinierung der Kommunikation zwischen den Gigapops dar, wie das Bild 7 zeigt. Diese Verbindung ist die interessantere der beiden Verbindungsarten, weil erst dadurch eine flächendeckende Kommunikation über das Internet2 ermöglicht wird. Dafür wird der *NSFvBNS (National Science*

Foundation's very high speed Backbone Network Service) verwendet. Gigapops werden miteinander durch dieses Netzwerk verbunden, das von der Regierung unterstützt wird.

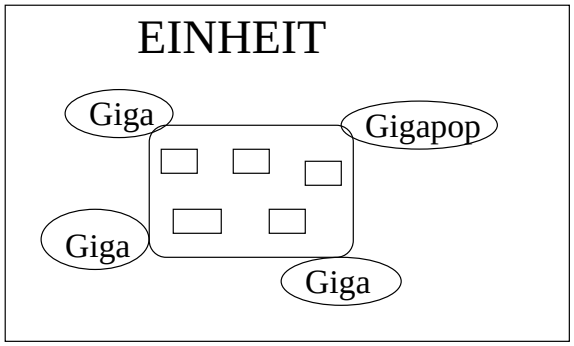

Abbildung 7: Verbindung der Gigapops, welche durch die CE koordiniert wird

Bemerkung: Die in der Einheit gezeichneten Kästchen repräsentieren Netzwerke, die für den Internet2-Datenfluß zuständig sind.

2.4 Aktivitäten der Gigapop-Betreiber

Die Betreiber von Gigapops müssen folgende Aufgaben erledigen:

- Organisation und Betrieb des Gigapops.
- Identifizierung und Sicherung eines Platzes für die Installation des Gigapops.
- Entwicklung eines Gigapop-Schemas, in dem sich der Gigapop mit der Einheit und den anderen Gigapops in Verbindung setzt.
- Installierung und Testen der Gigapop-Ausrüstung.
- Verbindung und Kontrolle der Anschlüße der Internet2-Mitglieder.
- Verbindung und Kontrolle der Anschlüße der sich in der Einheit befindenden Netzwerke.
- Herstellung des Kontaktes zwischen Kampus und Einheit.

2.5 Aktivitäten der Collective Entity (CE)

Folgende Aufgaben fallen in den Zuständigkeitsbereich der Collective Entity (CE):

- Organisation und Betrieb des Backbone-Netzes und Koordination der Gigapop-Betreiber.
- Vereinbarung und Überprüfung der Daten, die für die Netzwerke der Einheit zur Verfügung stehen.
- Verbindung der Netzwerke innerhalb der Einheit, wo hauptsächlich der Internet2-Verkehr fließt.

2.6 Aktivitäten der Kampus-Mitglieder

Die Universitäten sollen ihrerseits Arbeit leisten, damit sie über einen Gigapop Zugang zum Internet2 erhalten:

- Sie sollen mit anderen Interessenten kooperieren, um gemeinsam einen Gigapop in Betrieb zu nehmen.
- Sie sollen den etablierten Gigapop warten und verwalten.
- Sie sollen weiteren Benutzern Zugang zum Internet2 ermöglichen.

2.7 Erforderliche Arbeit für das ganze Projekt

Insgesamt sind für das ganze Projekt folgende Leistungen zu erbringen:

1. Spezialisten sollen detaillierte Implementierungen durchführen, in dem alle vorher besprochenen Prinzipien realisiert werden.
2. Industriepartner sollen teilnehmen, um notwendige Wissensquellen und benötigtes Material sicherzustellen.
3. Realisierung spezifischer Implementierungen und Ziele in der Art, daß sie entsprechend der erbrachten Leistung bewertet werden können.

3 Ausblick

Ein noch offener Aspekt ist die weitere Entwicklung von TCP. Es wird zur Zeit diskutiert, ob die durch IPv6 erforderlichen Modifikationen auf das Nötigste beschränkt werden sollen, oder ob eine völlig neue Protokollarchitektur entwickelt werden soll.

Literatur

[Team97a] Internet 2 Project Team (Hrsg.). *Preliminary Engineering Report*, Januar 1997.

[Team97b] Internet 2 Project Team (Hrsg.). *Project Description*, 1997.

[Thom96] S.A. Thomas. *IPng and the TCP/IP Protocols*. Wiley Computer Publishing. 1996.

[Zitt95] Martina Zitterbart. *Hochleistungskommunikation, Band 1: Technologie und Netze*. R.Oldenbourg Verlag. 1995.

Verteiltes Management im Internet — die "DISMAN Working Group" der IETF —

Matthias Franz

Kurzfassung

Durch immer größer werdende Netze wird ein zentrales Netzwerkmanagement immer problematischer. Aus diesem Grund wurde die DISMAN Working Group ins Leben gerufen. Sie soll die notwendigen Rahmenwerke erstellen, um verteiltes Management von Netzen zu erlauben. So entstanden erste Definitionen eines Netzwerkmanagement-Rahmenwerks und eines Rahmenwerks für verteiltes Management. Es wurde eine Script-MIB definiert, welche die Delegierung von Managementfunktionen an untergeordnete Managementsysteme, mit Hilfe von Scripts, ermöglicht. Um der Script-MIB von ihr benötigte Funktionalitäten zu bieten, wurden weitere MIBs definiert. Eine Event-MIB, mit der festgelegt werden kann, daß bestimmte Aktionen durchgeführt werden, wenn das zugehörige Ereignis ausgelöst wird, eine Expression-MIB für die Benutzung von Ausdrücken, eine Management-Target-MIB, um die Ziele für Managementoperationen zu bestimmen, und eine Notification-MIB, die Benachrichtigungen ermöglicht. Es bleiben aber noch Anforderungen, wie Sicherheit, unerfüllt.

1 Einleitung

Die heutigen Netze nehmen in Größe und Komplexität immer mehr zu. Dadurch wird jedoch ihre Verwaltung immer problematischer. Mit zentralem Management, wie es heute eingesetzt wird, ist diese Aufgabe fast nicht mehr zu bewältigen.

Durch die Zentralität wird ein Flaschenhals erzeugt, der sowohl zu Leistungsverlusten führt, als auch die Störanfälligkeit des Netzes vergrößert. Da jedes Gerät direkt von der Zentrale verwaltet wird, entscheidet ihre Belastung über die Situation im Netz. Fällt die Zentrale gar aus, ist das gesamte Netz ohne Aufsicht. Hinzu kommt noch die Belastung des Netzes durch die Kommunikation zwischen Gerät und Zentrale. Da beide meist ständig in Verbindung stehen müssen, werden im Netz viele Ressourcen verbraucht. Außerdem entstehen große Kosten, gerade über weite Strecken oder bei Wählverbindungen.

Als Lösung bot sich, wie auch schon in vielen anderen Bereichen, eine Dezentralisierung an. Verteiltes Management kann einige der genannten Probleme beseitigen.

Durch einen hierarchischen Aufbau werden die einzelnen Manager nicht mehr so stark belastet. Fällt einer von ihnen aus, wirkt sich das nur auf Teilnetze aus. Das restliche

Netz kann seine Arbeit unbehelligt fortsetzen. Da immer nur eine Verbindung zur nächst höheren Hierarchie-Ebene nötig ist, wird der Kommunikationsaufwand erheblich reduziert. Desweiteren können die einzelnen Manager viele Aufgaben eigenständig erledigen, so daß die Verbindungen nicht mehr ständig zur Verfügung stehen müssen.

Um zu untersuchen, wie dieses verteilte Management realisiert werden kann, wurde die DISMAN Working Group von der IETF ins Leben gerufen. Ein Überblick über deren Arbeit findet man in [IET97].

2 Zielsetzung und Rahmenwerk

Die oben genannten Probleme und Lösungsmöglichkeiten spiegeln sich auch in der Zielsetzung des DISMAN-Arbeitsgruppe wider:

- *Skalierbarkeit durch Aufgabenverteilung*
  Um auch große Netzwerke verwalten zu können, muß das Managementsystem Aufgaben an untergeordnete Manager verteilen.
- *Betrieb auch ohne ständige Verbindung oder bei Verbindung mit geringer Bandbreite*
  Die untergeordneten Managementstationen sollen ihre Aufgaben auch erfüllen können, wenn eine Verbindung zur übergeordneten Station nicht dauernd besteht oder unterbrochen wird.
- *Widerspiegelung der Unternehmensgrenzen und Abläufe*
  Unternehmen sind normalerweise hierarchisch strukturiert. Die verschiedenen Hierarchiestufen haben unterschiedliche Aufgaben, Verantwortungen und Rechte. Netzwerkmanagementsysteme können häufig an diese Hierarchien angelehnt werden.
- *Unterstützung moderner Systemarchitekturen*
  Durch die definierten Schnittstellen des verteilten Managements wird ein modularer Aufbau des Systems möglich, der eine große Flexibilität und eine hohe Wiederverwendung der einzelnen Komponenten ermöglicht.

Als Protokoll zwischen den Managementkomponenten wird das Simple Network Management Protocol (SNMP) verwendet, da es eine große Verbreitung aufweist.

Um diese Ziele zu erreichen, wurde zunächst die Rahmenwerke definiert [BGSW96].

2.1 Das Netzwerkmanagement-Rahmenwerk

Ein Netzwerkmanagement-System besteht aus mehreren Komponenten:

- mehrere Agenten
- mindestens eine Managementstation

- ein Managementprotokoll

Agenten sind Verarbeitungseinheiten (Prozesse), die Zugriff auf Magementwerkzeuge (*Management Instrumentation*) haben, d.h. die Managementinformationen liefern oder auf Auftrag modifizieren können.

Managed Elements sind Geräte wie z.B. Hosts, Router usw., die über einen Agenten verfügen (lokal oder entfernt) und daher durch Zugriff auf ihre *Managementinformationen* überwacht und gesteuert werden. Managementinformationen bestehen aus einer Auswahl von *Managed Objects* einer *Management Information Base* (MIB).

Managementstationen führen Managementanwendungen aus, die andere Komponenten überwachen und steuern. Diese Komponenten sind entweder *Managed Elements* oder *verteilte Managementstationen*.

Verteilte Managementstationen sind Managementstationen, die Anfragen von anderen Managern erhalten und diese bearbeiteten, indem sie Managementoperationen auf Agenten oder anderen Managern durchführen. Diese Anfragen können eine lange Zeit beanspruchen.

Verteilte Managementstationen werden oft auch als *mid-level-Manager* bezeichnet. Die einem mid-level-Manager übergeordneten Managementstationen heißen dann *higher-level-Manager*.

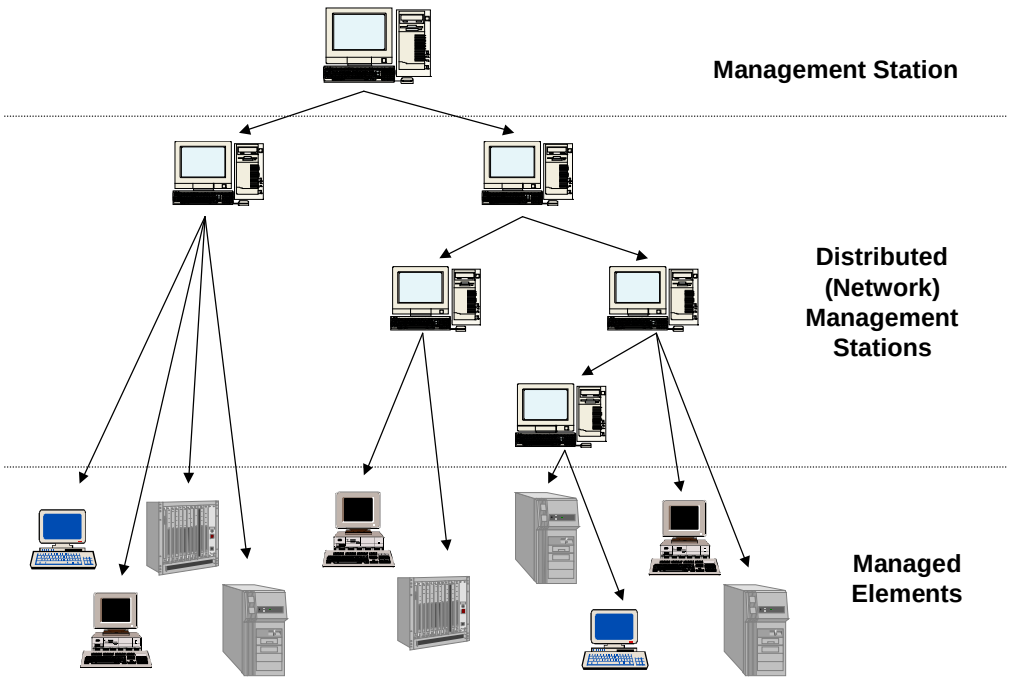

Abbildung 1: Beispiel für den Aufbau eines Managementsystems

2.2 Das verteilte Management-Rahmenwerk

Das verteilte Management-Rahmenwerk besteht aus Anwendungen und Dienstleistungen, die von den Anwendungen benötigt werden. Einige dieser Dienste sind grundlegend und werden daher näher betrachtet.

2.2.1 Verteilte Managementanwendungen

Eine verteilte Managementanwendung führt bestimmte Managementfunktionen durch. Dazu gehört meist das Überwachen und Steuern von Managed Elements. Dabei werden Funktionen wie Schwellwertabfragen (threshold polling), Abfragen historischer Daten (historical data polling) oder Suchen nach neuen Komponenten (discovery) ausgeführt.

Einige dieser Anwendungen werden in Standards spezifiziert werden, andere können anwendungsspezifisch sein. Dasselbe gilt für die benutzten Kommunikationsprotokolle.

2.2.2 Verteilte Managementdienste

Die verteilten Managementdienste sind eine Sammlung von Diensten, welche die Laufzeitumgebung der verteilten Managementanwendungen bilden. Sie sind ausschlaggebend für die Benutzbarkeit, Skalierbarkeit und Effizienz der Anwendungen. Einige der Dienste sind obligatorisch, andere optional. Aber sogar die obligatorischen Dienste erlauben verschiedene Ebenen, auf denen sie unterstützt werden können. Der Zugriff auf die Dienste erfolgt über MIB-Schnittstellen. Diese werden später beschrieben. Durch die Dienste wird echtes Unternehmensmanagement möglich, das es erlaubt, verteilte Managementfunktionen, ohne unnötigen Verwaltungsaufwand, unternehmensweit zu nutzen. Desweiteren werden die Managementanwendungen effizienter, da ein Dienst für alle lokalen Anwendungen Funktionen nur einmal ausführen oder Informationen nur einmal speichern muß. So wird auch das Schreiben von Anwendungen erleichtert, da viel Funktionalität schon in den Diensten steckt.

2.2.3 Grundlegende Dienste

- Der *Known Systems* Dienst:

  Dieser Dienst soll es möglich machen, Managementanwendungen auf einer Liste von Systemen auszuführen. Die einzelnen Systeme können von der Anwendung getrennt konfiguriert werden, was den Verwaltungsaufwand senkt, wenn sich die Liste ändert.

  Er stellt eine Liste aller Systeme, die das verteilte Managementsystem kennt und auf denen es Operationen ausführen kann, zur Verfügung. Für das Zusammenstellen der Liste gibt es mehrere Möglichkeiten: ein ständig laufender Prozeß zur automatischen Erkennung, durch Eintragung mit Hilfe von SNMP-SET-Requests, oder die Liste wird statisch vorgegeben. Zu jedem System werden auch Typ und Attributinformationen gespeichert. Auf dieser Dienst wird durch die `mgmtKnownSystemTable` in der *DISMAN MIB* zugegriffen.

- Der *Management Domains* Dienst:

  Dieser Dienst soll es ermöglichen, Managementanwendungen auf einer Liste aller Systeme eines bestimmten Bereichs durchzuführen. So bleiben die Anwendungen auf dem neuesten Stand, auch wenn sich die Bereiche ändern.

  Dazu stellt der *Management Domains* Dienst eine Liste zusammen, die eine Teilmenge der Liste des *Known Systems* Dienst ist. Sie wird durch Regeln oder Filter erzeugt, die diejenigen Systeme auswählen, die bestimmten Anforderungen entsprechen bzw. nicht entsprechen. Es kann mehrere Bereiche geben, die sich auch überlappen. Sie sollen Unternehmensgrenzen widerspiegeln, auf denen die Managementoperationen ausgeführt werden sollen. Die Regeln und Filter können mit der `mgmtRuleTable` der *DISMAN MIB* erzeugt werden.

- Der *Management Operation Targets* Dienst:

  Durch diesen Dienst sollen Managementoperationen auf die Systeme beschränkt werden, auf denen sie sinnvoll sind. Er ermöglicht es, Operationen auszuführen, ohne auf ein sich dynamisch veränderndes Netz Rücksicht nehmen zu müssen.

  Um seine Aufgabe zu erfüllen, stellt der *Management Operation Targets* Dienst ähnlich dem *Management Domains* Dienst eine durch Filter definierte Liste der Systeme zusammen, auf denen bestimmte Managementfunktionen ausgeführt werden können. Die zu diesem Dienst gehörende MIB wird in Abschnitt 4.3 beschrieben.

- Der *Delegation Control* Dienst:

  Oft verursachen Netzwerkmanagementanwendungen zusätzliche Belastungen für das Netz und dadurch Probleme. Dieser Dienst stellt Funktionen zur Verfügung, die den Ressourcenverbrauch einer verteilten Managementanwendung begrenzen. Für das sendende System als ganzes, für einzelne Endknoten oder für jede Managementanwendung einzeln können die Durchschnittsrate und Burst-Rate für Polling, für Benachrichtigungen und Broadcasts eingestellt werden. Außerdem gibt es noch ein "Notaus", damit eine delegierte Anwendung nicht unendlich weiter läuft, wenn sie von ihrem Auftraggeber vergessen wurde. Die zu diesem Dienst gehörende MIB wird in Abschnitt 3.1 beschrieben.

- Der *Scheduling* Dienst :

  Dieser Dienst erlaubt es, eine Zeit für die Ausführung von verteilten Managementanwendungen zu bestimmen. Es ist möglich, einen bestimmten Zeitpunkt zu wählen, die Ausführung periodisch zu wiederholen oder die Ausführung beim Auftreten eines Ereignisses zu beginnen. Auf Teile dieses Dienstes kann über die Script-MIB (siehe Abschnitt 3.1) zugegriffen werden, z.B. das Auslösen von Scripten durch Ereignisse. Die eigentlich zu diesem Dienst gehörende MIB wurde jedoch gerade erst mit der Schedule-MIB definiert [LeSc97a].

- Der *Reliable Notification* Dienst:

  Dieser Dienst sorgt für dafür, daß Benachrichtigungen garantiert korrekt beim Empfänger ankommen. Falls die Informationen, die ein Ereignis beschreiben, nicht in eine einzige PDU passen, wird ein eventID varbind definiert, d.h. es wird eine Zuordnung zwischen einer Variable und einem Wert definiert. Dadurch werden mehrere PDUs als zusammengehörig kennzeichnet. So kann der Empfänger

feststellen, ob die gesamte Benachrichtigung schon angekommen ist, oder ob noch PDUs fehlen. Dem Empfänger werden aber noch weitere Informationen geliefert, so daß er Duplikate erkennen kann. Ein Logging-Mechanismus gewährleistet ein Wiederholen der Benachrichtigung bei einem Kommunikationsfehler. Die zu diesem Dienst gehörende MIB wird in Abschnitt 4.4 beschrieben.

- Der *Notification Destinations* Dienst:

  Delegierte Managementfunktionen und *mid-level-Manager*, die Benachrichtigungen erzeugen dürfen, müssen wissen, wohin sie diese senden sollen. Außerdem müssen die Anzahl der Wiederholungen im Fehlerfall festgelegt werden, genauso wie die Durchschnitts- und Burst-Rate. Der *Notification Destinations* Dienst stellt dafür Funktionen zur Verfügung. Die zu diesem Dienst gehörende MIB findet sich ebenfalls in Abschnitt 4.4.

