Lehrstuhl für Kommunikationsnetze

Technische Universität München

Serverarchitektur zur netzunabhängigen Dienststeuerung in heterogenen Kommunikationsnetzen

Wolfgang Kellerer

Lehrstuhl für Kommunikationsnetze

Serverarchitektur zur netzunabhängigen Dienststeuerung in heterogenen Kommunikationsnetzen

Wolfgang Kellerer

Vollständiger Abdruck der von der Fakultät für
Elektrotechnik und Informationstechnik der Technischen Universität München
zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr.-Ing. Kurt Antreich
Prüfer der Dissertation:	1. Univ.-Prof. Dr.-Ing. Jörg Eberspächer
	2. Univ.-Prof. Dr. rer. nat. Dr. rer. nat. habil. Manfred Broy

Die Dissertation wurde am 13.08.2001 bei der Technischen Universität München
eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik
am 07.01.2002 angenommen.

Vorwort

Diese Arbeit ist im Laufe meiner Tätigkeit als wissenschaftlicher Assistent am Lehrstuhl für Kommunikationsnetze an der Technischen Universität München entstanden. Während dieser Zeit konnte ich neben vielfältigen Aufgaben in Forschung und Lehre als Projektleiter eines interdisziplinären Teams im Forschungsverbund FORSOFT wertvolle Erfahrungen sammeln.

Diese Interdisziplinarität prägte auch meine wissenschaftliche Arbeit auf dem Gebiet der Informations- und Kommunikationsdienste, das an einer Schnittstelle der Elektrotechnik zur Informatik liegt. Da sich der Nutzen eines Kommunikationssystems für die Teilnehmer über die angebotenen Dienste definiert, sind außerdem betriebswirtschaftliche Fragestellungen besonders interessant. Die vorliegende Arbeit spannt einen Bogen über verschiedene Teilaspekte der genannten Disziplinen.

Mein besonderer Dank gilt meinem Doktorvater Prof. Dr. Jörg Eberspächer, der durch wertvolle Diskussionen, konstruktive Anmerkungen und seine freundschaftliche Unterstützung in allen Phasen wesentlich zum Gelingen der Arbeit beigetragen hat.

Prof. Dr. Manfred Broy aus der Fakultät für Informatik möchte ich meinen herzlichen Dank für die Übernahme des Zweitgutachtens aussprechen.

Dank an alle Kolleginnen und Kollegen am Lehrstuhl. Sie haben entscheidend dazu beigetragen, daß diese Arbeit in einer kreativen und freundschaftlichen Atmosphäre entstehen konnte. Insbesondere gilt mein Dank meinem Projektkollegen Peter Sties, der mir während der gesamten Entstehung der Arbeit ein hilfreicher Diskussionspartner war. Darüber hinaus danke ich allen Kollegen aus der *Multimedia Services Group*, insbesondere Andrea Bör für den stets kreativen Gedankenaustausch in unserem gemeinsamen Forschungsumfeld. Wertvolle Diskussionen habe ich außerdem meinem Zimmerkollegen Josef Glasmann zu verdanken.

Ebenso möchte ich allen Studenten danken, die sich mit viel Einsatz an der prototypischen Erprobung der Systemkonzepte beteiligt haben.

Mein herzlichstes Dankeschön gilt meiner Frau Katharina, die mit ihrem großen Verständnis und ihrer Geduld wesentlich zum Gelingen dieser Arbeit beigetragen hat.

Fürstenfeldbruck, im August 2001 *Wolfgang Kellerer*

Kurzfassung

Die Kommunikationsinfrastruktur ist derzeit von einer wachsenden Heterogenität der Netze und der Teilnehmerendgeräte geprägt. Dies stellt insbesondere für neue Dienstanbieter, die ihre Informations- und Kommunikationsdienste für möglichst viele Teilnehmer zugänglich machen wollen, eine Herausforderung dar. Eine weitreichende Nutzbarkeit von Diensten erfordert einen Dienstzugang und eine Dienststeuerung, die unabhängig von einer spezifischen Netzinfrastruktur sind. Die Schlüsselkomponente für eine derartige netzunabhängige Dienststeuerung ist eine einheitliche Plattform, die die Konvergenz heterogener Netze auf Dienstebene verwirklicht. Von den bestehenden Dienstarchitekturen wird die Entkopplung von Dienstebene und Netzebene nur unzureichend erfüllt. Auch neue Ansätze berücksichtigen derzeit nur Teilaspekte oder erfordern eine spezielle Infrastruktur, z.B. eine Middleware, in den Netzelementen und in allen Endgeräten.

In der vorliegenden Arbeit wird eine neuartige Architektur für eine Serverplattform beschrieben, die es erlaubt, Informations- und Kommunikationsdienste unabhängig von Netzen und Teilnehmerzugängen zu steuern. Darüber hinaus können Dienste übergreifend über heterogene Kommunikationsnetze angeboten werden. Eine Anpassungsschicht im Server entkoppelt die Dienststeuerung von den Netzen und deren spezifischer Signalisierung. Die Beschreibung der Architektur erfolgt aus den verschiedenen Sichten eines entwickelten Systemmodells. Sie umfaßt die Struktur und das Verhalten von Steuerungskomponenten, ein Konzept zur Dienstprogrammierung, ein neues Signalisierungsprotokoll und Adaptoren für den Dienstzugang und die Kommunikationssteuerung.

In der Dienstarchitektur werden die drei Steuerungsbereiche Teilnehmer-, Dienst- und Kommunikationssteuerung unterschieden.

Die Teilnehmersteuerung realisiert den Zugang der Teilnehmer zum Server und übernimmt die Verwaltung persönlicher Profile. Eine XML-basierte Serverkomponente erlaubt einen einheitlichen Dienstzugang, der Informationen zum Beispiel im HTML-, WML- oder VXML-Format, unabhängig von den Darstellungsfähigkeiten des Endgerätes ausgeben kann.

Die Dienststeuerung selbst basiert auf einem abstrakten, objektorientierten Multimedia-Session-Modell und ist unabhängig von teilnehmer- und netzspezifischen Daten. Für die Programmierung der Dienstlogik wird eine XML-basierte Notation vorgeschlagen.

Es ist die Aufgabe der Kommunikationssteuerung, die Dienstbeschreibung für jeden Dienst auf eine konkrete Kommunikationsbeschreibung abzubilden. Die Adaptoren der Anpassungsschicht setzen die Server-interne Signalisierung auf Netz-APIs, wie z.B. Parlay, um. Zur Unterstützung der Kommunikationssteuerung bei der Auswahl geeigneter Adaptoren wurde eine zentrale Ressourcen-Verwaltung entwickelt, die nach dem Service Discovery-Prinzip arbeitet.

Für die Signalisierung innerhalb der Server-Architektur wurde das Session Control Protocol basierend auf dem Session Initiation Protocol der IETF spezifiziert. Neben rein funktionalen Gesichtspunkten wurde diese Wahl von der steigenden Bedeutung IP-basierter Signalisierung in der Standardisierung zukünftiger Kommunikationssysteme motiviert (z.B. bei UMTS). Neu ist insbesondere die Verarbeitung der zunächst abstrakten Dienstbeschreibung, die beim Durchlauf durch die Steuerungskomponenten der Dienstarchitektur schrittweise detailliert wird. Den Netzadaptoren werden Informationspfadbeschreibungen mit konkreten QoS-Parametern zur Einrichtung in den Netzen übergeben.

Wesentliche Teile der Serverarchitektur (Dienstzugang, Teile des Signalisierungsprotokolls, Ressourcen-Verwaltung) wurden prototypisch realisiert.

Summary

Today's network infrastructure is faced with an emerging heterogeneity of fixed and mobile networks and a variety of access devices. This is especially challenging for new service providers, who intend to offer their information and communication services to a mass of users. A network-spanning service usage needs a service access and a service control that are independent of any specific network infrastructure. The key component for this network independent service control would be a common platform for the development and provisioning of communication services to allow network convergence on a service layer. The decoupling of service and network layer is missing in conventional service architectures. Even new approaches only provide solutions to some problems, e.g. the standardization of a network-API without defining service architecture components. Others rely on a special kind of infrastructure, e.g. a complex middleware, which is needed in all network components and user terminals.

Our work describes a novel architecture for a server platform, which allows the control of information and communication services independent of networks and user access. Furthermore services can be realized spanning multiple, heterogeneous networks. An adaptation layer decouples the service control from the networks and their specific signaling protocols. The architecture is specified using an abstract system model. It contains functional components structured by three control areas, a concept for service programming, a new signaling protocol, and adaptors for service access and communication control.

Within the service architecture we separate the three areas user control, service control, and communication control. This decomposition concept emphasizes the user management, which becomes very important in a highly personalized service environment. To provide service access that is independent of the presentation characteristics of the terminals an XML-based web-server component has been developed that allows information output in different formats (e.g. HTML, WML, or VXML).

The service control is based on an object-oriented multimedia session model. All information included therein is independent of user or network specific data. We propose an XML-based notation for the programming of the service logic.

The communication control processes the abstract service model and maps it on a set of concrete communication parameters. The adaptors of the adaptation layer act as gateways between the server-internal signaling messages and network APIs, like Parlay. A resource registry complements the communication control in order to support an adaptive selection of network resources according to service discovery principles.

For the interaction of the components of the server architecture we specify the Session Control Protocol as a new signaling protocol based on the IETF Session Initiation Protocol. This selection has been motivated by the increasing importance of IP-based signaling protocols in the standardization of new communication systems, e.g. UMTS. A new signaling aspect is the processing of the service description. It is carried as a payload in the signaling messages. Each component of the service architecture adds details to the service description until finally information paths with a concrete QoS description are requested from the adaptors.

Some important parts of the server architecture have been realized in a prototype implementation. The mechanisms and the classification of the network specific communication control, which is performed by the adaptors, have been taken from studies with the Intelligent Network and with IP-based distribution networks that have been carried out in the context of this work.

Abkürzungen

ACTS	*EU-Förderprogramm Advanced Communication Technologies and Services*
API	*Application Programming Interface*
AR	*Access Resource (SAMSON)*
ASN.1	*Abstract Syntax Notation Number One*
ATM	*Asynchronous Transfer Mode*
BCSM	*Basic Call State Model (IN)*
B-IN	*Broadband Intelligent Network*
B-ISDN	*Broadband ISDN*
CAMEL	*Customized Applications for Mobile Network Enhanced Logic*
CC	*Communication Control (SAMSON); Call Control*
CCG	*Connectivity Connection Graph (SAMSON)*
CGI	*Common Gateway Interface*
CORBA	*Common Object Request Broker Architecture*
CP	*Communication Provider (SAMSON-Geschäftsmodell)*
CS	*Capability Set (IN; ISDN)*
CSM	*Communication Session Manager (SAMSON)*
CTI	*Computer Telephony Integration*
CtP	*Content Provider (SAMSON-Geschäftsmodell)*
CtResA	*Content Resource Adaptor (SAMSON)*
DAB	*Digital Audio Broadcasting*
DECT	*Digital Enhanced Cordless Telecommunications*
DPE	*Distributed Processing Environment*
DTD	*Document Type Definition (XML)*
DTMF	*Dual Tone Multi Frequency*
DVB	*Digital Video Broadcasting*
ETSI	*European Telecommunications Standards Institute*
FORSOFT	*Bayerischer FORschungsverbund SOFTware Engineering*
FTP	*File Transfer Protocol*
G-NA	*NA mit Netzrand-initiiertem Verbindungsaufbau (SAMSON)*
GPRS	*General Packet Radio Service*
GSM	*Global System for Mobile Telecommunications*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol (RFC 2616)*
IAM	*Initial Access Manager (SAMSON)*
ICM	*Initial Communication Manager (SAMSON)*
IETF	*Internet Engineering Task Force*
IMT2000	*International Mobile Telecommunications 2000*
IN	*Intelligent Network*
INAP	*Intelligent Network Application Protocol*
IP	*Internet Protocol (RFC 791); beim Intelligenten Netz: Intelligent Peripheral*

ISDN	*Integrated Services Digital Network*
ISM	*Initial Service Manager (SAMSON)*
ISO	*International Organization for Standardization*
IST	*EU-Förderprogramm Information Society Technologies*
ISUP	*ISDN User Part (SS#7)*
ITU-T	*International Telecommunication Union - Telecommunication Sector*
IuK	*Informations- und Kommunikations-(Dienst)*
JAIN	*JAVA APIs for Integrated Networks*
JINI	*Java Intelligent Network Infrastructure*
JTAPI	*Java Telephony API*
LAN	*Local Area Network*
LDAP	*Lightweight Directory Access Protocol (RFC 2251)*
MAC	*Media Access Control*
MAGIC	*Multiservice Applications Governing Integrated Control (RACE II R2044)*
MAP	*Mobile Application Part (SS#7)*
MCG	*Media Connection Graph*
MGC	*Media Gateway Controller*
MIME	*Multipurpose Internet Mail Extension*
MPEG	*Moving Pictures Expert Group*
MSC	*ITU-Message Sequence Chart*
NA	*Network Adaptor (SAMSON)*
NGN	*Next Generation Network*
NRA	*Network Resource Architecture (TINA)*
OSI	*Open System Interconnection*
OvQ	*Overall Quality*
PAC/PAS	*Protocol Agent Client/Server (SAMSON)*
PBX	*Private Branch Exchange*
PDA	*Personal Digital Assistant*
PIN	*PSTN Internet Notification*
PINT	*PSTN/Internet Interworking (RFC 2848)*
P-NA	*NA mit parallelem Verbindungsaufbau*
PSTN	*Public Switched Telephone Network*
QoS	*Quality of Service*
RACE	*EU-Förderprogr. R & D Advanced Communication Technologies in Europe*
RFC	*Request For Comments*
RM-ODP	*Open Distributed Processing Reference Model*
RR	*Resource Registry*
RTCP	*Real Time Transport Control Protocol*
RTP	*Real Time Transport Protocol*
SAMSON	*Server Architecture for network independent Mulitmedia Service cOntrol in heterogeneous communication Networks*

SC	*Service Control*
SCE	*Service Creation Environment*
SCF	*Service Control Function (IN)*
SDL	*ITU-Specification and Description Language*
SDP	*Session Description Protocol (RFC 2327)*
SIB	*Service Independent Building Block (IN)*
SIP	*Session Initiation Protocol (RFC 2543)*
S-NA	*NA mit sequentiellem Verbindungsaufbau (SAMSON)*
SP	*Service Provider (SAMSON-Geschäftsmodell)*
SPIRITS	*Service in the PSTN/IN Requesting InTernet Services*
SQ	*Service Quality*
SResA	*Special Resource Adaptor (SAMSON)*
SS#7	*Signaling System No. 7*
SSF	*Service Switching Function (IN)*
SSM	*Service Session Manager (SAMSON; TINA)*
SU	*Service User (SAMSON-Geschäftsmodell)*
SV	*Service Vendor (SAMSON-Geschäftsmodell)*
TCP	*Transmission Control Protocol (RFC 793)*
TINA	*Telecommunications Information Networking Architecture*
UA	*User Agent (SIP)*
UC	*User Control (SAMSON)*
UDB	*User Data Base*
UDP	*User Datagram Protocol (RFC 768)*
UI	*User Interface*
UML	*Unified Modeling Language*
UMTS	*Universal Mobile Telecommunication System*
UNI	*User Network Interface*
UP	*User Proxy (SAMSON)*
USG	*User Service Graph (SAMSON)*
VoIP	*Voice over IP*
VXML	*Voice eXtensible Markup Language*
WAP	*Wireless Application Protocol*
WML	*Wireless Markup Language*
xDSL	*Digital Subscriber Line*
XML	*eXtensible Markup Language*
XSL	*Extensible Stylesheet Language*

Inhaltsverzeichnis

Kapitel 1
Einführung

„The success of new service provision platforms will largely depend on their ability to blend with existing technologies“ IEEE Communications Magazine [GHH+99]

1.1 Ausgangssituation und Motivation

Durch den rasanten technologischen Fortschritt in der Übertragungstechnik wird die reine Übertragungskapazität bald nicht mehr der allein erfolgsentscheidende Faktor bei Kommunikationsnetzen sein. Vielmehr wird der wirtschaftliche und technische Nutzen von Kommunikationssystemen in Zukunft in hohem Maße von den **Diensten** bestimmt, die den angeschlossenen Teilnehmern zur Verfügung gestellt werden. Die Teilnehmer erwarten über die Dienste, die ein Kommunikationssystem anbietet, die Deckung ihres Kommunikations- und Informationsbedarfs [Ebe96].

Bisher waren die Dienste sehr eng mit der Netztechnologie verknüpft und wurden stets zusammen mit einem Kommunikationsnetz eingeführt, wie z.B. dem Telefonnetz oder dem Bildschirmtext-System. Um die enge Bindung von Dienst und Netz aufzuheben und damit die Einführung und Entwicklung neuer Dienste in bestehenden Systemen zu erleichtern, wurden bereits Anfang der 80er Jahre Architekturen, wie die Intelligenten Netze [FGK97] eingeführt. Doch diese erweisen sich bei genauerer Betrachtung als in sich geschlossen und nur für bestimmte Mehrwertdienste in der Telefonie geeignet. Eine vollständige **Entkopplung** der Dienste von den Netzen, die für eine getrennte Evolution der Dienste notwendig ist, vollzieht sich nur sehr langsam.

In den letzten Jahren hat allerdings in verschiedenen Bereichen der Telekommunikation ein deutlicher **Wandel** stattgefunden, der sich direkt oder indirekt auf die Dienste und die Diensterbringung auswirkt [EP00]. Während zuvor eher eine stetige Evolution der Kommunikationssysteme getrennt in unterschiedlichen Anwendungsbereichen zu beobachten war, sind die derzeitigen Entwicklungen stark von **Kooperation** und **Integration** zwischen verschiedenen Systemen geprägt. Wichtige Einflußfaktoren sind dabei nicht unbedingt ausschließlich technische Neuerungen, sondern auch marktpolitische Entscheidungen, wie die Deregulierung auf dem Telekommunikationsmarkt.

Beginnend 1984 in den USA wurden seit 1998 auch in Europa die (staatlichen) Monopole für die Telekommunikation aufgelöst [Wit98]. Die damit einhergehende **Deregulierung** und somit Liberalisierung des Telekommunikationsmarktes führte zu einem vorher nicht existierenden **Wettbewerb** zwischen den ehemaligen Monopolisten und den neuen Anbietern.

Attraktive **Dienste** für die Teilnehmer sind die entscheidenden Faktoren, um sich im Wettbewerb um neue Kunden auf dem Markt zu differenzieren. Der Wettbewerb zwingt die Anbieter, den Teilnehmern in immer kürzeren Abständen neue Dienste zur Verfügung zu stellen. Es ist daher Ziel, dieselben Dienste auf möglichst vielen Netzen anzubieten, ohne sie jedesmal neu implementieren zu müssen. Denn die Teilnehmer fordern nicht nur einfach zu bedienende, auf sie persönlich zugeschnittene, hoch funktionelle Dienste. Sie wollen diese auch **unabhängig von den Netzen** aufrufen, je nachdem, welche Anschlüsse gerade verfügbar sind.

Bestehende Systeme können diese Anforderung nach einem einheitlichen Dienstzugang nur unzureichend erfüllen. Die Bestrebungen Mitte der 90er Jahre, mit B-ISDN/ATM [Boc97] eine einheitliche Dienste- und Netzinfrastruktur zu schaffen, waren nicht erfolgreich. Auch beim künftigen Mobilfunkstandard der dritten Generation IMT2000/UMTS [HT01] ist es ungewiß, ob wirklich alle Dienste integriert werden können. Demgegenüber entstehen fortwährend neue, oftmals konkurrierende Netze und Steuerungsprotokolle, z.B. für *Voice over IP*: ITU-H.323 [OT99] und IETF-SIP [SR99]. Zudem werden viele der bestehenden Systeme weitergenutzt, weil sie bestimmte Dienstklassen besser erbringen (z.B. PSTN für Sprachtelefonie) oder zusätzliche Bandbreite bereitstellen (z.B. digitale Rundfunk-Verteilnetze). Bisher unbeachtete Systeme erfahren einen neuen Nutzen (z.B. *Powerline Communication*). Diese **Heterogenität** der Netzinfrastruktur im Teilnehmerzugang und im Übertragungsnetz ist bei der Erbringung von Diensten zu berücksichtigen. Sie stellt hohe Ansprüche an die Dienstplattform.

Auch im **Software Engineering** allgemein und speziell für die Informations- und Kommunikationstechnik haben sich Änderungen ergeben, die eine verbesserte Dienst- und Plattformentwicklung entscheidend unterstützen. Die Komplexität neuer Dienste kann durch den Einsatz von objektorientierten Sprachen, formalen Beschreibungstechniken und durch den konsequenten Einsatz von formalen Methoden im Entwicklungsprozeß drastisch reduziert werden [Bro97]. Dies ist insbesondere in einem kooperativen Umfeld verteilter Architekturen notwendig, in dem Systeme verschiedener Hersteller und Betreiber zusammenwirken. Um eine schnelle Dienstentwicklung ohne aufwendige Einarbeitung zu gewährleisten, gewinnen Standard-Software-Entwicklungsumgebungen (z.B. für Java) im Gegensatz zu proprietären, aber bisher vorherrschenden Entwicklungsumgebungen (*Service Creation Environments*) immer mehr an Bedeutung. Dies wird dadurch begünstigt, daß offene Programmierschnittstellen (APIs), eingebettet in Middlewareplattformen, proprietäre Protokollschnittstellen langsam verdrängen (z.B. Parlay [Par00]).

Bei der zukünftigen Entwicklung von Diensten sind alle oben beschriebenen Faktoren von hoher Bedeutung. Insbesondere die Existenz einer wachsenden Vielzahl von leistungsfähigen Netzen, Systemen, Standards und Endgeräten muß entsprechend berücksichtigt werden. Die Heterogenität stellt *die* Herausforderung dar, um neue Dienste effizient einzuführen und gleichzeitig die Charakteristika der bestehenden Systeme intelligent zu nutzen. Dazu ist es notwendig, die Steuerung von Diensten möglichst unabhängig zu machen von den spezifischen Technologien der bestehenden, heterogenen Kommunikationsnetze. In der vorliegenden Arbeit wird eine Lösung für diese Problemstellung in Form einer Serverarchitektur aufgezeigt.

1.2 Zielsetzung, Lösungsansatz und Merkmale der Serverarchitektur

Ziel der vorliegenden Arbeit ist es, eine Konvergenz der unterschiedlichen Kommunikationssysteme auf Dienstebene zu erreichen, um Dienste **unabhängig** von den Netzen und Teilnehmerzugängen steuern zu können und darüber hinaus Dienste **netzübergreifend** anbieten zu können. Als **Lösung** wird eine neuartige Serverarchitektur beschrieben, die die Dienststeuerung von der netzspezifischen Signalisierung entkoppelt. Die Dienststeuerung soll nicht nur **unabhängig** von einem Netztyp sein, sondern es sollen Dienste für die Teilnehmer in verschie-

denen, heterogenen Kommunikationsnetzen **übergreifend** erbracht werden. Dafür ist eine einheitliche Schnittstelle der Dienststeuerung zu den Netzen zu definieren. Das Modell der betrachteten Dienste oberhalb dieser Schnittstelle ist so flexibel zu gestalten, daß alle Arten von multimedialen Informations- und Kommunikationsdiensten mittels der Architektur steuerbar sind.

Eine Dienststeuerung ist ein hochkomplexes Kommunikationssystem, für das keine allgemeine Entwicklungsmethodik existiert [Kel98]. Bestehende Softwareentwicklungsmethoden erweisen sich als zu allgemein, da sie nicht auf die besonderen Bedingungen der Dienste eingehen, und sind deshalb entsprechend für die Dienstentwicklung zu spezialisieren.

Daher wurde ausgehend von obigem Lösungsansatz folgendes **Vorgehen** gewählt, um die Serverarchitektur als technische Lösung zu entwickeln. Die Analysephase ist durch die Validierung der Ideen mit Dienst-Szenarien geprägt [HKS97]. Um der Komplexität von Kommunikationssystemen Rechnung zu tragen, wird das Gesamtsystem aus ausgewählten Sichtweisen abstrakt beschrieben. Zur Umsetzung des entstandenen Modells werden existierende Ansätze und Systeme herangezogen und diese entsprechend angepaßt. Auf diese Weise wird die abstrakte Idee anschaulich und in einer bestehenden Umgebung schnell implementierbar gemacht. Da entsprechende mathematische Verifikationsmethoden für diese hochkomplexen Systeme noch Stand der Forschung sind (z.B. [Hin98b]), wird die Funktionsweise der Systemkomponenten mit Prototypen nachgewiesen. Insbesondere formale Beschreibungssprachen haben sich im Vorfeld der Arbeit für die Analyse, für Simulationen und für Rapid Prototyping als äußerst geeignet erwiesen [Hin98a, Kel99b, KIR96, VKK98].

Mit diesem Vorgehen wird in der vorliegenden Arbeit eine Serverarchitektur entworfen. Sie weist folgende **Merkmale** auf:

- Die **Dienststeuerung in heterogenen Netzen** wird durch **Adaptoren** realisiert. Diese arbeiten als Signalisierungsgateways zwischen dem Signalisierungsprotokoll der Serverarchitektur und der netzspezifischen Signalisierung. Sie ermöglichen eine Auswahl und Kombination verschiedener Kommunikationssysteme für die Diensterbringung.
- Für die Steuerung von Diensten in den unterschiedlichen Netzen wird auf bestehende, standardisierte APIs zurückgegriffen. Somit kann die Architektur einfach umgesetzt werden, ohne Änderungen in den Netzen zu erfordern.
- Die Serverarchitektur ermöglicht das Geschäftsmodell eines neuen, **Netzanbieterunabhängigen Dienstanbieters**.
- Die Serverarchitektur wird in drei getrennt verwaltete **Teilbereiche** unterteilt: Teilnehmersteuerung, Dienststeuerung und Kommunikationssteuerung.
- Die Teilnehmersteuerung regelt den **Teilnehmerzugang** und die **Teilnehmerprofilverwaltung** unabhängig von der Dienststeuerung. Sie unterstützt damit die bisher wenig berücksichtigte, aber derzeit wachsende Personalisierung und die Mobilität bei Diensten.
- **Objekt-orientierte Beschreibung** der Dienstzustände (*Call Model*) zur generischen Modellierung eines umfassenden Dienstspektrums, das nicht auf bestimmte Dienstklassen beschränkt ist.
- Dynamisch verfeinertes **Qualitäts-Modell**: Die Beschreibung des Dienstes wird bei der Verarbeitung durch die Serverkomponenten ausgehend vom Dienstaufruf des Teilnehmers per Dienstname bis zu einer detaillierten Kommunikationsdienst-Beschreibung schrittweise konkretisiert.

- Für die Signalisierung innerhalb der Dienststeuerung wird ein **neues Signalisierungsprotokoll** spezifiziert, das auf dem *Session Initiation Protocol* der IETF (SIP) basiert. Damit wird erstens durch die Ausrichtung auf das Internet eine einfache Integration der Architekturkomponenten in eine IP-basierte Infrastruktur ermöglicht. Zweitens können bestehende Netzkomponenten für die Signalisierung eingesetzt werden.
- Der Zugang zum Server ist durch die Verwendung von **XML-Technologie** [Mar00] und der Umsetzbarkeit von XML-Inhalten in verschiedene Ausgabeformate unabhängig von der Informationsdarstellung im Endgerät.
- Das ***Service Discovery***-Konzept wird auf die Dienststeuerung übertragen. Damit kann die Auswahl von Kommunikationssystemen unterstützt werden und die Serverarchitektur automatisch an eine sich ändernde Umgebung angepaßt werden.