3 Management Scripts

Um Managementfunktionen an *mid-level-Manager* zu delegieren, benötigt man die Möglichkeit, dort Anweisungsfolgen ausführen zu lassen. Für diese Aufgabe wurde eine Standardschnittstelle definiert, die Script-MIB [LeSc97b]. Sie wird dazu benutzt, Managementfunktionen, als Management-Scripts geschrieben in einer Management-Script-Sprache, auf dem *mid-level-Manager* auszuführen. Dabei stellt die MIB keine Anforderungen an die Sprache und erlaubt sogar die Benutzung compilierten Codes. Um die Delegierung von auf dem Internet-Managementrahmenwerk basierenden Managementfunktionen zu ermöglichen, muß die Script-MIB folgendes leisten können:

- die Übertragung von Management-Scripts zu einem *mid-level-Manager*;
- eine Möglichkeit, Management-Scripts aufzurufen, zu unterbrechen, fortzuführen und zu beenden;
- die Übertragung von Argumenten für die Management-Scripts;
- die Überwachung und Steuerung laufender Management-Scripts;
- die Übertragung der Ergebnisse der laufenden Management Scripts.

3.1 Struktur der Script-MIB

Die Objekte der Script-MIB werden in folgende Gruppen eingeteilt:

3.1.1 Die Sprach-Gruppe (smLanguage)

Die Sprach-Gruppe liefert Informationen über die Sprachen und die Spracherweiterungen, die vom *mid-level-Manager* unterstützt werden. Dies ist notwendig, da die MIB mit verschiedenen Sprachen und Versionen zusammenarbeiten soll. Daher können im Voraus keine Annahmen über die verwendete Sprache gemacht werden. Der *higher-level-Manager* benötigt aber Informationen, damit er entweder die für ein Script

benötigte Sprache und Version oder ein zur Sprache und Version passendes Script auswählen kann.

Die Sprachgruppe enthält zwei Tabellen und ein Verzeichnis:

1. Die `smLanguageTable` ist eine Liste aller Sprachen, die von dem *mid-level-Manager*, auf dem sich die MIB befindet, unterstützt werden.

2. Die `smExtensionsTable` listet alle zu einer gegebenen Script-Sprache verfügbaren Erweiterungen auf. Spracherweiterungen werden meist benötigt, um allgemeine Programmiersprachen um Managementfunktionalitäten zu erweitern. Diese Spracherweiterungen sind häufig Funktionsbibliotheken, die solche Fähigkeiten, wie das Versenden von SNMP-Anfragen, ermöglichen.

3. In der `smLanguageDir` werden alle bekannten Script-Sprachen aufgeführt.

3.1.2 Die Script-Gruppe (smScript)

Die Script-Gruppe dient zum Anfordern und Installieren der Management-Scripts, sei es von einem *higher-level-Manager* oder lokal.

Die Script-Gruppe besteht aus zwei Tabellen:

1. Die `smScriptTable` definiert Objekte, die die folgenden Operationen unterstützen:

   - Download von Scripts mit SNMP
     Diese Möglichkeit, ein Management-Script zum *mid-level-Manager* zu übertragen, geschieht nach dem *push*-Modell, d.h. der *higher-level-Manager* schiebt das Script zum *mid-level-Manager*. Der braucht sich nicht darum zu kümmern, wie das Script zu ihm kommt. Die Übertragung erfolgt mittels SNMP-SET-Request, so daß das Script in mehrere Teile aufgeteilt werden muß, um die maximale Paketgröße für SNMP-Pakete nicht zu überschreiten (siehe Abbildung 2).
   - Download von Scripts unter Angabe eines Uniform Ressource Locators
     Hier wird das *pull*-Modell verwendet, d.h. der *mid-level-Manager* besorgt sich das Script selbst. Dabei setzt der *higher-level-Manager* mit einem SNMP-SET-Request eine URL, also den Ort, an dem das Script abgeholt werden kann. Der *mid-level-Manager* muß nun mit Hilfe eines Agenten das Script von dem besagten Ort abholen. Dabei verwendet der Agent das in der URL angegebene Protokoll, z.B. FTP oder HTTP (siehe Abbildung 3).
   - Speichern von Scripts in lokalem, nicht flüchtigem Speicher
     Management-Scripts können lokal fest gespeichert werden, um nach einem Reset des *mid-level-Managers* den sofortigen Restart des Script zu ermöglichen, ohne daß es erst wieder vom *higher-level-Manager* geholt werden muß.
   - Lesen von Scripts aus lokalem nicht flüchtigem Speicher
   - Löschen von Scripts aus lokalem nicht flüchtigem Speicher

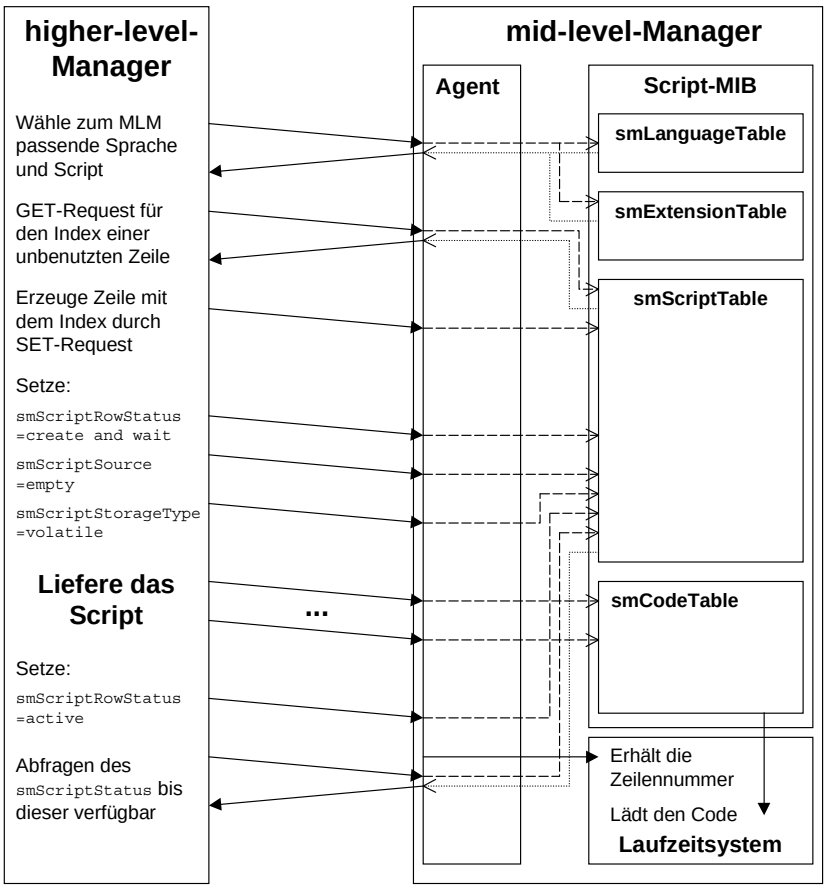

Abbildung 2: Laden eines Scrips mit dem *push*-Modell

- Auflisten permanenter Scripts
  Einige Scripts können unveränderlich vorgegeben werden.
- Einstellen, ob Script lesbar oder schreibbar
  Dies ermöglicht die Zugriffskontrolle auf Scripts über SNMP.

Ein Statusobjekt (`smScriptStatus`) ermöglicht einem Manager, den gegenwärtigen Status eines Scripts zu lesen. Es wird auch dazu benutzt, Fehlerinformationen für einen *higher-level-Manager* zur Verfügung zu stellen, wenn eine der oben genannten Operationen nicht durchgeführt werden kann.

Ein Quellobjekt (`smScriptSource`) gibt an, von wo das Script geladen werden kann. Ist in ihm eine URL angegeben, wird es von dort geholt, ansonsten lokal.

Von wo es lokal geladen werden soll, gibt das `smScriptStorageType`-Objekt an. Entweder wird das Script von einem Festspeicher gelesen oder aus der Codetabelle.

Ist das `smScriptStatus`-Objekt aktiv gesetzt, werden die Scripts vom *mid-level-Manager* automatisch ins Laufzeitsystem geladen. Dies beinhaltet das Holen des Scripts, eventuell von einer entfernten Station, das Übersetzen des Scripts, falls die verwendete Sprache das erfordert, und das Bereitstellen des Codes für das Laufzeitsystem.

Der Status dieses Ladevorgangs kann mit dem `smScriptStatus Objekt` abgefragt werden.

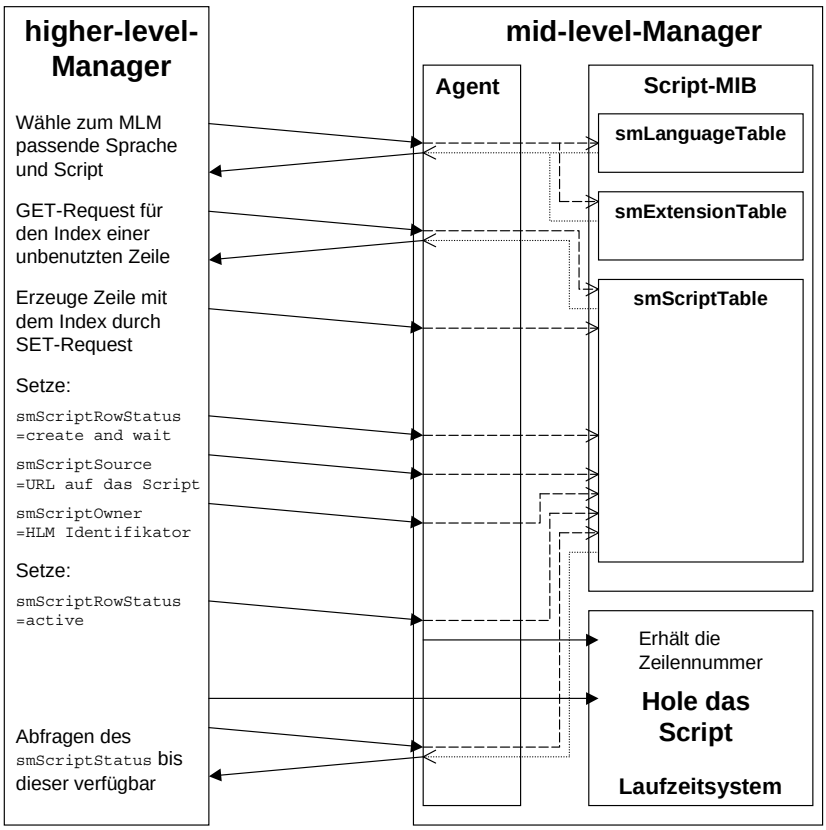

Abbildung 3: Laden eines Scrips mit dem *pull*-Modell

2. Die Codetabelle (`smCodeTable`) wird benutzt, um auf den Quellcode eines Scripts zuzugreifen und um ihn zu ändern. Sowohl Zugriff als auch Änderungen erfolgen mit SNMP. Dabei kann ein Script über mehrere Zeilen der Tabelle aufgeteilt sein, um den SNMP-Paketgrößenanforderungen zu genügen.

   Scripts können nur geändert werden, wenn das `smScriptAccess` Objekt schreiben zuläßt und der `smScriptRowStatus` anzeigt, daß das Script im Moment nicht benutzt wird. Das modifizierte Script wird dann wieder geladen, wenn der `smScriptRowStatus` auf aktiv gesetzt wird.

3.1.3 Die Script-Ausführungs-Gruppe (smLaunch)

Die Script-Ausführungs-Gruppe enthält zwei Tabellen:

1. Die `smLaunchTable` beschreibt vorkonfigurierte Scripts, die auf ein Ereignis warten, das sie auslöst. Ein Eintrag in diese Tabelle bindet ein Argument, eine Kontrollvariable und die Besitzer der von hier gestarteten Scripts an ein Script. Die Kontrollvariable gibt an, wie oft das Script gleichzeitig laufen darf.

2. In der `smRunTable` werden alle gerade laufenden oder vor kurzem beendeten Prozesse aufgelistet. Jede Zeile enthält Argument, Ergebnis, exit-Code, Statusinformationen, sowie Start- und Ende-Zeitstempel. Außerdem enthält die Tabelle drei schreibbare Objekte:

- Das `smRunLifeTime` Objekt gibt an, wie lange ein Script höchstens laufen kann, bevor es beendet wird.
- Das `smRunExpireTime` Objekt bestimmt die maximale Zeit, bevor ein Script, nachdem es seinen Durchlauf beendet hat, aus der `smRunTable` automatisch gelöscht wird.
- Mit Hilfe des `smRunAdminStatus` Objekt kann ein Script angehalten, fortgesetzt oder abgebrochen werden.

4 Weitere Aktivitäten der "DISMAN Working Grup"

4.1 Event-MIB

Die MIB unterstützt Ereignisauslöser und damit verbundene Aktionen. Mit ihr können sowohl Ereignisauslöser wie auch Aktionen definiert werden. Desweiteren kann man Verbindungen zwischen ihnen erstellen. Es können Informationen über Ereignisauslöser abgefragt werden, die Art und Wert der Auslösung bestimmt werden, ebenso die daraufhin versendeten Benachrichtigungen [Stew97a].

4.2 Expression-MIB

Diese MIB wird benutzt, um Ausdrücke aus MIB-Objekten zu definieren. Dabei wurde die MIB unter der Vorgabe entworfen, daß ein ausgewerteter Ausdruck wieder ein MIB-Objekt sein soll, und somit wie jedes andere Objekt benutzt werden kann. Die Operationen dieser MIB setzen voraus, daß OID-Fragmente benutzt werden können. Das sind Object Identifier, bei denen Teile, z.B. der übliche Präfix iso, fehlen [Stew97b].

4.3 Management-Target-MIB

Diese MIB erlaubt es, Zielsysteme für das Management anzugeben. Sie liefert eine Infrastruktur für andere MIBs in Form von Zielsystemen oder Gruppen von Zielsystemen, die Bestimmungsort für SNMP-Anfragen und -Benachrichtigungen sein können. Folgende Anforderungen und Ziele werden berücksichtigt:

- Alle für eine SNMP-Nachricht nötigen Angaben sollen im Selektor enthalten sein.
- Auswahlen sollen zu Bereichen einer logischen Einheit gehören, aber mit einer feineren Granularität in bezug auf die Zugriffsmöglichkeiten.
- Auswahlen sollen sowohl auf einzelnen Bereichen, als auch auf Gruppen von Bereichen möglich sein.
- Die Definition von Gruppen muß manuell und automatisch möglich sein.
- Dies alles soll sowohl für einen ziemlich leistungsstarken Manager als auch für einen *mid-level-Manager* oder ein beschränktes "self-managing-System" gelten.

Aus diesen Vorgaben wurde eine MIB definiert [Stew97c], die aus neun Tabellen und zwei Objekten besteht:

Zunächst gibt es Tabellen mit Einträgen über Systeme, die als Bestimmungsort von Managementoperationen für das lokale System von Bedeutung sind.

Mit Hilfe der *System Creation Table* (`mtaSystemCreationTable`) können neue Systeme hinzugefügt oder bestehende gelöscht werden. Neue Systeme werden hinzugefügt, indem sie einen Namen erhalten (`mtaSystemCreationName`) und ihr Status (`mtaSystemCreationStatus`) auf aktiv gesetzt wird. Gelöscht werden sie ebenfalls durch Setzen des Status. Jeder Eintrag hat zudem einen Index (`mtaSystemIndex`), mit dem er in der *System Table* erscheint, so lange er besteht.

In dieser *System Table* (`mtaSystemTable`) stehen nun Informationen über die zuvor eingetragenen Zielsysteme. Auf allen lokalen Systemen ist ein Objekt (`mtaSystemName`) definiert, das es erlaubt, den Namen eines Systems zu ändern, ohne daß sich der Index verändert.

Die folgenden Objekte werden nur auf Systemen benötigt, die andere Systeme verwalten oder hohe Anforderungen an die automatische Gruppendefinition stellen. Hier werden Informationen über den Algorithmus, mit dem dieser Eintrag erzeugt wurde, abgelegt (`mtaSystemAlgorithm`), sowie das Datum und der Zeitpunkt, wann er erstellt wurde (`mtaCreateTime`). Desweiteren können Informationen über Datum und Zeitpunkt des letzten erfolgreichen Managementkontakts mit dem System (`mtaSystemLastContact`) vorhanden sein. Weitere Informationen über Kontakte (`mtaSystemSysContact`), den Ort des Systems (`mtaSystemSysLocation`), dessen ID (`mtaSystemSysObjectID`) und Domain Name (`mtaSystemDomainName`) des Domain Name Services sind möglich.

Damit Managementanwendungen bemerken, wann sich Einträge von Systemen ändern, wann neue hinzukommen oder alte herausfallen, können sie das Objekt `mtaSystemLastChange` überwachen, das den Zeitpunkt der letzten Änderung bei den Systemeinträgen enthält. Bei Bedarf können die Managementanwendungen dann die Systemliste neu laden.

Für die Verwaltung von Adreßinformationen über SNMP-Manager oder SNMP-Agenten auf den verschiedenen Zielsystemen wird die *Adressing Table* (`mtaAddressingTable`) benutzt. Nur wenn ein Eintrag für das System in der `System Creation Table` aktiviert wird, erscheint auch hier ein Eintrag. Wird das System dort gelöscht, verschwindet er auch aus dieser Tabelle. Die Werte der Objekte eines Eintrags werden entweder manuell ausgefüllt oder von Anwendungen erzeugt.

Um mehrere Einträge für das gleiche System zu ermöglichen, gibt es einen Index (`mtaAddressIndex`), der dann die Unterscheidung der einzelnen Einträge erlaubt. Der Typ der Adresse (`mtaAddressType`) und die eigentliche Adresse (`mtaAddressAddr`) können ebenso angegeben werden wie die Portnummer (`mtaAddressPort`), die mit der Transportadresse verwendet wird.

Parameter für Übertragungswiederholungen können auch eingestellt werden. Sei es der Algorithmus (`mtaAddressRetransAlgorithm`), die minimale Zeit zwischen zwei Übertragungsversuchen (`mtaAddressRetransTimer`) oder die maximale Anzahl von Übertragungswiederholungen.

Um die Einträge aktivieren und löschen zu können, gibt es ein Statusobjekt (`mtaAddressStatus`). Wie bereits erwähnt, ist das Erstellen von Einträgen nur für aktivierte Systeme möglich.

Für Informationen über Ziele für Managementoperationen gibt es mehrere Tabellen:

Die *Target Creation Table* (`mtaTargetCreationTable`) ist, wie die *System Creation Table* bei Systemen, für die Erstellung von neuen Einträgen über Managementziele zuständig. Sie enthält auch die entsprechenden Objekte, wie den Namen (`mtaTargetCreationName`) für das System, einen Index (`mtaTargetIndex`), der zugeteilt wird, und einen Status (`mtaTargetCreationStatus`) für das Aktivieren und Löschen.

Für einen neuen Eintrag wird auch hier der Name angegeben und der Status auf aktiv gesetzt. Der Index wird dann in die *Target Table* übernommen. Wird ein Eintrag gelöscht, werden auch alle damit verbundenen Einträge in der *Target Table*, der *Scope Table*, der *Scope Community Table*, der *Group Rule Table* und der *Group Member Table* gelöscht.

Die *Target Table* enthält Einträge für die einzelnen Ziele. Jeder Eintrag für ein Ziel enthält dessen Namen (`mtaTargetName`), der auch geändert werden kann. Der Typ des Ziels (`mtaTargetType`) kann als individueller Bereich, als Gruppe oder als Gruppe von Bereichen und Gruppen festgelegt werden. Mit dem Zugriffsobjekt (`mtaTargetAccess`), kann die Zugriffsart für den Bereich des Ziels bestimmt werden: nur lesen, lesen und schreiben oder Benachrichtigungen sind möglich. Jedoch müssen alle Bereiche einer Gruppe den selben Zugriff haben. Sie müssen auch die selbe Versionsnummer der Zugriffskontrolle (`mtaTargetVersion`) haben.

Am Objekt `mtaTargetLastChange`, das den Zeitpunkt der letzten Änderung der Ziele enthält, können Anwendungen Veränderungen erkennen.

Die *Scope Table* (`mtaScopeTable`) enthält Einträge für die Ziele, deren Typ in der *Target Table* auf Bereich eingestellt ist. Ein Eintrag enthält dann die Indizes (`mtaScopeSystemIndex`) aller Systeme, die diesen Bereich unterstützen.

Die *Scope Community Table* (`mtaScopeCommunityTable`) enthält für jeden Bereich eine Zeichenkette (`mtaScopeCommunity`), die zur Authentisierung notwendig ist.

Zur Definition von Regeln für die Mitgliedschaft in einer Gruppe dient die *Group Rule Table* (`mtaGroupRuleTable`). Sie enthält Einträge für die Ziele, deren Typ auf Gruppe eingestellt ist. Jeder Eintrag enthält eine Regel (`mtaGroupRuleCondition`) als Integer-Objekt, die die Mitgliedschaft über die Indizes der Systeme definiert, und einen Status (`mtaGroupRuleStatus`) zum Aktivieren und Löschen von Einträgen.