Die wesentlichen Bestandteile der Serverarchitektur wurden in einer prototypischen Realisierung umgesetzt, um die Funktionsweise der hier vorgestellten Konzepte zu validieren.

1.3 Einordnung der Arbeit

An dieser Stelle wird kurz auf Ansätze und Systeme eingegangen, die eng mit der in dieser Arbeit vorgestellten Serverarchitektur zusammenhängen. Weitere Details zu diesen Architekturen und eine eingehende Analyse des Stands der Technik können in Kapitel 3 nachgelesen werden.

Intelligente Netze

Das Konzept der Intelligenten Netze (IN) [FGK97] wird seit den 80er Jahren in der Erbringung von leistungsfähigen Zusatzdiensten für die Telefonie eingesetzt. Hauptmerkmal ist die logische Trennung der Basisvermittlung (Basisdienst) von einer zentralisierten Steuerung von Zusatzdiensten wie z.B. *Freephone*, *Virtual Private Network*. Ziel war es, die Abhängigkeit der Dienstentwicklung von den Switch-Herstellern zu reduzieren und die Dienstentwicklung durch die Wiederverwendung von dedizierten Dienstbausteinen zu beschleunigen. Die IN-Architektur ist stark auf Telefoniedienste eingeschränkt und ermöglicht nur eine teilweise Unabhängigkeit der Dienststeuerung von der Netzinfrastruktur. Trotzdem ist das Grundkonzept der zentralen, vom Netz abgesetzten Dienststeuerung Grundlage jeder modernen Dienststeuerung. In diesem Sinn fließen die Erkenntnisse aus der Analyse der IN-Architektur in die vorliegende Arbeit ein.

TINA - Telecommunication Information Networking Architecture

Aus einer internationalen Initiative der wichtigsten Telekommunikationsfirmen (TINA-C) ist Mitte der 90er Jahre eine abstrakte Beschreibung einer umfassenden Dienst-, Netz- und Management-Architektur entstanden, die auf ATM-basierte Netze ausgerichtet ist [ILM99]. Wenngleich der tatsächliche Einsatz dieser Architektur bisher nicht nachgewiesen wurde, so sind viele darin enthaltenen Konzepte wegweisend für die weitere Entwicklung der Kommunikationssysteme. Die vorliegende Arbeit baut auf dem TINA-Konzept zur Trennung der Dienststeuerung in Teilnehmer-, Dienst- und Kommunikations-Session für die funktionale Dekomposition der Dienstarchitektur auf.

IETF Session Initiation Protocol (SIP)

In RFC 2543 wurde 1999 von der IETF ein Ende-zu-Ende Signalisierungsprotokoll für die Internet-basierte Multimediakommunikation standardisiert [SR99]. Das Protokoll ist durch eine universelle Verwendbarkeit für beliebige Dienste gekennzeichnet. Dies bewirken die Trennung von Transaktionsprotokoll und transportierter Dienstbeschreibung und ein einfacher Aushandlungsmechanismus. Daher wurde SIP in der vorliegenden Arbeit als Basis für das Signalisierungsprotokoll zur Dienststeuerung verwendet.

Parlay-API

Die *Parlay Group* wurde 1998 von namhaften Telekommunikationsfirmen gegründet [Par00]. Sie hat das Ziel, eine einheitliche Schnittstelle (Parlay-API) zu definieren, über die Dienstanbieter auf fremde Netze zugreifen können. Dem Parlay-API wird derzeit eine große Bedeutung für die zukünftige Dienstentwicklung beigemessen. Es ist auch ein wichtiger Faktor für die in dieser Arbeit vorgestellte Serverarchitektur, die oberhalb einer solchen Schnittstelle anzusiedeln ist, da ein Parlay-Interface erstmals einen standardisierten Zugang zu Netzen für unabhängige Dienstanbieter ermöglicht. Parlay definiert nur ein API, aber keine Komponenten für eine Dienststeuerung, z.B. für die Teilnehmerverwaltung oder den Teilnehmerzugang.

1.4 Beitrag der Arbeit

Die vorliegende Arbeit liefert aufgrund der beschriebenen Merkmale der Serverarchitektur Beiträge in wichtigen Gebieten der Kommunikationsnetze und des Software Engineerings:

- **Konvergenz der Netze**: Es wird ein Kommunikationssystem beschrieben, das auf Dienstebene durch eine strikte Entkopplung von Diensten und Netzen die Forderung nach der Konvergenz der Netze unterstützt. Die beschriebene Architektur ermöglicht es, Dienste unabhängig von der vorhandenen Infrastruktur anzubieten.
- **Dienstarchitekturen**: Die vorliegende Arbeit spezifiziert eine komponentenorientierte Architektur und ein objektorientiertes Dienstmodell für die Steuerung von Diensten über Netz-APIs. Sie bietet nicht nur eine neuartige, technische Lösung für die obige Zielsetzung, sondern zeigt auch allgemein nötige, abstrakte Komponenten und eine generelle Struktur einer idealen Dienststeuerung. Für Netz-APIs wie z.B. Parlay sind bisher keine Architekturen definiert, die wie hier Teilnehmersteuerung, Dienststeuerung und Kommunikationssteuerung integrieren.
- **Signalisierung für die Dienststeuerung**: Für die Kommunikation der Serverkomponenten wird ein neuartiges Signalisierungsprotokoll basierend auf dem IETF *Session Initiation Protocol* entwickelt. Da IP-basierte Signalisierung und speziell SIP in vielen Bereichen als Grundlage für die Signalisierung verwendet wird (z.B. bei der dritten Generation Mobilfunk), stellt das neue Protokoll einen Beitrag zur laufenden Standardisierung dar.
- **Software Engineering**: Um die Software-Architektur der Dienststeuerung auf Systemebene abstrakt darstellen zu können, wird ein Systemmodell vorgestellt. Dessen Sichtweisen sind auf die Spezifikation von Dienstarchitekturen abgestimmt.

Daneben zeigt die Übertragung von neuen Konzepten aus der Informationstechnik (das Metadatenformat XML [Mar00] und das *Service Discovery*-Prinzip) auf die Steuerung von Informations- und Kommunikationsdiensten, wie deren Vorteile für zukünftige

Kommunikationssysteme nutzbar gemacht werden können. Die Anwendung wurde in prototypischen Realisierungen gezeigt.

1.5 Gliederung

Kapitel 2 beschreibt das Umfeld von Informations- und Kommunikationsdiensten anhand von Begriffsdefinitionen, der Analyse der Netzinfrastruktur, Grundprinzipien von Dienstarchitekturen, der Dienstentwicklung und den Kosten im Dienstlebenszyklus. Daraus werden die generellen Merkmale von Dienstarchitekturen entwickelt. Anhand von Beispielszenarien werden konkrete Anforderungen an eine netzunabhängige Dienststeuerung abgeleitet, die im Schwerpunkt der weiteren Betrachtungen liegt.

Kapitel 3 gibt einen Überblick über existierende Architekturen und Ansätze zur Steuerung von Diensten und bewertet sie anhand der in Kapitel 2 aufgestellten Anforderungen. Dabei wird besonders auf bestehende Ansätze für netzunabhängige Architekturen eingegangen und der Stand der Technik analysiert, der in engem Zusammenhang mit der beschriebenen Serverarchitektur steht.

In Kapitel 4 wird das Konzept der Serverarchitektur erläutert, die sich in eine Dienstebene und in eine Anpassungsschicht unterteilt. Anschließend wird ein Modell für Dienstarchitekturen vorgestellt. Die zugehörigen Sichtweisen erlauben es, die Architektur aus verschiedenen Blickwinkeln abstrakt zu beschreiben. Das Geschäftsmodell unterscheidet die an einem Dienst beteiligten Aktoren und definiert externe Schnittstellen der Dienststeuerung. Aus der Informations-Sichtweise wird die Datenstruktur und aus der Komponenten-Sichtweise wird die Strukturierung in Komponenten mit den zugehörigen Schnittstellen beschrieben. Die Session-Sichtweise fokussiert die Teilnehmerinteraktionen. Aus der Betrachtung der Architektur aus der Kommunikations-Sichtweise folgen die Anforderungen an das Signalisierungsverfahren zwischen den Komponenten. Ein ausführliches Beispielszenario zeigt die Funktionsweise der Architektur und das Zusammenspiel der einzelnen Komponenten.

Aufbauend auf der abstrakten Modellierung der Dienstarchitektur in Kapitel 4 werden in Kapitel 5 die interagierenden Komponenten der Dienststeuerungsebene spezifiziert. Als erstes wird das neue Signalisierungsprotokoll SesCP (*Session Control Protocol*) beschrieben. Anschließend werden die Komponenten der Teilnehmerverwaltung und des Dienstzugangs, die Komponenten der Dienststeuerungsmodule und die Dienstlogik, sowie die Komponenten der Kommunikationssteuerung spezifiziert. Kapitel 5 schließt mit einem detaillierten Beispiel eines Signalisierungsablaufs.

In Kapitel 6 wird die Anpassungsschicht der Serverarchitektur beschrieben, welche die Einrichtung von Kommunikationsbeziehungen in den Netzen realisiert. Hauptkomponenten sind die Adaptoren für die Kommunikationssteuerung, für den Dienstzugang, für die Steuerung von zusätzlichen Ressourcen und für die Ressourcenverwaltung. Es wird eine Klassifikation vorgeschlagen und die Funktion der Adaptoren beschrieben.

Kapitel 7 gibt einen Überblick über die Spezifikation und die prototypische Realisierung ausgewählter Teile der Serverarchitektur. Zum Nachweis der Realisierbarkeit der Dienstarchitektur wurde ein Prototyp des Teilnehmerzugangs, der Ressourcenverwaltung und ein SDL-Simulator für das Signalisierungsprotokoll erstellt.

Kapitel 2
Dienstumfeld und Anforderungen an die Steuerung von Diensten

Um die in der vorliegenden Arbeit beschriebene Serverarchitektur in das technische Umfeld von Informations- und Kommunikationsdiensten einordnen zu können, ist es notwendig, die bestehende Situation anhand von Begriffen, generellen Systemeigenschaften und Entwicklungsmethoden auszuleuchten. Aus diesen Betrachtungen, die auch betriebswirtschaftliche Analysen miteinbeziehen, lassen sich schließlich die Anforderungen an eine netzunabhängige Dienststeuerung ableiten.

2.1 Multimediale Informations- und Kommunikationsdienste

Kaum ein Begriff in der Informations- und Kommunikationstechnik wird vielfältiger gebraucht als der Dienstbegriff. Daher ist es nicht möglich, *eine* allgemeingültige Definition anzugegeben. Der Dienstbegriff ist vielmehr für jedes Themenfeld festzulegen und abzugrenzen, um ein einheitliches Verständnis zu erreichen.

> Ein **Dienst** (*service*) im Kontext der Telekommunikation [I.112] ist eine Einrichtung in einem Kommunikationsnetz zum Austauch von Informationen, die damit für den Dienstbenutzer einen bestimmten Zweck erfüllt. Dieser Dienst wird dem Benutzer über einen entsprechenden Zugang zur Verfügung gestellt. Die grundlegenden Funktionen eines Dienstes sind Auf- und Abbau, Hinzunahme und Änderung von Kommunikationsbeziehungen zwischen Teilnehmern in einem Kommunikationsnetz und die Übertragung von Nutzinformationen.

Telekommunikationsdienste sind darüber hinaus dadurch gekennzeichnet, daß sie mindestens eine realzeitige Übertragung wie z.B. Sprachübertragung beinhalten. Werden verschiedene Informationselemente, d.h. realzeitige (Sprache, Video) und nicht-realzeitige (Text, Bild, Daten), in beliebiger Kombination zugelassen, dann spricht man von **Informations- und Kommunikationsdiensten**. Anstelle von Informationselementen spricht man auch von Medien, daher steht die Bezeichnung **Multimedia** für eben diese Kombination heterogener Informationselemente [SN95].

Ein multimedialer Informations- und Kommunikationsdienst (IuK-Dienst) ist damit durch die Einbeziehung mehrerer, verschiedener Medien gekennzeichnet. Er muß Mehrverbindungs- und Mehrteilnehmerrufe, Abhängigkeiten zwischen Medien, Aushandlung und dynamische Änderung von Parametern und Rufstruktur ermöglichen [Kel98]. Beispiele für IuK-Dienste

finden sich in Tabelle 2.1. In der vorliegenden Arbeit bezieht sich der Begriff Dienst auf multimediale Informations- und Kommunikationsdienste.

Ein Dienst besteht meist aus einer Grundfunktionalität (**Basisdienst**), z.B. dem Aufbau einer Wählverbindung für einen Sprachdialog, die bestimmte Dienstmerkmale (z.B. Bitrate) beschreibt. Daneben können ergänzende Dienstmerkmale zusätzliche Funktionen beschreiben, die nur in Verbindung mit der Grundfunktion genutzt werden können (z.B. Rufnummernanzeige, Anrufweiterleitung). Man spricht dabei von **Zusatzdiensten**.

Die Ausprägung der Dienstmerkmale in den zwei Bereichen Personalisierung und Mobilität haben beim derzeitigen Dienstangebot eine wichtige, wachsende Bedeutung und stellen erhöhte Anforderungen an die Systeme, die Dienste realisieren. **Personalisierung** bezeichnet das individuelle Zuschneiden von Diensten auf einen bestimmten Nutzer im Gegensatz zu einheitlichen Standarddiensten (z.B. dem analogen Fernsprechdienst). Netzprotokolle (z.B. zellularer Mobilfunk, Mobile IP) können **Endgeräte-Mobilität** gewährleisten, indem sie dem Teilnehmer erlauben, sein persönliches Endgerät überall zu benutzen. **Dienste-Mobilität** bedeutet, daß der Teilnehmer einen Satz von Diensten unabhängig vom Endgerät, Zugangsnetz und Kernnetz nutzen kann [Wil00]. Daneben ist für die Dienstnutzung auch die **persönliche Mobilität** relevant, bei der ein Teilnehmer auf verschiedenen Systemen unter einer einheitlichen Rufnummer erreichbar ist.

Dienstklasse	**Beispiele für IuK-Dienste**
Dialogdienste	Videotelefonie, Videokonferenz, *Chat*, Multiparty-Multimedia-Konferenz, verteilte interaktive Simulation, *Computer Supported Collaborative Work* (CSCW), Interaktive Spiele
Abrufdienste	Video-On-Demand, WWW-Zugriff, Katalogdienste, FTP *get*, Videoüberwachung
Zustelldienste	Multimedia-E-Mail, *Unified Messaging*, FTP *put*, Telefax
Verteildienste	Fernseh- und Rundfunkübertragungen, Near-Video-on-Demand

Tabelle 2.1: Beispiele für IuK-Dienste nach Dienstklassen

Da ein Dienst einen äußerst komplexen Sachverhalt beschreibt, dient die obige Definition nur dem groben Verständnis. Für eine genauere Beschreibung ist ein Dienst aus verschiedenen Blickwinkeln abstrakt zu betrachten. Ziel einer solchen **Klassifikation** ist es, exakter beschreibbare Teilfunktionen von Diensten zu identifizieren. So können genaue Anforderungen an die Systeme gestellt werden, die Dienste realisieren.

Am Zustandekommen eines Dienstes sind unterschiedliche Instanzen und handelnde Personen beteiligt. Da die Funktionen der handelnden Instanzen, wie z.B. „Teilnehmer“ oder „Dienstanbieter“, bei verschiedenen Diensten wiederholt auftreten, kann man einen Dienst nach den beteiligten **Rollen** klassifizieren. Auf Basis dieser Rollen können die Schnittstellen in einem Dienstsystem definiert werden (z.B. im Geschäftsmodell).

Die Kommunikationsbeziehungen zwischen den Rollen können nach der Art des Informationsflusses charakterisiert werden. Merkmale sind hier der **Medientyp** oder die **Dienstklasse**, die einen Informationsfluß hinsichtlich des Teilnehmernutzens charakterisiert. Tabelle 2.1 zeigt einige Beispiele für die vier Dienstklassen Dialogdienst, Zustelldienst, Abrufdienst und Verteildienst.

Wie bereits in der Einführung angesprochen, werden Dienste in unterschiedlichen Anwendungsbereichen eingesetzt.

> Unter einer **Anwendung** (*Application*) versteht man dabei den Teil eines Kommunikationssystems, der die Dienste in einen spezifischen Kontext einbettet, der über den reinen Informationsaustausch hinausgeht. Bei Anwendungen liegt der Schwerpunkt auf dem Informationsinhalt und der Aufbereitung. Sie greifen auf IuK-Dienste zu, um z.B. Informationskioske, Home-Banking, E-Commerce, *Distance Learning* oder Call-Center als typische Beispiele für Anwendungen zu realisieren [KQ98].

Das Zusammenwachsen der bisher getrennten Anwendungsbereiche Telekommunikation (z.B. Telefonie), Information (z.B. Internet-Dienste), Medien (z.B. Fernsehen) und Unterhaltung/ Entertainment (z.B. Spiele) bedeutet, daß Dienste entstehen, die kombinierte Merkmale aus den verschiedenen Bereichen aufweisen. Beispielsweise kann man sich interaktives Fernsehen aus den Basisdiensten Datenkommunikation und Videobroadcast zusammengesetzt vorstellen. Aus dieser **Dienste-Konvergenz** resultiert, daß die bisherige Bindung zwischen Diensten und der zugehörigen Infrastruktur aufgebrochen wird, indem in eine bestimmte Systemumgebung neue Basisdienste und Zusatzdienstmerkmale eingebracht werden. Diese Unabhängigkeit ist bei der Entwicklung neuer Systeme zu berücksichtigen. Eine universelle Dienststeuerung benötigt eine systemunabhängige Dienstbeschreibung.

2.2 Heterogenität der Netze

So wie konkrete Dienste Anforderungen an die Netze stellen, was zum Entstehen vieler verschiedener, dienstspezifischer Netze geführt hat, so stellt auch die bestehende heterogene Netzinfrastruktur Anforderungen an die Dienstsysteme. Hohe Kosten für Infrastruktur und das Fallen von Monopolen machen eine dienstspezifische Neuentwicklung und Neuinstallation von Netzen nicht rentabel.

Im Bereich der Telekommunikation besteht die Netzinfrastruktur aus einer Vielzahl an Festnetzen (z.B. ISDN [Boc97], B-ISDN [Boc97], xDSL) und an Mobilfunknetzen (z.B. GSM [EVB01], UMTS [HT01]). Daneben gewinnen Datennetze, insbesondere IP-basierte Datennetze, als lokale Netze (z.B. LAN/WirelessLAN, Bluetooth) oder Weitverkehrsnetze (Internet, GPRS [BVE99]), immer mehr an Bedeutung. Während die Telekommunikationsnetze verbindungsorientiert arbeiten, besteht der Vorteil bei den meisten Datennetzen in der verbindungslosen Paketorientierung, die eine effizientere Ausnutzung der vorhandenen Übertragungskapazität erlaubt. Zudem wird prognostiziert, daß der Datenverkehr in Zukunft weiter zunimmt [Jaj99].

Eine dritte Gruppe an Netzen kommt mit den analogen und digitalen Rundfunk-Verteilnetzen (z.B. Satellitennetze, DAB, DVB-C/-S/-T [Rei95]) hinzu. Diese werden in jüngster Zeit vor allem durch die Digitalisierung als wichtige Komponenten der Kommunikationsinfrastruktur neu entdeckt, da sie zusätzliche Bandbreite zum Teilnehmer über bereits installierte Kabel- oder Funknetze bereitstellen.

Diese derzeitige Vielfalt in der Netzinfrastruktur widerspricht dem Anfang der 90er Jahre vorhergesagten Trend zur vollständigen Vereinheitlichung der Netzinfrastruktur hin zu einem einzigen Netz, das alle Dienste an einem Netzanschluß anbietet. Dieses auf ATM basierende B-ISDN ist nie in großem Stil realisiert worden. Auch für die Zukunft ist kein Trend zu einer Integration zu erkennen, nachdem der Kommunikationsmarkt durch die Deregulierung zersplittert ist und der Wettbewerbsdruck kostspielige Umstellungen verhindert.

Im Bereich des Mobilfunk soll die dritte Generation, IMT2000 bzw. UMTS in Europa [HT01], das Dienste-integrierende Universalnetz werden. Aber auch im Mobilfunk werden die bestehenden Systeme koexistieren. Ein Grund ist die grundsätzlich begrenzte Übertragungskapazi-

tät in der Luft bei rasant steigenden Teilnehmerzahlen und Bandbreiteanforderungen durch mobile Anwendungen wie z.B. *Mobile Commerce* [Mül00].

Konvergenz der Netze

Die Koexistenz bestehender Netze bedeutet nicht, daß die Netze unverbunden nebeneinander stehen. Es sind vielfältige Bestrebungen zu einer Konvergenz der Netze vorhanden mit dem Ziel, Dienste netzübergreifend anbieten zu können. Netzübergänge, sogenannte ***Gateways***, erlauben zum Beispiel die Kopplung von Netzen und die Umsetzung von Diensten zwischen verschiedenen Netztechnologien oder unterschiedlichen Betreiberdomänen.

Weiter als einfache Netzkopplungen gehen die generellen Ansätze zur Konvergenz der Netze auf Netzebene oder auf Dienstebene. Auf Netzebene wird erwartet, daß sich in den nächsten fünf bis zehn Jahren ein robustes, paketorientiertes Transportnetz herausbildet, das eine Vielzahl von Diensten unterstützt. Diese Netze werden den meisten Verkehr tragen, wenngleich schmalbandige Netze anderer Technologien weiterhin koexistieren [Jaj99]. Vielfach wird IP als das Protokoll auf Vermittlungsschicht angesehen. Die Vision des mit ***Next Generation Network*** (NGN) bezeichneten Szenario beschreibt ein (IP-) paketbasiertes Kernnetz, für das bestehende Netze als Zugangsnetze wirken, die über entsprechende *Gateways* angeschlossen sind (Abbildung 2.1). Für alle Ansätze auf Netzebene sind Änderungen in der Infrastruktur notwendig, deren Durchführung derzeit nicht abzusehen ist.

Selbst die Einführung des **IP-Protokolls** als gemeinsame Vermittlungsschicht auf allen Netzen, Kernnetzen sowie Zugangsnetzen, stellt keine ideale Lösung dar. Einige Dienstparameter für Telekommunikationsdienste wie Qualitäts-Garantie, Sicherheit oder Gebührenberechnung können bisher nur durch ergänzende Mechanismen erbracht werden. Außerdem verlangt ein „All-IP-Netz“ intelligente, leistungsfähige Endgeräte.

Eine zweite Möglichkeit, Netze für eine übergreifende Diensterbringung zu integrieren, ist die **Kopplung auf Dienstebene**. Das bedeutet, daß die Netzwerkstruktur weitgehend unangetastet bleibt und die Kopplung der Systeme durch die Umsetzung von Signalisierungsdaten zwischen den Netzen erfolgt. Die existierenden Lösungen sind meist durch eine einseitige Kopplung geprägt, die auf einen bestimmten Anwendungsfall beschränkt ist. Beispiele sind die Verknüpfung von privaten Nebenstellenanlagen mit Rechnersystemen (z.B. CTI [CL00], WebIN

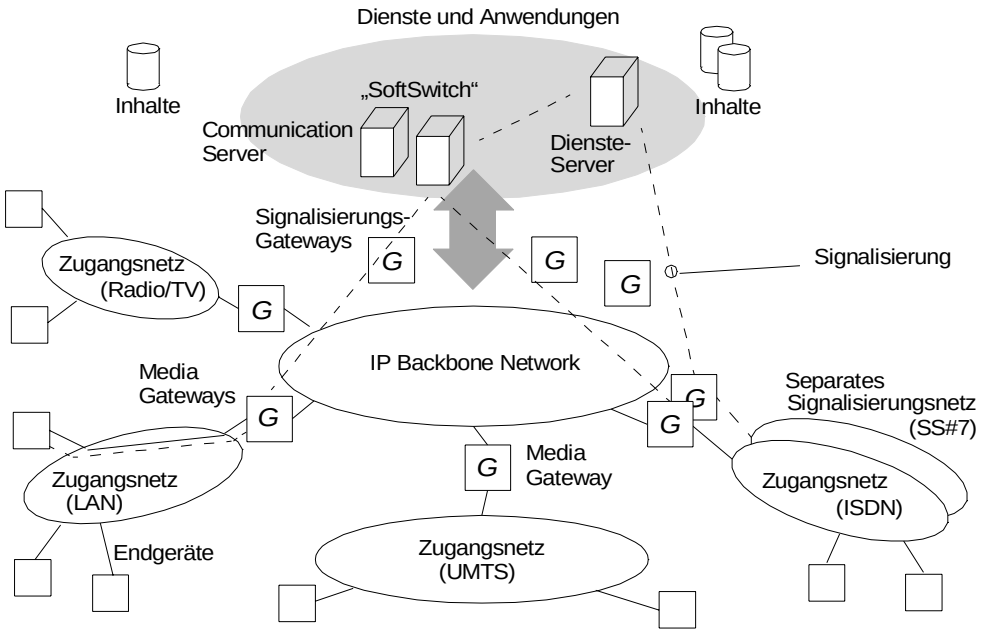


Abbildung 2.1: Next Generation Network [MU01]

[Low97]), oder die Anforderung von Telefoniediensten aus dem Internet [PC00]. Jüngste Bestrebungen standardisieren offene Schnittstellen zur Netzinfrastruktur, die es Dritten erlauben, steuernd auf die Netze zuzugreifen (z.B. Parlay [Par00]). Das in der vorliegenden Arbeit beschriebene System setzt auf diese letzteren Schnittstellen auf.

Eine Alternative ist der **Middleware-Ansatz**. Dabei wird eine einheitliche „Zwischenschicht“, die zwischen Netz- und Dienstebene liegt, als eine Art Signalisierungsplattform über die heterogene Netzinfrastruktur gelegt. Die Middleware stellt allen verteilten Systemkomponenten (z.B. Dienstkomponenten) Funktionalität bereit, die sie für die Zusammenarbeit (z.B. Diensterbringung) nutzen können. Dazu gehören grundlegende Kommunikationsdienste und unterstützende Funktionen, wie z.B. Datenbanken. Der grundsätzliche Nachteil der Middleware-Lösungen besteht darin, daß die Middleware-Software in jeder Komponente eines Kommunikationssystems vorhanden sein muß, d.h. auch in Endgeräten und in jedem Transitknoten. Außerdem sind die existierenden Middleware-Lösungen hinsichtlich Performance für Telekommunikationsdienste noch immer nicht ausgereift [BJM00].

Um entsprechend der Zielsetzung der vorliegenden Arbeit Dienste unabhängig in heterogenen Netzen steuern zu können, ist der Ansatz, die Integration der Netze auf Dienstebene zu realisieren, derzeit am meisten erfolgversprechend. Hier sind keine aufwendigen Änderungen in der Infrastruktur notwendig.

Endgeräte

Die Frage nach zukünftigen Endgeräten ist ebenso schwierig zu beantworten wie die Frage nach einem in Zukunft dominierenden Netz. Auch hier ist trotz der Bündelung von Dienstzugängen im Endgerät kein Trend zu einem universellen Endgerät zu beobachten. PDAs erlauben mittlerweile nicht nur die Terminverwaltung, sondern auch Web-Surfen und Email-Bearbeitung. PCs werden zu Telefonie-Endgeräten und in Fernsehern werden Web-Browser integriert. Nach der Meinung des Autors wird es in Zukunft weiter eine Vielzahl von zwar integrierten, aber differenzierten Endgeräten geben. Es wird nicht mehr nur der Dienst ausschlaggebend sein für die Endgeräteausstattung, sondern auch die Situation, in der der Teilnehmer das Endgerät nutzen will. Faktoren wie Portabilität versa Anzeigegröße, Bedienungsfülle über komplexe Benutzerschnittstellen versa einfache, vertraute Bedienung werden die Endgeräteausstattung bestimmen.

Aus diesen Gründen wird es in Zukunft auch kein Vorherrschen von ausschließlich intelligenten Endgeräten geben, auf deren Rechnerplattform beliebige Dienste-Software installiert werden kann. Die Diensterbringung wird sich auch zukünftig mit heterogenen Endgeräten auseinandersetzen müssen.

2.3 Dienstarchitektur: Grundstruktur und Aufgaben der Dienststeuerung

> Während man unter dem Begriff Dienst eine Einrichtung in einem Netz versteht, die dem Benutzer einen Kommunikationswunsch erfüllt, bezeichnet eine **Dienstarchitektur** (*Service Architecture*) die Struktur und Arbeitsweise dieser Einrichtung. Eine Dienstarchitektur ist ein Satz von Modellierungskonzepten, Prinzipien und Bedingungen für den Entwurf von Dienstsystemen. Zur Dienstarchitektur gehören die Struktur der Komponenten, aus denen sie besteht, und die Art der Wechselwirkungen zwischen den Komponenten. Ziel ist die möglichst flexible Einrichtung und Handhabbarkeit von Diensten während deren Lebenszyklus [Hun95].

2.3.1 Aufgaben

Die **Dienststeuerung** stellt die Intelligenz in einem Kommunikationssystem dar, die den Ablauf eines Dienstes regelt. Wie man der Definition des Dienstbegriffes entnehmen kann, besteht die Funktion der Dienststeuerung darin, entsprechend dem Dienstzweck Informationsbeziehungen zwischen den Dienstteilnehmern aufzubauen, zu ändern und abzubauen. Weitere Aufgaben unterstützen diese Kernfunktion. Abbildung 2.2 stellt die verschiedenen Funktionen dar, die eine Dienststeuerung zu erfüllen hat [KSS00, Kel98, P909, Kri97].

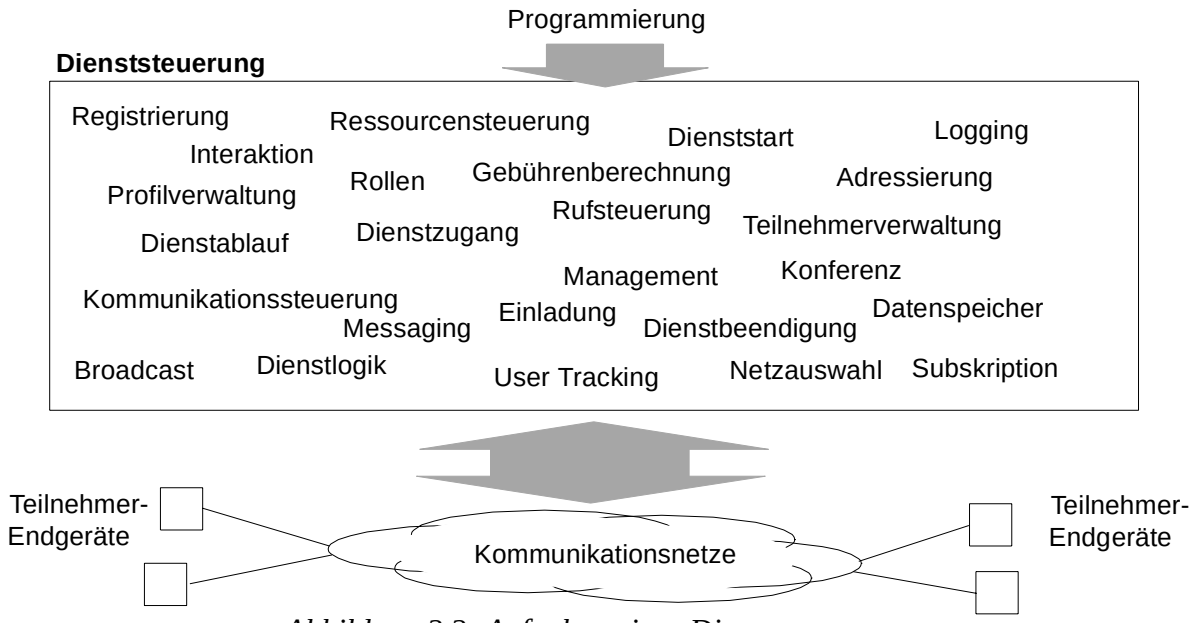


Abbildung 2.2: Aufgaben einer Dienststeuerung

Diese Aufgaben können je nach Relevanz für den Teilnehmer, den Dienst oder die Netze folgendermaßen gruppiert werden:

Teilnehmer-bezogene Aufgaben

- Teilnehmerzugang: Ein Teilnehmer meldet sich und sein Endgerät bei der Dienststeuerung an und hinterlegt sein Profil (Registrierung).
- Authentisierung und Autorisierung im Teilnehmerzugang: Teilnehmer und Dienststeuerung weisen sich gegenseitig als die richtigen und berechtigten Partner aus; meist geschieht dies über Paßwörter und/oder Verschlüsselungsmechanismen.
- Verwaltung des Teilnehmerprofils (Konfiguration, Personalisierung): Der Teilnehmerdatensatz kann durch den Teilnehmer und durch die Dienststeuerung ausgelesen und modifiziert werden.
- Gebührenberechnung: In einem kommerziellen Umfeld ist die Dienstnutzung in Rechnung zu stellen.
- *Directory Service*: Die Dienststeuerung verwaltet eine Teilnehmerdatenbank zur Auffindung der Teilnehmerdatensätze und für die Adreßauflösung.

Dienst-bezogene Aufgaben

- Dienstzugang: Der Teilnehmer hat die Möglichkeit, unter verschiedenen Diensten eines Anbieters auszuwählen. Ihm wird danach eine Schnittstelle zur Verfügung gestellt, über die er den Dienst mit seinen Daten (z.B. gewünschte Kommunikationspartner) aufrufen kann.
- Verwaltung des Dienstprofils: Dienstparameter (z.B. minimale Verbindungsqualität) können während der Dienstnutzung oder unabhängig davon geändert werden.
- Dienststeuerung anhand der Dienstlogik: Eine Dienst-Session beschreibt alle einem Dienst zugehörigen Steuerungsvorgänge. Ihre aktuellen Parameterwerte legen den Dienstzustand fest. Die Dienstlogik bestimmt den Ablauf der Dienststeuerung (z.B. Reaktion auf Ereignisse, Hinzunahme neuer Teilnehmer).
- Reaktion auf Ereignisse: Die Dienststeuerung hat nicht nur auf direkte Teilnehmeranforderungen zu reagieren, sondern auch auf externe Ereignisse, wie die Änderung von Verbindungsparametern, Ablauf des Benutzerkontos bei PrePaid-Diensten oder auf bestimmte Zeitereignisse.
- Hinzunahme / Quittierung von Teilnehmern
- Verwaltung von Teilnehmerrollen und -rechten: Bei der Dienststeuerung sind die Berechtigungen (Einladen von Teilnehmern, Sperren von Teilnehmern) in Abhängigkeit von den Rollen (Konferenzleiter, Initiator), die ein Teilnehmer in einem Dienst einnimmt, zu prüfen.
- Datenspeicherung z.B. von Teilnehmeradressen.

Kommunikationsressourcen-bezogene Aufgaben

- Steuerung (Einrichtung und Verwaltung) von Kommunikationsbeziehungen in den Kommunikationsnetzen
    - Steuerung von Zwei- oder Mehrteilnehmerverbindungen
    - Steuerung von Zustelldiensten, z.B. Email, Fax
    - Steuerung von Verteildiensten, z.B. unidirektionale Aussendung von Informationen an eine unbeschränkte Teilnehmerzahl
- Unterstützung der Teilnehmer-Interaktion
- Abstimmung der Fähigkeiten
    - der Endgeräte
    - der Netze

  Parameter wie Darstellung, Kodierungsformate und Übertragungskapazitäten sind zwischen den am Dienst beteiligten Einheiten abzugleichen.
- Auswahl von Netzkomponenten: Falls die Dienststeuerung nicht auf einem bestimmten Kommunikationssystem beruht, sind geeignete Übertragungskomponenten auszuwählen. Dabei kann es sich um Übertragungsabschnitte (Netzressourcen) sowie um Spezialressourcen (z.B. Kodierungs-Konverter, Konferenzbrücken) handeln.

Allgemeine Aufgaben

- Logging: Speicherung ausgewählter Aktionen für die Zugriffsüberwachung oder die Fehlerkontrolle.
- Management: Ein Dienst muß einfach einzurichten, zu aktivieren und zu deaktivieren sein, d.h. der Dienstlebenszyklus (siehe Abschnitt 2.4.1) ist mit möglichst geringem (finanziellen) Aufwand zu unterstützen.

2.3.2 Grundstruktur: Partitionierung und Schnittstellen

Die Dienststeuerung wird zur Erfüllung der oben skizzierten Aufgaben durch die Struktur der Dienstarchitektur entscheidend unterstützt. Ein Charakterisierungsmerkmal von Dienstarchitekturen sind interne und externe Schnittstellen. Interne Schnittstellen partitionieren die Dienststeuerung in funktional abgegrenzte, leicht handhabbare Komponenten. Dadurch wird die Erweiterbarkeit der Architektur, die Wiederverwendbarkeit von Komponenten und der Zukauf von Komponenten fremder Hersteller unterstützt. Externe Schnittstellen ermöglichen die Zusammenarbeit mit unterschiedlichen Kommunikationssystemen (z.B. Interworking mit anderen Dienstanbietern, Programmierung von Diensten). Die Schnittstelle zu den Teilnehmern kann je nach der Struktur und der Verteilung der Komponenten der Dienststeuerung eine interne oder eine externe Schnittstelle sein.

Darüber hinaus lassen sich die Schnittstellen nach Funktionsebenen einteilen, je nachdem ob sie eine horizontale oder eine vertikale funktionale Trennung (Partitionierung) unterstützen. Eine vertikale Trennung bezeichnet Schnittstellen für die Zusammenarbeit zwischen Systemkomponenten auf derselben funktionalen Ebene, während die horizontale Partitionierung das Interworking unterschiedlicher Ebenen beschreibt. Man unterscheidet idealerweise zwischen den Anwendungen, der Dienstebene, der Netzebene und den Kommunikationsnetzen (siehe Abbildung 2.3, [Kel98]). In manchen Systemen wird zusätzlich eine Ressourcenebene abgespalten, die Aufgaben wie Ruf- und Ressourcensteuerung übernimmt, z.B. [Kni93].

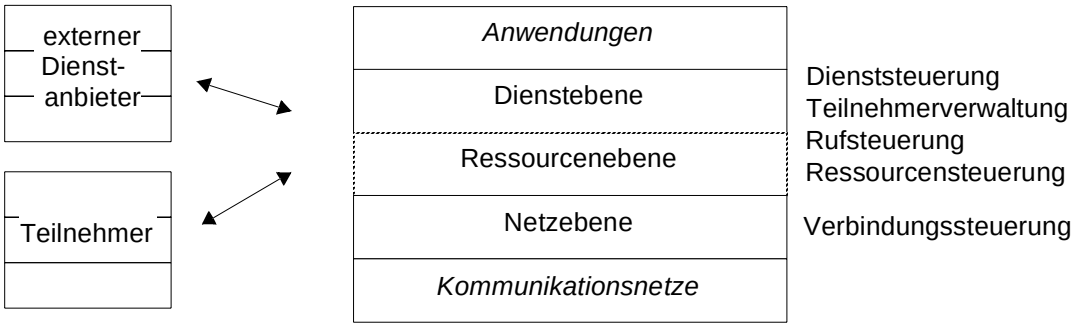


Abbildung 2.3: Funktionsebenen von Kommunikationssystemen

Für eine universelle Dienststeuerung ist die funktionale Trennung zwischen der Dienstebene und der Netzebene (z.B. durch APIs in der Ressourcensteuerung) das wichtigste Merkmal, um eine getrennte Evolution von Diensten und Netzen zu ermöglichen. An der (vertikalen) Schnittstelle zu den Teilnehmern ist die Funktionalität eines Dienstes sichtbar. Durch sie erfolgt die Differenzierung zu anderen Dienstanbietern. Eine vertikale Schnittstelle zu externen Dienstanbietern erweitert das eigene Dienstangebot und die erreichbare Teilnehmerzahl.

2.3.3 Intelligenzverteilung

Ein weiteres Charakteristikum einer Dienstarchitektur ist der physikalische Ort der Intelligenz im Netz, die die Steuerungsaufgaben wahrnimmt. Man unterscheidet generell zwischen Protokoll-basierten (a), Netz-zentralen (b), Endgeräte-basierten (c) und Middleware-basierten (d) Lösungen (Abbildung 2.4).

Die Dienste des PSTN und des ISDN werden in der OSI-Schicht-3, dem Vermittlungsprotokoll, realisiert. Bei diesen Protokoll-basierten Dienststeuerungen (a) muß die Dienste-Software auf allen Netzknoten vorhanden sein. Für den Vorteil hoher Leistungsfähigkeit und Skalierbarkeit für viele Teilnehmer muß der Nachteil aufwendiger Erweiterbarkeit in Kauf genommen werden. Beim Netz-zentralen Ansatz (b) ist die steuernde Intelligenz an einem Ort im Netz konzentriert, zu dem alle die Anfragen geleitet werden. Die zentrale Anordnung erleichtert die Verwaltung und die Erweiterbarkeit um neue Dienste. Da alle Dienstaufrufe von einer zentralen Instanz bearbeitet werden, kann die Leistungsfähigkeit des Systems ein problematischer Faktor bei der Realisierung sein.

Das extreme Gegenbeispiel zum Netz-zentralen Ansatz ist eine rein Endgeräte-basierte Lösung (c), bei der das Netz nur eine Rolle als Datenübertragungssystem spielt. Dieser Ansatz wird im Internet favorisiert und ist durch hohe Innovativität und Flexibilität gekennzeichnet, da die Dienstentwicklung nicht mehr bei den Netzbetreibern liegt, sondern theoretisch von jedem Teilnehmer durchgeführt werden kann. Das Problem ist hierbei die schwierige Verteilung der Software auf die Endgeräte und die konsistente Durchführung von Änderungen.

Die Einführung einer einheitlichen Middleware (d) kann Grundlage für eine verteilte Dienstarchitektur sein, wie bereits in Abschnitt 2.2 beschrieben. Ist die Middleware in jeder beteiligten Einheit (Endgerät, Netzknoten) installiert, wird die Dienstentwicklung von einer einfachen Erweiterbarkeit und Administrierbarkeit profitieren, da die Middleware von der spezifischen Infrastruktur abstrahiert. Noch ist aber die Leistungsfähigkeit der existierenden Middleware-Plattformen für Telekommunikationsdienste nicht erreicht.

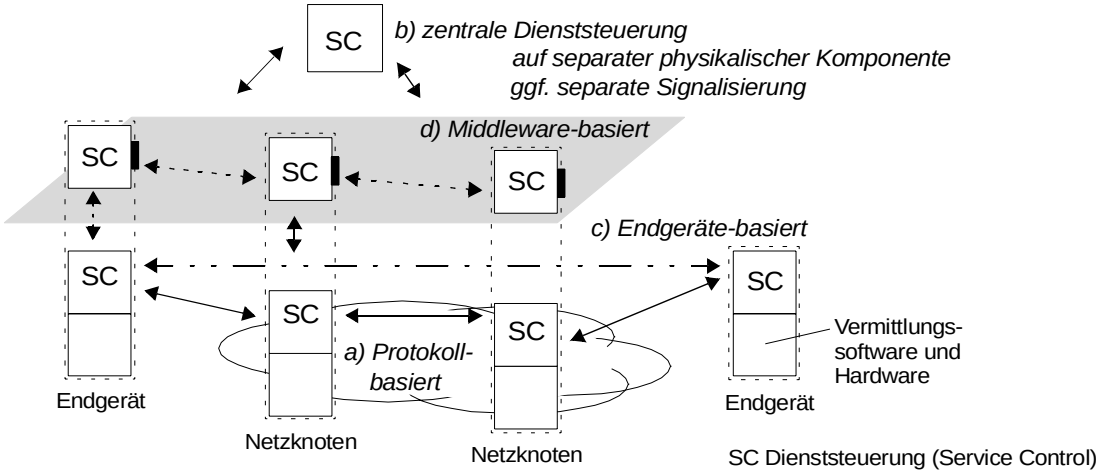


Abbildung 2.4: Alternative Ansätze zur Lokalisierung der Dienststeuerung

2.4 Service Engineering

Der gesamte Prozeß der Dienst- und Systementwicklung, der Diensteinrichtung, der Dienstausführung und des Dienstmanagements, d.h. alle Aktivitäten die den Lebenszyklus von Diensten betreffen, wird unter dem Begriff ***Service Engineering*** zusammengefaßt. Da Dienste fast ausschließlich in Software realisiert werden, ist *Service Engineering* ein Spezialgebiet des Software Engineering [Ree95]. Die eigenständige Betrachtung dieses Entwicklungsvorganges

ist, ebenso wie der Begriff „Service Engineering“, eine noch junge Disziplin [HZ98]. Bisher waren die Dienste mit den Netzen noch zu eng verknüpft. Die Entwicklung einer Dienststeuerung, der Schwerpunkt der vorliegenden Arbeit, ist *eine* Aufgabe im *Service Engineering*. Weitere Aufgaben sind die Entwicklung von einzelnen Diensten oder die betriebswirtschaftliche Planung. Typische Aufgaben und besondere Systemmerkmale, die das *Service Engineering* innerhalb des Software Engineerings kennzeichnen, sind[1]:

- Grundprinzip der verteilten Systeme
- Reaktivität
- Unterstützung eines kurzen Dienstlebenszyklus
- Teilnehmerorientierung (Bedienbarkeit) und hohe Teilnehmerzahl/-interaktion
- Berücksichtigung der Mobilität von Teilnehmern und Endgeräten
- Abbildung der (komplexen) Zustände der zugrunde liegenden Vermittlungssysteme
- Behandlung ungewünschter Interaktionen zwischen gleichzeitig aktiven Diensten (*Feature Interaction*) [Kel99a]
- Interoperabilität mit bestehenden Kommunikationsnetzen und Systemen
- Kommunikationsmechanismen, z.B. Signalisierungsprotokolle
- Einhaltung von Sicherheitsstandards und Zuverlässigkeit
- Werkzeuge zur Dienstentwicklung (*Service Creation Environments*)

Um die Entwicklung der Dienststeuerung, die in der vorliegenden Arbeit vorgenommen wird, als eine der Aufgaben des *Service Engineering* einzuordnen, werden im Folgenden einige charakteristische Aspekte der Softwareentwicklung für Dienste vorgestellt.

2.4.1 Betrachtung des Dienstlebenszyklus unter Kostengesichtspunkten

Für Dienste kann ebenso wie für andere Softwaresysteme ein spezifischer Lebenszyklus angegeben werden [AC227, P103]. Abbildung 2.5 zeigt den schematischen Dienstlebenszyklus eines Dienstes von der Idee bis zur Auflösung. Den typischen Softwareentwicklungsphasen in der Anforderungsanalyse und der Dienstdefinition schließen sich die Phasen des Dienstbetriebes an. Ein installierter und getesteter Dienst kann für verschiedene Benutzer eingerichtet werden (Parametrisierung und Subskription). Die eigentliche Ausführung des Dienstes kann je nach den Nutzungsvereinbarungen beliebig oft erfolgen. Die Bereitstellungsphase endet, wenn der Dienst, d.h. die Software, aus dem Kommunikationssystem entfernt wird. Die oben beschriebenen diensttypischen Anforderungen des Service Engineering sind im gesamten Dienstlebenszyklus zu berücksichtigen.

Durch den harten Wettbewerb sind für den Erfolg einer Dienstarchitektur nicht nur rein technische, sondern auch betriebswirtschaftliche Belange von Bedeutung. Aus betriebswirtschaftlicher Sicht handelt es sich bei einem Dienst um eine Dienstleistung, die in einer vorhandenen Systemwelt (Dienstarchitektur, Netzinfrastruktur, Teilnehmerendgeräte, etc.) realisiert wird. Die Nutzung eines Dienst-Systems muß dabei entsprechend finanziell bewertet werden. Bei

1. Das Teilprojekt C2 „Softwaretechnik für Kommunikationssysteme: Service Engineering“ des bayerischen Forschungsverbundes FORSOFT (HTTP://www.forsoft.de/) beschäftigte sich schwerpunktmäßig mit der Softwareentwicklung für IuK-Dienste. Die hier beschriebenen Ausführungen basieren auf den Projektergebnissen von C2, z.B. [KSS00].

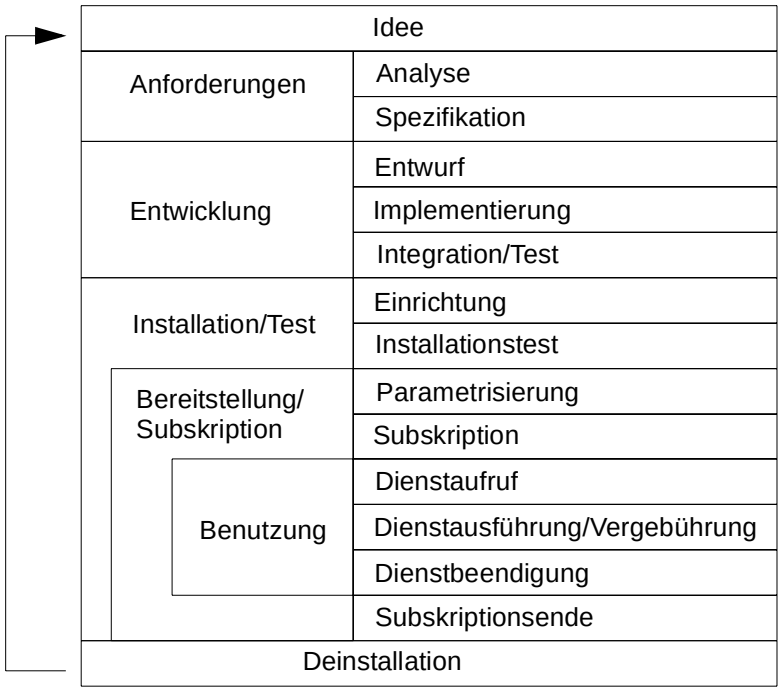


Abbildung 2.5: Dienstlebenszyklus

der Dienstleistung handelt es sich um ein immaterielles Gut. Der Kunde erfährt bei deren Inanspruchnahme einen Nutzen, für den er bereit ist, ein Entgelt zu bezahlen [JK99].

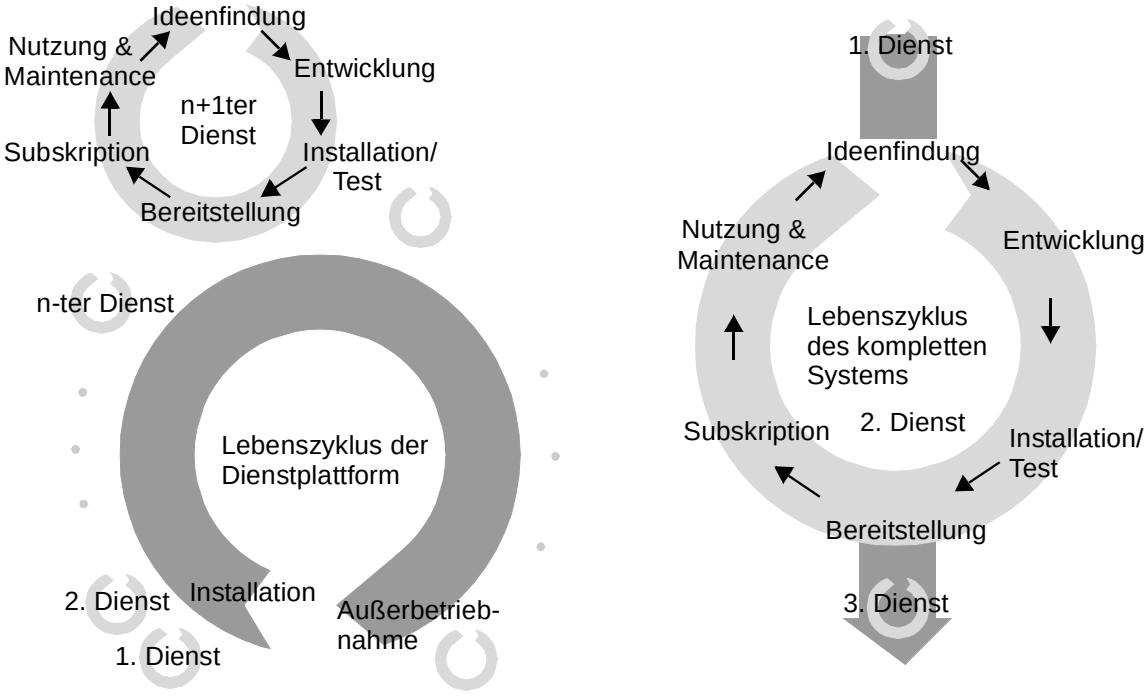
Der allgemeine Dienstlebenszyklus aus Abbildung 2.5 läßt sich in die drei Phasen Vorleistungszyklus, Betreibungszyklus und Nachleistungszyklus unterteilen [JK99]. Die Aktivitäten des Vorleistungszyklus schaffen die Voraussetzung für den Betreibungszyklus. Dazu gehören die Phasen Requirements, Design, Entwicklung und Installation. Hier anfallende Kosten sind einmalige Kosten, während der Betreibungszyklus wiederkehrend anfallende Kosten beschreibt. Diese haben daher einen besonders hohen Einfluß auf die Gesamtkosten. Zum Betreibungszyklus gehören alle Aktivitäten zur Vermarktung und zur Kundenbetreuung. Der Nachleistungszyklus beinhaltet die Aktivitäten zum Abbruch bzw. zur Beendigung des Dienstes und beschreibt einmalig anfallende Kosten. Tabelle 2.2 faßt die Leistungen und die Kosteneinflußgrößen in den drei Phasen zusammen. Wir gehen dabei von einem Dienst aus, der in einem bestehenden System realisiert werden soll.