In der *Group Member Table* (`mtaGroupMemberTable`) gibt es für jede Gruppe einen Eintrag. Dieser enthält ein Objekt (`mtaGroupMemberManual`), das angibt, ob die Gruppe manuell erstellt wurde oder mit Hilfe von Regeln, und ein Statusobjekt (`mtaGroupMemberStatus`) zum Aktivieren und Löschen von Einträgen.

4.4 Notification-MIB

Diese MIB erlaubt die Kontrolle und das Mitschreiben von SNMP-Benachrichtigungen wie Traps und Informs. Sie liefert die Infrastruktur für andere MIBs, indem sie die

Erzeugung von Benachrichtigungen steuert und angibt, wohin sie verschickt werden sollen. Außerdem verfügt sie über Zähler und eine lokale Mitschreibfunktion. So bietet sie einen für alle gemeinsamen Mechanismus, um den Verlust von Traps zu verhindern. Es wird Versendern von Benachrichtigungen extrem nahe gelegt, diese MIB zu benutzen [Stew97d].

5 Ausblick

Der neue Ansatz, die Managementfunktionen verteilt durchzuführen, befindet sich noch in seinem Anfangsstadium. Die bisherigen Definitionen greifen daher hauptsächlich auf schon im Netzmanagement eingesetzte Protokolle wie SNMP zurück. Dies ist auch sinnvoll, um einen möglichst problemlosen Übergang vom zentralen hin zum verteilten Netzwerkmanagement zu gewährleisten. In Zukunft sollen jedoch auch andere Verteiltechniken, wie CORBA oder HTTP hinzukommen. Anfänge sind, z.B. beim *pull*-Modell der Script-MIB, bereits zu erkennen.

Da man sich auf neues Gebiet wagte, hat man zunächst auch jegliche Sicherheitsaspekte außer acht gelassen. Für einen praktischen Einsatz in großen Netzen dürfte aber entscheidend sein, daß nur autorisierte Personen oder Rechner Managementfunktionen delegieren können, was letztendlich nichts anderes ist, als auf entfernten Rechnern Programme (Management-Scripts) ablaufen zu lassen, die großen Einfuß auf das System und das Netz als ganzes haben. Deshalb wird es unausweichlich sein, die vorhandenen Ansätze im Hinblick auf die Sicherheit zu erweitern.

Welche Bedeutung dem verteilten Management zugewiesen wird, wird durch das Tempo, in dem an den Entwürfen gearbeitet wird, deutlich. Regelmäßig erscheinen neue verbesserte Ausgaben der Entwürfe. Inkompatibilitäten der einzelnen MIBs werden verringert, neue Funktionalität wird hinzugefügt.

Literatur

[BGSW96] A. Bierman, M. Greene, B. Stewart und S. Waldbusser. Distributed Management Framework. Internet Draft, Internet Engineering Task Force (IETF), DISMAN Charter, 27. Dezember 1996.

[IET97] Distributed Management (DISMAN) Charter. Veröffentlicht im World Wide Web unter der Adresse http://www.ietf.org/html.charters/disman-charter.html, 1997.

[LeSc97a] D.B. Levi und J. Schoenwaelder. Definitions of Managed Objects for Scheduling Management Operations. Internet Draft, Internet Engineering Task Force (IETF), DISMAN Charter, 30. Dezember 1997.

[LeSc97b] D.B. Levi und J. Schoenwaelder. Definitions of Managed Objects for the Delegation of Management Scripts. Internet Draft, Internet Engineering Task Force (IETF), DISMAN Charter, 21. November 1997.

[Stew97a] B. Stewart. Event MIB. Internet Draft, Internet Engineering Task Force (IETF), DISMAN Charter, 26. März 1997.

[Stew97b] B. Stewart. Expression MIB. Internet Draft, Internet Engineering Task Force (IETF), DISMAN Charter, 3. Juni 1997.

[Stew97c] B. Stewart. Management Target MIB. Internet Draft, Internet Engineering Task Force (IETF), DISMAN Charter, 26. März 1997.

[Stew97d] B. Stewart. Notification MIB. Internet Draft, Internet Engineering Task Force (IETF), DISMAN Charter, 26. März 1997.

Management im Internet: die 'SNMPv3 Working Group' der IETF

Jörg Förster

Kurzfassung

In den folgenden Kapiteln wird die Arbeitsweise vom 'Simple Network Management Protocol Version 3' (SNMPv3) beschrieben. Es werden zunächst die grundlegenden Begriffe erläutert und ein kurzer Einblick in SNMPv1 und dessen Nachteile gegeben. Danach werden Änderungen, die in SNMPv2 gegenüber SNMPv1 gemacht wurden, aufgezeigt. Anschließend wird das Rahmenwerk von SNMPv3 und einige spezifische Mechanismen von SNMPv3 vorgestellt. Aufbauend auf diesen werden noch einige Applikationen angesprochen, um abschließend eine Bewertung von SNMPv3 durchzuführen.

1 Einleitung

In heutiger Zeit ist es nötig, daß man schnell und effizient den Betrieb eines Netzwerkes überwachen und steuern kann. Dies ist mit dem 'Simple Network Management Protocol' (SNMP) möglich. Um die Vorgehensweise zu erklären, ist es nötig, einige grundlegende Begriffe einzuführen.

1.1 Das Management im Internet: Grundlegende Begriffe

Das Rahmenwerk von SNMP besteht aus Agenten, Managern und einem Netzwerkmanagementprotokoll. Ein Agent überwacht die einzelnen Systemressourcen und verwaltet die anfallenden Daten in einer Art Datenbank. Diese Datenbank wird 'Management Information Base' (MIB) genannt. In der MIB sind alle wichtigen Einträge enthalten und bilden eine Art Baum (vgl. Abb. 1). Einen solchen Eintrag bezeichnet man als 'Managed Object' (MO). In der MIB sind die einzelnen MOs mittels einer ASN.1-Beschreibung definiert. Diese sieht dann zum Beispiel folgendermaßen aus:

```
sysDescr OBJECT-TYPE
              SYNTAX  DisplayString (SIZE (0..255))
              ACCESS  read-only
              STATUS  mandatory
              DESCRIPTION
                      "A textual description of the entity.  This value
                      should include the full name and version
```

```
                identification of the system's hardware type,
                software operating-system, and networking
                software.  It is mandatory that this only contain
                printable ASCII characters."
        ::= { system 1 }
```

In diesem Falle heißt das MO `sysDescr` und beschreibt den Hardwaretyp, die verwendete Systemsoftware und die Netzwerksoftware (s. `DESCRIPTION`). Das Objekt ist vom Typ `DisplayString`, was nichts anderes als ein OctetString ist. Das Objekt hat eine Größe von 0 bis 255 Zeichen. Es wird zudem vorgeschrieben, daß das Objekt nur ausgelesen werden kann (s. `ACCESS`). Zudem wird durch den Status festgelegt, ob dieses Objekt auf jeden Fall in einer Implementierung enthalten sein muß oder nicht. Dies ist hier der Fall (`STATUS mandatory`). Als letztes wird dann noch der Baum angegeben, in dem das Objekt ist, und an welcher Stelle des Baumes, sozusagen welches Blatt, das Objekt ist. Ein gemeinsames Netzmanagementprotokoll, das sowohl der Agent als auch der Manager verwendet, dient zur Übertragung der Daten zwischen dem Agenten und dem Manager. Der Manager hat nun die Möglichkeit, Anfragen an den Agenten zu stellen, die dieser dann bearbeitet und dem Manager das Ergebnis zurückliefert. Zudem kann der Manager bei entsprechenden Objekten Werte setzen. Dies geschieht, indem er dem Agenten mitteilt, daß er dem entsprechenden Objekt den gewünschten Wert zuweisen möchte. Der Agent gibt dann gegebenenfalls den Wunsch an die entsprechende Ressource weiter. Diese bearbeitet den Befehl und liefert dann entweder einen Fehler, sofern etwas nicht korrekt ist, oder den gesetzten Wert als Ergebnis zurück.

1.2 SNMP Version 1 (SNMPv1)

SNMPv1 wird zum Transport von Managementinformationen genutzt. Das Protokoll ermöglicht einem Netzwerkadministrator die interaktive Abfrage von Parametern oder die Überwachung von bestimmten Netzwerkzuständen. Auf der Basis von SNMP werden alle für eine Netzwerkmanagementapplikation relevanten Daten zwischen dem Manager, Agenten und den Netzwerkkomponenten übermittelt.

1.3 Nachteile von SNMPv1

Um besser auf die Nachteile von SNMPv1 eingehen zu können, ist es nötig, den groben Ablauf einer Abfrage bzw. eines Kommandos aufzuzeigen.

Eine SNMP-Abfrage kann alle MOs eines Agenten abfragen. Dies sieht dann allgemein folgendermaßen aus:

```
snmpget hostname community Objektname
```

Ausgehend davon kann man sich einfach vorstellen, wie ein entsprechendes Kommando zum Setzen einer Variablen aussieht:

```
snmpset hostname community Objektname Wert
```

[1]

[1] Eventulle Optionen wie 'timeouts' bzw. Angabe der Versionsnummer etc. wurden absichtlich aus Übersichtlichkeitsgründen weggelassen.

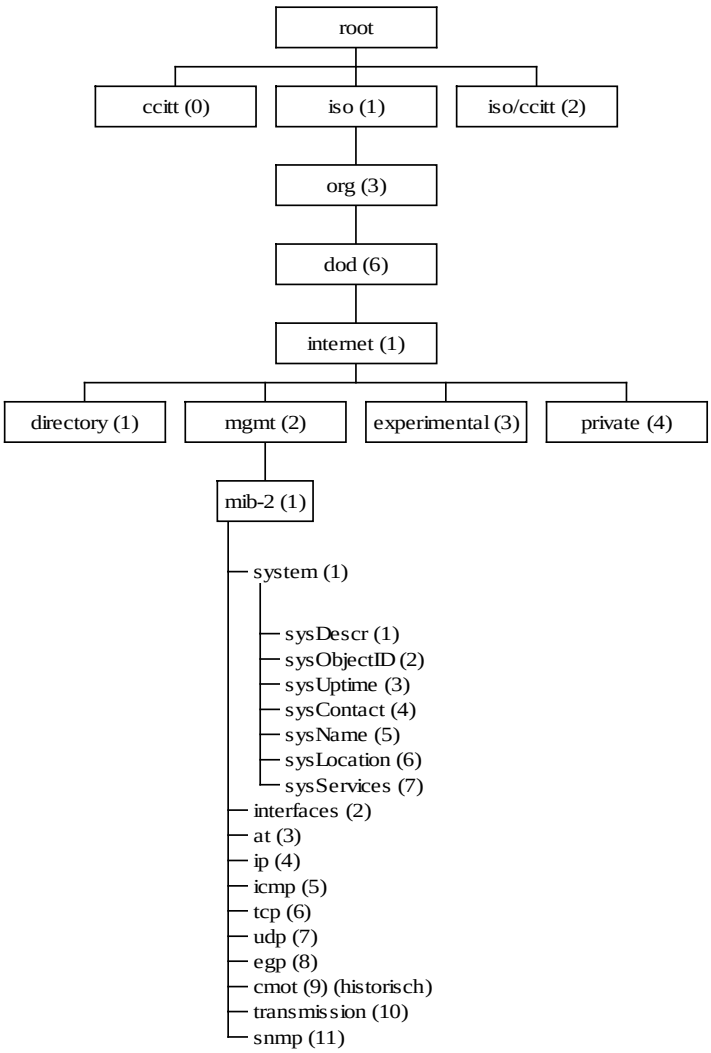

Abbildung 1: schematische Darstellung des MIB II Baumes

Wie man leicht erkennt, ist der einzige Sicherungsmechanismus, der benutzt wird, die 'community'. Im allgemeinen gibt es nur zwei 'communities', die eingesetzt werden. Zum einen ist das **public**, sprich, das Datum ist für alle zugänglich, zum anderen **secret**, das heißt, daß nur die Kommandos bearbeitet werden, die 'community' **secret** angegeben haben. Daraus läßt sich schließen, daß im Endeffekt kein wirklicher Sicherungsmechanismus in SNMPv1 vorhanden ist (vgl. [HeGr93], S. 305: Vorspiegelung einer Community).

Im Falle einer Abfrage ist diese Sicherheitslücke nicht ganz so relevant, da man durch reines Abfragen keinen Schaden anrichten kann. Im Falle des `snmpset` ist dies allerdings nicht mehr ganz so.

Der Systemverwalter hat keine Chance, wichtige Daten so zu schützen, daß man sie zwar auslesen, aber nicht abändern kann. Dem Systemverwalter bleibt aus diesem Dilemma nur eine Möglichkeit. Beim Konfigurieren und Starten kann es die Möglichkeit geben, den communities eigene Namen zu geben. Dies hat dann zur Folge, daß man nicht **secret** (wie normalerweise), sondern einen anderen Namen nimmt. So wird es dem 'normalen Benutzer' erschwert, sensible Daten zu verändern. Allerdings ist dies auch nur

ein scheinbarer Schutz, da man durch reines Mithören der 'SNMP-PacketDataUnits' (SNMP-PDUs) und dem Wissen, wie diese aufgebaut sind, den community-Namen herausfinden kann.

1.4 SNMP Version 2 (SNMPv2): Änderungen zu SNMPv1

SNMPv2 ist eine Weiterentwicklung von SNMPv1. Im wesentlichen wurde es mit folgenden Erweiterungen versehen, von denen hier nur wenige genauer ausgeführt werden [HeGr93]:

- Erweiterungen zur 'Structure of Management Information'
- Kommunikation zwischen Managerstationen
- Sicherheit
  Die 'Secure' SNMP-Arbeitsgruppe erarbeitete einen Lösungsvorschlag, wie man dem oben erwähnten Sicherheitsproblem in SNMPv1 entgegenwirken kann. Sie schlug vor, daß der MD5-Algorithmus zur Authentifikation benutzt wird. Dieser Algorithmus bildet aus einer Art Prüfsumme und einem Zeitstempel einen 16 Byte langen individuellen Schlüssel für jedes SNMPv2-Paket. Durch diesen Schlüssel ist der Empfänger in der Lage zu überprüfen, ob das Paket verfälscht wurde und von wem es abgeschickt wurde. So kann verhindert werden, daß ein Netzbeobachter die übertragene Nachricht leicht verändern kann. Zudem besteht hier die Option, die Nachrichten zu verschlüsseln. Als Verschlüsselungsmechanismus ist der DES (Data Encryption Standard) [DES77] des 'National Institute of Standard' vorgesehen. Hier ist zu erwähnen, daß der Netzwerkadministrator die Möglichkeit hat, die einzelnen Sicherheits- und Authentifizierungsmechanismen durch Parties zu konfigurieren. Durch diese Parties ist es möglich, unterschiedliche Zugriffsrechte und die Sichtweise auf die MIB individuell festzulegen. Mit anderen Worten, für jede Party wird ein bestimmter Teilbereich des MIB-Baumes konfiguriert (MIB-View), so daß eine Station nur die für sie vorgesehenen Objekte lesen bzw. beschreiben kann.
  Zusätzlich wurden Zeitstempel für die Lebensdauer einer SNMP-Nachricht eingeführt, die verhindern sollen, daß Wiederholungen einer gültigen Nachricht zu einem späteren Zeitpunkt erfolgen.
- Übertragung größerer Informationsmengen in einer PDU
  Als Ergänzung zu den schon aus SNMPv1 bekannten Befehlen **Get**, **GetNext** und **Set** wurde der **GetBulk**-Request eingeführt. Er ermöglicht es, einen kompletten MIB-Baum mit nur einer Anfrage zu lesen. Die Anzahl der zu bearbeitenden Daten wird somit verringert und die Belastung des Netzes wird wesentlich reduziert.
- Erweiterte Fehlersignalisierung
- Verwendung unterschiedlicher Transportdienste

Genauere Erklärungen zu den einzelnen Punkten sind in den RFCs 1441–1452 zu finden und wurden aus Gründen der Thematik hier weggelassen.

2 SNMP Version 3 (SNMPv3)

Die SNMPv3-Arbeitsgruppe versucht soweit wie möglich, die Konzepte, technische Elemente und Dokmentationen der SNMPv2u und SNMPv2* Ansätze zu übernehmen. SNMPv2u und SNMPv2* sind dabei Ansätze, die versuchten, mehr Sicherheit und andere Verbesserungen für SNMP zu entwickeln. Allerdings konnte man sich nicht einig werden, wie man die Verbesserungen zu SNMPv1 hinzufügen sollte. So kam es, daß die 'SNMPv2 Working Group' sich nicht auf einen einzigen Ansatz einigen konnte. Dies hatte dann die obengenannten zwei Ansätze zur Folge. Die SNMPv3 Arbeitsgruppe setzte sich schließlich folgende Ziele [MuCO97]:

- Definition von SNMPv3-Modulen und -Schnittstellen
- Spezifikation vom 'Message Processing' und 'Control Module'
- Spezifikation des 'Security Model Module'
- Spezifikation des 'Local Processing Module'
- Spezifikation des SNMPv3-'Proxy'

2.1 Elemente der Architektur

In diesem Abschnitt werden die verschiedenen Elemente der Architektur beschrieben und benannt. Es gibt drei Arten der Benennung

- Benennung der 'Entities' (Einheiten)
- Benennung der 'Identities' (Identitäten)
- Benennung der 'Management Information' (Managementinformation)

auf die im folgenden näher eingegangen wird.

2.1.1 Benennung der 'Entities'

Eine SNMP-'Entity' ist eine Applikation dieser Architektur. Jede dieser 'Entities' besteht aus einer SNMP-'Engine' und einer oder mehreren verbundenen Applikationen.

Abbildung 2 zeigt Details einer SNMP-'Entity' und ihrer Komponenten.

2.1.2 Die SNMP-Engine

Die SNMP-Engine bietet Dienste zum Senden und Empfangen von Nachrichten, Authentisierung und Verschlüsseln von Nachrichten sowie zum Überwachen der Zugriffe auf MOs. Es gibt eine eins-zu-eins Assoziation zwischen einer SNMP-Engine und der SNMP-Entity welche die Engine enthält.

Die SNMP-Engine besteht, wie aus Abbildung 2 leicht zu ersehen ist, aus einem Dispatcher, einem 'Message Processing Subsystem', einem 'Security Subsystem' und einem 'Access Control Subsystem'.

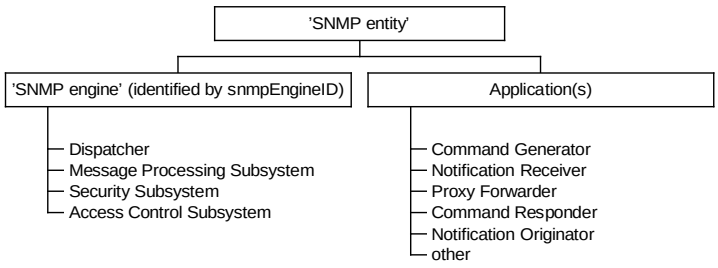

Abbildung 2: Details und Aufbau einer SNMP-Entity

Der Dispatcher
Der Dispatcher hat in der SNMP-Engine die Aufgabe, zeitgleich verschiedene Versionen von SNMP-Nachrichten zu unterstützen. Er bewerkstelligt dies, indem er SNMP-Nachrichten empfängt und verschickt. Aus den eingehenden Nachrichten entnimmt er die Version und reicht sie an das entsprechende 'Message Processing Model' weiter. Er stellt weiterhin eine abstrakte Schnittstelle für SNMP-Applikationen zur Verfügung, um deren PDUs zu verschicken. Ein weiteres abstraktes Interface für den Transport von PDUs an entfernte SNMP-Entities stellt er auch zur Verfügung.

Das Message Processing Subsystem
Das Message Processing Subsystem (MPS) bereitet die zu versendenen Nachrichten auf und extrahiert die Daten aus empfangenen Nachrichten. Das MPS bedient sich dabei mehrerer Message Processing Models, wie es in Abbildung 3 dargestellt ist.