Phase	**Leistungen**	**Kosteneinflußgrößen**
Vorleistungs-zyklus	Konzeption, Marktforschung Programmierung und Test Systemanpassung /-erweiterung Installation, Schulungsmaßnahmen	Umfang und Komplexität der Software Zukauf von Systemkomponenten Wechselwirkungen mit dem bestehenden System Marktgröße
Betreibungs-zyklus	Marketing Kundenregistrierung/ -betreuung Wartung des Systems	Marktgröße Zahl der Nutzer Datenmenge und Komplexität der Systemumgebung
Nachleistungs-zyklus	Vertragskündigung Herstellung Ausgangszustand	Anzahl bestehender Verträge Umfang der Tests

Tabelle 2.2: Leistungen und Kosteneinflußgrößen im Dienstlebenszyklus

Vergleich der Kosten bei IN-Diensten und herkömmlichen Lösungen

Die Fallstudie zum *Total Cost of Ownership* von Diensten im Intelligenten Netz [JK00] analysiert die anfallenden Kosten im IN-Lebenszyklus und identifiziert die Kostentreiber in einem Vergleich mit herkömmlichen Lösungen für die Bereitstellung derselben Dienste. Letztere zeichnen sich dadurch aus, daß die Dienste-Software dabei in oder an einem Vermittlungsknoten eng verwoben mit der Vermittlungssoftware implementiert ist. Das IN-System stellt demgegenüber einen Switch-unabhängigen, Netz-zentralen Ansatz dar. Die IN-Architektur bietet mit der IN-Plattform, die generische Bausteine (SIBs) anbietet, eine Verkürzung der Entwicklungszeit für Dienste. Allerdings sind höhere Aufwendungen für die einmalige Bereitstellung der Plattform zu tätigen. Der Vorleistungszyklus beim IN ist also durch Plattformentwicklung und einen darauf aufsetzenden, einfacheren Entwicklungszyklus gekennzeichnet (Abbildung 2.6). Für weitere Dienste muß nur dieser durchlaufen werden. Bei herkömmlichen Systemen ist für jeden Dienst der volle Vorleistungszyklus zu durchlaufen. Abbildung 2.7 verdeutlicht, wie sich die höheren Installationskosten für die IN-Plattform bei der Entwicklung mehrerer neuer Dienste amortisieren.



IN-Dienstlebenszyklus *Dienstlebenszyklus herkömmlicher Systeme*

Abbildung 2.6: Dienstlebenszyklen bei IN und bei herkömmlichen Systemen

Die durchgeführte Studie zum Vergleich der IN-Architektur mit herkömmlichen Architekturen hat gezeigt, daß der Einfluß einer Architektur auf den Gesamtaufwand groß ist. Dies gilt insbesondere, wenn mehrere Dienste auf einer Plattform entwickelt werden. Gut strukturierte Architekturen können die Kosten im Dienstlebenszyklus somit entscheidend senken. Ein Mehraufwand bei der Spezifikation der Architektur lohnt sich hier.

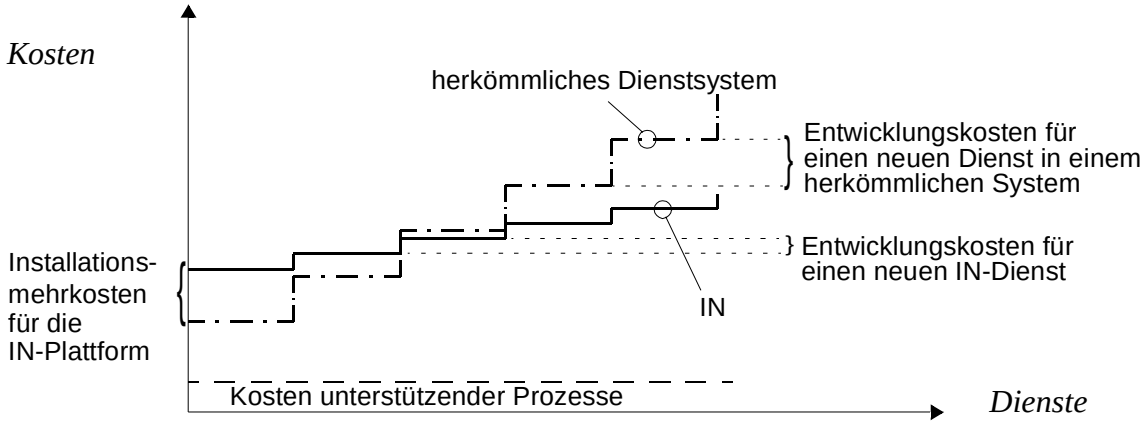


Abbildung 2.7: Vergleich der Kosten für die Dienstentwicklung beim IN mit herkömmlichen Systemen

Es hat sich außerdem gezeigt, daß die Verwaltungskosten (Betreibungszyklus) stark in die Kostenrechnung eingehen, da sie wiederkehrend mit jedem Dienstaufruf bzw. jeder Subskription anfallen. Auch dies ist bei zukünftigen Architekturen durch eine geeignete Konzeption der Teilnehmerverwaltung zu berücksichtigen.

2.4.2 Dienstentwicklung: Vorgehensmodell

Die Dienstentwicklung beschreibt die Phasen Anforderungsanalyse und Dienstdefinition des Lebenszyklus. Um diese Phasen methodisch zu unterstützen, sind bestehende Prozeßmodelle der Softwareentwicklung wie das klassische Wasserfallmodell, das Spiralmodell [Som87], das V-Modell [V01], oder der *Rational Unified Process* [Kru99] an die speziellen Anforderungen der Dienstentwicklung anzupassen. Für die Entwicklung ist typisch, daß ein Dienst selten „from-the-scratch“ als eigenständiger Dienst vollständig neu entwickelt wird. Dies war nur in den Anfangszeiten der Telekommunikation der Fall, als für jeden Dienst ein eigenes Netz konstruiert wurde. Die heutige Dienstentwicklung ist am treffendsten mit einem **kombinierten Vorgehensmodell** zu beschreiben, bei dem ein *Top-Down*-Ansatz mit einem *Bottom-Up*-Ansatz kombiniert ist.

Beim idealen *Top-Down*-Vorgehen wird ausgehend von einer Idee ein Dienstsystem neu entwickelt, ohne daß auf Randbedingungen wie vorhandene Systeme und einsetzbare Technologien Rücksicht genommen werden muß. Bei der Realisierung einer neuen Dienstidee ist meist auf vorhandene Systeme (z.B. Netze, Dienstarchitekturkomponenten) aufzusetzen oder es sind Komponenten zu integrieren (z.B. Abrechnungssystem). Die Eigenschaften und Einschränkungen bestehender Systeme beeinflussen dabei den zu entwickelnden Dienst erheblich. Oft ergibt sich die Entwicklungssituation, daß eine bestehende Systemplattform für neue Dienste geöffnet werden soll. In diesem Fall, da nicht die Dienstidee sondern das existierende System den Ausgangspunkt der Entwicklung bildet, spricht man von einem *Bottom-Up*-Ansatz. Ein Beispiel ist die Erweiterung des digitalen Rundfunks um das Angebot von Informationsdiensten, z.B. Web-Zugriff über DVB-T[1].

Abbildung 2.8 gibt einen Überblick über ein kombiniertes Vorgehensmodell für den Prozeß der Dienstentwicklung [KSS00]. Besonderer Schwerpunkt liegt hier auf den frühen Phasen, da

1. Im Rahmen des Teilprojektes C2 von FORSOFT wurde dieser Dienst unter Verwendung des hier beschriebenen Vorgehensmodells realisiert [SEK99].

dort die richtungsweisenden Entscheidungen getroffen werden. Fehlentscheidungen können in den folgenden Phasen nur noch kostspielig korrigiert werden. Neu an diesem Entwicklungsprozeß ist die Szenarienbildung für die Beschreibung von Dienstideen. Diese Schwerpunktbildung ist auch verstärkt in der objektorientierten Softwareentwicklung zu beobachten (Stichwort: *Use Case*). Darüber hinaus dienen Klassifikationen (siehe Abschnitt 2.1) dazu, die Komplexität der Anforderungsbeschreibung zu reduzieren. Bevor eine Umsetzung in Software erfolgt, ist der Entwicklungsaufwand abzuschätzen, um eine erste Entscheidung über deren Durchführbarkeit treffen zu können (siehe folgender Abschnitt). Aufgrund der Heterogenität der existierenden Dienstarchitekturen (siehe Kapitel 3) ist es im Rahmen des Prozeßmodells nur schwer möglich, das weitere Vorgehen allgemeingültig zu beschreiben. Falls keine „Existierenden Komponenten“ vorhanden sind und eine Dienstarchitektur neu entworfen werden muß, bietet es sich an, wohldefinierte Architekturen als Muster zu verwenden. Der Schwerpunkt der vorliegenden Arbeit liegt auf der Spezifikation einer Dienstarchitektur, die eine möglichst universelle und effiziente Dienstentwicklung erlaubt.

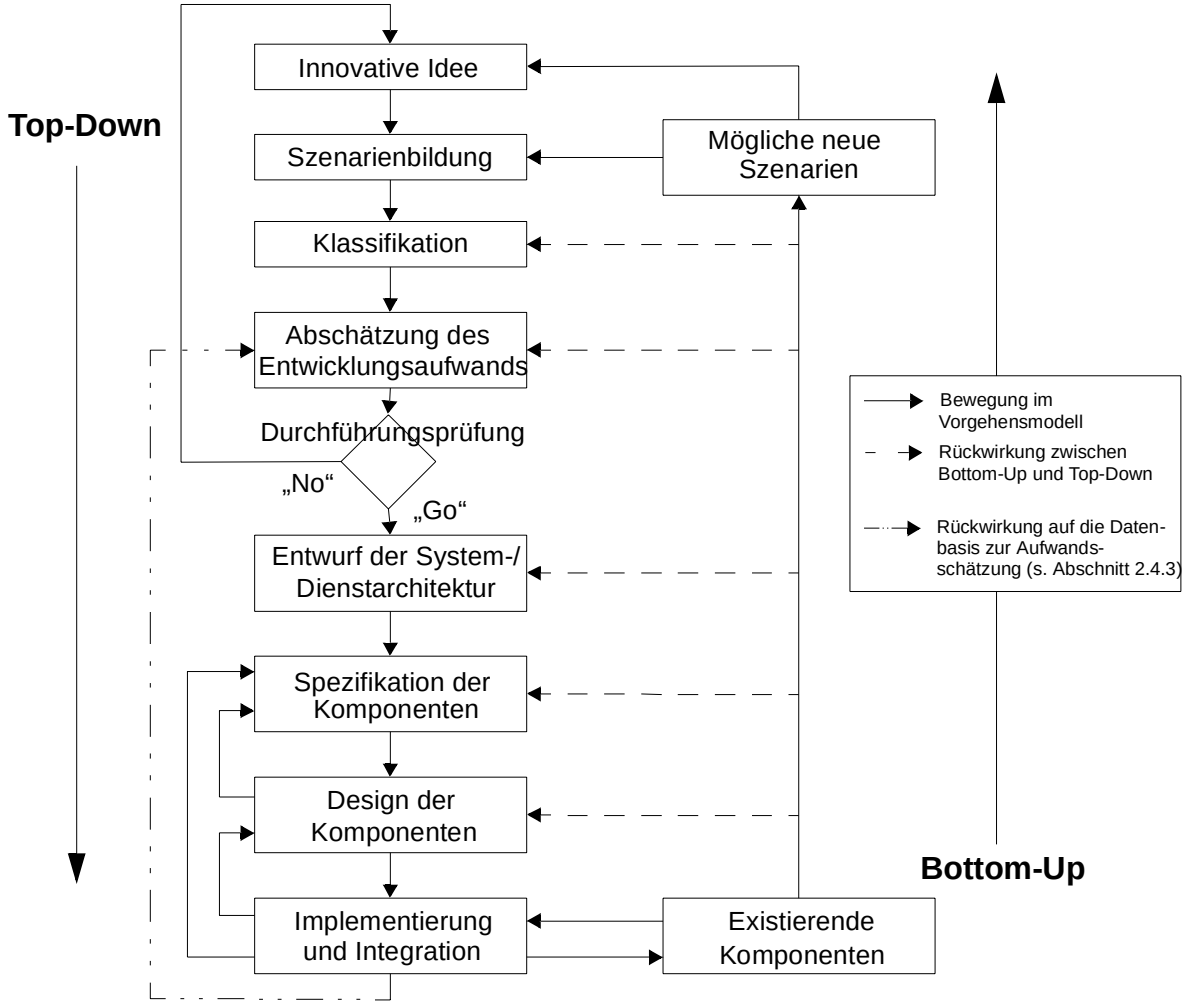


Abbildung 2.8: Vorgehensmodell für die Dienstentwicklung

2.4.3 Eine integrierte Methode für die Kostenschätzung

Um eine fundierte Aussage über die Durchführbarkeit einer Dienstentwicklung treffen zu können, sind mehrere Kostenmanagementmethoden zu einem integrierten Kostenmanagement zu kombinieren. Dabei werden nicht nur die anfallenden Kosten, sondern auch die zu erwartenden Umsätze betrachtet. Die Gegenüberstellung aller Einnahmen und Ausgaben zu einem Zeitpunkt bildet dann die Entscheidungsgrundlage. Als das Ergebnis einer im Rahmen dieser

Arbeit durchgeführten Studie [JK99] für ein System nach dem Prinzip der Intelligenten Netze (IN) wurden zwei Instrumente für die Kostenschätzung mit einem Instrument zur dynamischen Kostenrechnung verknüpft. Mit der *Function Point*-Methode können die Entwicklungskosten und die Integrationskosten eines Dienstes ausgehend von der Requirements-Beschreibung geschätzt werden (*Top-Down*). Die Methode des *Target Costing* [KW95, S. 311] basiert auf der Bestimmung der erzielbaren Einnahmen für einen Dienst, aus denen dann die Grenze für den Gesamtaufwand für eine Dienstentwicklung errechnet werden kann (*Bottom-Up*). Die dynamische Kostenrechnung durch die Kapitalwertmethode [KW95, S. 171] bewertet die Kosten über den Zeitverlauf und bringt die Entscheidungshilfe für verschiedene Zeitszenarien. Bild 2.9 stellt den Zusammenhang dieser drei Instrumente dar. Im Rahmen der Studie wurde die *Function Point*-Methode für IN-Dienste angepaßt.

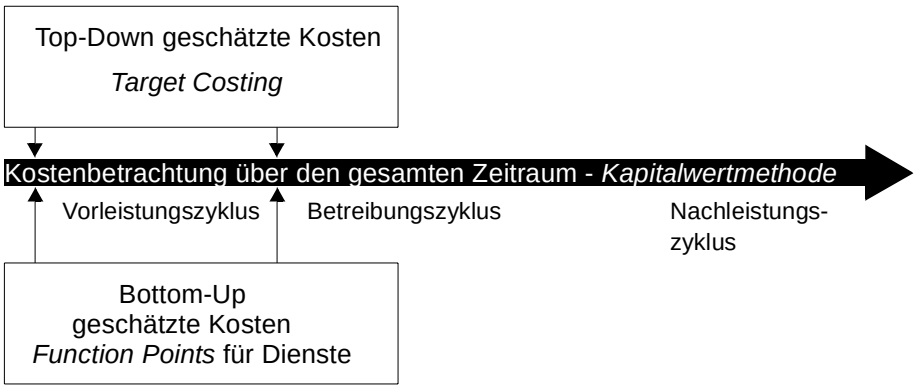


Abbildung 2.9: Integriertes Kostenmanagement [JK99]

Kostenschätzung mit der Function Point Methode

Bei der allgemeinen *Function Point*-Methode wird der zu erwartende Aufwand für die Softwareentwicklung aus den Funktionen abgeleitet, die das Softwaresystem bieten soll. Jeder Funktionstyp (z.B. Datenbankmodul, Graphische Oberfläche) hat dabei einen bestimmten Punktewert, der sich nach der Komplexität der Funktion richtet. Die Summe der Punktewerte gibt die ungewichteten *Function Points*. Aufwandsbeeinflussende Parameter werden über einen technischen Komplexitätsfaktor berücksichtigt, der sich aus einer Bewertung von 14 Einflußgrößen nach ihrer konkreten Bedeutung (0...5) ergibt (z.B. Transaktionsrate). Die ungewichteten *Function Points* werden mit dem technischen Komplexitätsfaktor multipliziert und es ergeben sich die gewichteten *Function Points*. Über eine Erfahrungsdatenbank können die Punkte z.B. in Personenmonate umgerechnet werden, um den Entwicklungsaufwand anzugeben.

In einem IN-System ist die Summe aus der Anzahl an Knoten und an Kanten im SIB-Graphen das Indiz für den Funktionsumfang des Dienstes. Für den technischen Komplexitätsfaktor wurden zwölf aufwandbeeinflussende Größen festgelegt, die mit Werten von 0 (kein Einfluß) bis 5 (hoher Einfluß) belegt werden können: Änderungen an der Dienststeuerplattform (SCF), Änderungen an der Netzplattform (SSF), Protokollanpassungen (INAP), Datenbankanpassungen, zusätzliche neue Komponenten (IP), Transaktionsrate (Anrufe/Zeit), Verflechtung der Komponenten, Dezentralität (Verteilung von Komponenten), *Online Updates/User Interface*, Interaktionen zwischen Diensten, *Billing/Ticketing*, Infrastruktur (Netz). Der Gesamtaufwand ergibt sich wiederum aus dem Produkt der *Function-Points* (Knotenanzahl und Kantenanzahl) und dem technischen Komplexitätsfaktor. Zusätzlich sind je nachdem, ob eine IN-Dienstplattform bereits besteht, noch die Kosten für die IN-Plattform zuzurechnen.

2.4.4 Folgerungen für Dienstarchitekturen

Im *Service Engineering* wird der Lebenszyklus von Diensten unter Berücksichtigung der besonderen Randbedingungen des technischen und betriebswirtschaftlichen Umfeldes betrachtet. Diese umfassende Betrachtung wird bei der Softwareentwicklung von Dienstsystemen oft vernachlässigt. Die oben angestellten Untersuchungen verdeutlichen aber den Zusammenhang von Designentscheidungen für eine Dienstarchitektur hinsichtlich des Dienstentwicklungsprozesses und insbesondere des Kostenmanagements.

Die Dienstentwicklung hängt entscheidend von der Systemplattform ab. Umfassende Dienstarchitekturen können die nötigen Voraussetzungen schaffen, um verschiedene Dienste auf einer Plattform zu etablieren. Die Vergleichsstudie zwischen einer IN-Architektur und einer proprietären Dienstimplementierung hat gezeigt, daß die richtige Gestaltung einer Dienstarchitektur (hier nach dem IN-Konzept) den Entwicklungsvorgang beschleunigt. Es können viele Dienste auf einer Plattform erstellt werden, ohne das Basissystem jedesmal neu erstellen zu müssen. Damit können die Gesamtentwicklungskosten bezogen auf mehrere Dienste erheblich gesenkt werden.

Typisch sind zudem die Kosten, die während der Benutzung eines Dienstes entstehen. Die Studien haben gezeigt, daß diese Kosten einen erheblichen Anteil an den Gesamtkosten haben, da sie wiederkehrend anfallen. Auch hier helfen Managementkomponenten und Teilnehmerverwaltungseinheiten den teuren menschlichen Verwaltungsaufwand (z.B. in Call Centern) zu begrenzen.

2.5 Merkmale einer universellen Dienstarchitektur

Die vorangegangenen Betrachtungen von Diensten und ihrem technischen und betriebswirtschaftlichem Umfeld beinhalten vielfältige Anforderungen, die von einem System zur Dienststeuerung zu bewältigen sind und in Zukunft eine große Rolle bei der Diensterbringung spielen werden. Telekommunikationsdienste haben sich von einfachen Standard-Sprachübermittlungsdiensten zu umfassenden, multimedialen Informations- und Kommunikationsdiensten gewandelt, die in einer persönlichen Ausprägung möglichst überall für die Teilnehmer verfügbar sein sollen.

Doch nicht nur aus der Komplexität der Dienste an sich, sondern auch aus der Kommunikations-Infrastruktur erwachsen neue Anforderungen an die Dienstarchitekturen. Die derzeitige Situation zeigt eine äußerst heterogene Landschaft aus unterschiedlichen Netzen und Endgeräten, die von verschiedenen, konkurrierenden Betreibern angeboten werden. Da die Herausbildung einer einheitlichen Infrastruktur als unwahrscheinlich gilt, kann derzeit eine Integration dieser Systeme für ein umfassendes Angebot nur auf Dienstebene, entkoppelt von den Netzen erfolgen.

Eine ideale Dienstarchitektur hat unter diesen Gesichtspunkten die folgenden Merkmale aufzuweisen (siehe auch [ILM99]).

- **Breites Dienstspektrum:** Eine ideale Dienstarchitektur sollte nicht an einen bestimmten Dienst oder an eine spezifische Dienstklasse gebunden sein. Es müssen sowohl die bestehenden Standard-Basisdienste (z.B. Telefonie) als auch beliebige neue Dienste unterstützt werden. Insbesondere komplexe Multimedia-Dienste (verschiedene Medien, mehrere Teilnehmer) und Dienste, die verschiedene Anwendungsbereiche kombinieren (z.B. Interaktives Fernsehen, Click-To-Dial) vervollständigen das breite Dienstspektrum.

- **Schnelle Dienstentwicklung und Diensteinführung:** Kurze Evolutionszyklen sorgen für eine schnelle Erweiterung um neue Dienste. Dafür ist ein möglichst generisches Dienstmodell Voraussetzung, das die Beschreibung beliebiger, neuer Dienste erlaubt. Es ist von der Architektur zu unterstützen, ohne Änderungen an den Komponenten zu erfordern.
- **Personalisierung:** Es soll die Möglichkeit bestehen, Dienste auf die speziellen Ansprüche der Teilnehmer oder Teilnehmergruppen zuzuschneiden. Zudem hat eine Teilnehmerverwaltung für jeden Teilnehmer jederzeit den Zugang zu seinen Diensten und zur Änderung seiner persönlichen Einstellungen sicherzustellen.
- **Dienste-Mobilität:** Teilnehmer wollen ihre Dienste unabhängig von ihrem Aufenthaltsort, ihrem Netzzugang und ihrem Endgerät nutzen. Darüber hinaus ist eine einheitliche Adressierung eines Teilnehmers sicherzustellen (Unterstützung persönlicher Mobilität).
- **Horizontale Partitionierung (Netzunabhängigkeit):** Eine Dienstarchitektur benötigt exakt definierte horizontale Schnittstellen, um eine funktionale Trennung verschiedener Aufgabenbereiche zu unterstützen, wie z.B. Dienststeuerung, Ressourcensteuerung und Netzsteuerung. Zum Beispiel ermöglicht eine horizontale Partitionierung die Unabhängigkeit der Dienstarchitektur von einer speziellen Netzinfrastruktur.
- **Netzübergreifende Dienststeuerung:** Eine einheitliche Schnittstelle schafft eine netzübergreifende Unabhängigkeit von heterogenen Kommunikationsnetzen. Dienste können auf unterschiedlichen Netzen angeboten werden.
- **Unterstützung verschiedener Rollen in der Diensterbringung (vertikale Partitionierung):** An der Diensterbringung können verschiedene Rollen wie Teilnehmer, Dienstanbieter, Dienstbetreiber und Dienstprogrammierer beteiligt sein. Eine Dienstarchitektur sollte daher geeignete Schnittstellen anbieten, die eine kooperative Teilnahme an der Diensterbringung erlauben. Ein Beispiel ist die Programmierung von Diensten durch den Dienstanbieter, durch Dritte oder durch die End-Teilnehmer selbst. Zur vertikalen Partitionierung zählt insbesondere das Interworking mit anderen Dienstarchitekturen.
- **Leistungsfähigkeit und Skalierbarkeit:** Um alle Teilnehmerzugriffe mit gleichbleibender Qualität abwickeln zu können, müssen Dienstarchitekturen hohe Leistungsanforderungen erfüllen. Sie müssen so strukturiert sein, daß sie für steigende Teilnehmerzahlen ausbaubar sind.
- **Sicherheit:** Die wachsende Menge an Teilnehmern (End-Teilnehmer und Drittanbieter) an einem Dienst stellt erhöhte Anforderungen an die Sicherheit bezüglich Authentisierung, Autorisierung, Abrechnung und Datenschutz.
- **Management:** Dienstarchitekturen müssen Fehlermanagement, Konfigurationsmanagement und weitere Management-Funktionen für eine einfache Einführung, Aktivierung, Deaktivierung und Auflösung von Diensten unterstützen.

2.6 Besondere Anforderungen an eine netzunabhängige Dienstarchitektur

Die Unabhängigkeit der Diensteuerung von verschiedenen heterogenen Kommunikationsnetzen wurde als entscheidende Eigenschaft einer neuen Dienstarchitektur in der bestehenden heterogenen Infrastruktur identifiziert. Sie bildet den Schwerpunkt der vorliegenden Arbeit. Eine vollständig netzunabhängige Dienststeuerung ermöglicht es den Betreibern einer Dienst-

plattform, Dienste auf unterschiedlichen Netzen und in Kombination unterschiedlicher Netze anzubieten. Dabei ist die Dienstbeschreibung entkoppelt von allen Infrastruktur-spezifischen Details. Um eine Dienstplattform einfach einsetzen zu können, sollen keine Änderungen an der Infrastruktur notwendig sein.

Diese charakteristischen Eigenschaften netzunabhängiger Dienstarchitekturen werden im Folgenden an typischen Szenarien illustriert:

Beispiel 1: Multimedia-Tele-Konferenz

Eine Videokonferenz mit drei Teilnehmern soll gestartet werden. Dem Initiator („Alex“) ist unbekannt, wo und in welchen Netzen sich seine beiden Partner befinden. Er übergibt seinem Dienstanbieter den „Konferenz-“wunsch („MyConference“) und die Namen der gewünschten Teilnehmer („Bernd“, „Christa“). Der Dienstbetreiber löst die Namen in eindeutige Bezeichnungen auf, erhält damit die Netzadressen aus einer Datenbank und wählt die Netzressourcen aus, die benötigt werden, um die gewünschte Kommunikationsbeziehung (Konferenz) einzurichten.

Alex und Bernd werden über ISDN verbunden und Christa wird über ihr H.323-VoIP-System (Internet) zugeschaltet. Für die Steuerung der Konferenz auf Verbindungsebene kann auf das Merkmal „Dreier-Konferenz“ im ISDN zurückgegriffen werden. Auf dessen Steuerung hat der Dienstanbieter über eine entsprechende Schnittstelle Zugriff, so daß die Auswahl eines separaten Konferenzservers entfällt. Der Dienstanbieter sorgt lediglich dafür, daß Teilnehmerin Christa über einen entsprechenden ISDN/H.323-Gateway verbunden wird.

Fazit: Durch einen möglichst einfachen Dienstzugang bleibt der Vorgang der Netzauswahl für die Teilnehmer transparent. Das Netz ist nicht nur als reine Verbindungssteuerung zu sehen, sondern soweit möglich, werden die speziellen Merkmale der Netze verwendet.

Beispiel 2: Mobiler Informationsabruf

Während einer Telefonkonferenz zwischen einem H.323-Multimedia-PC im Firmen-Intranet und einem GSM-Telefon (mobiler Teilnehmer) beschließen die beiden Teilnehmer, eine Multimediapräsentation aus dem Internet zu Rate zu ziehen. Während der H.323-Teilnehmer diese über seinen Internet-Anschluß direkt empfangen kann, erhält der mobile Teilnehmer die Präsentation auf seinem ebenfalls mitgeführten Laptop über das terrestrische digitale Rundfunknetz DVB-T.

Fazit: Eine netzunabhängige Dienststeuerung ist insbesondere für die Dienstnutzung durch mobile Teilnehmer wichtig, da sich hier die aktuell verfügbare Infrastruktur (Netze, verfügbare Endgeräte) selbst für denselben Teilnehmer in Abhängigkeit vom Aufenthaltsort ändern kann. Deutlich zeigt sich auch hier, daß die Teilnehmer sich weder um die Netzauswahl noch um die Kombination bestimmter Netze zu kümmern haben.

Beispiel 3: Steuerung des Zugriffs Dritter auf Informationsressourcen

In Anlehnung an das Szenario aus Beispiel 2 ist auch vorstellbar, daß der mobile Teilnehmer seinem Geschäftspartner, zu dem er unterwegs ist, vorab den Zugriff auf eine multimediale Produktsimulation ermöglicht. Alle Details der Netzauswahl und Ressourcensteuerung, um die Daten zum Partner zu bringen, werden von der Dienststeuerung übernommen.

Fazit: Hier wird die Unabhängigkeit der Dienststeuerung besonders deutlich, indem je nach Situation geeignete Netze ausgewählt werden. Um dieses Szenario zu ermöglichen, dürfen

keine speziellen Änderungen in der Infrastruktur (z.B. Endgeräte) notwendig sein, da der Geschäftspartner nicht notwendigerweise ein registrierter Nutzer der Dienststeuerung ist. Die Ressourcen für die Dienstausführung brauchen nicht dieselben zu sein wie die für den Zugang des Dienstinitiators zur Dienststeuerung.

Beispiel 4: Teilnehmerprofilverwaltung

> Über seinen zentralen Kalender kann ein Geschäftsmann seine globale Erreichbarkeit automatisch konfigurieren. Dabei kann er die Änderung seiner persönlichen Einstellungen über unterschiedliche Kommunikationssysteme vornehmen (WWW, Email, Voice).

Fazit: Die zentrale Verwaltung der Teilnehmerprofile ermöglicht einen schnellen Zugriff auf die Teilnehmereinstellungen. Der Teilnehmer wird durch sein Profil in der Dienstebene vertreten. Die Teilnehmerverwaltung kann über unterschiedliche Netze erreicht werden.

Beispiel 5: Interactive Game Community

> Der Dienstanbieter aus Beispiel 3 möchte seinen Kundenkreis erweitern und bietet den Zugang zu einem zentralen Spieleserver an, über den mehrere Teilnehmer gleichzeitig über unterschiedliche Netze und Endgeräte (z.B. PC, TV mit Set-Top-Box, Spielekonsole) an einem Netzspiel partizipieren können. Um diesen Dienst anbieten zu können, sind lediglich kleine Änderungen an der Dienstlogik notwendig (Maximale Teilnehmeranzahl, Spieldauer = Sessiondauer). Netzaspekte oder Teilnehmerprofile sind nicht zu berücksichtigen.

Fazit: Die Dienstlogik ist unabhängig von Infrastruktur-spezifischen Details.

Beispiel 6: Netzauswahl

> Der Dienstanbieter aus Beispiel 2 hat zusätzlich einen Nutzungsvertrag für UMTS abgeschlossen und verfügt über einen entsprechenden Steuerungszugang. Nun kann dem mobilen Teilnehmer die Multimedia-Präsentation auch über UMTS übertragen werden, wenn festgestellt wird, daß der kostengünstige DVB-T Übertragungskanal durch den Broadcast anderer Informationsdaten überlastet ist. Am Dienst selbst ändert dies nichts.

Fazit: Die Dienststeuerung kann um beliebige Netze erweitert werden, ohne daß Änderungen an bestehenden Diensten vorgenommen werden müssen. Auswahlparameter für Netze können neben statischen Eigenschaften auch dynamische Parameter umfassen, wie z.B. Auslastung oder Preise.

Anforderungen

Wie die Szenarien gezeigt haben, lassen sich die in Abschnitt 2.5 aufgestellten Merkmale konkretisieren, um spezifische Anforderungen an die in der vorliegenden Arbeit entwickelte netzunabhängige Dienstarchitektur aufzustellen. Leistungsfähigkeits-, Sicherheits- und Managementanforderungen stehen nicht im Fokus der Arbeit und gehen daher in die folgenden Betrachtungen nicht weiter ein. Eine Zusammenarbeit mit anderen Dienstanbietern (vertikale Partitionierung) ist zwar grundsätzlich vorgesehen, wird aber im Folgenden nicht weiter behandelt.

Horizontale Partitionierung. Für die Realisierung der Netzunabhängigkeit ist eine funktionale Trennung zwischen Dienstebene und Netzebene erforderlich (Beispiele 1, 2 und 3). Die

zugehörige Schnittstelle ist so anzulegen, daß Kommunikationsbeziehungen in unterschiedlichen Netzen und in deren Kombination gesteuert werden können (Beispiel 2).

Um unterschiedliche Netzanbieter zuzulassen, sind darüber hinaus nicht nur die Details der Netzinfrastruktur von der Dienststeuerung abzuschirmen, sondern es dürfen auch spezifische Merkmale der Dienststeuerung keine besonderen Anforderung an die Netze stellen. Das heißt, die Dienstarchitektur darf keine aufwendigen Änderungen in der existierenden Infrastruktur (z.B. Einzug einer Middleware) oder spezielle Plattformen (z.B. intelligente Endgeräte) erfordern (Beispiel 3).

Eine sinnvolle Schnittstelle sollte so gestaltet sein, daß generische Funktionen der Netze (**Netzdienste**) berücksichtigt werden können (z.B. Endgeräte-Mobilität bei Mobilfunknetzen oder Konferenzfähigkeit bei ISDN) und nicht in der Dienstebene aufwendig realisiert werden müssen (Beispiel 1).

Adaptivität. Um nicht nur ausgewählte Netze der bestehenden Infrastruktur zu berücksichtigen, sind Mechanismen vorzusehen, die es ermöglichen, jederzeit Dienste über neue Netze zu steuern (Beispiel 6). Darüber hinaus sollte die Dienstarchitektur auch adaptiv auf Änderungen in der Netzinfrastruktur reagieren können. Die Dienststeuerung muß bei der Auswahl geeigneter Netze stets über deren Verfügbarkeit, Auslastung, ihre Qualitätsparameter und Netzfunktionalität informiert sein (Beispiel 6).

Dienstspektrum und Dienstentwicklung. Netzunabhängige Dienstarchitekturen sind nicht auf eine Klasse von Diensten (z.B. Telefonie oder Fernsehen) in einem Netz festgelegt, sondern ermöglichen ein sehr breites Spektrum an Diensten. Um dies in der Dienstentwicklung berücksichtigen zu können, ist ein generisches Dienstmodell zugrundezulegen, in dem Dienste unabhängig von Infrastruktur-spezifischen Details beschrieben werden können (Beispiel 5). Das Dienstmodell bildet die Grundlage für die Spezifikation und Programmierung von Diensten, die für die Teilnehmer zur Dienstausführung instanziert werden. Um die Wiederverwendbarkeit von Diensten nicht nur für verschiedene Netze, sondern auch für alternative Teilnehmer zu gewährleisten, ist es zudem sinnvoll, das Dienstmodell unabhängig von Teilnehmer-spezifischen Details zu machen.

Personalisierung und Dienste-Mobilität. Die Teilnehmer verlangen Dienste, die auf ihre persönlichen Wünsche zugeschnitten sind und auf die sie auch in einem mobilen Umfeld jederzeit zugreifen können (Beispiele 2 und 3). Um dieser Anforderung in einem heterogenen Umfeld Rechnung zu tragen, ist eine eigenständige Teilnehmerverwaltung vorzusehen, die den Teilnehmern einen Netz- und Endgeräte-unabhängigen Zugang zu ihren persönlichen Einstellungen und zu ihren Diensten ermöglicht (Beispiel 4). Die Teilnehmerverwaltung schirmt die Dienststeuerung von Teilnehmer-spezifischen Details ab (z.B. Endgerät, Signalisierungsprotokoll, sowie auch Registrierung, Vergebührung) und ist getrennt von der Dienstausführung zu realisieren (Beispiele 3 und 4).

Zusammenfassung der Anforderungen

- Strikte Trennung von Dienststeuerung und Netzinfrastruktur
- Netzübergreifende Dienststeuerung
- Unabhängigkeit des Systems von spezieller Infrastruktur
- Berücksichtigung spezieller Netzfunktionen (Netzdienste) durch die Dienststeuerung
- Einfache Erweiterbarkeit des Systems um neue Netze

- Adaptivität hinsichtlich sich ändernder Netzinfrastruktur und Erweiterungen der Netzfunktionalität
- Generisches Dienstmodell
- Unabhängiger Teilnehmer- und Dienstzugang getrennt von der Dienststeuerung

Diese Anforderungen dienen als Grundlage für die Spezifikation der Serverarchitektur in der vorliegenden Arbeit. Sie sind gleichzeitig die Kriterien für die Analyse und den Vergleich von Dienstarchitekturen in Kapitel 3.

Kapitel 3
Vergleich und Bewertung von Architekturen zur Dienststeuerung

Ziel des folgenden Kapitels ist es, den Stand der Technik von Dienstarchitekturen zu erläutern und diese anhand des zuvor aufgestellten Funktionskataloges zu bewerten. Zunächst werden grundlegende Architekturkonzepte und Standards diskutiert, um die Anwendung der in Kapitel 2 vorgestellten Prinzipien zu zeigen und die Evolutionsschritte bei der Entwicklung der Dienstarchitekturen herauszustellen (Abschnitte 3.1 bis 3.5). Die Bewertung erfolgt zum einen aus der Sicht eines Dienstanbieters und zeigt die Möglichkeiten, die diesem von der Dienstarchitektur bereitgestellt werden. Hier werden speziell die Merkmale horizontale Partitionierung und Dienstentwicklung/Dienstspektrum herangezogen. Aus Teilnehmersicht sind zum anderen die Merkmale einer flexiblen Teilnehmerverwaltung wie Personalisierung und Dienste-Mobilität wichtig. Abschnitt 3.6 geht dann speziell auf Architekturansätze und Projekte ein, welche die in dieser Arbeit gestellte Aufgabe der netzunabhängigen Steuerung von Diensten behandeln. Diese Ansätze werden auf ihre Realisierung der Netzunabhängigkeit nach den in Abschnitt 2.6 festgelegten Anforderungen untersucht.

Bei der Beschreibung von Dienstarchitekturen lassen sich zwei grundlegende Evolutionslinien unterscheiden (siehe auch [Kel98, KQ98]):

Eine Gruppe von Dienstarchitekturen kommt aus dem Anwendungsbereich der Telekommunikation. Hier liegt der Schwerpunkt auf den Netzen und den darin enthaltenen Einheiten (Server) zur Steuerung von Diensten (Intelligenz in den Netzen, Abschnitt 3.1 - 3.3). Die Evolution geht von rein dienstspezifischen Systemen (z.B. das analoge Telefonnetz) über Dienste-integrierende Systeme (ISDN, B-ISDN) hin zu Ansätzen, die eine Trennung von Diensten und Netzen beinhalten. Das Konzept der Intelligenten Netze (IN) beschreibt eine Dienststeuerung für Zusatzdienste in der Telefonie, die in einem Netz-zentralen Server getrennt von Vermittlungsknoten realisiert ist. Hingegen erlauben Konzepte für verteilte Dienststeuerungen wie z.B. TINA eine größere Dienstvielfalt und entkoppeln durch die Einführung einer Middleware die Netzinfrastruktur deutlicher von der Dienststeuerung.

Die zweite Gruppe betrachtet das Netz als bloßen Transportkanal für beliebige Daten und sieht die Dienststeuerung in den Endgeräten. Ansätze dieser Gruppe kommen aus dem Internet-Bereich (Abschnitt 3.4). Internet-basierte Dienstarchitekturen sind durch ihren stark dezentralen Charakter gekennzeichnet, da die Dienststeuerung in den Endgeräten erfolgt. Eine Entkopplung von Dienststeuerung und Netz ist somit erreicht. Jedoch müssen für Telekommunikationsdienste zusätzliche Funktionen (z.B. für eine Ende-zu-Ende Qualitätsgarantie, für

Adressumsetzung oder für die Vergebührung) im Netz bereitgestellt werden, die wieder eine Abhängigkeit zwischen Dienst und Netz schaffen. Die Prinzipien von Internet-basierten Dienstarchitekturen werden anhand der Architekturen zur Internet-Telefonie erläutert (SIP, H.323, und Megaco).

Durch die Verschmelzung der Bereiche Telekommunikation und Internet entsteht darüber hinaus eine dritte Gruppe von Dienstarchitekturen, die eine Entkopplung von Diensten und Netzen umfassender verwirklichen, als dies in den einzelnen Bereichen der Fall ist. Diese Architekturen weisen dedizierte Programmierschnittstellen auf, sogenannte APIs (Parlay, JAIN), die einen Zugriff externer Dienstanbieter auf unterschiedliche Netze unterstützen (Abschnitt 3.5).

Aus vielen der bisher aufgezählten Architekturen sind weiterführende Ansätze entstanden, die netzunabhängige und netzübergreifende Dienstarchitekturen beschreiben (Abschnitt 3.6). Agenten-orientierte Konzepte oder programmierbare Netze bringen neue Lösungsmöglichkeiten, deren Eigenschaften anhand von Beispielarchitekturen bewertet werden.

3.1 Verteilte, protokollbasierte Dienstarchitekturen

Viele heutige Kommunikationssysteme, wie zum Beispiel das analoge Telefonsystem oder das X.25-Paketdatennetz, bestehen aus einem monolithischen System, das auf einem spezialisierten Netz nur einen Dienst anbietet. Die Digitalisierung der Informationskodierung, des Informationstransports und der Vermittlungstechnik hat dazu geführt, daß an einem Teilnehmer-Netzanschluß (UNI) mehrere Dienste angeboten werden können. Das *Integrated Services Digital Network* (ISDN) [Boc97] bietet verschiedene Dienste wie Sprachtelefonie, Telefax oder Datentransfer über leitungsvermittelte 64kbit/s-Verbindungen. Für multimediale Dienste wurde Mitte der achziger Jahre das Breitband-ISDN (B-ISDN) standardisiert. Die Standardisierung geht von geschlossenen Netzen aus und sieht demzufolge keine Schnittstellen zu externen Anbietern vor.

Beim Schmalband-(64kbit/s-)ISDN ist die Dienststeuerung in das Signalisierungsprotokoll der Vermittlungsschicht (Schicht 3 des OSI-Modells) [Q.931] eingebettet. Es liegt eine verteilte Dienststeuerung vor, da der Q.931-Protokollautomat in jedem Vermittlungsknoten vorhanden ist. Demzufolge ist für jeden neuen Dienst die Software aller Vermittlungsknoten anzugleichen, was mit einem erheblichen Aufwand verbunden ist.

3.1.1 Breitband-ISDN

Das Breitband-ISDN wurde evolutionär aus dem Schmalband-ISDN entwickelt und von der ITU-T in der Q.2xxx-Serie und sehr ähnlich dazu vom ATM-Forum [ATMF] standardisiert. Die Signalisierung der ersten Evolutionsstufe (*Capability Set* 1, [Q.2931]) entspricht architektonisch vollständig der des Schmalband-ISDN und unterstützt Punkt-zu-Punkt Verbindungen mit fester Konfiguration der Dienstparameter.

Im *Capability Set* 2 wird eine funktionale Trennung eingeführt, die die Rufsteuerung [Q.2982] von der Verbindungssteuerung [Q.2983] trennt, um neue Funktionen wie Punkt-zu-Mehrpunkt Verbindungen, Mehrfachverbindungen, variable Übertragungsbitraten und die Aushandlung von Verbindungsparametern zu unterstützen (Abbildung 3.1). Die Dienststeuerung bleibt aber eng mit der Verbindungssteuerung und der Netzinfrastruktur gekoppelt. Das Signalisierungsprotokoll der Rufsteuerung bestimmt auch beim CS 2 die Dienstfunktionalität, aber im Gegensatz zu CS 1 beschränkt sich die Rufsteuerung auf die Zugangsvermittlungsknoten. Trotzdem bedeutet dies einen hohen Aufwand bei der Einführung neuer Dienste. Die ITU-T beschreibt in den Empfehlungen I.130, I.140 und I.210 ein Vorgehen für die Dienstentwicklung für ISDN-

Systeme [I.130]. Dabei sind für jeden Dienst neue Protokolle zu entwickeln bzw. bestehende anzupassen, die auf allen Vermittlungs- bzw. Rufsteuerknoten implementiert bzw. integriert werden müssen. Diese aufwendige Dienstentwicklung ist nur möglich, da das ganze System in der Hand eines einzigen Betreibers liegt. Für den Teilnehmer existiert ein breites Spektrum an unterstützten Diensten wie Breitband-Videokonferenz oder Multimedia-Dokumentenabruf [Boc97], die dieser über sein spezielles Endgerät an seinem Netzanschluß aufrufen kann. Da es sich bei diesen Diensten um Standarddienste handelt, sind personalisierte Dienste und die Unterstützung von Mobilität kaum realisierbar.

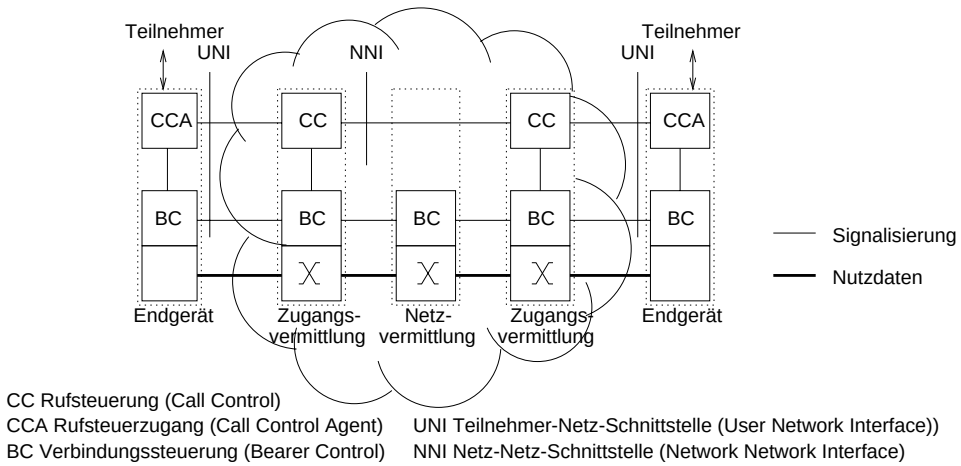


Abbildung 3.1: Architektur des B-ISDN (CS-2)

3.1.2 Die Signalisierungsarchitekturen MAGIC und AMSA

Ausgehend von der oben beschriebenen Standardisierung wurden die protokollorientierten ISDN-Architekturen in verschiedenen Forschungsprojekten weiterentwickelt. Das RACE Projekt MAGIC (*Multiservice Applications Governing Integrated Control*) führt die Ressourcensteuerungsebene als eine weitere funktionale, horizontale Trennung ein (Abbildung 3.2). Ein objektorientiertes Rufmodell bildet die Schnittstelle zu einer Endgeräte-basierten Dienststeuerung [Kni93]. Anstelle der bisher üblichen Beschreibung des Rufzustandes durch einen einzigen Zustandsautomaten enthält das objektorientierte Rufmodell verschiedene Objekte wie Teilnehmer, Dienstkomponente und Abbildungselement, deren Gruppierung und deren Parameterbelegung den Rufzustand festlegen. Eine Zustandsänderung wird durch Hinzufügen, Modifizieren oder Löschen eines oder mehrerer Objekte beschrieben. Dieses Rufmodell erleichtert dem Dienstanbieter die Beschreibung komplexer, multimedialer Dienste unabhängig vom Netz. Die Dienststeuerung umfaßt nur die Einrichtung dieser Beschreibung im Netz. Die Dienststeuermodule in den Teilnehmerendgeräten, die diese Beschreibung erzeugen, sind nicht weiter ausgeführt.

Die Architektur AMSA [Mül96] erweitert die MAGIC-Architektur, indem die funktionale, horizontale Trennung weiter verfeinert wird. Größter Unterschied ist die Zentralisierung der Ruf- und der Ressourcensteuerung in AMSA. Über ein an MAGIC angelehntes objektorientiertes Rufmodell bietet die Architektur einen „Universaldienst“ an, der von verschiedenen, in der Arbeit nicht detailliert beschriebenen Dienststeuerungen verwendet werden kann.

Fazit

Zusammenfassend läßt sich feststellen, daß protokollbasierte Dienstarchitekturen auf die Marktsituation eines vorherrschenden Monopol-Anbieters, des nationalen Telekommunikationsbetreibers, zugeschnitten sind, ohne eine Zusammenarbeit mehrerer Betreiber zu berück-

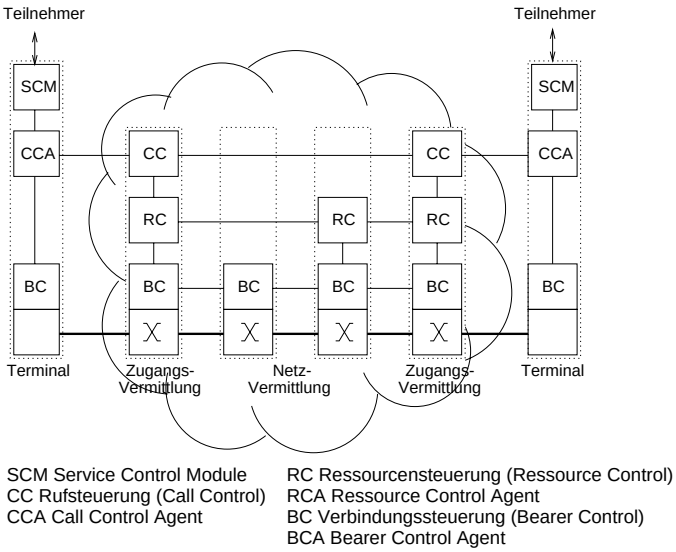


Abbildung 3.2: MAGIC-Architektur

sichtigen. Die Dienstarchitekturen sind daher für alternative Netze oder für eine netzübergreifende Dienststeuerung schlecht geeignet.

3.2 Dienstarchitekturen mit getrennter Dienst- und Rufsteuerung

Um Zusatzdienste für die Telefonie anbieten und schnell realisieren zu können, wurden Architekturen geschaffen, in denen die Steuerung der Zusatzdienste von der Steuerung des Basisdienstes getrennt ist. Das Konzept der Intelligenten Netze (IN) beschreibt ein System, das seit den achtziger Jahren für öffentliche Netze Anwendung findet. Es wird im Folgenden als Beispiel für derartige Architekturen analysiert.

3.2.1 Intelligente Netze

Das *Intelligent Network* (IN) [FGK97] ist ein Architekturkonzept, das die realzeitige Ausführung von Zusatzdiensten in einer verteilten, protokollbasierten Dienstumgebung gewährleistet. Im Gegensatz zu herkömmlichen Lösungen, die eine solche Dienststeuerung direkt an einem Vermittlungsknoten realisieren, sieht das IN-Konzept einen zentralen Steuerserver vor, der über ein eigenes Signalisierungsprotokoll mit allen relevanten Vermittlungsknoten verbunden ist. Auf diese Weise wird eine einfache und schnelle Dienstentwicklung unterstützt, da Dienste nur an einer Stelle im Netz implementiert werden. Die Motivation für die Einführung des IN war, Unabhängigkeit von den Herstellern der Vermittlungssysteme zu erzielen. An eine Dienstanbieter-übergreifende Zusammenarbeit wurde nicht gedacht.

Neben einer Standardisierung durch die ITU-T in der Q.12xx Serie [Q.12xx] existieren weitere Standardisierungen für Intelligente Netze bei Bellcore (*Advanced Intelligent Network*) und bei ETSI. Die folgenden Aussagen stützen sich auf die ITU-T Standards. Die Evolution der IN-Standards drückt sich in *Capability Sets* aus. Derzeit gültig ist CS 2. Die *Capability Sets* unterscheiden sich vor allem in der Funktionalität, die dem Steuerserver im IN-Signalisierungsprotokoll zur Verfügung steht, und im Dienstmodell. Für eine Analyse nach den oben aufgestellten Kriterien reicht es aus, sich auf die Grundprinzipien zu beschränken, die in beiden *Capability Sets* gleich sind.

Abbildung 3.3 zeigt das Prinzip der IN-Architektur. Bestimmte Vermittlungsknoten werden um *Service Switching Functions* (SSF) erweitert. Diese bilden die Zustände des Basistelefon-

dienstes, der von den Vermittlungsknoten erbracht wird, in ein *Basic Call State Model* (BCSM) ab. Dies erlaubt eine einheitliche Sicht auf die Vermittlungsvorgänge. Für bestimmte Ereignisse (z.B. Wahl einer bestimmten Rufnummer) können Triggerpunkte im BCSM gesetzt werden, deren Auslösen einen Aufruf an die Dienststeuerung (*Service Control Function*, SCF) im zentralen Server bewirkt. Die Meldungen, die zwischen SSF und SCF zur Dienststeuerung ausgetauscht werden, sind im Signalisierungsprotokoll INAP (*Intelligent Network Application Part*) [Q.1228] festgelegt. Die INAP-Meldungen werden dabei über ein von der Basistelefonie getrenntes Signalisierungsnetz, das Signalisierungssystem Nr. 7 (SS#7) übertragen. Die SCF kann über das INAP-Protokoll Zusatzdienste steuern, indem sie z.B. geänderte Rufnummernziele setzt, Ansagen mittels zusätzlicher Ressourcen (*Intelligent Peripheral*, IP) schaltet, die Vergebührung beeinflußt oder neue Triggerpunkte setzt. Die *Service Management Function* (SMF) stellt die Funktionen zur Einrichtung und Verwaltung der Dienste in der Dienststeuerung (SCF) bereit. Die *Service Data Function* (SDF) dient der Speicherung Dienst-bezogener Daten (z.B. Rufnummern). Eine *Service Creation Environment Function* (SCEF) stellt eine Programmierumgebung für IN-Dienste dar.

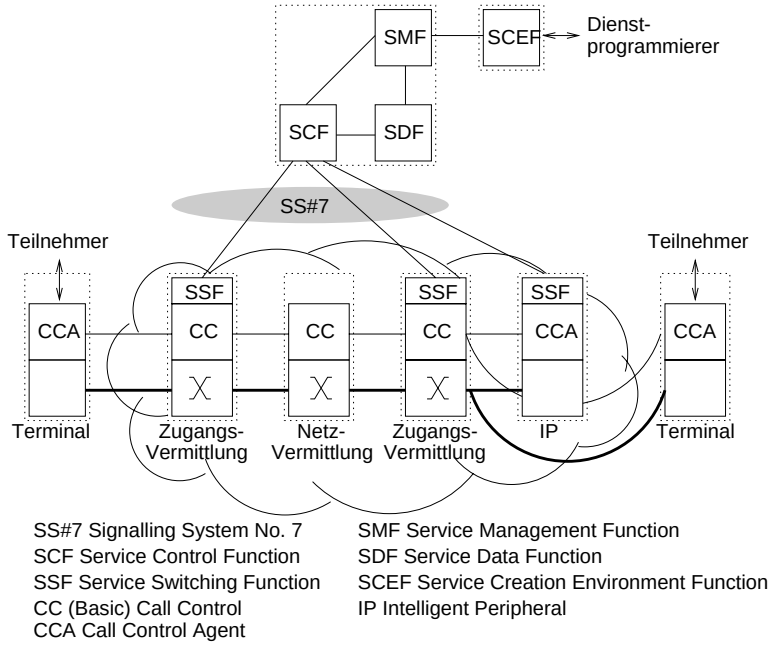


Abbildung 3.3: IN-Architektur

Wie beim ISDN sind Netz- (SSF) und Dienstbetreiber (SCF) eine Einheit, da die Dienststeuerung nur bedingt unabhängig vom Netz ist. Die Funktionalität der Dienststeuerung ist eng an das Rufmodell in der *Service Switching Function* und an den INAP-Meldungssatz gekoppelt, d.h. die Funktionalität wird von den Eigenschaften der Netze (PSTN, ISDN, GSM) bestimmt. Zudem erschweren verschiedene, proprietäre Erweiterungen des INAP-Protokolls die Zusammenarbeit verschiedener Betreiber.