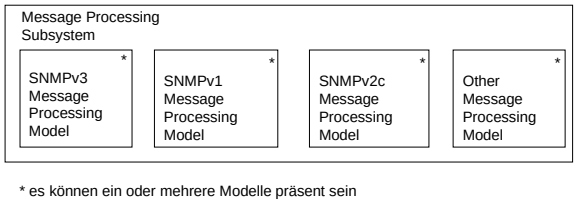

Abbildung 3: Details und Aufbau des Message Processing Subsystems

Jedes Message Processing Model definiert dabei für jede SNMP-Nachricht ein versionsspezifisches spezielles Format und koordiniert die Preparation und Extraktion eines jeden solchen versionsspezifischen Nachrichtenformats.

Das Security Subsystem
Das Security Subsystem bietet Sicherheitsdienste an. Es bietet Authentisierung und Geheimhaltung von Nachrichten und beinhaltet für diesen Zweck verschiedene Sicherheitsmodelle, wie es in Abbildung 4 dargestellt ist. Jedes Sicherheitsmodell beschreibt die Bedrohungen, gegen welche es schützen soll, die Ziele seiner Dienste und das verwendete Sicherheitsprotokoll.

Das Sicherheitsprotokoll definiert Mechanismen, Prozeduren und MIB-Daten, die für die Geheimhaltung und Authentisierung benutzt werden.

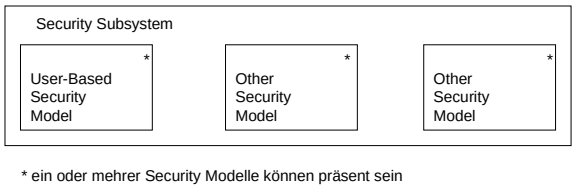

Abbildung 4: Details und Aufbau des Security Subsystems

Das Access Control Subsystem
Das Access Control Subsystem bietet Authentisierungsdienste an, wie sie ein oder mehrere der Access Control Modelle vorschreiben. Ein 'Access Control Model' aus Abbildung 5 definiert dabei genau eine Zugriffsentscheidungsfunktion unter Berücksichtigung der Zugriffsrechte.

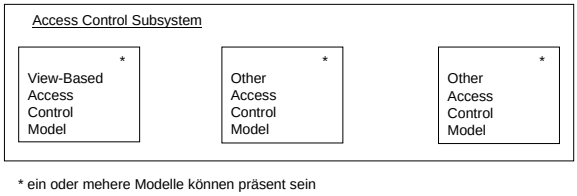

Abbildung 5: Details und Aufbau des Access Control Subsystems

2.1.3 Applikationen

Es gibt viele Arten von Applikationen. Einige sollen hier nun aufgeführt werden:

- *command generators*, die Managementdaten überwachen und manipulieren;
- *command responders*, die den Zugang zu den Managementdaten bieten;
- *notification originators*, die asynchrone Nachrichten initiieren;
- *notification receivers*, die asynchrone Nachrichten verarbeiten;
- *proxy forwarders*, die Nachrichten zwischen den Entities weiterleiten.

Diese Applikationen benutzen alle die Dienste, die ihnen die SNMP-Engine zur Verfügung stellt.

Der SNMP-Manager

Eine SNMP-Entity, die ein oder mehrere 'command generators' und/oder 'notification receiver'-Applikationen beinhaltet, wurde der Tradition folgend SNMP-Manager genannt. Den Aufbau und die Arbeitsweise eines solchen Managers zeigt Abbildung 6.

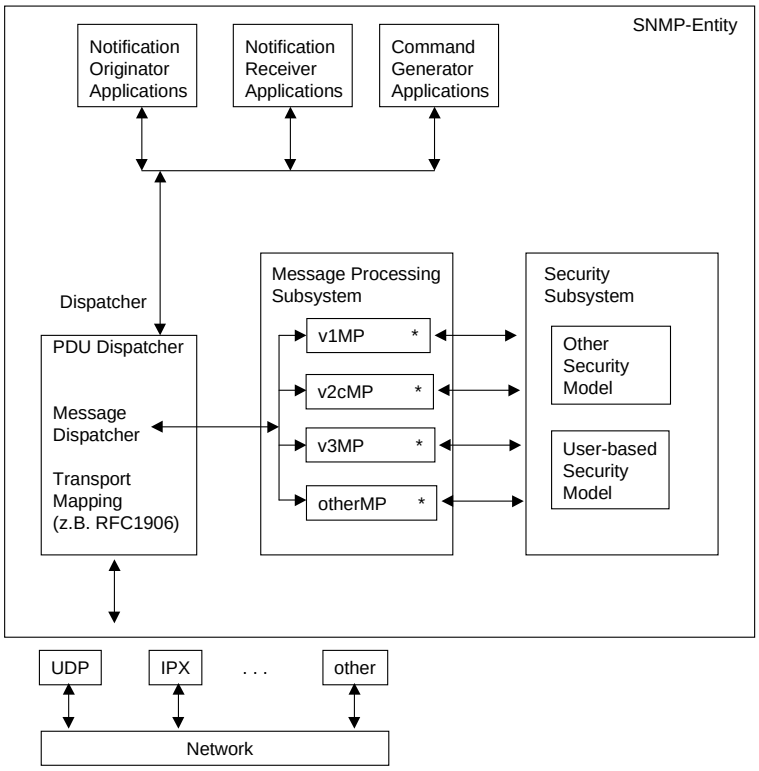

Abbildung 6: Details und Aufbau des SNMP-Managers

Der SNMP-Agent

Eine SNMP-Entity, die ein oder mehrere 'command responder' und/oder 'notification originator'-Applikationen beinhaltet, wurde der Tradition entsprechend SNMP-Agent genannt. Den Aufbau und die Arbeitsweise eines solchen Agenten zeigt Abbildung 7.

2.1.4 Benennung der Identities

Die Benennung der Identitäten beruht im großen und ganzen auf dem 'principal', dem 'securityName' und der modellabhängigen 'securityID'.

Der *principal* ist dabei der Auftraggeber, in dessen Namen ein Dienst angeboten oder eine Verarbeitung stattfinden soll. Ein Auftraggeber kann unter anderem

- ein Einzelner, der in einer speziellen Art und Weise handelt,
- mehrere, die zusammen in einer speziellen Art und Weise handeln,
- eine oder mehrere Applikationen oder
- Kombinationen davon

sein.

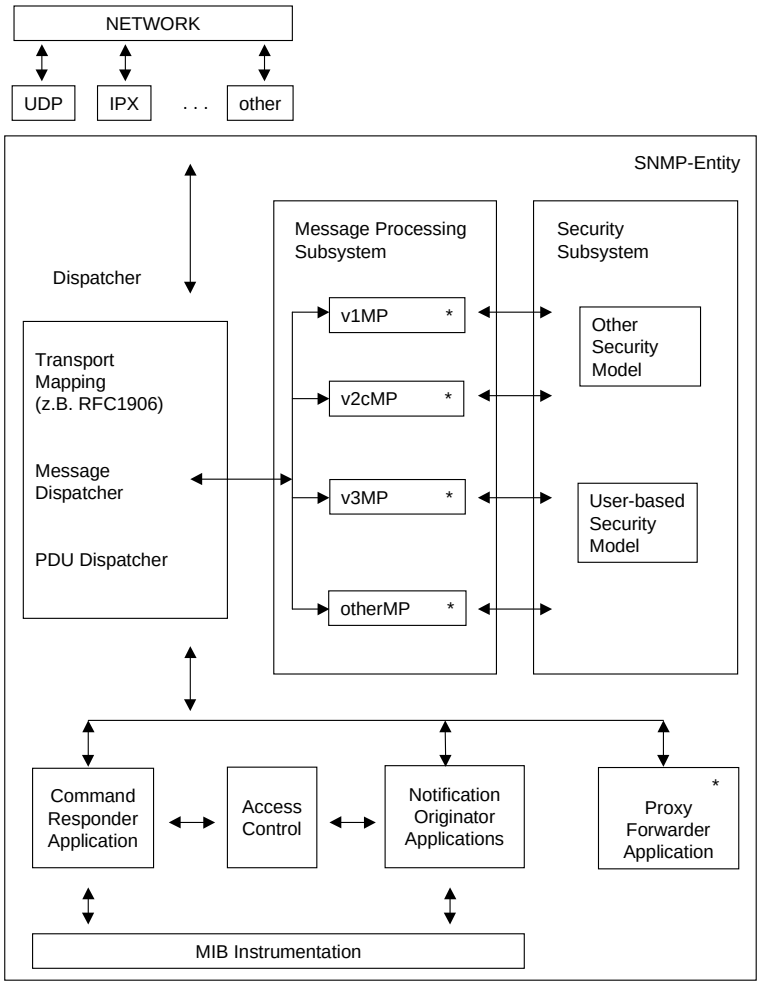

Abbildung 7: Details und Aufbau des SNMP-Agenten

Der *securityName* ist die Klartextdarstellung des Auftraggebers (principal). Er hat ein modellunabhängiges Format und kann außerhalb eines speziellen Sicherheitsmodells benutzt werden.

Die modellabhängige *securityID* ist eine modellspezifische Repräsentation des 'securityName' in einem speziellen 'securityModel'. Diese 'securityIDs' können im Klartext dargestellt werden und haben eine modellabhängige Syntax.

Der Zusammenhang wird aus Abbildung 8 ersichtlich. Hinzuzufügen ist noch, daß die Umwandlung zwischen 'securityID' und 'securityName' in den Zuständigkeitsbereich des entsprechenden Sicherheitsmodells fällt. Abbildung 8 verdeutlicht dies.

2.1.5 Benennung der 'Mangement Information'

Die Managementinformation ist in einer SNMP-Entity vorhanden, in der eine 'command responder'-Applikation lokalen Zugang zu den unter Umständen vielfältigen Kontexten hat. Diese Applikation benutzt eine 'contextEngineID', die gleich der 'SNMPEngineID' ihrer zugehörigen SNMP-Engine ist. Der SNMP-Kontext oder kurz Kontext ist eine Sammlung von Managementinformationen, die für eine SNMP-Entity zugänglich sind. Ein Element der Managementinformation kann dabei in mehreren Kontexten existieren (s. Abbildung 9).

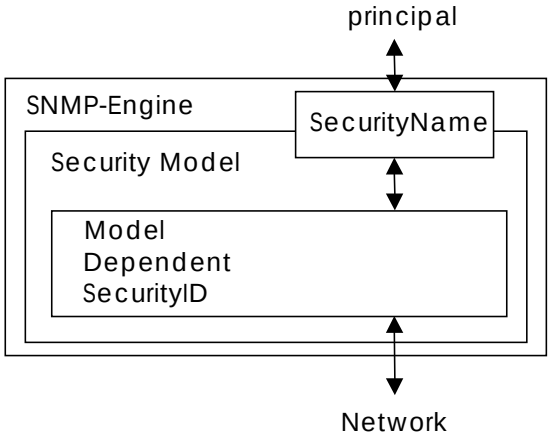

Abbildung 8: Details und Arbeitsweise des Security Models

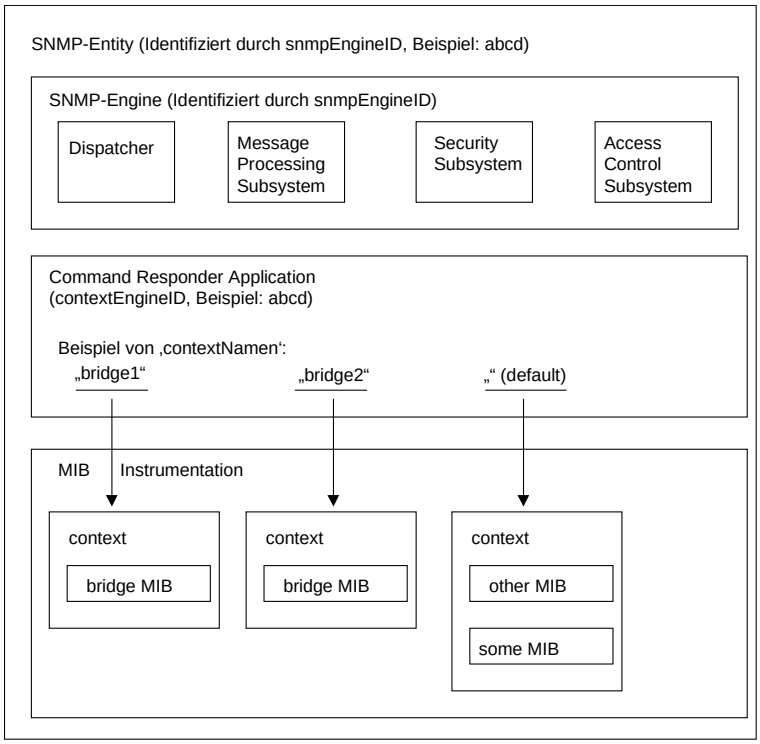

Abbildung 9: Beispiel zur Bennenung von Managementinformationen

Normalerweise gibt es viele Instanzen eines MO-Typs innerhalb einer Managementdomäne. Der Einfachheit halber erlaubt es die Identifizierungsinstanz, die durch das MIB-Modul spezifiziert wird, nicht, daß eine Instanz hervorgehoben gegenüber den anderen innerhalb einer Managementdomäne ist. Es erlaubt eher jeder Instanz, nur innerhalb eines Gültigkeitsbereiches oder Kontextes identifiziert zu werden. Innerhalb einer Managementdomäne gibt es mehrere dieser Kontexte. Oft ist ein Kontext eine physikalische oder logische Einheit. Obwohl ein Kontext auch mehrere Einheiten umfassen, ein Teilbereich einer einzelnen Einheit oder sogar ein Teilbereich mehrerer Einheiten sein kann, ist ein Kontext immer definiert als eine Untermenge einer einzelnen SNMP-Entity. Aus diesem Grund muß der 'contextName' und die 'contextEngineID' neben dem Objekttyp und der Instanz identifiziert werden, um einen einzelnen Eintrag

einer Managementinformation innerhalb einer Managementdomäne zu identifizieren. Diesen Zusammenhang soll folgendes Beispiel erläutern. Das Managed Object **ifDescr** (s. [McKa94]) ist definiert als eine Beschreibung einer Netzwerkschnittstelle. Um die Beschreibung der ersten Netzwerkschnittstelle von Gerät **device-X** zu identifizieren, sind vier Informationen nötig:

- die snmpEngineID der SNMP-Entity, welche Zugriff auf die Mangementinformation des Gerätes **device-X** hat,
- der 'contextName' (in diesem Fall **device-X**),
- der MO-Typ (**ifDescr**) und
- die Instanz ("1").

Jeder Kontext hat (mindestens) eine eindeutige Kennung innerhalb der Managementdomäne. Das gleiche Element der Managementinformation kann dabei allerdings mehrere eindeutige Kennungen haben. Dies tritt dann auf, wenn ein Element einer Managementinformation in mehreren Kontexten existiert oder ein Kontext mehrere eindeutige Kennungen hat.

Die Kombination von 'contextEngineID' und 'contextName' identifiziert eindeutig den Kontext in einem Verwaltungbereich.[2]

Die 'contextEngineID'
Innerhalb eines Verwaltungsbereiches Identifiziert eine 'contextEngineID' eindeutig eine SNMP-Entity, die eine Instanz eines Kontextes mit einem speziellen 'contextName' realisieren kann.

Der 'contextName'
Der 'contextName' wird benutzt, um einen Kontext zu benennen. Jeder Kontext muß dabei eindeutig innerhalb einer SNMP-Entity sein.

Die 'scopedPDU'
Eine 'scopedPDU' ist ein Datenblock, der eine 'contextEngineID', einen 'contextName' und eine SNMP-PDU enthält. Diese PDU beinhaltet Informationen, die eineindeutig innerhalb eines Kontextes eines Verwaltungsbereiches benannt sind. Diese eindeutige Benennung ist durch die Kombination von 'contextEngineID' und 'contextName' gegeben. Mehr Informationen über SNMP-PDUs sind in [CMRW96c] zu finden.

Die 'maxSizeResponseScopedPDU'
Die 'maxSizeResponseScopedPDU' ist die maximale Größe, die eine 'scopedPDU' innerhalb einer 'response message' haben darf.[3]

[2]Beachte: Es können mehrere eindeutige Kombinationen von 'contextEngineID' und 'contextName' existieren, die eindeutig den selben Kontext definieren.

[3]Beachte: Die Größe der scopedPDU beinhaltet NICHT die Größe des SNMP-Nachrichtenkopfs.

Das 'securityLevel'
Die hier vorgestellte Architektur kennt drei Sicherheitsstufen (wie auch bereits in SNMPv2 vereinbart wurde):

- ohne Authentisierung und ohne Geheimhaltung (`noAuthNoPriv`)
- mit Authentisierung aber ohne Geheimhaltung (`authNoPriv`)
- mit Authentisierung und mit Geheimhaltung (`authPriv`)

Die Reihenfolge ist so, daß `noAuthNoPriv` die niedrigste und `authPriv` die höchste Wertigkeit hat.

2.2 Erstellen und verschicken von Managementnachrichten

In diesem Abschnitt wird beschrieben, wie Nachrichten verarbeitet (wie wird die ein / ausgehende Nachricht verarbeitet) und versandt werden und wie sie lokal verarbeitet (welche Module passiert die Nachricht intern) werden.

2.2.1 Versenden und Verarbeiten von Nachrichten

Der Dispatcher ist eine der Schlüsselfiguren in einer SNMP-Engine. Er hat, wie schon unter 2.1.2 aufgezeigt, die Aufgabe, die verschiedenen Requests den jeweiligen versionsspezifischen 'Message Processing Models' zuzuweisen und die PDUs an verschiedene Applikationen zu verteilen. Um Nachrichten zu verschicken, gibt eine Applikation eine PDU, d.h. die Daten, die nötig sind, um die Nachricht aufzubereiten und zu verschicken, an den Dispatcher. Die Applikation legt dabei fest, welches versionsspezifische 'Message Processing Model' und welches 'Security Model' benutzt werden soll. Sobald die Nachricht soweit bearbeitet ist, versendet der Dispatcher die Nachricht. Bei eingehenden Nachrichten hingegen stellt der Dispatcher fest, um welche SNMP-Version es sich handelt und gibt die Nachricht dann dem entsprechenden versionsspezifischen 'Message Processing Model' weiter. Dieses extrahiert dann die Komponenten der Nachricht und koordiniert die Ausführung des entsprechenden Sicherheitsdienstes. Nach der versionsspezifischen Verarbeitung stellt der PDU-Dispatcher die Applikation, welche die PDU zur Verarbeitung empfangen soll, fest und leitet die Nachricht weiter. Während all dieser Arbeiten erfaßt der Dispatcher Statistiken über die SNMP-Nachrichten und das Verhalten der SNMP-Engine in MOs, um diese einer entfernten SNMP-Entitiy zugänglich zu machen. Wie schon in den Abbildungen 6 und 7 zu sehen ist, können parallel mehrere 'Message Processing Models' vorhanden sein. Aus Gründen der Thematik wird hier allerdings nur auf das 'SNMPv3 Message Processing Model' eingegangen. In [SFDC90] und [CMRW96a] sind Angaben zu den 'Message Processing Models' zu SNMPv1 und SNMPv2c zu finden.

Das 'SNMPv3 Message Format'
In diesem Abschnitt wird das 'SNMPv3 Message Format' (v3MF) und das dazugehörige 'SNMPv3 Message Processing Model' (v3MP) vorgestellt.

```
SNMPv3MessageSyntax DEFINITIONS IMPLICIT TAGS ::= BEGIN
       SNMPv3Message ::= SEQUENCE {
           -- identify the layout of the SNMPv3Message
           -- this element is in same position as in SNMPv1
           -- and SNMPv2c, allowing recognition
           msgVersion INTEGER { snmpv3 (3) },
           -- administrative parameters
           msgGlobalData HeaderData,
           -- security model-specific parameters
           -- format defined by Security Model
           msgSecurityParameters OCTET STRING,
           msgData  ScopedPduData
       }
       HeaderData ::= SEQUENCE {
           msgID      INTEGER (0..2147483647),
           msgMaxSize INTEGER (484..2147483647),

           msgFlags   OCTET STRING (SIZE(1)),
                      --  .... ...1   authFlag
                      --  .... ..1.   privFlag
                      --  .... .1..   reportableFlag
                      --              Please observe:
                      --  .... ..00   is OK, means noAuthNoPriv
                      --  .... ..01   is OK, means authNoPriv
                      --  .... ..10   reserved, must NOT be used.
                      --  .... ..11   is OK, means authPriv
           msgSecurityModel INTEGER (0..2147483647)
       }
       ScopedPduData ::= CHOICE {
           plaintext    ScopedPDU,
           encryptedPDU OCTET STRING  -- encrypted scopedPDU value
       }
       ScopedPDU ::= SEQUENCE {
           contextEngineID  OCTET STRING,
           contextName      OCTET STRING,
           data             ANY -- e.g., PDUs as defined in RFC1905
       }
   END
```

Im folgenden sollen die wichtigen Punkte aus der oben aufgeführten ASN.1-Beschreibung näher erläutert und dabei auch das v3MP erklärt werden. Die **msgVersion** ist für die Identifizierung der Version nötig und wird in diesem Falle auf snmpv3(3) gesetzt. Die **msgID** wird für die Koordinierung der Anfragen und Antworten genutzt. Zudem wird sie bei v3MP für die Koordinierung der Verarbeitung durch verschiedene Modelle der Untersysteme innerhalb der Architektur genutzt. Die Werte der msgID sollten so generiert werden, daß keine nochmalige Benutzung irgendeines noch ausstehenden Wertes vorkommt.[4] Die **msgMaxSize** wird dafür genutzt, um der antwortenden Engine mitzuteilen, wie groß die Antwort auf eine Frage maximal sein kann.

Das **msgFlags**-Feld der Nachricht zeigt an, wie eine Nachricht zu verarbeiten ist. Ein Flag dieses Feldes ist das **authFlag**. Es gibt an, ob die Nachricht vom Empfänger

[4]Beachte: Die Request-Id in einer PDU wird von einer SNMP-Applikation zur Identifizierung der PDU genutzt. Die msgID wird allerdings von der Engine genutzt, um die Nachricht, welche die PDU enthält, zu identifizieren. Die Engine kann unter Umständen gezwungen sein, die Nachricht zu identifizieren, auch wenn die Entschlüsselung der PDU fehlschlägt.

authentifiziert werden muß (Wert 1) oder nicht (Wert 0). Ist der Wert 1, so muß das securityModel des Senders den securityNamen identifizieren, unter dessen Auftrag die Nachricht geschickt wurde. Zudem muß es unter einer securityModel-spezifischen Art und Weise genügend Daten für den Empfänger zur Verfügung stellen, damit dieser in der Lage ist, die Kennung zu authentisieren. Das **privFlag** ist das zweite Flag dieses Feldes. Sollte dieses Flag gesetzt sein, so muß das securityModel des Senders auch die scopedPDU in einer SNMP-Nachricht vor Offenlegung schützen (z.B. durch Verschlüsseln). Sollte das privFlag gesetzt sein, so ist es eine Grundvoraussetzung der SNMP-Architektur, daß dann auch die Authentisierung erforderlich ist. Als letztes Flag des msgFlag-Feldes ist das **reportableFlag** zu nennen. Es gibt an, ob eine ReportPDU zum Sender zurückgeschickt werden muß oder nicht.[5]

Das **msgSecurityModel** gibt an, welches der securityModels der Sender nutzte, um die Nachricht zu generieren. Dieses Modell muß dann der Empfänger nutzen, um die Sicherheitsverarbeitung der Nachricht durchzuführen. Das **msgSecurityParameter**-Feld der SNMPv3-Nachricht wird für die Kommunikation zwischen den securityModel-Modulen des Senders und des Empfängers genutzt. Die Daten dieses Feldes werden vom v3MP nicht interpretiert sondern nur vom securityModel verwendet, das durch das msgSecurityModel-Feld definiert wurde.

Das **scopedPduData**-Feld enthält entweder die scopedPDU in Klartext (privFlag=0) oder eine encryptedPDU (eine verschlüsselte PDU), die durch einen OCTET STRING dargestellt und von dem securityModel des Empfängers wieder entschlüsselt wird, um eine scopedPDU in Klartext zu erhalten. Die **scopedPDU** beinhaltet Informationen, um den verwalteten eindeutigen Kontext und eine PDU zu identifizieren. Die OIDs in der PDU beziehen sich auf MOs, die zugänglich im spezifizierten Kontext sind. Die **contextEngineID** definiert eineindeutig, innerhalb einer Verwaltungsdomäne, eine SNMP-Entity, die eine Instanz eines Kontextes mit einem besonderen 'contextName' realisieren kann. Bei eingehenden Nachrichten wird die contextEngineID benutzt, um herauszufinden, an welche Applikation die scopedPDU weitergeleitet werden muß. Bei zu versendenen Nachrichten setzt das v3MP die contextEngineID auf den Wert, der in der Anfragenachricht bereitgestellt wurde. Das **contextName**-Feld in Verbindung mit der contextEngineID identifiziert den speziellen Kontext, dem die Managementinformation der PDU zugedacht ist. Der 'contextName' ist eindeutig innerhalb einer SNMP-Entity, die durch die 'contextEngineID' festgelegt ist. Das **data**-Feld zuletzt enthält die PDU. Unter anderem steht in der PDU der Typ, den das v3MP benutzt hat. Das v3MP spezifiziert, daß die PDU einem der in [CMRW96c] definierten Typen entspricht.

2.2.2 Die lokale Verarbeitung von Nachrichten und der Zugriff auf Managementinformationen

Lokale Verarbeitung kann in einer SNMPv3-Engine auftreten, wenn sie als Agent auftritt und einen SNMPv3-Request, der von einer SNMPv3-Engine, die als Manager auftritt, beantwortet. Dieser Request ist entweder vom Typ GetRequest, GetNextRequest, GetBulkRequest oder SetRequest.[6]

[5] Das reportableFlag muß immer gesetzt werden, sofern es sich um eine requestPDU (z.B. getrequest) oder eine bestätigte notification-typePDU (z.B. informPDU) handelt.

[6] Im Folgenden wird des öfteren von SNMPv2 Operationen geredet, da diese aus den SNMPv2 Dokumenten übernommen wurden.

Lokale Verarbeitung tritt nur dann auf, wenn der Request sich auf die lokale Managementinformation bezieht, die durch die contextEngineID in der scopedPDU festgelegt wird. Das Message Processing Modul bestimmt dabei, ob die contextEngineID zu der lokalen SNMPv3-Engine gehört. Wenn dem so ist, leitet das Modul die scopedPDU and das 'Local Processing' Modul weiter.

Lokale Verarbeitung tritt fast immer auf, wenn eine Antwort generiert werden muß (manchmal auch bei Fehlermeldungen oder anderen außergewöhnlichen Anzeigen). Diese Antwort nennt man dann **responsePDU**. Lokale Verarbeitung kann zudem aus der Generierung eines Errorreports resultieren. Diese nennt man dann eine **reportPDU**.

Der 'Local Configuration Datastore'
Um das Modell zu implementieren, muß jede SNMPv3-Engine sich ihre eigenen Informationen über den Zugriff, Rechte und Vorschriften bezüglich ihrer Menge von Informationen merken. Diese Menge wird, weil es lokal gespeicherte Informationen sind, der SNMPv3-Engine's Local Configuration Datastore (LCD) genannt.

Um einem SNMPv3-Engine's LCD eine 'remoteConfiguration' (Konfiguration durch eine entfernte Managementstation) zu ermöglichen, müssen Teile des LCD als MOs zugänglich sein. Das MIB-Modul, welches in [WiPM97] enthalten ist, definiert diese MOs.

Elemente des Modells
Dieser Abschnitt definiert, wie die Zugriffskontrolle, die in diesem LPM angewandt wird, realisiert wird.

Ein **groupName** identifiziert eine **group** von keinem oder mehreren 'security entities', unter dessen Namen auf SNMP-MOs zugegriffen werden kann. Es ist die Aufgabe des 'Message Processing and Control Module' (MPC) herauszufinden, für welchen 'groupName' eine eingehende SNMP-Anfrage bearbeitet werden muß.

Das SNMPv3 Local Processing Model verlangt, daß der 'groupName' an das Local Processing Module als Eingabe weitergeleitet wird, wenn eine PDU an dieses zur Verarbeitung weitergereicht wird.

Das **Level of Security** (LoS) identifiziert die Sicherheitsstufe (in bezug auf die Sicherheit der Nachricht), die für die Übermittlung der SNMP-Nachricht genutzt wurde. Es ist die Aufgabe des MPC herauszufinden, welches LoS für eine eingehende SNMP-Nachricht benutzt wurde. Es ist zusätzlich die Aufgabe des Local Processing Module, das LoS zu beachten, wenn es die Zugriffskontrolle anwendet.

Dieses Local Processing Model kennt drei verschiedene LoS:

1. ohne Authentisierung und ohne Geheimhaltung,
2. mit Authentisierung aber ohne Geheimhaltung und
3. mit Authentisierung und mit Geheimhaltung

Dieses Local Processing Model verlangt, daß das LoS an das Local Processing Module als Eingabe weitergeleitet wird, wenn die Nachricht an das 'local processing module' weitergereicht wird.

Der **SNMP context** ist eine Sammlung von Managementinformationen, auf die ein SNMP-Agent zugreifen kann. Ein Element der Mangementinformation kann dabei in mehreren Kontexten existieren.

Die **scopedPDU** enhält den **Naming-Scope** (Namensbereich), auf den sich die SNMP-PDU bezieht. Dabei identifiziert der Namensbereich den SNMP-Agenten und den Kontext, auf den der Agent (lokalen) Zugriff hat, eineindeutig.

Der Namensbereich enthält die **contextEngineID**, die wiederum eineindeutig den SNMP-Agenten identifiziert, der (lokalen) Zugriff auf die Managementinformation besitzt, die durch den 'contextName' und die SNMP-PDU festgelegt wird.

Der **'Naming-Scope'** besteht aus dem 'contextName', welcher eineindeutig den SNMP-Kontext identifiziert, auf den der SNMP-Agent, an den die SNMP-PDU geschickt wurde, Zugriff hat und der schon oben erwähnten 'contextEngineID'.

Die **Access Policy** dieses 'Local Processing Models' beschreibt die Zugriffsrechte der Gruppen. Zu den Zugriffsrechten, die durch das LoS gegeben sind, gibt es dann noch die '**read-view**' und die '**write-view**'. Die 'read-view' ist eine Menge von Objektinstanzen, auf die die Gruppe lesend zugreifen kann. Auf Objekte wird lesend zugegriffen, wenn eine Verarbeitung nach einem get-, getnext- oder getbulk-Befehl ansteht und wenn traps verschickt werden. Die 'write-view' gibt an, auf welche Objektinstanzen schreibend zugegriffen werden kann. Objekte werden dann beschrieben, wenn eine set-Operation zu verarbeiten ist.

Elemente der Prozedur
In diesem Abschnitt wird beschrieben, welche Prozeduren des Local Processing Moduls, das dieses Local Processing Model implementiert, abgearbeitet werden, wenn eine scopedPDU bearbeitet oder generiert wird.

Im folgenden wird nun beschrieben, welche Prozeduren abgearbeitet werden, wenn eine scopedPDU empfangen wird.

1. Der PDU-Teil der scopedPDU wird bearbeitet. Sollte der serialisierte PDU-Wert (nach [CMRW96e]) nicht der Serialisierung der PDU-Werte entsprechen, so wird der **snmpInASNParseErrs counter** (s. [CMRW96b]) inkrementiert und die empfangene scopedPDU wird ohne weitere Bearbeitung verworfen. Eine Fehleranzeige wird an das aufrufende Modul zurückgesendet.

2. Der SNMPv2-Operationstyp wird aus der ASN.1-Marke festgestellt, die mit der PDU-Komponente assoziiert ist.

3. Sollte der SNMPv2-Operationstyp entweder eine SNMPv2-Response-, -Trap-, -Inform- oder -Report-Operation sein, so wird der **snmpV3Unauthorized-Operations counter** erhöht. [7]

4. Das LCD wird nach Informationen über den SNMP-Kontext (identifiziert durch den contextName) gefragt. Sollte die Information nicht vorhanden sein, so wird

[7] Es ist zu beachten, daß normalerweise keine solche PDU das MPC zu diesem Modul verlassen sollte. Auf dies weisen die Autoren des [WiPM97] auf Seite 8 hin. Allerdings ist hierzu noch keine andere Lösung veröffentlicht worden.

der **snmpV3LpStatsUnknownContexts counter** erhöht und ein Report (s. [CMRW96c]) generiert. Die empfangene scopedPDU wird nicht weiter bearbeitet und verworfen.

5. Das LCD wird über Informationen betreffend die SNMP-Group befragt. Sollte diese Information nicht vorhanden sein, so wird der **snmpV3LpStatsUnknownGroups counter** erhöht. Es wird zusätzlich ein Report generiert (s. [CMRW96c]) und die empfangene scopedPDU verworfen.

6. Wird eine SNMPv2-Operation vom Typ get, getnext, getbulk oder set empfangen, so werden die entsprechenden Sicherheitsmechanismen überprüft. Treten dort keine Fehler auf, so werden die Operationen bearbeitet. Tritt jedoch ein Fehler auf, so wird eine Response-PDU generiert, die den **error-index** auf 0 und den **error-status** auf **authorizationError** (s. [CMRW96c]) gesetzt hat.

7. Sollte die SNMPv2-Operation ein Typ sein, der in keine der oben erwähnten Kategorien fällt, so ist es ein unbekannter PDUtyp. In diesem Fall wird der **snmpV3UnauthorizedOperations counter** erhöht und die scopedPDU verworfen.

Eine genau Spezifikation, welche Prozeduren bei der Generierung abgearbeitet werden, ist in [WiPM97] nicht aufgeführt.

2.3 Sicherheitsvorkehrungen in SNMPv3

In den beiden nächsten Abschnitten wird das 'User-Based Security Model (USM) und das 'View-Based Access Control Model' (ACM) näher beschrieben.

2.3.1 Das 'User-Based Security Model' (USM)

Im folgenden werden kurz die wichtigsten Aspekte des USM nach [BlWi97] beschrieben.

Bedrohungen

- Die grundsätzlichen Bedrohungen, gegen die das SNMP Security Model schützen soll, sind **Modification of Information** (modifizierte Informationen) und **Masquerade** (Maskierungen).

- Sekundäre Bedrohungen, vor denen dieses Modell begrenzten Schutz bietet, sind **Disclosure** (Enthüllung) und **Message Stream Modifikation** (Veränderung des Nachrichtenstroms).

- Letztendlich gibt es noch zwei Bedrohungen vor denen das SNMP Security Model nicht schützen muß. Diese sind das **Denial of Service** (Dienstverweigerung) und die **Traffic Analysis** (Verkehrsanalyse).

Ziele und Anforderungen
Aus den vorher aufgeführten Bedrohungen ergeben sich folgende Ziele.

1. Verifizierung einer jeden eingehenden Nachricht, daß diese nicht während der Übertragung verändert wurde,

2. Verifizierungen der Identität des Benutzers, unter dessen Namen die SNMP-Nachricht generiert worden sein soll,

3. Detektion von SNMP-Nachrichten, deren Inhalt älter als die erlaubte Lebensdauer ist und

4. Schutz gegen die Entschlüsselung einer Nachricht, sofern dies nötig ist.

Um zu den oben aufgeführten Zielen ein sicheres Netzmanagement zu bieten, wurde dieses SNMP-Sicherheitsmodell so geplant, daß auch die folgenden Aspekte berücksichtigt wurden.

1. Sollten die Erfordernisse für ein effektives Management in Zeiten hoher Netzlast nicht mit denen eines sichereren einhergehen, so sollte das Modell das effektive bevorzugen.

2. Weder das Sicherheitsprotokoll noch der zugrunde liegende Sicherheitsmechanismus sollten auf der Verfügbarkeit eines anderen Netzwerkdienstes aufbauen.

3. Ein Sicherheitsmechanismus sollte der grundlegenden Philosophie des SNMP-Managements keine Änderungen auferlegen.

Sicherheitsdienste
Die Sicherheitsdienste, die benötigt werden, um die oben aufgezeigten Ziele zu erreichen, sind folgende:

- **Datenintegrität**
  ist die Eigenschaft, daß Daten weder verändert oder in einer unauthorisierten Art und Weise zerstört wurden, noch daß die Reihenfolge der Daten stärker verändert wurde, als es unter normalen Umständen möglich ist.

- **Authentisierung der Datenquelle**
  ist die Eigenschaft, daß die angeforderte Identität des Benutzers, der die Daten generiert hat, bestätigt ist.

- **Datenvertraulichkeit** ist die Eigenschaft, daß die Information nicht verfügbar oder entschlüsselbar für unauthorisierte Einheiten oder Prozesse sind.

- **Pünktlichkeit einer Nachricht und begrentzer Replay-Schutz** ist die Eigenschaft, daß eine Nachricht, die außerhalb eines Zeitfensters (150 sek. s.[BlWi97]) zwischen Generierung und Ankunft ankommt, nicht akzeptiert wird.

Die Modulorganisation
Das hier beschriebene Sicherheitsprotokoll ist in drei unterschiedliche Module aufgeteilt. Jedes dieser Module hat seine eigene spezifische Verantwortung, so daß sie zusammen die Ziele und Sicherheitsdienste (wie oben beschrieben) erfüllen. Im folgenden werden nun die drei Module genannt und kurz beschrieben. Das **Authentisierungsmodul** muß für die Datenintegrität und die Authentisierung der Datenquelle sorgen. Das **Zeitmodul** muß vor Datenverzögerung und Replay schützen. Das **Datenschutzmodul** muß Schutz vor Entschlüsselung der Nutzdaten der Nachricht gewährleisten.

Das Zeitmodul ist dabei auf das 'User-Based' Sicherheitsmodell fixiert, während mehrere Authentisierungs- und Datenschutzmodule vorgesehen sind.

An dieser Stelle sollte noch auf zwei Protokolle bezüglich der Authentisierung und ein Datenschutzprotokoll eingegangen werden.

Das in [BlWi97], Abschnitt 6, definierte Authentisierungsprotokoll HMAC-MD5-96 muß von jedem 'User-based' Sicherheitsmodell unterstützt werden. Es basiert grob gesagt auf den MD5-'hash'-Funktionen (s. [Rive94]) und dem in [KrBC97] beschriebenen HMAC-Modus. Es kürzt die Ausgabe zudem auf 96 Bits. Dieses Protokoll wird durch **usmHMACMD5AuthProtocol** identifiziert. Zusätzlich sollte noch das HMAC-SHA-96 (s. [BlWi97] Abschnitt 7) unterstützt werden. Dieses Protokoll benutzt die SHA-'hash'-Funktion [SHA95] und den HMAC Modus (s. [KrBC97]) und kürzt die Ausgabe auch auf 96 Bits. Das Protokoll wird durch **usmHMACSHAAuthProtocol** identifiziert.

Das in [BlWi97], Abschnitt 8, beschriebene Datenschutzprotokoll **CBC-DES Symmetric Encryption Protocol** wird durch **usmDESPrivProtocol** identifiziert. Im Moment ist es das einzige Protokoll, das im 'User-based'-Sicherheitsmodell benutzt wird. In Zukunft kann es aber durch andere ersetzt oder ergänzt werden.

Schutz vor Replays, Delays und Umadressierung
Um vor Replays, Verzögerungen und Umadressierungen zu schützen, werden einige Mechanismen benutzt. Bevor diese jedoch kurz erklärt werden, sollte der Begriff der **'authorative'-SNMP-Engine** eingeführt werden:

Eine 'authoritive'-SNMP-Engine ist eine der SNMP-Engines, die an der Kommunikation beteiligt sind. Diese Aufgabe fällt bei einer SNMP-Nachricht, die eine Antwort erfodert, dem Receiver zu. Anderenfalls ist der Sender die 'authorative'-SNMP-Engine.

Der Mechanismus, der vor Verzögerungen oder Replays schützen soll, nutzt die Zeitstempel, die in jeder generierten Nachricht enthalten sind. Die SNMP-Engine evaluiert diese Indikatoren (Zeitstempel) und entscheidet dann, ob eine empfangene Nachricht rechtzeitig ankam. Ist dies nicht der Fall, so kann die SNMP-Engine überprüfen, ob die Nachricht, im Vergleich zur vorhergehenden Nachricht des Senders, ähnlich verspätet ist, um daraus Rückschlüsse auf die Gültigkeit zu ziehen. Schlägt dies alles fehl, so wird die Nachricht als nicht authentisch erachtet.

Eine SNMP-Engine muß auch einen Mechanismus besitzen, der eingehende Nachrichten mit den erwarteten Anfragen vergleicht. Dabei muß sie jede Antwort verwerfen, die keiner ausstehenden Anfrage zugeordnet werden kann.