Der potentielle Funktionsumfang von Zusatzdiensten ist somit auf die Möglichkeiten des Rufmodells (siehe Abbildung 3.4) und der INAP-Meldungen beschränkt. Mögliche Zusatzdienste reichen beim Capability Set 2 von *Freephone*-Diensten über *Virtual Private Network*-Dienste bis zu Konferenzdiensten.

Die Dienstentwicklung wird beim IN durch ein Bausteinkonzept entscheidend vereinfacht. Dienste setzen sich aus modularen Bausteinen, sogenannten *Service Independent Building Blocks* (SIB), zusammen. Diese werden durch einen Ablauf-Graphen zu einem Dienst verbunden. Für die Dienstentwicklung existieren umfangreiche *Service Creation Environments*

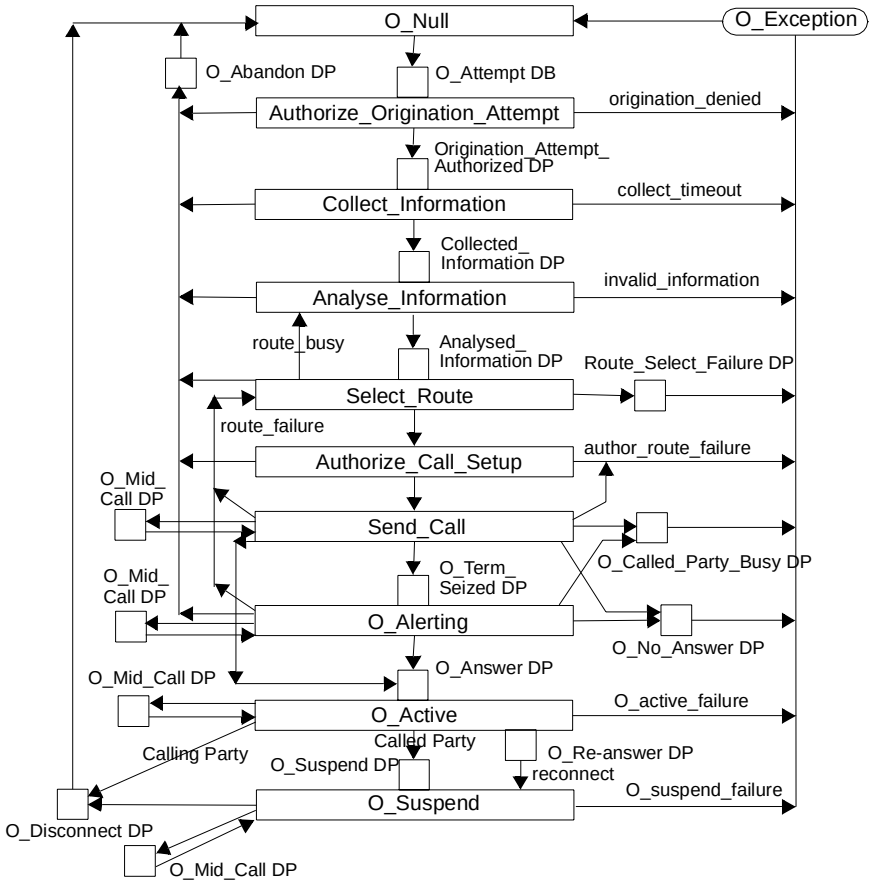


Abbildung 3.4: BCSM IN CS-2 Rufende Seite mit Triggerpunkten [Q.1228]

(SCE), die ein komfortables Programmieren der Dienstlogik zulassen. Die SIBs sind parametrisierbar und erlauben das Einrichten personalisierter Dienste sowie persönlicher Mobilität durch persönliche Rufnummern. Viele IN-Anbieter stellen einen Web-basierten Teilnehmerzugang zur Verfügung, über den die Einstellungen selbst modifiziert werden können.

Der Dienstzugang für die Teilnehmer beschränkt sich beim IN nur auf das Netz, in dem ein IN-System (SSF) implementiert ist. Dienste können nicht direkt aufgerufen werden, sondern das Vorliegen eines Zustandes, der einen Aufruf an die SSF auslöst, wird durch Triggerpunkte (*Detection Point*s/DP in Abbildung 3.4) erkannt.

3.2.2 Broadband Intelligent Network

In [VH98] werden Untersuchungen und eine prototypische Implementierung aus dem ACTS Projekt INSIGNIA (AC068) beschrieben, die das IN-Konzept auf eine Steuerung von multimedialen Diensten erweitern. Basis ist ein komplexes Multimedia-Rufmodell. Dabei zeigt sich das Problem, daß zum einen der Zustandsraum für das Rufmodell schnell sehr stark anwächst und kaum mehr beherrschbar ist. Die wachsende Komplexität ist bereits im Zustandsautomaten für das BCSM der rufenden Seite im IN-CS2 zu erkennen, das auf Sprachdienste beschränkt ist. Zum anderen ist es schwer möglich, eine generische Menge an SIBs zu finden, mit der möglichst alle zukünftigen Dienste entwickelt werden können.

Fazit

Bei der IN-Architektur wird die Dienststeuerung von einem Rufmodell bestimmt, das die Vermittlungsvorgänge eines speziellen Netzes modelliert. Daher kann keine Netzunabhängigkeit

erzielt werden. Der zentrale Ansatz für die Dienststeuerung erweist sich für eine effiziente Dienstentwicklung als günstig.

3.3 Verteilte, Middleware-basierte Dienstarchitekturen

Bei den hier beschriebenen Architekturen wird die Dienststeuerung fast vollständig von der Netzinfrastruktur entkoppelt, indem eine Middleware-Schicht (siehe auch Abschnitt 2.2) eingezogen wird, über die die verteilten Komponenten der Dienststeuerung kommunizieren. Verschiedene Projekte (z.B. EURESCOM [P103], ANSA [Lin93], RACE R2049 [Tri95], TINA) verwenden den Middleware-Ansatz als zentralen Bestandteil der Dienstarchitektur. Die TINA-Architektur ist der bekannteste Vertreter dieser Klasse.

Telecommunications Information Networking Architecture TINA

Während das Konzept der Intelligenten Netze aus Überlegungen resultiert, die ausgehend von einer vorhandenen Netzinfrastruktur, d.h. *Bottom-Up*, entstanden sind, wurde bei der *Telecommunications Information Networking Architecture* [TINA-C] ein generischer Ansatz verfolgt, bei dem versucht wurde, die Vorteile aus der Telekommunikation und der Informationstechnik zu kombinieren. Tatsächlich vereint TINA das Steuerungskonzept des IN, und die Managementkonzepte und Abstraktionsebenen aus dem *Telecommunication Management Network* [M.3010]. TINA basiert auf dem *Open Distributed Processing*-Referenzmodell (RM-ODP) [X.901] und beschreibt Dienste durch das Prinzip der objektorientierten, verteilten Programmierung als interagierende Komponenten, die Dienstlogik und Daten beinhalten und ihre Funktionen über eine Middleware bereitstellen.

Ziel der TINA-Initiative war es, eine globale Architektur für multimediale Telekommunikationssysteme basierend auf modernster Software-Technologie zu schaffen. TINA wurde dazu von 1993 bis 1997 von einem TINA-Core Team bestehend aus Mitarbeitern der größten Firmen der Telekommunikations- und Informationstechnik und zahlreichen TINA-Auxiliary-Projects grundlegend spezifiziert [ILM99] und danach in verschiedenen TINA Working-Groups weiterentwickelt [BHG00]. Die Arbeiten der TINA-Initiative selbst wurden Ende 2000 offiziell eingestellt. Die Ergebnisse werden in vielen unabhängigen Projekten weitergeführt.

Ausgangspunkt der TINA-Architektur bildet ein Geschäftsmodell (*Business Model*), das alle an einem Dienst beteiligten Rollen darstellt. Über dieses Modell werden die Schnittstellen zwischen den verschiedenen Bereichen der TINA-Architektur definiert (Abbildung 3.5). Der *Retailer* koordiniert den Zugang der Teilnehmer (*Consumer*) zu den Diensten. Er kann Dienste selbst anbieten oder diese an *Third Party Service Provider outsourcen*. Der Aufbau von Verbindungen wird von *Connectivity Providern* ausgeführt. Der *Broker* spielt eine Vermittlerrolle

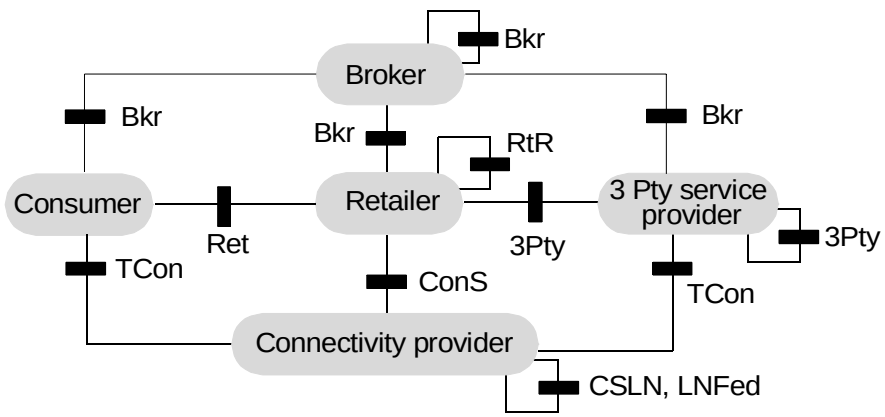


Abbildung 3.5: TINA Business Model [ILM99]

zur Auffindung von Dienstanbietern. Bisher sind von TINA-C lediglich die Schnittstellen Ret, ConS und TCon explizit spezifiziert worden [BHG00]. *Broker* und *Third Party Service Provider* sind in TINA nicht detailliert beschrieben. Für die anderen Rollen wurden in TINA explizit Komponenten spezifiziert, deren Verhalten durch ihre Schnittstellen definiert ist.

Die TINA-Architektur basiert auf einem Softwaremodell, das für alle Komponenten (Endgeräteapplikationen, Dienstserver, Vermittlungsknoten) gleich gilt und das es dem Dienstanbieter erlaubt, Dienste aus verteilten, interagierenden Komponenten zusammenzusetzen. Zur Strukturierung werden drei Hauptbestandteile unterschieden: Die *Computing Architecture* beschreibt die verteilte Verarbeitungsumgebung (*Distributed Processing Environment*, DPE), die Middleware. TINA-DPE ist eine Erweiterung der Middleware-Technologie CORBA der *Object Management Group* [OMG]. Die *Service Architecture* beschreibt eine komponentenbasierte Plattform für die Bereitstellung von Diensten. Die Steuerung und Verwaltung der Netzressourcen wird in der *Network Resource Architecture* beschrieben. Im Folgenden wird nur auf die im Zusammenhang mit dieser Arbeit relevanten Eigenschaften von TINA eingegangen.

Die TINA-*Service Architecture* definiert konkrete Komponenten, die für alle Dienste verwendet werden oder in denen die spezifische Dienstlogik realisiert wird. Zur weiteren funktionalen Unterteilung werden die Signalisierungsabläufe in Sessions unterteilt. TINA unterscheidet eine *Access Session*, die den Teilnehmerzugang abwickelt, eine *Service Session*, die die Dienstausführung steuert, und eine *Communication Session*, in der die Netzressourcen koordiniert werden. Abbildung 3.6 zeigt die Komponenten der TINA-Architektur und deren Zusammenhang mit den Sessions.

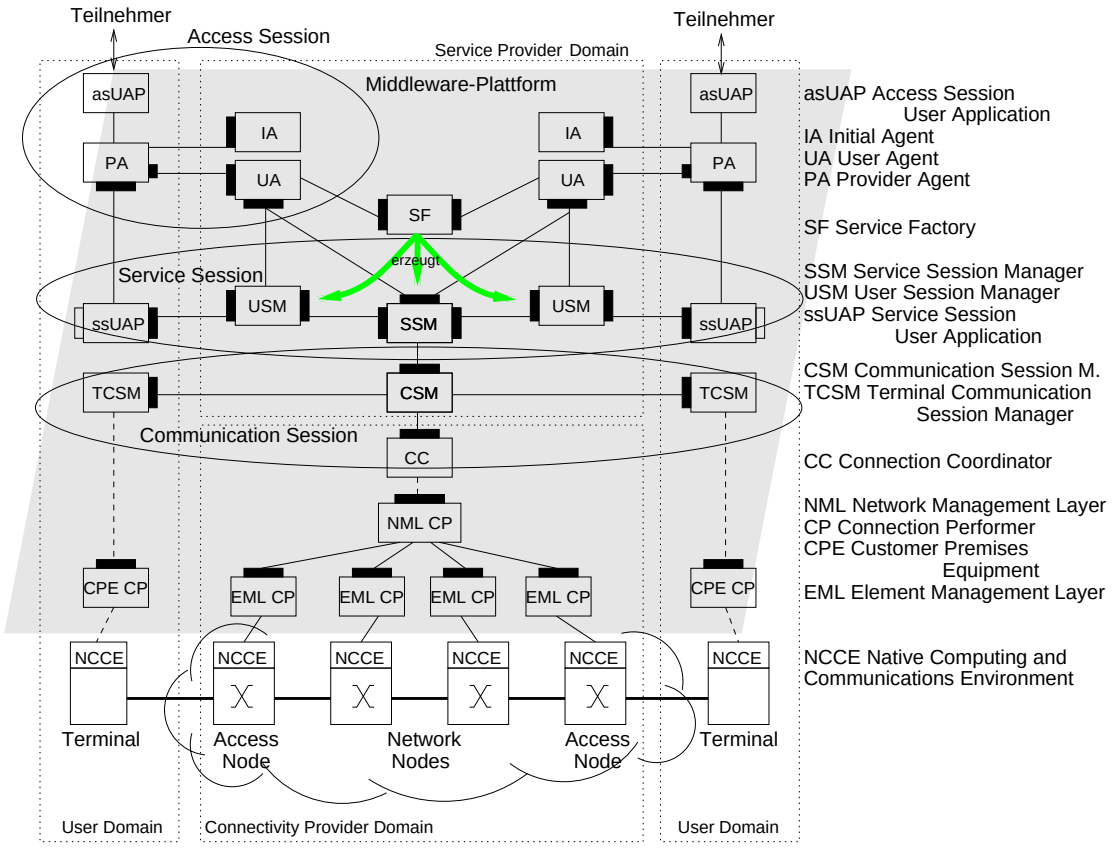


Abbildung 3.6: TINA-Dienstarchitektur

Die zentrale Komponente der *Service Session*, d.h. der Dienststeuerung, ist der *Service Session Manager* (SSM). Er wird für jeden aktiven Dienst instanziert, um den Dienstablauf zu steuern. TINA enthält keine Angaben, wie die Dienstlogik zu realisieren ist, gibt aber ein objektorientiertes Modell an, das die Zustände der *Service Session* beschreibt. Dieses Modell ist dem objektorientierten Rufmodell aus MAGIC ähnlich (Abschnitt 3.1.2). Im Gegensatz zu einem starren, Zustands-basierten Rufmodell, wie es z.B. vom IN bekannt ist, enthält der TINA-*Service Session Graph* (Abbildung 3.7), der im SSM verwaltet wird, Teilnehmerobjekte (*Session Members*), die über Informationsflüsse (*Stream Bindings*) miteinander verbunden sind. Objekte, deren Parameter und Abhängigkeiten können mit diesem Modell dynamisch erzeugt und gelöscht werden. Dies bietet eine wesentlich flexiblere Beschreibung des Dienstzustandes, als es mit dem IN-Rufmodell möglich ist, und unterstützt ein breites Dienstspektrums. Da TINA selbst keine detaillierten Angaben über die Dienstentwicklung macht, wurden in Projekten wie ACTS TOSCA [AC237] oder ACTS SCREEN [AC227] Mechanismen entwickelt, wie durch den Einsatz von SDL und die Anwendung von Vererbungsmechanismen Dienstobjekte programmiert werden können. Für jeden Dienst ist ein *Service Session Manager* sowie ein *User Session Manager* je Teilnehmer auf Betreiberseite und eine *Service Session User Application* auf Teilnehmerseite zu programmieren (siehe Abbildung 3.6).

Durch die flexible Dienstbeschreibung stehen einem TINA-Dienstanbieter alle Klassen von IuK-Diensten offen. Da die TINA-Dienstarchitektur generell unabhängig von der darunterliegenden Netztechnologie ist, können die Dienste unabhängig von netzspezifischen Details entwickelt werden.

In TINA wird die Steuerung von Kommunikationsbeziehungen in der *Network Resource Architecture* (NRA) in der ConS (*Connectivity Service*)-Schnittstelle beschrieben. Leider ist die NRA auf die Unterstützung von ATM-Verbindungen ausgerichtet, was die Netzauswahl stark einschränkt [MC00] und keine netzübergreifende Dienststeuerung zuläßt. Existierende Studien zur Erweiterung der NRA auf Internet-Verbindungen werden in Abschnitt 3.6.2 beschrieben.

Durch eine separate Realisierung von Teilnehmerzugang und Dienstzugang in den Komponenten der *Access Session* kann Dienste-Mobilität, d.h. ein Zugriff auf die Dienste von verschiedenen Endgeräten aus, erreicht werden. Allerdings ist dabei sicherzustellen, daß die zugehörigen TINA-Komponenten für die *Access Session* und für die *Service Session* auf jedem partizipie-

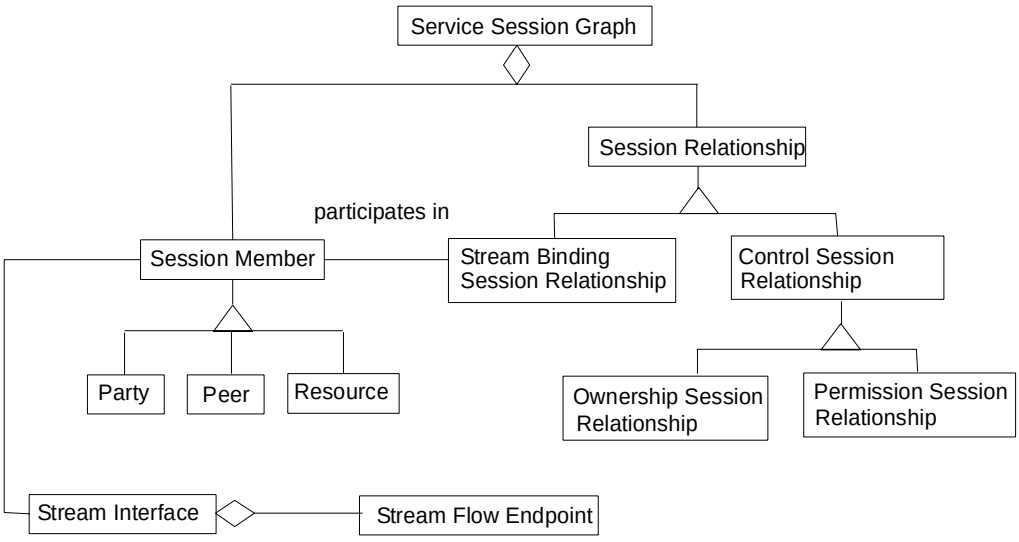


Abbildung 3.7: TINA Service Session Graph [Kri97]

renden Endgerät vorhanden sind. Der Teilnehmer kann die Dienststeuerung daher nicht von einem beliebigen Endgerät aus anrufen.

Fazit

Die TINA-Architektur zeichnet sich durch eine ganze Reihe der geforderten Merkmale aus: Sie beinhaltet ein mächtiges, objektorientiertes Dienstmodell. Die Dienststeuerungsvorgänge laufen unabhängig von der Netzinfrastruktur ab. TINA enthält eine separate Teilnehmerverwaltung für den Teilnehmer- und den Dienstzugang. Nachteilig wirkt sich dagegen aus, daß TINA auf ATM-Netze ausgerichtet ist und daher keine netzübergreifende Dienststeuerung unterstützt wird. Außerdem erfordert TINA eine Middleware-Plattform zur Verteilung der Komponenten, die sowohl alle Netzinfrastrukturkomponenten als auch die Endgeräte betrifft.

3.4 Verteilte, Endgeräte-basierte Dienstarchitekturen: Internet-Architekturen

Das Internet steht als Synonym für Netze, die auf dem IP-Protokoll basieren, und stellt selbst bereits eine offene, verteilte Plattform für die Bereitstellung von Diensten dar. Das IP-Protokoll ist dabei die verbindende Zwischenschicht auf heterogener Transporttechnologie (z.B. *IP over Ethernet*, *IP over ATM*). Durch seinen steilen Aufstieg vom Datennetz zu einem weltumspannenden Informations- und auch Kommunikationsnetz ist die Internet-Architektur als Infrastruktur für eine universelle Diensterbringung von speziellem Interesse.

Das traditionelle Internet bietet eine flexible Umgebung für Dienste wie z.B. E-Mail, File-Transfer, und WWW. Dienste werden durch Applikationsprogramme in den Endgeräten oder in Servern realisiert, die spezielle Ende-zu-Ende Signalisierungs- und Übertragungsprotokolle verwenden (z.B. FTP, HTTP auf UDP/TCP). Dadurch ist die Dienstrealisierung unabhängig von der Netztechnologie. Die Netzbetreiber müssen sich nur um die Bereitstellung des IP-Protokoll-Stacks und nicht um die Dienste kümmern. Der Aufwand, Programme in den Endgeräten zu aktualisieren, ist gering, verglichen mit der Aktualisierung jedes Vermittlungsknotens beim ISDN. Erleichtert wird die Aktualisierung durch Dienste im Internet, die einen einfachen Download von Software ermöglichen.

Die bisherige Beschreibung bezog sich auf die herkömmlichen Standard-Datendienste des Internet. Anders stellt sich die Situation dar, wenn man Telekommunikationsdienste oder Multimediadienste betrachtet. Da das Internet aus einem verbindungslosen best-effort Paketnetz besteht, sind Funktionen wie Qualitätsgarantie, Sicherheit oder Gebührenberechnung aufwendig einzuführen. Dies hat Auswirkungen auf die Unabhängigkeit der Dienststeuerung in den Endgeräten vom Transportnetz, da im Netz dienstspezifische Komponenten (z.B. H.323-*Gatekeeper*, siehe unten) eingerichtet werden müssen.

Im Folgenden werden drei Dienstarchitekturen näher betrachtet, die multimedial erweiterte Telekommunikationsdienste im Internet bereitstellen. Die Analyse von H.323 und SIP stützt sich dabei auf die Ergebnisse der Studie [GKM01a,b].

3.4.1 Internet-Dienstarchitektur gemäß H.323

Die *Study Group* 16 der ITU-T hat die Empfehlung H.323 im Dezember 1996 als Standard für realzeitige Videokonferenzen über Lokale Netze, in denen keine Qualität garantiert ist, verabschiedet. H.323 wird seitdem beständig erweitert. Diesen Ausführungen liegt die Version H.323 v.2 (1999) zu Grunde [H.323]. Die Dienststeuerung bei H.323 ist entsprechend dem Internet-Konzept Endgeräte-basiert. Teilnehmer-Registrierung und Zugangskontrolle innerhalb einer Domäne werden von einem sogenannten *Gatekeeper*, einem Server im Netz, über

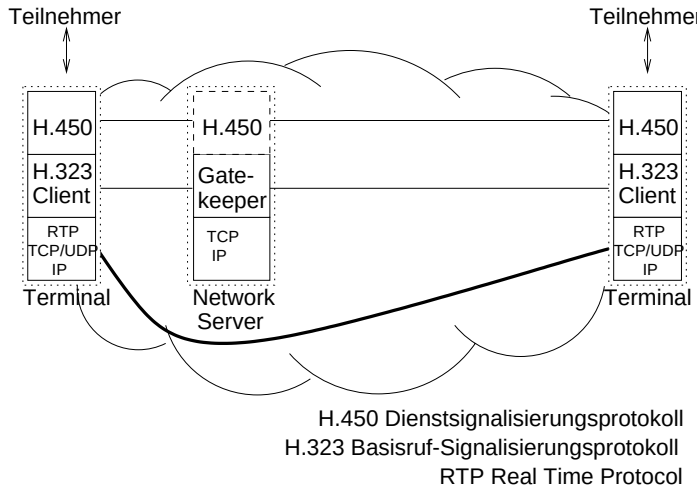


Abbildung 3.8: H.323-Architektur

das RAS-Protokoll *(Registration, Admission and Status)* durchgeführt. Die Übertragung von realzeitigen Medien verwendet die Protokolle RTP und RTCP [OT99]. Ein Ziel von H.323 ist die Zusammenarbeit mit bestehenden Telekommunikationsnetzen, speziell dem ISDN. Daher sind die Mechanismen stark an das Q.931-Protokoll angelehnt.

H.323 definiert im H.225.0-Protokoll einen Basisdienst für multimediale Konferenzen. Das H.245-Protokoll regelt die Aushandlung der Parameter des Basisdienstes (z.B. Kodierungsformate). Für die Erweiterungen des Basisdienstes können mit dem H.225.0-Protokoll Zusatzdienste signalisiert werden. Die entsprechenden Nachrichten dafür sind im Standard H.450 spezifiziert [H.450]. Diese Zusatzdienste realisieren z.B. *Call Transfer* oder *Call Hold*. Zusatzdienste werden prinzipiell Ende-zu-Ende signalisiert, ohne eine Instanz im Netz zu benötigen [KK99]. Für manche Dienste, z.B. Rufbearbeitung bei abgeschaltetem Endgerät, sind allerdings Server im Netz notwendig.

Das Spektrum möglicher Dienste beschränkt sich bei H.323 zwar auf standardisierte Zusatzdienste, aber es können über diese hinaus auch proprietäre Zusatzdienste entwickelt werden. H.450.1 stellt hierfür ein Framework bereit, das den Transport von Dienstanforderungen und eine Verfahrensweise für unbekannte Dienste regelt [KK99].

Für die Implementierung der Dienststeuerung definiert H.323 eine Architektur, bei der separate Zustandsautomaten für den Basisdienst (*Basic Call*) und jeden einzelnen aktiven Zusatzdienst vorgesehen sind. Diese modulare Architektur erlaubt dem Dienstanbieter eine einfache Implementierung der Software für neue Zusatzdienste, ohne daß der Zustandsautomat des Basisdienstes verändert werden muß. Im Gegensatz dazu würde die Komplexität bei Systemen mit nur einem Zustandsautomaten für jeden Zusatzdienst sehr stark anwachsen. Ähnlich wie beim ISDN erfordert jeder Zusatzdienst damit eine eigene Standardisierungsprozedur, um die Konsistenz der Endgeräte- und Server-Software zu gewährleisten. Dies mag zwar hemmend auf einen schnellen Dienstlebenszyklus wirken, aber nur so kann ein konsistentes Interworking zwischen den Endgeräte-basierten Dienststeuerungen unterschiedlicher Hersteller garantiert werden (vertikale Partitionierung). Zur horizontalen Partitionierung sind in der H.323-Architektur keine Schnittstellen vorgesehen, damit Dienste in (lokalen) Netzen fremder Anbieter gesteuert werden können.

Personalisierte Dienste für die Teilnehmer können nur durch das Einstellen von Parametern bei den Standard-Zusatzdiensten realisiert werden (z.B. *Call Forwarding*-Rufnummer). Ein weitergehendes persönliches Zuschneiden von Diensten wie z.B. beim IN ist nicht vorgesehen.

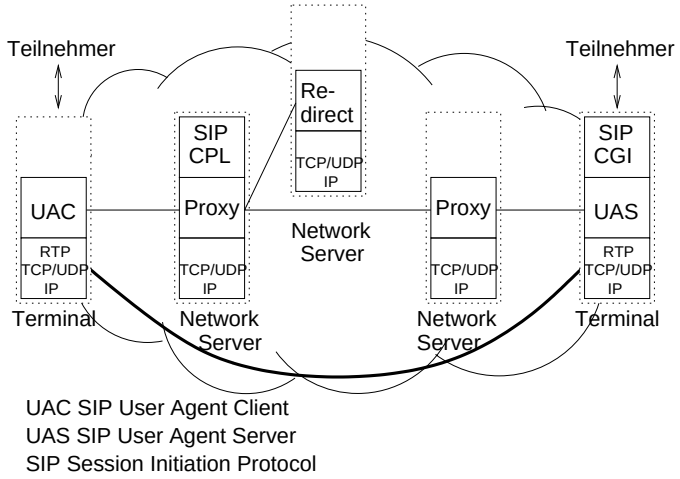


Abbildung 3.9: SIP-Architektur

Bei Endgeräte-basierten Systemen können personalisierte Dienste überdies nur vom eigenen Endgerät aus, in dem die Dienste und Dienstparameter liegen, aufgerufen werden.

3.4.2 Internet-Dienstarchitektur gemäß SIP

Die IETF *Multiparty Multimedia Session Control Working Group* (MMUSIC) hat parallel zu H.323 eine Steuerungsarchitektur für Multimediadienste im Internet entwickelt. Die Basis bildet das *Session Initiation Protocol* (SIP), das 1999 als RFC 2543 standardisiert wurde [HSS99]. Die Architektur wird durch weitere IETF RFCs und insbesondere IETF Drafts ergänzt, die von der IETF SIP Working Group[1] verwaltet werden.

Die Dienststeuerung in SIP ist ebenfalls Endgeräte-basiert. Man unterscheidet Dienstinitiatoren (*User Agent Client*) und gerufene Teilnehmer (*User Agent Server*). Ähnlich wie bei H.323 übernehmen Server im Netz die Aufgabe der Teilnehmerregistrierung (*SIP Registrar*) und zur Adreßauflösung (SIP-Proxy-Server, SIP-Redirect-Server) (Abbildung 3.9). Die Übertragung von realzeitigen Medien basiert ebenfalls auf RTP/RTCP. Es werden bei SIP im Gegensatz zu H.323 keine separaten Protokolle für die Registrierung oder für die Parameteraushandlung verwendet. Beide Funktionen werden zusammen mit der eigentlichen Dienstsignalisierung allein durch das SIP-Protokoll erbracht. Dies reduziert den Nachrichtenaustausch.

Die Dienstsignalisierung erfolgt in SIP zweistufig. Das SIP-Protokoll [HSS99], speziell die Meldung INVITE, enthält alle Teilnehmeradressen und weitere Angaben, die der Protokollverarbeitung in den SIP-Servern (Proxy-Server, *User Agent*-Server) dienen, wie z.B. Session-Bezeichner und unterstützte Funktionen. Die Beschreibung des Dienstes selbst wird zwischen den Teilnehmern als Nutzdaten in einer SIP-Meldung übertragen. Die IETF empfiehlt dafür das *Session Description Protocol* [HJ98], das kein Protokoll im eigentlichen Sinne ist, sondern eine strukturierte Beschreibung der Dienstzusammensetzung aus verschiedenen Medien und zugehörigen Verbindungen. Da diese Dienstbeschreibung für das SIP-Protokoll transparent ist und im Normalfall nur in den *User Agents* interpretiert wird, können prinzipiell beliebige Dienstbeschreibungen transportiert werden. SIP ist somit nicht auf multimediale Telekommunikationsdienste im engeren Sinne beschränkt, sondern kann ebenfalls Informationsabrufdienste oder Verteildienste signalisieren.

1. HTTP://www.ietf.org/html.charters/sip-charter.html

SIP sieht keine Schnittstelle vor, die eine netzunabhängige Steuerung erlaubt, da sich die Protokoll-Parameter und insbesondere die SDP-Dienstbeschreibung auf Internet-Protokolle für die Übertragung der Medien beziehen. Andere Dienstbeschreibungen erlauben z.B. die Signalisierung von nicht IP-basierten Diensten, die durch spezielle SIP-*User Agents* aus dem Internet heraus in externen Netzen gesteuert werden. Das PINT-Konzept [PC00] beispielsweise verwendet SIP, um Dienste im PSTN anzustoßen. Der PINT-Nutzer (SIP-Client) signalisiert die Dienstbeschreibung an ein PINT-Gateway (SIP-Server), das den Dienst im PSTN ausführt. Dienstbeispiele sind *Click-to-Dial*, *Click-to-Fax*.

Im Internet kann ein SIP-Dienstanbieter ähnlich wie bei H.323 Telefonie-Zusatzdienste realisieren. Da SIP selbst nur für die Initialisierung und Beendigung von Diensten gedacht war, sind Protokollerweiterungen für Zusatzdienste notwendig, die während einer Dienstausführung aufgerufen werden. SIP erlaubt die Definition von neuen SIP-Meldungen und neuen Meldungsparametern. Bis vor kurzem wurde bei SIP der Ansatz verfolgt, mit wenigen zusätzlichen Meldungen und Parametern auszukommen und alle Zusatzdienste darauf aufzubauen. Dies kann leicht zu Interoperabilitäts-Problemen aufgrund von Interpretationsfehlern führen, wenn Dienste nicht explizit signalisiert werden können. Im März 2000 ist daher ein IETF Draft erschienen, der in einem *Call Control Framework* eine jeweils eigene Definition von Diensten vorsieht [Cam00]. Leider bietet SIP keinerlei Angaben für die Implementierung der Dienststeuerung wie bei H.323, so daß die Auflösung der Komplexität für die Zusatzdienste dem Implementierer überlassen bleibt.

Neben Protokollerweiterungen, die der Standardisierung von Zusatzdiensten bei H.323 entsprechen, bietet SIP die Möglichkeit, Dienste zusammengesetzt aus mehreren SIP-Meldungen in den Netz-seitigen Servern oder im Endgerät zu implementieren. Dafür werden spezielle Programmiersprachen vorgesehen. SIP-CGI und SIP-Servlets erlauben es den Dienstanbietern (Server-Administratoren), Dienste ähnlich wie bei HTTP mit CGI-Skripten bzw. Java-Skripten zu programmieren [RLS99].

Speziell für die Dienstprogrammierung durch die Teilnehmer selbst wurde die *Call Processing Language* (CPL) [LS00] geschaffen, die aus Sicherheitsgründen nur einen beschränkten Leistungsumfang bietet. Mit allen Programmiersprachen ist es möglich, personalisierte Dienste anzubieten.

Die Standardisierung dieser Programmierkonzepte beschränkt sich derzeit auf Vorschläge in IETF Drafts. Da in keinem Fall eine Schnittstelle zwischen den SIP-Protokollautomaten und Dienstprogrammen definiert ist, sind die Programmier-Konzepte an sich auch für H.323 anwendbar. Alle Dienste werden im Endgerät oder durch SIP-Meldungen in den durchlaufenen Servern ausgelöst. Ein anderer, Endgeräte-unabhängiger Teilnehmer- oder Dienstzugang ist nicht vorgesehen.

3.4.3 Dienstarchitektur mit Megaco/H.248

Das *Media Gateway Control Protocol* (Megaco) wurde gemeinsam von der ITU-T (H.248) und der IETF (RFC 3015) zur Steuerung multimedialer Dienste im Internet standardisiert [CGR00]. Ziel ist die Kopplung von durchschaltevermittelten Telekommunikationsnetzen (z.B. PSTN) und deren Diensten mit dem Internet. Media Gateways am Netzübergang steuern die Dienste im Internet (z.B. VoIP).

Das Megaco zugrundeliegende Architekturkonzept basiert auf der Dekomposition der Gateways in eine Steuerungskomponente (*Media Gateway Controller,* MGC) und in die *Media Gateways*, an denen die Informationsflüsse enden. Mit Megaco wird eine offene Schnittstelle (Protokoll) zwischen Steuerung und Gateway standardisiert. Die *Media Gateways* übernehmen

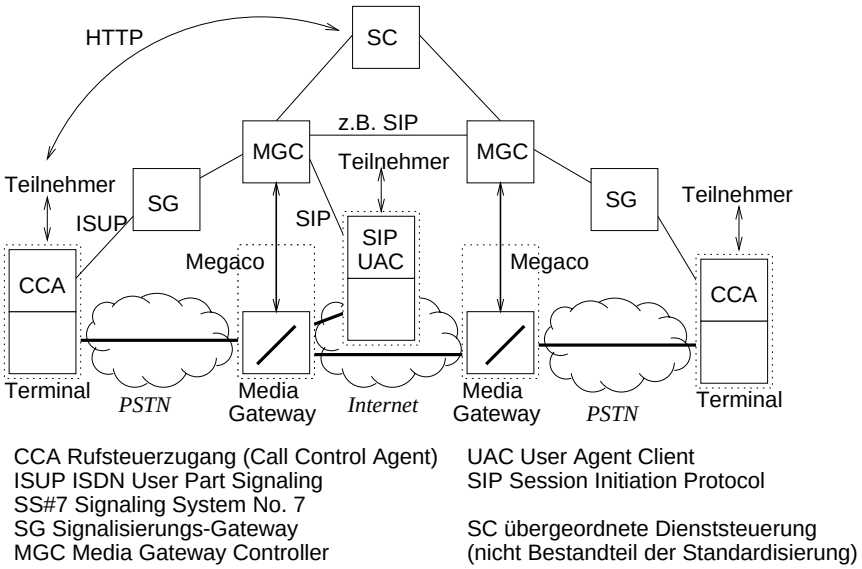


Abbildung 3.10: Megaco-basierte Architektur

die Steuerung der Nutzdaten in verschiedenen Netzen und können eine Vielzahl unterschiedlicher Informationsflüsse in unterschiedlicher Kombination und Verbindungstopologie verarbeiten. Überdies erlauben Media Gateways auch eine Umkodierung oder die Umsetzung von Medien, z.B. Text in Sprache. Megaco ermöglicht auf diese Weise eine Entkopplung der Dienstebene von der Netzebene, um Dienste in unterschiedlichen Netzen zu steuern. Allerdings spezifiziert Megaco nur das Steuerungsprotokoll für die Gateways. Im Gegensatz zu H.323 und SIP beinhaltet es keine Dienstarchitektur, in der konkrete Komponenten und Protokolle für den Dienststeuerablauf festgelegt werden.

Das *Megaco Connection Model* erlaubt Rückschlüsse auf die Art der Dienste, die ein Dienstanbieter in den MGC oder in einer übergeordneten Dienststeuerung (Abbildung 3.10) realisieren kann. Es erlaubt die Verknüpfung heterogener Informationsflüsse in beliebiger Anzahl und in beliebiger Kombination und ist damit für Multimedia-Dienste geeignet.

Die Dienst-Signalisierung, d.h. der Dienstzugang des Teilnehmers oder die Signalisierung zwischen verschiedenen MGCs, ist ebenfalls nicht standardisiert. Drei Möglichkeiten für den Dienstzugang kommen in Betracht: Taylor [Tay00] beschreibt ein Szenario, in dem z.B. SIP, H.323 oder INAP gekoppelt über Signalisierungs-Gateways die Dienstsignalisierung zum MGC übernehmen. Diese Protokolle legen damit die möglichen Dienste fest bzw. sie legen die Möglichkeiten der Teilnehmer fest, Dienste aufzurufen. Außerdem können Dienste auch durch Ereignisse in den *Media Gateways* ähnlich wie beim IN durch Triggerpunkte ausgelöst werden. Zur Realisierung eines dienstunabhängigen Teilnehmerzugangs ist es außerdem denkbar, einen Dienst in einer übergeordneten Dienststeuerung vom Teilnehmer aus über HTTP durch HTML-Seiten zu starten.

In Abbildung 3.10 ist das Modell einer Megaco-basierten Dienstarchitektur dargestellt. Da Megaco keine Dienstarchitektur beschreibt, ist es völlig offen, wie Dienstprogrammierung, Dienstpersonalisierung oder Dienstemobilität realisiert werden.

Fazit

Das Internet ermöglicht als paketorientierte Transportschicht eine netzunabhängige Dienststeuerung auf unterschiedlichen Netzen. Für multimediale Telekommunikationsdienste sind aber zusätzliche Mechanismen erforderlich, um eine vergleichbare Qualität und Funktionalität im Internet bereitstellen zu können, wie man sie vom Telefonnetz gewohnt ist. Die Architektu-

ren von H.323 und SIP realisieren die entspechenden Funktionen. Die ergänzenden Mechanismen binden ihre Dienstarchitekturen aber stark an die IP-Netztechnologie.

Die Architekturen unterstützen ein breites Spektrum an Multimediadiensten und beinhalten flexible Protokolle für die Sessionsteuerung. Daher sind ihre Protokolle für die zukünftige Dienst-Signalisierung äußerst interessant. Einen weiteren Baustein für die Signalisierung bildet das Protokoll Megaco/H.248. Es eignet sich besonders zur Steuerung von Multimedia-Diensten über Gateways in heterogenen Netzen.

3.5 API-basierter Ansatz

Die Analyse der bisherigen Dienstarchitekturen hat gezeigt, daß noch große Defizite bei der horizontalen Trennung der Dienstebene gegenüber den Netzen bestehen. In vielen Fällen sind zwar Schnittstellen zwischen einer Dienststeuerung und einer Kommunikationssteuerung standardisiert. Allerdings wird die Zusammenarbeit zwischen Dienstanbietern, die im Rahmen der Deregulierung ihre Dienste über Netze fremder Betreiber steuern wollen, und Kommunikationsanbietern, die eine Schnittstelle zur Verfügung stellen, nicht unterstützt. Offene Programmierschnittstellen (APIs), die von Kommunikationsanbietern bereitgestellt werden, z.B. Parlay oder JAIN, schließen diese Lücke.

3.5.1 Parlay

Ziel der Parlay-Group, die im April 1998 gegründet wurde, ist die Definition eines offenen und gesicherten APIs, das externen Dienstanbietern den Zugriff auf netzinterne Informationen erlaubt sowie die Steuerung und Überwachung von Netzressourcen ermöglicht [PAR00]. Das Parlay-API ist als eine einheitliche Schnittstelle zu unterschiedlichen Kommunikationsnetzen angelegt, um die Entwicklung von Diensten über verschiedene Netze zu vereinfachen. Derzeit gültig ist die Version 2.0, auf die sich die weiteren Ausführungen beziehen.

Abbildung 3.11 zeigt die Parlay-Architektur und die darin spezifizierten Schnittstellen. Dienstanbietersysteme, in der Abbildung mit *Client Application* bezeichnet, greifen über das Parlay-API, realisiert in einem Parlay-Gateway, auf die Netzressourcen zu. Das Parlay-API bietet die zwei Schnittstellenkategorien Parlay-*Framework Interface* und Parlay-*Service Interface* an. Das Parlay-*Framework Interface* stellt Funktionen für einen gesicherten Zugang, für das Management sowie für die Auswahl und den Zugriff auf die Steuerungsfunktionen bereit. Es enthält aber keine Funktionen zur individuellen Teilnehmerverwaltung.

Die Steuerungsfunktionen, die den Zugriff auf die Netzressourcen ermöglichen, werden in den *Service Interfaces* angeboten. Es handelt sich bei den Parlay-*Services* nicht um Dienste für Teilnehmer, sondern um die Netzfunktionen der steuerbaren Netzressourcen, die im Parlay-API gekapselt werden. Die Schnittstellen (*Resource Interfaces*) zu den unterschiedlichen Netzressourcen, werden in Parlay 2.0 nicht spezifiziert. Parlay-*Services* werden über eine interne Schnittstelle beim Framework registriert. Ihre Funktionen sind in die fünf Gruppen *Call Control Services*, *User Interaction Services*, *Mobility Services*, *Messaging Services* und *Connectivity Services* unterteilt. Jede Gruppe umfaßt verschieden umfangreiche Schnittstellen, die jeweils eine Klasse von Netzdiensten abbilden. Die angebotenen Schnittstellen sind dabei teilweise sehr netzspezifisch (z.B. *INAP1 Call Control Service*, *CAMEL Application Part Call Control Service*). Andere Schnittstellen bieten generische Rufmodelle, von einem einfachen, zwei Teilnehmer umfassenden Ruf bis zu einem *Multimedia Conference Call Control Service*. An den Schnittstellen werden sowohl Steuerungsbefehle an die Netzressourcen (Rufaufbau, *Call Forwarding*) als auch Ereignisse von den Netzressourcen (z.B. Rufnummer gewählt, Teilnehmer belegt) ausgetauscht. Ähnlich wie bei TINA sind die Meldungen der Schnittstellen an

interagierende Komponenten gekoppelt, die für einen Dienstaufruf (Session) instanziert werden (z.B. Parlay-*Call Control Manager*). Mit dem Parlay-API wird aber im Vergleich zum TINA-Ansatz ein pragmatischer Ansatz verfolgt, bei dem die Schnittstellen und deren Funktionen sehr stark an existierenden Netzinfrastrukturen orientiert sind. Der Fokus liegt auf dem PSTN/IN, dem Internet und den Mobilfunknetzen.

Über das Parlay-API bieten Kommunikationsanbieter dem Dienstanbieter je nach Ausprägung der angebotenen Parlay-*Service Interfaces* eine modulare Schnittstelle, mit der dieser seine Dienste über eine heterogene Netzinfrastruktur steuern kann. Die Definition einer Dienstarchitektur ist nicht Bestandteil der Parlay-Spezifikation und auch nicht Ziel der Parlay-Group, d.h. eine Struktur der Dienststeuerung, Dienstkomponenten (z.B. Funktionen zur Teilnehmerverwaltung), ein einheitliches Dienstmodell oder Richtlinien zur Dienstprogrammierung sind nicht spezifiziert. Enthält ein Parlay-Gateway mehrere *Service Interfaces*, so bleibt auch der Auswahlmechanismus der Dienststeuerung überlassen. Die Parlay-Interfaces sind unabhängig von einer bestimmten Implementierung mit UML beschrieben, so daß die Art des Kommunikationszugriffs der *Client Application* auf die Parlay-Interfaces von der Implementierung des Kommunikationsanbieters abhängt (z.B. CORBA-Middleware).

Ähnlich wie bei Megaco ergeben sich zwei Möglichkeiten, wie ein Teilnehmer in einer Parlay-Umgebung seine Dienste startet. Entweder werden Dienste über den Teilnehmer-Netz-Zugang angestoßen, indem Ereignisse über das Parlay-API an eine Client Application gemeldet werden. Die andere Möglichkeit besteht in einer separaten Dienst-Signalisierung zwischen Teilnehmer und Dienststeuerung, z.B. über HTTP. Durch die fehlende Spezifikation einer Dienstarchitektur bleibt die Realisierung von personalisierten Diensten oder von Dienste-Mobilität der Ausgestaltung der *Client Application* überlassen.

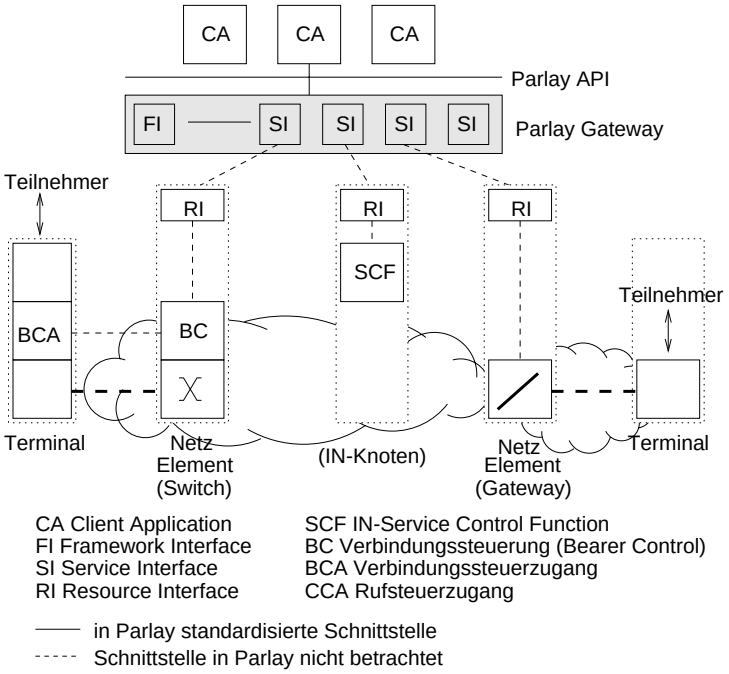


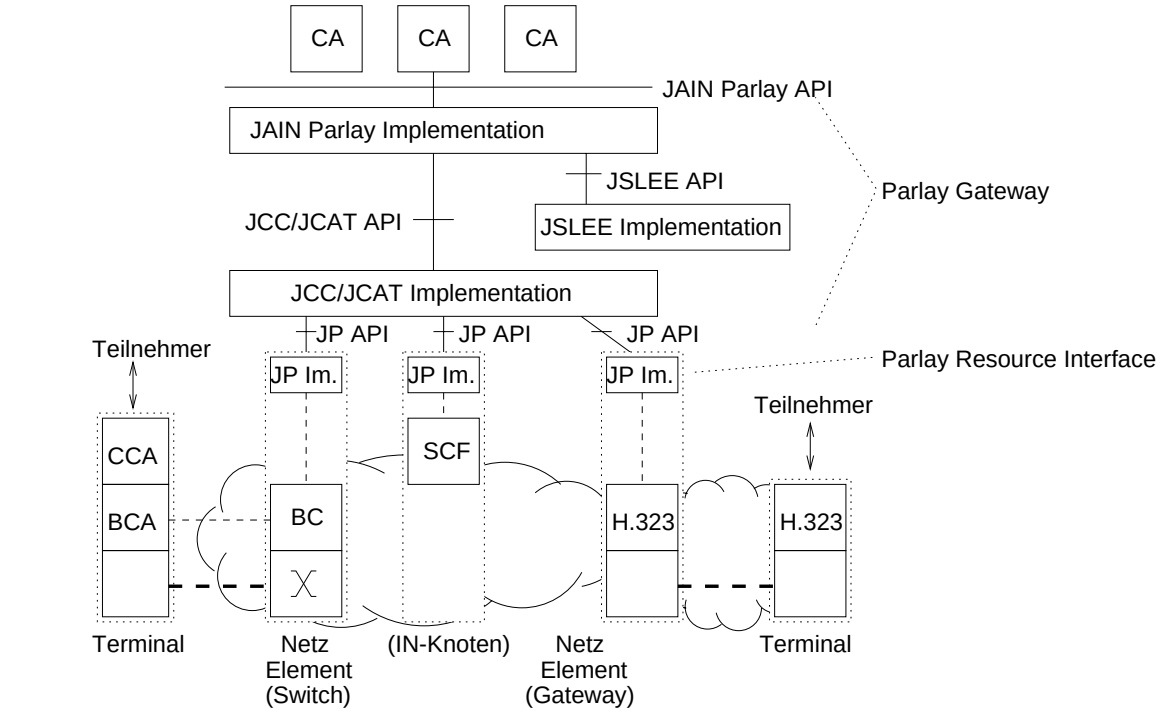
Abbildung 3.11: Parlay-Architektur

3.5.2 Java APIs for Integrated Networks (JAIN)

Das JAIN-Konzept [BBD00, JAIN] wurde erstmals 1998 von SUN Microsystems als *Java Advanced Intelligent Network* vorgestellt und kurz darauf in *Java APIs for Integrated Networks* umbenannt, um den Schwerpunkt klarzustellen. Im Prinzip ist JAIN eine Java-Spezifi-

kation des Parlay-APIs in der Version 1.2. JAIN legt somit die Programmiertechnik und damit auch die Kommunikationstechnik zwischen der *Client Application* und den *Parlay-Interfaces* fest. Die Vorteile in der Java-Realisierung liegen in der Portabilität der Dienste durch die einheitliche Java-Programmier- und Ablaufumgebung. Durch das JAVA-Komponenten-Konzept (*Java-Beans*) können Schnittstellen vergleichbar mit Bausteinen einfach und sogar zur Laufzeit in Applikationen integriert werden, was eine erhebliche Erleichterung der Dienstentwicklung mit sich bringt. Im Vergleich zum IN erfolgt die Dienstentwicklung mit Java in einer weit verbreiteten Programmiersprache und nicht mit proprietären *Service Creation Environments*.

Neben dem Parlay-API, das eine gesicherte Schnittstelle für externe Dienstanbieter bereitstellt, bietet JAIN weitere APIs auf zwei niedrigeren Abstraktionsstufen an, wie in Abbildung 3.12 dargestellt. Auf unterster Ebene bilden die netzspezifischen *JAIN Protocol APIs* die Signalisie-



AS Application Server
JSLEE JAIN Service Logic Execution Environment
JCC JAIN Call Control
JCAT JAIN Coordination and Transaction
JP API JAIN Protocol API
JP Im. JAIN Protocol Implementation

SCF IN-Service Control Function
BC Verbindungssteuerung (Bearer Control)
BCA Verbindungssteuerzugang
CCA Rufsteuerzugang

Abbildung 3.12: JAIN-Architektur

rungsmeldungen unterschiedlicher Netze, wie z.B. ISDN/ISUP, Megaco, H.323 für die Verwendung in einer Java-Umgebung ab. Sie entsprechen damit einer Implementierung der *Parlay Resource Interfaces.* Das kombinierte *JAIN Call Control (JCC) API* und *JAIN Coordination and Transactions (JCAT) API* bietet dagegen Netz- und Protokoll-unabhängige Funktionen für die Rufsteuerung und die Rufverarbeitung. Beispiele hierfür sind die Überwachung, Initiierung, Beantwortung, Verarbeitung und Manipulation von multimedialen Mehr-Teilnehmer-Rufen übergreifend auf unterschiedlichen Netzen. Diese Schnittstelle wird erst durch die Kombination mit der Funktionalität des *JAIN Service Logic Execution Environment (JSLEE)*, die netzunabhängige, nicht Ruf-bezogene Funktionen für Management und Sicherheit (vgl. Parlay-*Framework Interface*) anbietet, zu einem Java-basierten Parlay-API, das einem externen Dienstanbieter bereitgestellt werden kann.