Diese beiden Mechanismen schützen vor authentifizierten Nachrichten, die allerdings nicht rechtzeitig eingegangen sind. Allerdings schützen die beiden Mechanismen nicht

vor unautorisierter Löschung oder Unterdrückung von Nachrichten. Zusätzlich ist es einer SNMP-Engine mit diesen Mechanismen nicht möglich, ein Reihenfolgevertauschung zu erkennen, wenn alle beteiligten Nachrichten innerhalb des erlaubten Zeitfensters eintreffen. Es gibt allerdings Mechanismen, die unabhängig von den oben aufgeführten, im Sicherheitsprotokoll definiert wurden, um Reihenfolgevertauschungen, Wiederholungen, Löschungen oder Unterdrückungen innerhalb von Set-Operationen zu detektieren (z.B. snmpSetSerialNo [CMRW96b]).

In jeder Nachricht ist ein, bezüglich der 'authorative'-SNMP-Engine, eindeutiger Bezeichner, der mit dem Sender oder dem beabsichtigten Empfänger der Nachricht zusammenhängt.

Es ist anzumerken, daß eine Report-, Response- oder Trap-Nachricht, die von einer 'authorative'-SNMP-Engine an eine 'non-authorative'-SNMP-Engine geschickt wird, wiederholt werden kann und an eine andere 'non-authorative'-SNMP-Engine geschickt wird. Die zweite "non-authorative'-SNMP-Engine' kann dann möglicherweise ihre Vorstellung bezüglich des Zeitstempels erneuern. Dies wird allerdings nicht als Gefahr behandelt. In diesem Fall wird eine Report- oder Response-Nachricht vom 'Message Processing Model' verworfen, da keine ausstehenden Anfragen vorhanden sein sollten. Ein Trap hingegen kann möglicherweise akzeptiert werden. Dies ist jedoch auch keine echte Gefahr, da die zweite SNMP-Engine nicht darauf vorbereitet ist, diese Trap zu empfangen, aber es ihr erlaubt ist, die Managementinformationen, die der Trap enhält, zu lesen.

Elemente des Modells

In diesem Abschnitt werden die Definitionen, die benötigt werden, um das Sicherheitsmodell zu realisieren, zusammengefaßt und kurz erläutert.

Die Managementoperationen, die dieses Sicherheitsmodell benutzen, nutzen eine Menge von definierten `user identities`. Dies bedeutet, daß jede SNMP-Engine, die eine authorisierte Managementoperation für diesen Benutzer ausführen will, diesen Benutzer kennen muß. Zusätzlich muß eine SNMP-Engine, die mit einer anderen SNMP-Engine kommunizieren will, Wissen über einen Benutzer dieser SNMP-Engine besitzen. Dies beinhaltet das Wissen über die anwendbaren Attribute des Benutzers. Die Attribute sind wie folgt definiert:

Der **userName** gibt den Namen des Benutzers an. Der **securityName** ist ein String, der den Benutzer in einer sicherheitsmodellunabhängigen Form repräsentiert. Das Attribut **authProtocol** gibt an, ob Nachrichten dieses Benutzers authentifiziert werden können. Ist dies der Fall, so gibt es den Authentifikationsprotokolltyp an (HMAC-MD5-96 oder HMAC-SHA-96 2.3.1). Sollte die Authentisierung möglich sein, so gibt der **authKey** den (privaten) Authentifikationsschlüssel, der im Authentifikationsprotokoll genutzt wird, an.[8] Der authKey ist nicht via SNMP zugänglich. Die Länge des authKey wird durch das verwendete Protokoll bestimmt. Der einzige Weg, den Authentifikationsschlüssel entfernt zu erneuern/ändern, wird durch die beiden Attribute **authKeyChange** und **authOwnKeyChange** ermöglicht. Dies sollte in einer sicheren Art und Weise ablaufen, so daß keine Schutzvorrichtung eingeführt werden

[8] BEACHTE: Der Authentifikationschlüssel eines Benutzers wird normalerweise von Engine zu Engine unterschiedlich sein.

muß. Das **privProtocol** ist ein Indikator, ob eine Nachricht eines Benutzers verschlüsselt wurde. Ist dies der Fall, so gibt privProtocol den Typ des Protokolls an. Eines der möglichen Protokolle ist das in [BlWi97] Kapitel 8 beschriebene `CBC-DES Symmetric Encryption Protocol`. Als letztes Attribut wäre noch der **privKey** zu erwähnen. Dies ist der Schlüssel, der zur Ver-/Entschlüsselung der Nachricht benutzt wird.[9] Dieser Schlüssel ist wie der 'authKey' nicht via SNMP zugänglich, sondern kann nur (analog zu authKey) via **privKeyChange** oder **privOwnKeyChange** geändert werden. Dies sollte, wie auch schon beim 'authKey' erwähnt, in einer sicheren Art und Weise ablaufen.

Um Wiederholungen zu verhindern, enthält jede SNMP-Engine drei Objekte.

1. **snmpEngineID**, die eineindeutig eine SNMP-Engine innerhalb einer Domäne identifiziert;
2. **snmpEngineBoots**, die Anzahl der Reboots/Rekonfigurationen, seit die snmpEngineID das letzte Mal konfiguriert wurde;
3. **snmpEngineTime**, die vergangen Sekunden seit der letzten Erhöhung von 'snmpEngineBoots'.

Jede non-authorative SNMP-Engine hat dafür Sorge zu tragen, daß sie sich auf die authorative SNMP-Engine synchronisiert.

Zusätzlich wird die **msgAuthoritativeEngineID** (Darstellung der snmpEngineID der authorative SNMP-Engine) in einer authentischen Nachricht genutzt, um Wiederholungen von Nachrichten zwischen zwei SNMP-Engines an eine andere SNMP-Engine zu vermeiden. Weiterhin gibt es noch die **msgAuthoritativeEngineBoots** und **msgAuthoritativeEngineTime**, die vor Angriffen mit verspäteten Nachrichten schützen. Diese beiden Werte stellen die obengenannten Werte 'snmpEngineBoots' und 'snmpEngineTime' der authorative SNMP-Engine dar.

2.3.2 Das 'View-Based Access Control Model' (VACM)

In diesem Abschnitt wird beschrieben, wie der Zugriff (read, write, notify) auf ein spezielles Objekt (Instanz) erlaubt ist [WiMc97]. Das hier beschriebene Modell ist ein Teil des 'Access Control Subsystems'.

Die Zugriffskontrolle

Die Zugriffskontrolle findet in einer SNMP-Entity statt, wenn eine Modifikationsanfrage (z.B. Änderung einer Kontrolltabelle eines Agenten) einer SNMP-Entity bearbeitet werden. Sie tritt auch auf, wenn eine SNMP-Benachrichtigung generiert wird. Das VACM definiert nun eine Menge von Diensten, die eine Applikation nutzen kann, um die Zugriffsrechte zu überprüfen. Um diese Dienste erbringen zu können, benötigt das VACM Teile des LCD (2.2.2). Es sollte jedoch einer SNMP-Entity möglich sein, Teile des LCD aus der Entfernung zu konfigurieren. Deshalb wurden diese Teile durch MOs, definiert in der `View.based Access Control Model Configuration MIB`, zugänglich gemacht.

[9] BEACHTE: Dieser Schlüssel des Benutzers wird normalerweise von Engine zu Engine unterschiedlich sein.

Elemente des Modells
In diesem Abschnitt werden die Definitionen beschrieben, die benötigt werden, um den Zugriffskontrolldienst, der von diesem VACM bereitgestellt wird, zu realisieren.

Eine **Group** (Gruppe) ist eine Menge (0 oder mehrere) von Tupeln `<securityModel, securityName>`, in deren Auftrag auf SNMP-MOs zugegriffen werden kann. Die Gruppe definiert dabei die Zugriffsrechte, die für alle 'securityNames' der Gruppe verfügbar sind. Eine Gruppe wird durch ihren **groupName** identifiziert.

Das 'Access Control Module' nimmt an, daß der 'securityName' bereits authentifiziert wurde und bietet keine weitere Authentifikation an. Das VACM benutzt das 'securityModel' und den 'securityName' als Eingabe für das 'Access Control Module', um daraus den 'groupName', als Funktionswert davon, zu bestimmen. Aus diesem ergeben sich dann die Zugriffsrechte.

Innerhalb einer Gruppe können für unterschiedliche Zugriffsrechte für Mitglieder bezüglich des 'securityLevels' definiert werden. Dabei identifiziert das 'securityLevel' das Sicherheitslevel (vgl. [HaPW97]), das angenommen wird, wenn die Zugriffsrechte geprüft werden.

Das VACM definiert eine **vacmContextTable**, die alle lokal verfügbaren Kontexte (s. 2.2.2) durch ihre 'contextName's auflistet.

Aus Sicherheitsgründen ist es oft nützlich, wenn man die Zugriffsrechte einiger Gruppen auf eine Untermenge der Managementinformationen einer Domäne begrenzt. Um diese Fähigkeit zu bieten, ist der Zugriff auf einen Kontext via einer **MIB view** eingeführt worden. Diese stellt eine spezielle Menge von 'Managed Object Types' innerhalb eines Kontextes dar. So kann nun durch diese 'MIB view' festgelegt werden, auf welche MO's diese Gruppe welche Zugriffsrechte hat.

Da die 'Managed Object Types' und deren Instanzen durch eine baumartige Benennung (s. [CMRW96d]) identifiziert werden, ist es bequemer, eine 'MIB view' als eine Kombination von **view subtrees**[10] zu definieren. Jeder dieser 'view subtrees' ist ein Teilbaum innerhalb eines MO-Namensbaumes. Es kann nun passieren, daß eine sehr große Anzahl von 'view subtrees' erforderlich wird, um einen speziellen Bereich zugänglich zu machen. Dies ist durch folgendes Beispiel leicht nachzuvollziehen: Es sollen alle Spalten in einer konzeptuellen Zeile einere 'MIB-Table' dargestellt werden. Dies würde eine große Anzahl von 'view subtrees' erfordern, da jede in einem eigenen Unterbaum (einer pro Spalte) auftritt. Da die Formate jedoch gleich sind, kann die benötigte Menge an Unterbäumen leicht in eine Struktur zusammengefaßt werden. Diese Struktur nennt man eine **ViewTreeFamily** (Familie von 'view subtrees').

Das VACM bestimmt die Zugriffsrechte einer Gruppe. Die Rechte werden durch **read-view**, **write-view** und **notify-view** bezüglich eines speziellen Kontextes und der verwendeten 'securityModel' und 'securityLevel' dargestellt. Die 'read-view' repräsentiert dabei eine Menge von Objektinstanzen, auf die die Gruppe lesend zugreifen darf. Die 'write-view' repräsentiert eine Menge von Objektinstanzen, auf die eine Gruppe schreibend zugreifen darf. Die 'notify-view' ist die Menge von Objektinstanzen, innerhalb der eine Gruppe Benachrichtigungen senden darf.

[10] Ein 'view subtree' ist die Menge aller MIB-Objektinstanzen, die den gleichen ASN.1 'Object Identifier' Prefix haben.

Elemente der Prozedur
In diesem Abschnitt werden die Prozeduren beschrieben, die ein VACM enthalten muß, wenn es die Zugriffsrechte für eine Applikation prüfen soll.

Im folgenden wird beschrieben, wie der **isAccessAllowed Prozeß** abläuft. Abbildung 10 zeigt den Entscheidungsbaum dieses Prozesses. Der 'isAccessAllowed' Prozeß erhält

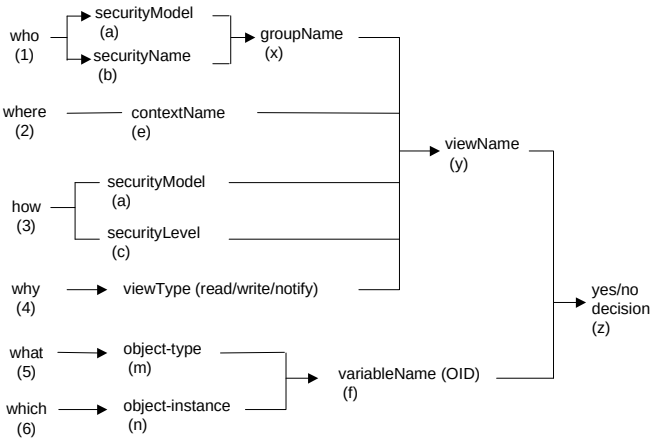

Abbildung 10: Entscheidungsgraph der Zugriffskontrolle des VACM

die sechs folgenden Eingabeparameter:

- `(a) securityModel -- benutztes Sicherheitsmodell`
- `(b) securityName  -- Principal, der den Zugriff möchte`
- `(c) securityLevel -- das Sicherheitslevel`
- `(d) viewType      -- read-, write- or notifiy-view`
- `(e) contextName   -- Kontext, der den 'variableName' enthält`
- `(f) variableName  -- OID des MOs bestehend aus Objekttyp (m) und Objektinstanz (n)`

In diesem Prozeß werden nun die einzelnen Schritte wie in Abbildung 10 dargestellt, abgearbeitet. Exemplarisch soll hier nur der erste Schritt beschrieben werden. Das 'who' (1), das durch das 'securityModel' (a) und den 'securityName' (b) dargestellt wird, wird als Index (a,b) für die 'vacmSecurityToGroupTable' benutzt. Dieser Index beschreibt einen Eintrag in dieser Gruppe, der die Gruppe enthält, die durch den 'groupName' (x) dargestellt wird.

Analog verlaufen dann die weiteren Schritte bis hin zur ja/nein-Entscheidung (z).

3 Anwendungen basierend auf SNMPv3

In den folgenden Abschnitten werden kurz fünf SNMP-Applikationstypen beschrieben die in [LeMS97] erklärt werden.

- Applikationen, die SNMP-Get-, -GetNext-, -GetBulk- und/oder -Set-Anfragen initiieren ('command generators' (Befehlsgeneratoren))
- Applikationen, die auf eine Anfrage antworten ('command responders' (Befehlsbeantworter))
- Applikationen, die Benachrichtigungen generieren ('notification originators' (Benachrichtigungsinitiator))
- Applikationen, die Benachrichtigungen empfangen ('notification receiver' (Benachrichtigungsempfänger))
- Applikationen, die SNMP-Get-, -GetNext-, -GetBulk- und/oder -Set-Anfragen weiterleiten ('proxy forwarder' (Proxy-Applikationen))

3.1 Die Befehlsgenerator-Applikation

Ein Befehlsgenerator initiiert SNMP-Get-, -GetNext-, -GetBulk- und/oder -Set-Anfragen. Er verarbeitet auch die Antworten auf eine generierte Anfrage.

3.2 Die Befehlsbeantworter-Applikation

Ein Befehlsbeantworter empfängt SNMP-Get-, -GetNext-, -GetBulk- und/oder -Set-Anfragen, die an das lokale System gerichtet ist. Die Applikation wird dann die sich eignenden Protokolloperationen unter Benutzung der Zugriffskontrolle ausführen und eine Antwortnachricht generieren. Diese wird an den Initiator der Anfrage zurückgeschickt.

3.3 Die Benachrichtigungsinitiator-Applikation

Ein Benachrichtigungsinitiator überwacht ein System bezüglich spezieller Ereignisse oder Umstände. Er generiert Traps und/oder Informationsnachrichten aufgrund der Ereignisse bzw. Umstände. Ein Benachrichtigungsinitiator muß einen Mechanismus besitzen, der herausfindet, wohin eine Nachricht geschickt werden soll und welche SNMP-Version und Sicherheitsparameter zu verwenden sind. Für diesen Zweck gibt es eine spezielle MIB, die in [LeMS97] beschrieben ist.

3.4 Die Benachrichtigungsempfänger-Applikation

Ein Benachrichtigungsempfänger lauscht, um Benachrichtigungen zu entdecken und eine Antwort darauf zu generieren, sofern die empfangene Nachricht eine 'Inform PDU' enthält.

3.5 Die Proxy-Applikation

Ein Proxy leitet SNMP-Nachrichten weiter[11]. Der Proxy kann dazu benutzt werden, um SNMP-Nachrichten einer Transportart in eine andere Transportart umzuwandeln. Dies beinhaltet die Umwandlung einer SNMP-Anfrage in eine nicht auf SNMP-basierende Anfrage. Zusätzlich bietet der Proxy die Möglichkeit, Werte zusammenzufassen, so daß es möglich ist, entfernt liegende Werte in einem MO zusammenzufassen.

4 Zusammenfassung und Bewertung

In diesem Abschnitt wird nun noch einmal SNMPv3 kurz zusammengefaßt und versucht, die Architektur zu bewerten.

4.1 Zusammenfassung

SNMPv3 ist ein modulares Rahmenwerk. Es umfaßt die Bereiche des Transports sowie die Bereiche des Zugriffsrechts und der Geheimhaltung. Die 'SNMPv3 Working Group' legt dabei viel Wert darauf, so transparent und flexibel wie möglich zu sein. Zudem war ein Grundprinzip, daß soviel wie möglich von den Strukturen des SNMPv2u und SNMPv2* übernommen werden sollte. Letztendlich führte dies zu der Veröffentlichung der Arbeit als 'Proposed Standard RFC' [Mund97].

4.2 Bewertung

SNMPv3 ist eine ausgereifte Rahmenstruktur für die Belange des Netzmanagements. Es ist mit dieser Struktur sowohl möglich, ein kleines (LAN) wie auch ein sehr großes (WAN) Netzwerk zu überwachen und zu managen. Es wurden die Nachteile von SNMPv1, SNMPv2u und SNMPv2* beseitigt und somit ein leistungsfähiges Paket entwickelt. Durch die Modularität bietet sich den Firmen die Möglichkeit, auf ihren bereits existierenden Komponenten aufzubauen und diese gegebenfalls umzurüsten. In bezug auf die Sicherheit der Nachrichten und des Zugriffs auf MOs ist zu sagen, daß dort die Lücken geschlossen wurden. Zusammenfassend läßt sich sagen, daß dieser Ansatz der 'SNMPv3 Working Group' gelungen ist und, bedenkt man die Variationsmöglichkeiten, wohl recht schnell in die Praxis umgesetzt werden wird.

[11] BEACHTE: Die Implementation eines Proxys ist optional!

Literatur

[BlWi97] U. Blumenthal und B. Wijnen. User-Based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3), November 1997.

[CHPW97] J. Case, D. Harrington, R. Presuhn und B. Wijnen. Message Processing and Dispatching for the Simple Network Management Protocol (SNMP), November 1997.

[CMRW96a] J. Case, K. McCloghrie, M. Rose und S. Waldbusser. Introduction to Community-based SNMPv2 (RFC1901), Januar 1996.

[CMRW96b] J. Case, K. McCloghrie, M. Rose und S. Waldbusser. Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2) (RFC1907), Januar 1996.

[CMRW96c] J. Case, K. McCloghrie, M. Rose und S. Waldbusser. Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2) (RFC1905), Januar 1996.

[CMRW96d] J. Case, K. McCloghrie, M. Rose und S. Waldbusser. The Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2), Januar 1996.

[CMRW96e] J. Case, K. McCloghrie, M. Rose und S. Waldbusser. Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2) (RFC1906), Januar 1996.

[DES77] Data Encryption Standard, 1977.

[HaPW97] D. Harrington, R. Presuhn und B. Wijnen. An Architecture for Describing SNMP Management Frameworks, Oktober 1997.

[HeGr93] Mathias Hein und David Griffiths. *SNMPv2 in Theorie und Praxis.* Thomson Publishing. 1993.

[KrBC97] H. Krawczyk, M. Bellare und R. Canetti. HMAC: Keyed-Hashing for Message Authentification, Februar 1997.

[LeMS97] D. Levi, P. Meyer und B. Stewart. SNMPv3 Applications, November 1997.

[McKa94] K. McCloghrie und F. Kastenholz. Evolution of the Interfaces Group of MIB-II (RFC1573), Januar 1994.

[MuCO97] R. Mundy, J. Curran und M. O'Dell. SNMP Version 3 (SNMPv3) Charter, November 1997.

[Mund97] R. Mundy. SNMPv3 Working Group — A View from the Chair, Dezember 1997.

[Rive94] R. Rivest. Message Digest Algorithm MD5, April 1994.

[SFDC90] M. Schoffstall, M. Fedor, J. Davin und J. Case. A Simple Network Management Protocol (SNMP) (RFC1157), Mai 1990.

[SHA95] Secure Hash Algorithm. NIST FIPS 180-1. Veröffentlichung im WWW unter der Adresse
http://csrc.nist.gov/fips/fip180-1.txt (ASCII)
http://csrc.nist.gov/fips/fip180-1.ps (Postscript), April 1995.

[WiMc97] B. Wijnen und K. McCloghrie. View-Based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP), November 1997.

[WiPM97] B. Wijnen, R. Presuhn und K. McCloghrie. Local Processing Model for Version 3 of the Simple Network Management Protocol (SNMP), Mai 1997.

Telecommunications Management Network: Die Q3-Schnittstelle

Hassèn Abdi

Kurzfassung

In diesem Seminar werden die wichtigsten Merkmale des sogenannten Telecommunications Management Network (TMN) behandelt. Das TMN beinhaltet unterschiedliche Aspekte, die in verschiedenen Abschnitten zitiert werden. Im ersten Abschnitt werden die Gründe und den Bedarf nach TMN-Netzwerken kurz beschrieben und anschließend das TMN grob vorgestellt. Im zweiten Abschnitt findet man eine Repräsentation der funktionalen Architektur eines TMN, die durch verschiedene Funktionsblöcke, funktionale Komponenten und Referenzpunkte beschrieben wird. Der dritte Abschnitt stellt die verschiedenen Schichten und die Funktionsverteilung des TMN vor. Die TMN-Architektur ist also in logischen Schichten gegliedert, die im Laufe der Zeit verfeinert werden, um die benötigten Dienste abdecken zu können. Abschnitt 4 behandelt die physikalische Architektur eines TMN, die aus mehreren TMN-Elementen besteht, die wiederum in den im folgenden zitierten TMN-Blöcken enthalten sind. Abchließend wird in Abschnitt 5 die von ITU-T genormte Q3-Schnittstelle betrachtet. Sie stellt ein Interface für das TMN zum Telekommunikationsnetz dar.

1 Einleitung

Die Nachfrage und Bedarf nach Telekommunikationsdiensten sind in den letzten Jahren sehr schnell gestiegen und wachsen ständig weltweit exponentiell, so daß die Telekommunikationsnetzwerke sehr groß und komplex geworden sind [Sell95]. Diese große Ausdehnungen und Verbreitung von Netzwerken machen es sehr schwer, die ausgefallenen und gestörten Komponenten zu lokalisieren und zu manipulieren und die entsprechenden Gegenmaßnahmen einzuleiten. Daher ist eine gezielte Infrastruktur sehr notwendig und muß vorhanden sein, die alle Bereiche der Kundenwünsche von der technischen Realisierung bis zum Betrieb des Netzes abdeckt. Diese Infrastruktur soll durch ein Managementnetzwerk mit Standardprotokollen, Schnittstellen und standardisierter Architektur hergestellt werden. Das Managementnetzwerk wird von der *International Telecommunications Union — Telecommunications Standardization Sector (ITU-T)* als **Telecommunications Management Network (TMN)** bezeichnet. Das TMN benötigt für den Nachrichtenaustausch und die Informationsammlung in Telekommunikationnetzwerken eine bestimmte Schnittstelle an den Netzwerkkomponenten. Die wichtigste genormte Schnittstelle hierbei ist die Q3-Schnittstelle, über die eine Netzleiteinrichtung mit den Netzkomponenten für den Betrieb kommunizieren kann. Das

TMN soll eine Menge von Funktionen für den Betrieb, die Verwaltung und die Instandhaltung des Telekommunikationsnetzes bereitstellen. Diese Funktionen können in zwei Gruppen eingeteilt werden: die Basisfunktionen und die erweiterten Funktionen. Die Basisfunktionen sind:

- Managementfunktionen wie Konfigurationsmanagement, Leistungsmanagement, Fehlerbehandlung, Accounting-Management und Sicherungsfunktionen
- Kommunikationsfunktionen zwischen den verschiedenen Funktionseinheiten
- und Planungsfunktionen, unter anderem auch die physikalische Planung des Netzes und Arbeitsplanung.

Die Basisfunktionen dienen als Grundlage für die Erweiterungsfunktionen, diese sind:

- Dienstmanagement
- Wiederherstellung des Netzes
- Rekonfiguration und
- Bandbreitenzuteilung.

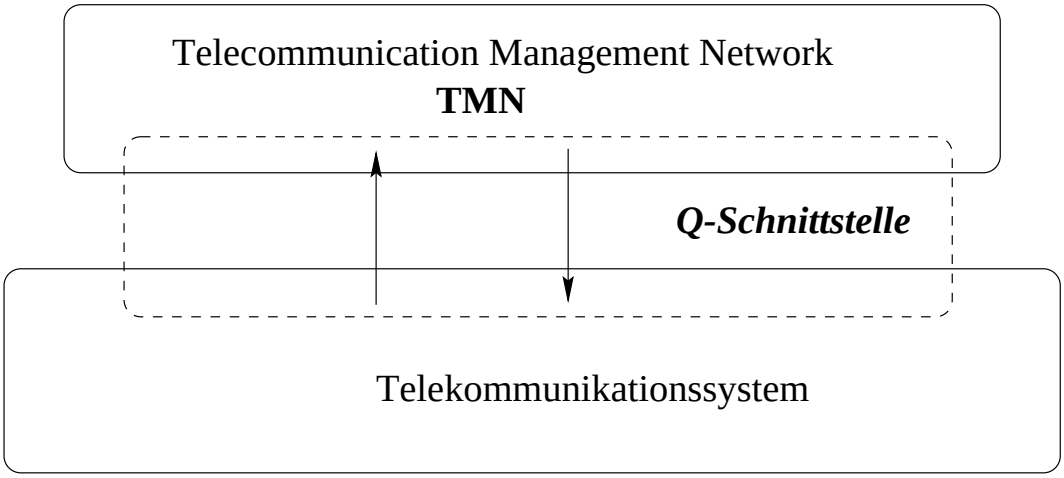

Abbildung 1: TMN, Q-Schnittstelle und Telekommunikationssystem.

Das TMN ist als physikalisch getrenntes Netz zum Management von Telekommunikationsnetzen und -diensten aller Art zu betrachten. Der Gebrauch eines generischen Informationsmodells und standardisierter Schnittstellen für Managementzwecke etabliert ein herstellerunabhängiges Management von Netzen, Netzelementen, Diensten und damit des gesamten Telekommunikationsgeschäfts. Die physikalische Trennung des TMN vom Kommunikationsnetz und -dienst bringt wesentliche Vorteile:

- Das Management wird unabhängiger von der Last im Nutznetz und erzeugt umgekehrt in ihm auch keine Last.
- Informationen über das Netz und/oder den Dienst werden getrennt und somit unabhängig von den Netzeinrichtungen geführt und ermöglichen zu jeder Zeit Abfragen beispielsweise über den Netzzustand.

- Notwendige Arbeiten an den Netzelementen können während der verkehrsschwachen Zeiten (außerhalb der normalen Arbeitszeit) selbständig durchgeführt werden (z.B. Konfigurationen, Auswertung von Verkehrsmessungen u.a.).

Wichtige Prinzipien der TMN-Architektur sind:

- die Verteilungen von Managementaufgaben auf verschiedene Subsysteme im TMN (dezentral);
- der Austausch von Managementinformationen zwischen den Subsystemen;
- der Zugriff auf TMN-Komponenten und Netzelemente über standardisierte Managementschnittstellen und standardisierte Kommunikationsprotokolle;
- der Einbezug bestehender Einrichtungen.

Die TMN-Architektur wird in eine funktionale, informationsbezogene und physikalische Architektur gegliedert. Alle diese Architekturen ergänzen sich im Sinne einer differenzierten Betrachtungsweise des TMN.

2 Die funktionale Architektur

Von seiner funktionalen Seite her ist das TMN als ein eigenständiges Netzwerk zu betrachten. Seine Hauptaufgabe besteht darin, daß es Telekommunikationsnetzwerke verwalten soll. Um seine Aufgaben zu erfüllen, benutzt es in der Tat Teile des Telekommunikationsnetzwerkes. Man kann also sagen, daß manche Teile des TMN ein im Telekommunikationsnetzwerk eingebettetes logisches Netzwerk sein können. Das TMN ist so konzipiert, um die Informationen vom Telekommunikationsnetzwerk und seinen Diensten zu transportieren und zu verarbeiten. Diejenigen Funktionen, die direkt für das Management gebraucht werden, liegen vollständig innerhalb des TMN. Manche andere liegen teilweise außerhalb des TMN, wobei nur deren Aspekte von der TMN-Architektur erfaßt werden, die für das Management erforderlich sind.

Die Funktionalität des Managementsystems kann in drei Blöcke eingeteilt werden, diese sind nämlich der *Betriebssystemfunktionsblock* (*Operation Systems Function*: OSF), der *Vermittlungsfunktionsblock* (*Mediation Function*: MF) und der *Kommunikationsfunktionsblock* (*Data Communication Function*: DCF). Neben diesen Funktionsblöcken besitzt das TMN Teile der *Netzwerkelementfunktionsblöcke* (*Network Element Function* : NEF), der *Bedienstationsfunktionsblöcke* (*Workstation Function* : WSF) und der *Q-Adapter-Funktionsblöcke* (*Q Adaptor Function*: QAF) des Telekommunikationsnetzwerkes.

2.1 Die Funktionsblöcke des TMN

Im folgenden sind die TMN-Funktionsblöcke mit ihren Funktionen aufgeführt. In Abbildung 2 ist die TMN-Architektur mit den Blöcken gezeigt.

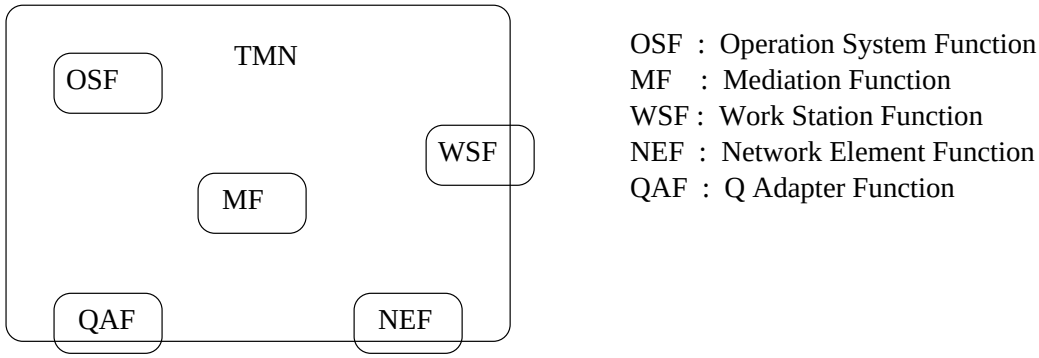

Abbildung 2: Die TMN-Funktionsblöcke.

- Der Operation Systems Function Block (OSF) beinhaltet die Netzleitfunktionen und verarbeitet Managementfunktionen mit Bezug zum Telekommunikationsnetz. Die Managementfunktionen sind etwa Beobachtung, Koordinierung und Kontrolle von Telekommunikationsfunktionen oder von TMN-Funktionen selbst. Zur Unterstützung der Kommunikationsschichten werden von der ITU-T vier OSF-Blöcke definiert. Die unterstützten Schichten sind die *Business Management Layer*, die *Service Management Layer*, die *Network Management Layer* und die *Network Management Element Layer*.

- Der Network Element Function Block (NEF) enthält die Netzelementfunktionen. Diese sind z.B. Vermittlungsfunktionen, die für den eigentlichen Zweck der Telekommunikation implementiert wurden. Sie sind ein Gegenstand der Managementfunktionen des TMN und sie befinden außerhalb des TMN. Die NEF kommunizieren mit dem TMN zum Zwecke der eigenen Beobachtung und Kontrolle. Sie unterstützen also das TMN in dessen eigenen Aufgaben und liegen innerhalb des TMN.

- Der Q Adapter Function Block (QAF) wir dazu benötigt, daß er nicht TMN-konforme Instanzen an TMN-Funktionsblöcke anschließt, die sich ähnlich zu den NEF und SOF verhalten. Als weitere Aufgabe übersetzen (syntaktische Umsetzung) die QAF die Managementinformationen zwischen einem TMN-Referenzpunkt und einem Referenzpunkt außerhalb des TMN.

- Der Work Station Function Block (WSF) ist für den TMN-Benutzer ein benutzerfreundliches User Interface zur Interpretation der Managementinformationen, die innerhalb des TMN liegen.

- Der Mediation Function Block (MF) ist dazu implementiert, um die Kommunikationen zwischen den verschiedenen TMN-Funktionsblöcken miteinander über verschiedene Schnittstellen und Referenzpunkte zu ermöglichen. Dieser Block kann als Umsetzfunktion betrachtet werden. Als *Mediation Functions* kommen beispielsweise Informationstransportfunktionen wie Protokollkonvertierung, Routing und Informationsverarbeitungsfunktionen wie Ausführung, Speicherung und Filterung vor.

2.2 Die funktionalen Komponenten

Zur Ergänzung der TMN-Funktionsblöcke ist noch eine Menge von funktionalen Komponenten implementiert worden. Diese Komponeneten sind:

- Die Managementapplikationsfunktion (*Management Application Function*) MAF implementiert die TMN-Managementdienste und die diese Dienste unterstützenden TMN-Managementfunktionen. Sie agieren je nach Managementaufgabe entweder als Manager oder als Agent. Diese Funktionen werden in Abhängigkeit des TMN-Funktionsblocks, der sie enthält, als MF-MAF, OSF-MAF, NEF-MAF oder QAF-MAF bezeichnet. Die NEF-MAF und die QAF-MAF haben die Rolle eines Agenten, die MF-MAF und OSF-MAF unterstützen beide Rolle (Manager und Agent). Die OSF-MAF spielt eine zentrale Rolle, da sie über die größte Managementintelligenz verfügt. Sie wertet hereinkommende Informationen aus (z.B. Alarme), verarbeitet sie weiter(z.B. Alarmkorrelation, -statistiken) und leitet weitere Reaktionen ein (z.B. Benachrichtigung von anderen Managementinstanzen, automatische Rekonfiguration).

Funktionsblöcke	Funktionale Komponente	Assoziierte MCF
OSF	MIB, OSF-MAF, (A/M), HMA	MCFx, MCFq3, MCFf
(darunterliegende) OSF	MIB, OSF-MAF, (A/M), ICF, HMA	MCFx, MCFq3, MCFf
WSF	PF	MCFf
NEFq3	MIB, NEF-MAF (A)	MCFq3
NEFqx	MIB, NEF-MAF (A)	MCFqx
MF	MIB, MF-MAF (A/M), ICF, HMA	MCF3, MCFqx, MCFf
QAFq3	MIB, QAF-MAF, (A/M), ICF	MCFq3, MCFm
QAFqx	MIB, QAF-MAF, (A/M), ICF	MCFqx, MCF

Tabelle 1: Relationen zwischen TMN-Funktionen und funktionalen Komponenten

- Die Managementinformationsbasis (*Mangement Information Base*) MIB repräsentiert den Satz der *Managed Objects* in einem Managementsystem und ist der konzeptionelle Aufbewahrungsort von Managementinformationen.

- Die Informationskonvertierungsfunktion (*Information Conversion Function*) ICF wird zur Übersetzung des Informationsmodells an einer bestimmten Schnittstelle in ein Informationsmodell an einer anderen Schnittstelle benötigt, was sich beispielsweise in der Konvertierung von Objektdarstellungen auswirkt. Diese Übersetzung kann syntaktischer oder semantischer Art sein.

- Die Darstellungsfunktion (*Presentation Function*) PF beinhaltet alle Funktionenen zur benutzerfreundlichen Darstellung sowie zur Modifikation von Managementinformation. Sie führt generelle Operationen aus zur Übersetzung von Managementinformationen aus dem TMN-Informationsmodell in ein darstellbares Format an der Benutzerschnittstelle. Diese Übersetzung kann auch in der umgekehrten Richtung erfolgen.

- Die Mensch-Maschine-Anpassung (*Human Machine Adaptation*) HMA führt die Konvertierung des MAF-Informationsmodells in das PF-Informationsmodell durch. Anschaulich ausgedrückt verpackt die HMA Daten in Fenster oder Masken, fügt Information zur eindeutigen Interpretation der Information aus der PF hinzu oder reorganisiert diese Information. Zudem unterstützt die PF die Feststellung der Benutzeridentität (Authentifizierung) und die Zuteilung seiner Benutzerrechte (Autorisierung).

- Die Nachrichtenaustauschfunktion (*Message Communication Function*) MCF muß in allen TMN-Funktionsblöcken enthalten sein, die über eine physikalische Schnittstelle verfügen. Die MCF macht TMN-Funktionsblöcke kommunikationsfähig, also fähig zum Austausch von Managementinformation. Die MCF beinhaltet einen Protokollstack zum Anschluß der TMN-Funktionsblöcken an die Datenkommunikationsfunktionen DCF (*Data Communication Function*). DCF-Funktionen werden von den anderen TMN-Funktionsblöcken zum Austausch von Managementinformationen benötigt. Sie haben als primäre Rolle die Bereitstellung eines Transportverfahrens mit Routingfunktionen sowie Funktionen zur Zwischenspeicherung und Zusammenarbeit der Daten zwischen den TMN-Funktionsblöcken. In der Tabelle 1 sind die Beziehungen zwischen den TMN-Funktionsblöcken und den funktionalen Komponenten zusammengefaßt.

2.3 Die Referenzpunkte

Für den Informationsaustausch zwischen den nicht-überlappenden TMN-Funktionsblöcken wurden die sogenannten TMN-Referenzpunkte als konzeptionelle Punkte definiert. Die TMN-Referenzpunkte sind in folgenden Klassen klassifiziert: q-, f-, x-, g- und m-Referenzpunkte. Die Referenzpunkte q, f und x liegen innerhalb bzw. am Rand, g und m sind aber außerhalb des TMN.

	NEF	OSF	MF	QAFq3	QAFqx	WSF	Non-TMN
NEF		q3	qx				
OSF	q3	q3,x	q3	q3		f	
MF	qx	q3	qx		qx	f	
QAFq3		q3					m
QAFqx			qx				m
WSF		f	f				g
Non-TMN				m	m	g	

Tabelle 2: Die durch TMN-Referenzpunkte hergestellte Relation zwischen TMN-Funktionsblöcken.

- x-Referenzpunkte sind zwischen den OSF verschiedener TMN zur Verbindung eingelegt und sie erlauben einen Einblick bzw. einen Eingriff in eine fremde Managementdomäne.

- f-Referenzpunkte treten immer in Zusammenhang mit der WSF auf, welche die Bedienbarkeit eines TMN ermöglicht.

- q-Referenzpunkte verbinden, zum eigentlichen Management, die TMN OSF-, MF-, NEF- und QAF-Funktionsblöcke entweder direkt oder über DCF. So werden also Verbindungen zwischen Manager und Agent oder zwischen zwei Managern ermöglicht.
- g-Referenzpunkte stellen die Benutzerschnittstelle und liegen außerhalb des TMN.
- m-Referenzpunkte erlauben den Anschluß proprietärer Netzelemente oder proprietärer Managementinstanzen und liegen außerhalb des TMN. Sie stellen eine Einbindung herstellerspezifischer Einrichtungen in ein TMN sicher.

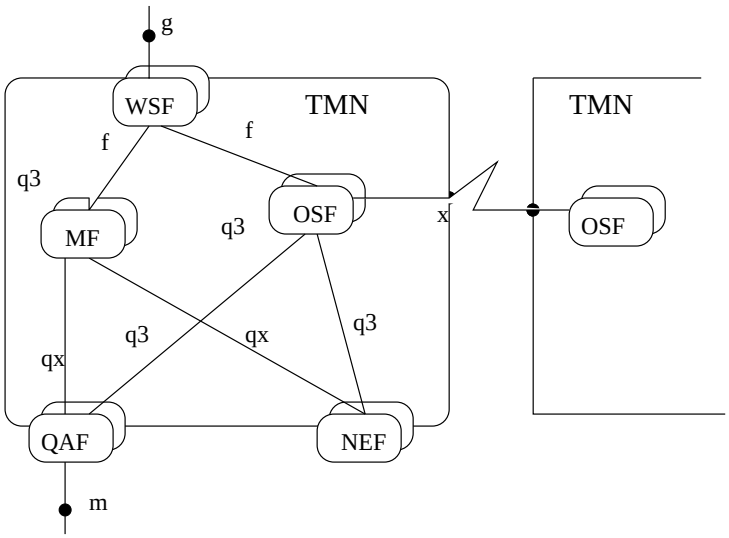

Abbildung 3: Referenzpunkte und Funktionsböcke des TMN.

3 Die Schichten und die Funktionsverteilung des TMN

3.1 Logische geschichtete Architektur

Ähnlich wie das Telekommunikationssystem, das in sieben ISO/OSI-Schichten geteilt ist, bildete sich früh das Bedürfnis heraus, bestimmte Ressourcen des TMN zu Domänen zusammenzufassen und unter verschiedenen Gesichtspunkten zu gliedern. Diese Domänen können ausgehend von den OSI-Management-Funktionen wie folgt aufgeteilt werden:

- Fault Management.
- Accounting Management.
- Configuration Management.
- Performance Management.

- Security Management.

Die Domänen können unterschiedliche Aspeke von identischen Objekten beinhalten. Im TMN-Kontext bildet dann der Satz der *Managed Objects* zusammen mit dem Manager eine Managementdomäne. Wenn man diese geteilten Managementdomänen als Schichten betrachtet, so spricht man von der sogenannten logischen Schichtenarchitektur LLA (*Logical Layered Architecture*). Nach diesem Kozept der Domänen kann man jeder Netzleitfunktion OSF einen bestimmten Managementbereich zuordnen, den man mit dem Begriff F-Domäne bezeichnet.

3.2 Verfeinerung der Netzleitfunktion

Die Verfeinerung der OSF ist heutzutage sehr notwendig, da die Managementsysteme meistens zwar einige Managementprobleme lösen, sie aber nur eine Teilansicht eines Managementkomplexes geographischer oder technischer Art geben können. Diese Sicht ist also beschränkt, da sie nur ein Teilproblem lösen kann. Diese geringe Beschränkungen werden dadurch gelöst, indem man eine weitere geschickte Verfeinerung der OSF in folgende Bereiche eingeführt hat (siehe auch Abbildung 4):

- B-OSF (Business Operation System Function)
- S-OSF (Service Operation System Function)
- N-OSF (Network Operation System Function)
- E-OSF (Element Management Operation System Function)

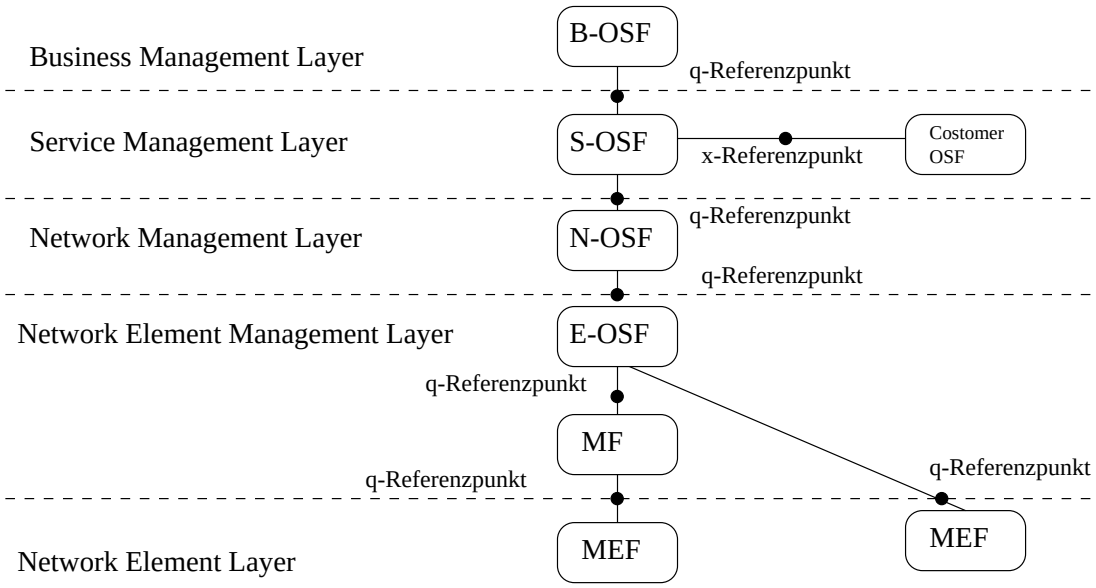

Abbildung 4: TMN-Managementschichten.

Die B-OSF stellt eine globale Geschäftssicht dar, trifft Vereinbarungen mit anderen Betreibern und behandelt finanzielle Aspekte. Die S-OSF behandelt Dienstaspekte, stellt die Dienstqualität sicher und wird bei Bedarf durch die Customer-OSF ergänzt,

die eine Dienstsicht für Kunden ermöglicht. Die N-OSF ist für das Netzmanagement verantwortlich und ermöglicht eine globale Netzsicht. Schließlich stellt die E-OSF eine detaillierte Netzsicht zur Verfügung inklusive Überwachung aller Netzelementfunktionen NEF, auf welche direkt oder über die Umsetzungsfunktion MF zugegriffen wird. Bei TMN-Systemen besteht die Möglichkeit, daß Managementschichten übersprungen werden können. Z.B. kann die B-OSF sich einen sofortigen Überblick über die aktuelle Situation im Netz verschaffen und, wenn es notwendig ist, einen direkten Eingriff in die E-OSF ausführen.

Das Überspringen von Managementschichten wurde mit "Bypass of layers" bezeichnet. Dieses Prinzip wurde sehr viel diskutiert, da es Vor- und Nachteile aufweist. Bei einigen Fällen besteht die Möglichkeit, daß die B-OSF auf E-OSF, die mit anderen Aufträgen befaßt ist, eingreift. Dies verzögert die Reaktion der E-OSF auf die B-OSF-Aufträge in einem Moment, in dem die B-OSF ein flexibles und schnelles Reagieren und Eingreifen braucht. Das Business Management sowie das Service Managment stehen bei einem im modernen Telekommunikationsmanagement geforderten, umfassenden Netzmanagement mit Kundenausrichtungen im Mittelpunkt und sind in den oberen generischen Managementschichten der OSF-Hierarchie-Stufen angesiedelt. Die Netzelementfunktion, die eine spezifische Managementschicht bildet, ist als Mittel zum Zweck ganz unten zu finden. Im Zusammenhang mit den sogenannten Mehrwertdiensten VAS (Value-Added Service), die Zusatzdienste in bestehenden Telekommunikationsnetzen realisieren, sind die B-OSF und die S-OSF von Bedeutung. Es folgen daher die Definitionen der Business- und der Dienstmanagementschicht mit den zugehörigen OSF.

- Die Business-Managementschicht besitzt Verantwortung für das ganze Unternehmen und ist die Schicht, in der Vereinbarungen zwischen den Betreibern getroffen werden. Diese Schicht bringt normalerweise Zielsetzungen hervor, aber sie kann für solche Fälle der zentrale Punkt für Handlungsbedarf werden, wo ausführende Handlungen gewünscht werden. Diese Schicht ist Teil des Gesamtmanagements eines Unternehmens und viele Weckselwirkungen mit anderen Managementsystemen sind nötig.

Ähnlich verhält sich mit der Definition der B-OSF:

- Einige TMN-Implementationen können die B-OSF enthalten, die mit dem ganzen Unternehmen befaßt ist und eine Gesamtkoordination des Unternehmens ausführt.

Die Dienstmanagementschicht wird in der ITU-T-Empfehlung wie folgt definiert:

- Die Dienstmanagementschicht ist für die vertragsmäßigen Dienstaspekte verantwortlich, die bestehenden oder potentiellen, neuen Kunden zur Verfügung gestellt werden. Sie hat sechs prinzipielle Rollen:

  1. Kundenkontakt und Zusammenarbeit mit anderen Verwaltungen;
  2. Zusammenarbeit mit anderen Dienstanbietern;
  3. Zusammenarbeit mit der Netzmanagementschicht;
  4. Unterhalt von statistischen Daten;

5. Zusammenarbeit mit der Business-Managementschicht;
6. Aktionen zwischen den Diensten.

Die S-OSF ist mit Dienstpaketen eines oder mehrerer Netze befaßt und führt normalerweise eine Schnittstellenfunktion (über den Referenzpunkt x) zum Kunden hin aus.

4 Die physikalische Architektur

In den letzten beiden Kapiteln wurden die funktionale TMN-Architektur mit allen zugehörigen TMN-Funktionen und die TMN-Informationsarchitektur inclusive der LLA beschrieben [Diet96]. Als drittes Element in den Sichtweisen der TMN-Architektur ist nun die physikalische TMN-Architektur zu beschreiben. Sie besteht aus mehreren TMN-Elementen, die wiederum die beschriebenen TMN-Funktionsblöcke enthalten.

4.1 Die TMN-Elemente

Entsprechend der TMN-Funktionsblöcken, die verschiedene Funktionen enthalten, existieren auch die folgenden TMN-Elemente:

- die Netzführungseinrichtung (*Operation System*) OS;
- das Datenkommunikationsnetz (*Data Communication Network*) DCN;
- die Bedienstation (*Work Station*) WS;
- die Umsetzungseinrichtung (*Mediation Device*) MD;
- das Netzelement (*Network Element*) NE;
- der Q-Adapter (*Q Adapter*) QA.

Die Netzleitfunktion OS wird meistens mit Betriebssystem bezeichtnet, da sie die zentrale TMN-Intelligenz darstellt. Die OS kommunizieren über standardisierten Schnittstellen Q3 oder Qx mit den Netzelementen. Die Kommunikation zwischen den OSs und den NEs erfolgt über das sogenannte Datenkommunikationsnetz DCN, das z.B. als ein X.25-Netz vorkommt. In manchen Fällen trägt die Umsetzungseinrichtung zur Kommunikation bei und hilft den OSs in bestimmten Situationen. Die QA, die auf der Seite des TMN eine Standardschnittstelle Q3 oder Qx anbieten, dienen dazu, daß sie bestehende Einrichtungen z.B. proprietäre Netzelemente oder weitere Netzführungszentralen mit herstellenspezifischen Subnetzen zum TMN verbinden. Ein OS kann auch mit einem Partner-OS, einer anderen Managementdomäne eines anderen TMN Managementinformationen über die X-Schnittstelle austauschen. Die Bedienung der OS wird mit Bedienstationen WS sichergestellt, die über die F-Schnittstelle auf die OS oder die MD zugreifen.

In der Tabelle 3 sind alle möglichen Standard-Funktionen (M: Standard-Relation) und optionale Funktionen (O: optionale Relation) gezeigt und der Zusammenhang der TMN-Elemente zu den TMN-Funktionsblöcken deutlich gemacht. Die TMN-Funktionsblöcke wurden in der Abbildung 3 dargestellt. Bild 5 zeigt eine vereinfachte physikalische TMN-Architektur und die vorhandenen TMN-Schnittstellen, die im folgenden Abschnitt beschrieben werden.

	NEF	MF	QAF	OSF	WSF
NE	M	O	O	O	O
MD		M	O	O	O
QA			M		
OS		O	M	M	O
WS					M

Tabelle 3: Die Relationen zwischen TMN-Elementen und TMN-Funktionsblöcken

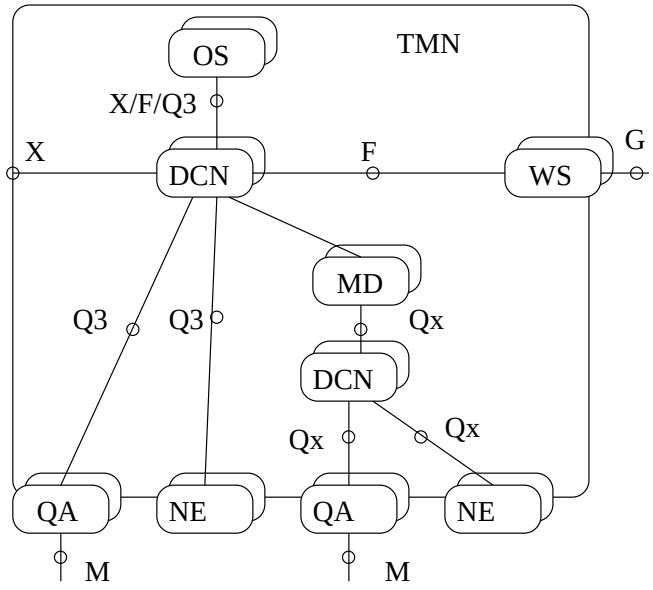

Abbildung 5: Beispiel für eine vereinfachte phusikalische Architektur.

4.2 Die TMN-Schnittstellen

Die bis jetzt bei der TMN-Architektur vorhandenen Schnittstellen sind die folgenden: Qx, Q3, X und F und sind in der Tabelle 4 gezeigt. Jede TMN-Schnittstelle ermöglicht einen Zugang zu den verschiedenen TMN-Elementen, deren Anzahl auf der ersten Blick sehr groß erscheint. Sie folgt aus der Anzahl der möglichen TMN-Funktionsblöcken in den TMN-Elementen, aus denen sich eine entsprechende Zahl von Referenzpunkten gibt. Es besteht eine 1:1-Verbindung zwischen den TMN-Referenzpunkten und den TMN-Schnittstellen. Der Gebrauch dieser Schnittstellen ist notwendig, wenn bei einer physikalischen TMN-Implementierung externe Verbindung zu den TMN-Elementen innerhalb oder außerhalb des TMN erforderlich sind.

Die Q3-Schnittstelle, die als Schnittstelle zwischen OS und MD, NE und QA vorhanden ist, ist am weitesten fortgeschritten und in den ITU-T-Empfehlungen Q.811 und Q.812 definiert (Siehe Kapitel 5). Die anderen TMN-Schnittstellen wie X und F sind ihre Studien noch nicht ganz abgeschlossen. Die Qx-Schnittstelle lokalisiert sich zwischen MD und NE oder QA und sie wurde als eine (abgemagerte) Q3-Schnittstelle ohne deren volle Funktionalität definiert. Ähnlich lassen sich die X- und die F-Schnittstelle beschreiben. Obwohl die F-Schnittstelle ihre eigene ITU-T-Empfehlung besitzt, bleibt sie jedoch sehr allgemein gehalten und gibt keine genaue Definition der OSI-Dienstelemente oder der verwendeten Protokolle. Da die TMN-Elemente WS und OS auf Elementen der Informationstechnik basieren, die über eigene interne Schnittstellen verfügen, ist der

	Qx	Q3	X	F
NE	<O	O	O>	O
MD	<O	O>	O	O
QA	<O	O>	O	O
OS	<O	O	O>	O
WS				M

Legende
M mandatory (muß vorhanden sein)
O optional (kann vorhanden sein)
<...> eine der Relationen in eckiger Klammer muß vorhanden sein

Tabelle 4: Die Relationen zwischen TMN-Elementen und TMN-Schnittstellen

F-Schnittstelle keine große Hoffnung auf eine baldige Implementierung zu geben. Zur X-Schnittstelle gibt es bislang nur Arbeitspapiere, die in der nächsten Zeit zu einer eigenen Empfehlung führen werden. Es kann davon ausgegangen werden, daß die X-Schnittstelle eine gewisse Ähnlichkeit zur Q3-Schnittstelle haben wird, allerdings mit wesentlich stärker ausgeprägten Sicherheitsmechanismen, da die X-Schnittstelle zwischen zwei verschiedenen TMN-Managementdomänen liegt.

5 Die Q3-Schnittstelle

Seit ein paar Jahren bemühen sich viele internationale Institute für die Standardisation der sogenannten Q3-Schnittstelle. Erst im Jahr 1996 wurden Standardmodelle von den Instituten ITU-T und ETSI (European Telecommunications Standardization Institute) für die Q3-Schnittstelle veröffentlicht [Pete96]. Die von der ITU-T veröffentlichten Modelle unterstützen generelle Managementfunktionen, die spezielle Aspekte für das Management von Vermittlungsstellen haben. Von der Seite der ETSI wurden einige stabile Modelle ausgegeben. Alle diese Modelle können bis jetzt nur wenige Aspekte der Vermittlungsstellen unterstützen und somit stellen die entwickelten Q3-Modelle nur eine abstrakte und generische Sicht eines Netzwerkelements (NE) dar, und es kann von denen nicht erwartet werden, daß sie alle spezifische Merkmale einer Vermittlungsstelle abdecken. Alle diese Q3-Modelle haben eine abstrakte Managementsicht gemeinsam, die in drei verschiedene Funktionsebenen aufgeteilt werden kann: Teilnehmermanagement, Verkehrsmanagement (Traffic Management) und das Managemet von Systeme, die über eigene Ressourcen verfügen.

- Im Kontext der Funktionsebene des Teilnehmermanagements ist der Kunde ein Benutzer von Telekommunikationsdiensten und die Dienstgeber sind die Besitzer der Informationen, die mit dem Kunden assoziiert sind. Diese Informationenen sind etwa Kundenname, Accountnummer und Kundendienst und werden von den Q3-Objektmodellen direkt mit der Bereitstellung der Dienste (Informationstransfer, Datenvermittlung) für den Kunden in Zusammenhang gebracht.

- Die Q3-Objektmodelle für das Verkehrsmanagement beschreiben die Austauschfunktionen für das Management von Fernsprechverbindungen des öffentlichen Fernsprechsystems PSTN (Public Switched Telephone Network) und des ISDN

TNM	TM	SRM
PSTN-Dienstbereitstellung ISDN-Dienstbereitstellung CENTREX-Management (CENTRal EXchange) Leitungstests	Rufwegewahl Verkehrskontrolle Verkehrsmessung Leistungsüberwachung	Alarmüberwachung Ereignisbehandlung Logging Scheduling V5-Schnittstellenmanagement Ressourcennutzungs- überwachung Sicherheitsmanagement OSI Schichten-Management Software-Management

Legende TNM: Teilnehmermanagement,
TM: Traffic Management,
SRM: Systemressourcenmanagement.

Tabelle 5: Die von Q3 abgedeckten Funktionsebenen

(Integrated Service Digital Network). Die Ziele des Verkehrsmanagements bestehen darin, daß soviele Anrufe wie möglich ausgeführt werden können, wobei gleichzeitig die vorhandenen Ressourcen effizient ausgenutzt werden. Die Bestimmung der Informationsquelle und -senke sowie Wegwahl und Behandlung von Ausnahmesituationen sind weitere wichtige Aufgaben, die von diesen Funktionen erledigt werden.

- Die anderen Ebenen des Managements von Systemen, wo die Q3-Objektmodelle verfügbar sind, sind die V5-Interfacekonfiguration (genormte Schnittstelle, welche die Konfiguration, koordinierte Bereitstellung und das Starten der V5-Schnittstellen und die Überwachung der Vermittlung von V5-Informationskanälen enthält), Fehlermanagement (Fault Management), Nutzungsüberwachung, Alarmüberwachung, Software- und OSI-Schichtenmanagement.

Die Gesamtheit dieser spezifischen Q3-Objetmodelle für Vermittlungsstellen basiert auf generischen Managementinformationen und wurde in den ITU-T X.700 Empfehlungen spezifiziert. Von diesen Basisstandards besteht die Möglichkeit, daß andere Objektklassen für Vermittlungsstellen abgeleitet werden können.

6 Ausblick

Die Q3-Schnittstelle steht noch unter Entwicklung und sie wird zur Zeit von der ITU-T und ETSI erforscht. Die ITU-T hat eine Arbeit über ein Objektmodell für die ATM-Switching NEs gestartet und sollte bald diese Arbeit vorstellen. Von der Seite der ETSI wird eine Arbeit für das sogenannte *Charging Management* angegangen. Da einige wichtige Funktionalitäten innerhalb der Verwaltung von Vermittlungsstellen fehlen, sollte diese Schnittstelle weiterentwickelt und standardisiert werden.

Literatur

[Diet96] T. Dietz. Telecommunication Management Network. In S.Dresler, G. Schäfer und H. Wiltfang (Hrsg.), *Netzwerkmanagement und Hochgeschwindigkeitskommunikation XIV*, Nr. 23/96 der Internen Berichte, S. 1–16. Fakultät für Informatik, Universität Karlsruhe, Karlsruhe, 1996.

[Pete96] W. J. Petermüller. Q3 Object Models for the Management of Exchanges. *IEEE Communications Magazine* 34(3), März 1996, S. 48–60.

[Sell95] R. Sellin. *TMN – Die Basis für das Telekom-Management der Zukunft.* Nr. 15 der TTK (Taschenbuch der Telekommunikation). R. v. Decker's Verlag G. Schenk, Heidelberg. 1995.