Bei der Bewertung anhand des Merkmalskatalogs treffen für JAIN grundsätzlich dieselben Aussagen zu wie für Parlay. Lediglich durch die Verwendung von Java unterstützt JAIN über Parlay hinausgehend eine einfache Dienstentwicklung mit einer Standard-Programmiersprache und die Portabilität von Dienstimplementierungen auf alle Java-Plattformen.

3.6 Netzunabhängige Dienstarchitekturen

Die bisher erläuterten Dienstarchitekturen bieten teilweise gut strukturierte Lösungen für die Steuerung multimedialer Dienste. Jedoch beschränkt sich die Entkopplung von Dienstebene und Netzebene oft nur auf einen Netztyp. Darüber hinaus sind keine Mechanismen für die Zusammenarbeit externer Dienstanbieter mit Kommunikationsanbietern vorgesehen. Die beiden jüngst entwickelten Standards Parlay und JAIN lösen dieses Problem: es wird ein API definiert, über das externe Dienstanbieter auf die unterschiedlichen Kommunikationsnetze von Netzbetreibern zugreifen können. Allerdings standardisiert Parlay/JAIN keine Dienstarchitektur.

Die Herausforderung, eine netzunabhängige Dienststeuerung zu entwickeln, mit der IuK-Dienste über heterogene Netze gesteuert werden können, bildet den Schwerpunkt der vorliegenden Arbeit. Dies war auch Ziel verschiedener Forschungsprojekte. Die daraus entstandenen Lösungsansätze werden im Folgenden analysiert und anhand des Anforderungskatalogs aus Abschnitt 2.6 bewertet. Sie stützen sich in vielen Fällen auf die eben beschriebenen, standardisierten Architekturen.

3.6.1 IN-basierte Architekturen

Um die Konvergenz heterogener Netze zu unterstützen, sind im Umfeld der Intelligenten Netze Ansätze entstanden, die insbesondere auf eine Kopplung der IN-Architektur mit dem Internet abzielen. Unter dem Stichwort „**WebIN**“ werden Systeme verstanden, bei denen die Dienstmanagementfunktion (SMF) eines IN-Systems über das Internet bzw. über ein Intranet angesteuert werden kann [Low97]. Der *Service Node* „I@C“ der Firma Siemens erlaubt beispielsweise die Verwaltung von Teilnehmereinstellungen (z.B. Rufumleitung) durch die Benutzer und sogar die Erstellung der Dienstlogik über Java-Skripten auf einem herkömmlichen Web-Browser [Fre99]. Derartige Firmen-Lösungen zur Integration von Internet und IN basieren meist auf proprietären, d.h. firmenspezifischen Implementierungen ohne offene Schnittstellen.

In verschiedenen Aktivitäten der IETF werden offene Schnittstellen und Protokolle für die Zusammenarbeit von PSTN/IN und Internet standardisiert. Unter der Bezeichnung **PINT** *(PSTN/Internet Interworking)* [PC00] wird ein Protokoll basierend auf IETF SIP (siehe Abschnitt 3.4.2) standardisiert, das einen Aufruf von Diensten im PSTN aus dem Internet erlaubt. Das PINT-Protokoll transportiert eine Dienstbeschreibung von einem PINT-Client (Web-Server) zu einem PINT-Gateway, der normalerweise auf einem IN-System aufsetzt und die Dienstausführung steuert. Typische PINT-Dienste sind *Click-to-Dial*, *Click-to-Faxback*.

Genau spiegelbildlich aufgebaut ist die **PIN**- *(PSTN Internet Notification)* Architektur [BCC00]. Hier wird ein Internet-Protokoll spezifiziert, das Dienstanforderungen eines IN-Steuerknotens an einen IP-basierten Server überträgt, um Dienste wie *Internet Call Waiting* zu realisieren. In **SPIRITS** *(Service in the PSTN/IN Requesting InTernet Services)* wird eine Architektur beschrieben, die die PINT-Architektur mit einem PIN-Ansatz kombiniert [SFL00]. Das heißt, SPIRITS erweitert die PINT-Architektur um Komponenten und zugehörige Schnittstellen, die PSTN/IN initiierte Dienste in PINT-Clients aufrufen (z.B. *Internet Call Waiting*, *Internet Caller-ID Delivery*, *Internet Call Forwarding*). Allen drei Ansätzen ist gemeinsam,

daß derzeit lediglich die Komponenten und Protokolle für das Internet spezifiziert werden, die Anbindung der Internet-Gateways an die entsprechenden PSTN/IN Komponenten jedoch dem Implementierer überlassen bleibt.

Während die obigen Ansätze die IN-Steuerknoten (SSF) dazu benutzen, spezielle Dienste (z.B. „PINT-Dienste“) zu steuern, haben verschiedene andere Projekte zum Ziel, die IN-Dienste selbst übergreifend über PSTN und Internet zu nutzen. Das Projekt **INSeCT** [DV00, CDG00] sowie das EURESCOM Projekt **P916** [P916] arbeiten an einer Anbindung von H.323-Internet-Telefonie-Systemen an das IN. Aufgaben sind hierbei der Transport von INAP über IP und die Anbindung des H.323-*Call Models* an das IN-*Call Model*.

Fazit

Die oben beschriebenen Ansätze realisieren zwar eine Integration heterogener Netze (PSTN/IN und Internet/VoIP), aber nicht die geforderte Unabhängigkeit der Dienststeuerung von der Netzinfrastruktur, wie sie z.B. Parlay bereits bietet. Da sich die Ansätze auf zwei Netztypen (PSTN/IN und Internet) beschränken, sind die Möglichkeiten zur Dienststeuerung für einen Dienstanbieter eingeschränkt (z.B. IN-*Call Model*). Das Parlay-API bietet hier bereits mehr Möglichkeiten und erlaubt zudem externe Anbieter, so daß die Ansätze als Grundlage für eine netzunabhängige Dienststeuerung kaum in Frage kommen.

3.6.2 Weiterentwicklung der TINA-Ansätze

Im Gegensatz zum IN verfügt die TINA-Architektur über eine umfassende Dienststeuerung, mit der ein nahezu unbegrenztes Dienstspektrum netzunabhängig modelliert und gesteuert werden kann. Die Notwendigkeit einer einheitlichen, auf allen beteiligten Infrastruktureinheiten vorhandenen Middleware und die starke Ausrichtung der Kommunikationssteuerung auf ATM-Netze (TINA-NRA) erfordern bestimmte, derzeit nicht vorhandene Voraussetzungen von der Netzinfrastruktur zur Realisierung eines TINA-Systems. Die Integration existierender Netze wird nicht unterstützt. Daher wurde schon bald nach Lösungen und Migrationsstrategien gesucht, TINA in die bestehende Netzinfrastruktur einzubinden (z.B. [P508]). Im Projekt CAMOUFLAGE [HKG97] wurde eine Anpassung für TINA an das B-ISDN entwickelt, in der die Middleware-Kommunikation in die B-ISDN Signalisierung umgesetzt wird. Die TINA-IN Work-Group [TWG99] spezifiziert einen IN-Zugang zur TINA-Dienststeuerung in einer *IN-TINA Adaptation Unit* (ITAU) [BCL00]. Das *Call Control Interface* der ITAU basiert dabei auf dem der Parlay-Spezifikation. [KSN97] beschreibt einen Ansatz wie die TINA-NRA für die Unterstützung von Verbindungen im Internet erweitert werden kann. Insgesamt zeigt sich, daß bei TINA durch die logische Trennung der Kommunikationssteuerung von der Dienststeuerung bereits ein wichtiger Weg hin zur Netzunabhängigkeit beschritten werden kann. Darüber hinaus existieren weitere Ansätze, die Architekturen oder Prototypen beschreiben, in denen eine (modifizierte) TINA-Dienststeuerung in unterschiedlichen Netzen eingesetzt wird (z.B. [EKG97, CA99])[1].

Die Architektur des TINA-C erfüllt mit der Dienstarchitektur, die eine entkoppelte Dienst- und Kommunikationssteuerung sowie eine separate Teilnehmerverwaltung enthält, bereits eine ganze Reihe der gestellten Anforderungen (siehe Abschnitt 3.3). Während TINA selbst nur auf ATM-ähnliche Netze abzielt, verdeutlichen die oben vorgestellten Projekte, daß die TINA-

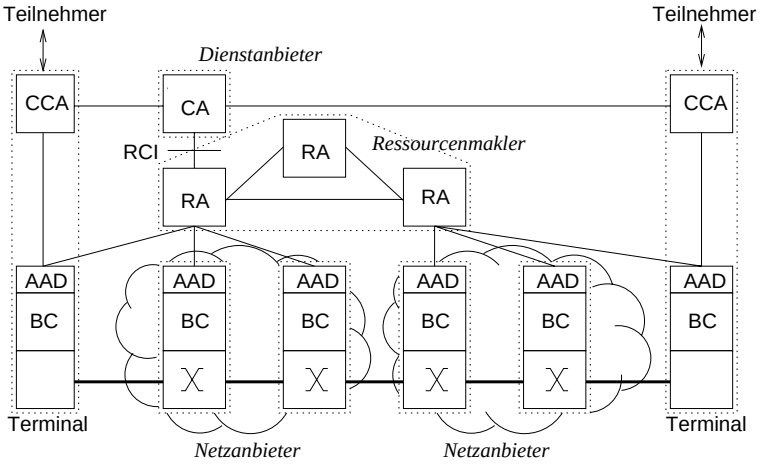
1. Grundsätzlich ist bei allen diesen Ansätzen, die eine Anbindung der TINA-Architektur für spezielle Netze spezifizieren, anzumerken, daß es derzeit kein TINA-System als etabliertes Kommunikationssystem gibt. Nach Beendigung der TINA-Aktivitäten sind zudem Teile von TINA immer noch nicht spezifiziert [MC00].

Dienststeuerungsmechanismen für heterogene Netze geeignet sind und eine netzübergreifende Steuerung von Diensten auf Basis der TINA-Konzepte realisierbar ist. Da die bisherigen Ansätze spezifisch auf je einen Netztyp abzielen, fehlen Mechanismen für eine globale Berücksichtigung der heterogenen Netzinfrastruktur insbesondere zur Adaptivität. Da TINA auf einem Middleware-Konzept beruht, sind zudem Änderungen an der Infrastruktur notwendig, die Teilnehmer herkömmlicher Endgeräte von einem Dienstzugang ausschließen.

3.6.3 Agentenunterstützte Ressourcensteuerung

Der Einfluß neuer Software-Konzepte wirkt sich auch auf die Entwicklung von Dienstarchitekturen aus. Das Konzept der „Software-Agenten“ ist eine relativ unscharfe Bezeichnung für eine Reihe von unterschiedlichen Ansätzen. Man versteht unter **Agenten** kooperationsfähige Komponenten, die Code beinhalten, um stellvertretend für eine Kommunikationseinheit (z.B. Dienst, Dienstanbieter, Teilnehmer, Kommunikationsanbieter) aufgrund von Ereignissen in einer bestimmten Umgebung zu handeln [GP01].

In [Que00] wird die Signalisierungsarchitektur AUREUS als Agenten-basierte Erweiterung der Architekturen MAGIC und AMSA (siehe Abschnitt 3.1.2) beschrieben, in der statische Agenten dazu dienen, die Realisierung von Kommunikationsbeziehungen in den Netzen unterschiedlicher Anbieter auszuhandeln. AUREUS ist keine Dienststeuerung im eigentlichen Sinne, da eine komplette Dienstbeschreibung die Ausgangslage darstellt und der Teilnehmerzugang und die Steuerung von vordefinierten Diensten (Dienstlogik) nicht betrachtet werden. Die Dienstbeschreibung entspricht dem objektorientierten Rufmodell aus [Mül96] bzw. der MAGIC-Architektur, das um einige zusätzliche Parameter ergänzt wurde, um Preise und Prioritäten ausdrücken zu können. Kern ist die Abbildung dieser Universaldienst-Beschreibung auf unterschiedliche Netzressourcen unter Beachtung von Preis und QoS. Dafür werden spezielle Architekturkomponenten spezifiziert, die als Agenten die Interessen der Teilnehmer (Einrichtung der Dienstbeschreibung im Netz zu günstigen Preisen) und die Interessen der Ressourcenanbieter (Verbindungen im Netz, Nutzung von Konferenzbrücken, etc.) vertreten und eine Lösung aushandeln. Bild 3.13 zeigt das Architekturmodell.



SCM Dienststeuerungsmodul (Service Control Module)
CCA Rufsteuerzugang (Call Control Access)
CA Rufagent (Call Agent)
RA Ressourcen Agent (Resource Agent)
AAD Agentenadapter (Agent Adapter)
BC Verbindungssteuerung (Bearer Control)
RCI Resource Control Interface

Abbildung 3.13: Agentenunterstützte Ressourcensteuerung

Die Architektur beschreibt eine vollständige Trennung von Dienstkomponenten und Netzen. An die bestehende Infrastruktur werden keine besonderen Anforderungen gestellt, außer daß für jeden Anbieter von Netzressourcen ein Agent in der Architektur vorhanden sein muß, der die Kommunikationssteuerung übernimmt. Allerdings zielt die Architektur eher auf die Diensterbringung über unterschiedliche Betreiberdomänen ab als über heterogene Netze, da die beschriebenen Modelle auf ATM-Netze ausgerichtet sind. Grundsätzlich könnten auch heterogene Netztypen durch die Agenten vertreten werden.

Die Realisierung von Adaptivität ist an dieser Architektur besonders hervorzuheben. Es können nicht nur neue Ressourcenanbieter hinzugenommen werden (Agenten), sondern die Agenten sind speziell darauf ausgelegt, sich ändernde Parameter (QoS, Preise) in die Ressourcensteuerung dynamisch einzubringen.

3.6.4 Programmierbare Netze

Unter Programmierbaren Netzen versteht man Ansätze, bei denen Teile der Steuerungssoftware in Netzknoten bzw. in Netz-seitige Server über dedizierte Schnittstellen direkt eingebracht werden können, um damit eine bestimmte Funktionalität (z.B. einen IuK-Dienst) durch das Netz zu erbringen. Um programmierbare Netze in der bestehenden Infrastruktur aufzubauen, sind meist erhebliche Änderungen in der Infrastruktur zur Schaffung einer programmierbaren Plattform vorzunehmen. Im Vergleich zum IN läuft die Dienstlogik hier direkt auf den (ausgewählten) Netzknoten. Nur die Verteilung der Logik wird zentral gesteuert.

Gbaguidi et al. (EPFL Lausanne) beschreiben in [GHP99, GHH99] eine programmierbare Architektur zur netzübergreifenden Dienststeuerung über Internet und Telekommunikationsnetze. Darin wird ein Dienst durch interagierende Komponenten erbracht, die auf die Elemente der Dienstplattform (Endgeräte, Netzkonten, Server, Gateways, etc.) verteilt werden. Die Kommunikation zwischen den Dienstkomponenten erfolgt über eine Java-basierte Middleware, die mit den Dienststeuerkomponenten zu einer *Java Service Layer* erweitert wird.

Dienste können von autorisierten Dienstnutzern (Teilnehmer) selbst aus Dienstmerkmalen zusammengesetzt werden, die als *Java Beans* realisiert sind. Eine *Service Factory* des Dienstbetreibers generiert daraus die Komponenten einer Dienstinstanz und verteilt diese auf die Elemente der Dienstplattform. Letztere werden um *Trigger Event Listener* erweitert, damit sie bestimmte Ereignisse (z.B. Wahl einer Rufnummer) erkennen und diese an eine Dienstkomponente melden, um damit einen Dienst zu starten. Dienste können also nicht direkt durch eine Teilnehmeranforderung, sondern nur mittelbar über Triggerpunkte aufgerufen werden. Bild 3.14 zeigt den Aufbau dieser Dienststeuerung.

Diese Dienstarchitektur eignet sich prinzipiell für alle Netzarten, vorausgesetzt die Netzkomponenten/Endgeräte sind in der Lage, entsprechende Ereignismeldungen zu generieren. Sonst muß die Infrastruktur um diese Funktion erweitert werden. Ebenso ist eine programmierbare (Java) Middleware vorzusehen. CORBA allein erfüllt diese Anforderungen nicht. Sind diese Funktionen überall vorhanden, dann bietet die Architektur die in Kapitel 2 geforderte Entkopplung von Dienststeuerung und Netzinfrastruktur. In der Verteilung der Dienstkomponenten auf die Elemente der Plattform ist eine gewisse Adaptivität hinsichtlich der Infrastruktur realisiert, die allerdings vor der Dienstnutzung in der *Service Factory* durchgeführt wird. Adaptivität für eine Dienstausführung ist nicht möglich, da die Dienstkomponenten für jeden Dienst bereits auf den entsprechenden Netzelementen angeordnet sind.

Die Komposition der Dienste aus Dienstmerkmalen erlaubt theoretisch beliebige Dienste. Das potentielle Dienstspektrum ist abhängig von der Auswahl an Dienstmerkmalen. Unter Umstän-

den müssen für neue Dienste laufend neue Dienstmerkmale erstellt werden. Leider wird nichts über die angebotenen Merkmale oder das zugrundeliegende Dienstmodell ausgesagt.

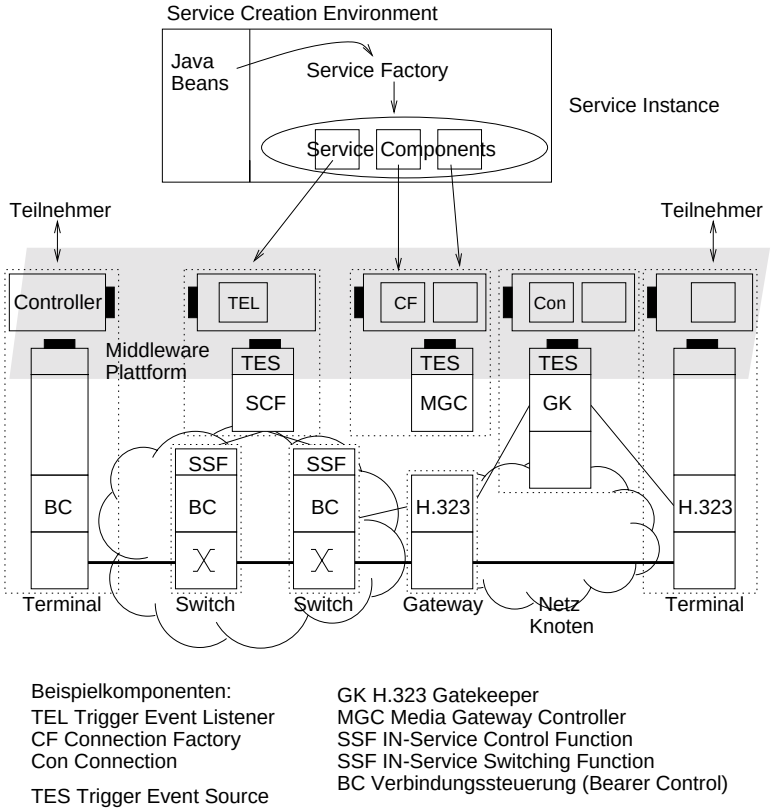


Abbildung 3.14: Dienstarchitektur der EPFL

Fazit

Ähnlich der TINA-Architektur stellt die Architektur der EPFL ein System interagierender Komponenten auf einer Middleware dar. Dienste können hier aber in unterschiedlichen Netzen gesteuert werden. Auch die Realisierung des Dienstzuganges unterscheidet sich. Dieser kann aus beliebigen Netzen erfolgen. Geeignete Triggerpunkte sind Voraussetzung. Eine Strukturierung der Dienstarchitektur in Steuerungsbereiche wird vermißt.

3.6.5 Internet-basierte Architektur: ICEBERG

Das ICEBERG-Projekt [WRC00] der Universität Berkeley beschreibt eine Architektur zur Steuerung von integrierten Daten- und Telefondiensten, bei der Server im Internet die Rolle der Dienststeuerung und des Kommunikationsnetzes übernehmen. Das Internet arbeitet als *Core-Switch*. Zur netzübergreifenden Steuerung von Diensten können beliebige Netze und Endgeräte an das Internet über Adaptoren angeschlossen werden.

Bild 3.15 gibt einen Überblick über die ICEBERG-Architektur. Dienstanbieter werden durch iPoPs *(ICEBERG Point of Presence)* repräsentiert. *ICEBERG Access Points* (IAP) sind Gateways, über die die Netze oder die Endgeräte an die iPoPs angeschlossen werden. Die iPoPs beinhalten netzunabhängige Funktionen für den Teilnehmerzugang und für die Dienststeuerung.

Der Teilnehmerzugang enthält Funktionen für die Teilnehmerprofilverwaltung, für die Adreßauflösung und für eine Reaktion auf Änderungen im Teilnehmerzugang. Somit ermöglicht er eine personalisierte und Endgeräte-unabhängige (Dienste-Mobilität) Dienststeuerung.

Ein *Call Agent* regelt die Dienststeuerung je Dienst für einen iPoP. Zunächst wird eine Dienst-Session mit dem *Session Initiation Protocol* (SIP) bilateral zwischen den Teilnehmern aufgebaut. Die gemeinsame Dienstbeschreibung wird danach vollständig dezentral verwaltet. Dafür wird ein eigenes Signalisierungsprotokoll definiert, bei dem die Zustände einer Dienst-Session regelmäßig zwischen den Teilnehmern (repräsentiert durch *Call Agents*) ausgetauscht werden (*Lightweight Call Session Protocol,* LCSP) [WJK00]. Dies vermeidet Session-Abbrüche durch den Ausfall zentraler Komponenten, ein wichtiger Anspruch im ICEBERG-Projekt.

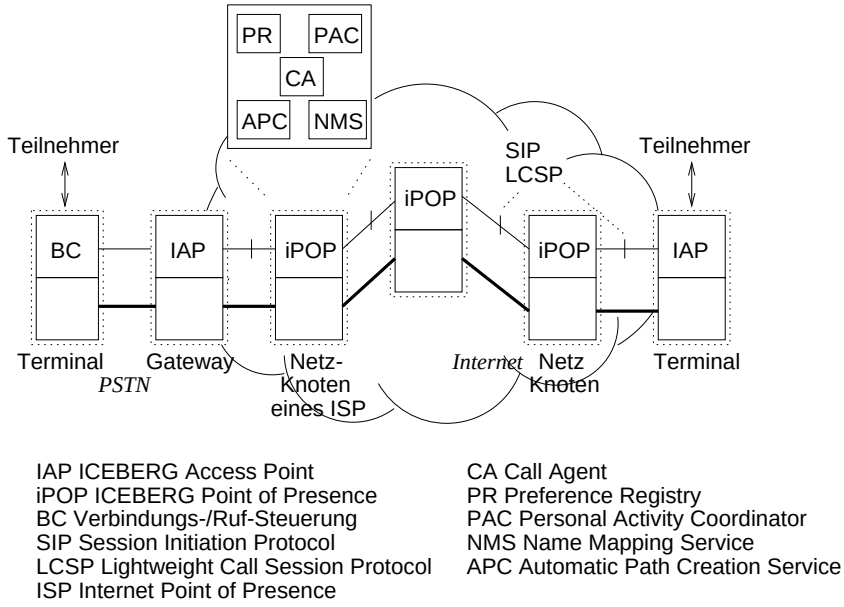


Abbildung 3.15: ICEBERG-Architektur

Durch die explizite Entkopplung unterschiedlicher Netze mittels *ICEBERG Access Points* und die Entkopplung der Dienstsignalisierung von netzspezifischen Details, realisiert durch das ICEBERG-Signalisierungsprotokoll wird eine vollständige Trennung der Dienststeuerung von den Netzen erreicht. Einschränkend im Hinblick auf eine allgemeine Verwendbarkeit z.B. durch netzunabhängige Dienstanbieter ist die Tatsache, daß der Datentransport zwischen den Netzen nur über das Internet erfolgt und eine qualitätsbasierte IP-Infrastruktur Voraussetzung ist.

ICEBERG sieht die Realisierung beliebiger, netzübergreifender Dienste vor. Allerdings werden die Zustände eines Dienstes nur durch eine einfache Tabelle modelliert. Parameter wie Medienattribute, Rechte und Rollen fehlen. Die Mächtigkeit der Dienstbeschreibung erscheint daher nicht so groß als erwartet. Da neben SIP und LCSP keine Schnittstelle zwischen IAP und iPoP offengelegt ist, ist eine dynamische Reaktion auf Änderungen im Zugangsnetz und die dynamische Integration neuer Netze fraglich

Fazit

Die ICEBERG-Architektur realisiert eine funktionelle Dienststeuerung für IuK-Dienste in heterogenen Netzen. Da nicht nur die Signalisierung, sondern auch die Nutzdaten über das Internet übertragen werden, ist eine qualitätsbasierte IP-Infrastruktur Voraussetzung.

3.7 Diskussion und Fazit

Um die abschließende Bewertung zu erleichtern, wird die Analyse der Dienstarchitekturen anhand der wichtigsten Vergleichskriterien in der nachstehenden Tabelle 3.1 zusammengefaßt.

Fast alle betrachteten Architekturen weisen eine zumindest teilweise Entkopplung der Dienstebene von der Netzebene auf. Allerdings ist deutlich zu erkennen, daß sich die Ansätze darüber hinaus schwerpunktmäßig auf die Realisierung einzelner Anforderungen aus Abschnitt 2.6 konzentrieren. Die unterschiedlichen Schwerpunkte resultieren nicht zuletzt aus den verschiedenen Zielsetzungen, die den Architekturen zugrunde liegen.

Nicht dargestellt ist die Art der unterstützten Dienste. Während viele der Architekturen ein breites Dienstspektrum ermöglichen, beschränken sich die Ansätze, die aus einer direkten Weiterentwicklung des IN resultieren (z.B. PINT, SPIRITS), stark auf Telefoniedienste.

Inwieweit die Dienste tatsächlich unabhängig von der Netzinfrastruktur beschrieben werden können, zeigt sich am deutlichsten beim **Dienstmodell**. Hier ist die TINA-Architektur am weitesten entwickelt, da diese auf einer objektorientierten Modellierung der Dienstarchitektur basiert. Vielen Ansätzen fehlt eine solche generische Dienstbeschreibung.

Zur Netzunabhängigkeit der Dienststeuerung gehört nicht nur eine generische Dienstbeschreibung, sondern auch die Trennung der Dienstebene von der Netzebene in der Signalisierungsarchitektur (**horizontale Partitionierung**). Diese ist bei den meisten Dienstarchitekturen vorhanden. Lediglich bei den ISDN- und IN-Architekturen sowie den VoIP-Architekturen bestehen Abhängigkeiten durch Protokolle und Komponenten (z.B. SSF). Einige Architekturen erlauben es, Dienste sogar übergreifend über mehrere, beliebige Netze zu steuern (z.B. ICEBERG, EPFL). Andere Architekturen sind dabei auf mehrere Domänen eines Netztyps oder wenige, bestimmte Netze beschränkt. Dies ist historisch bedingt (z.B. TINA oder MAGIC auf ATM-Netze) oder in der Zielsetzung der Architekturen so festgelegt (z.B. SPIRITS, PINT).

Eine strikte Trennung bei gleichzeitiger Unterstützung nahezu beliebiger Netztypen ist insbesondere bei den Architekturen festzustellen, die Programmierschnittstellen zur Entkopplung bereitstellen (z.B. Parlay, JAIN). Hier sind außerdem die Auswirkungen auf die Infrastruktur (**Unabhängigkeit von spezieller Infrastruktur**) am geringsten, da die Anbindung heterogener Netze über Adaptoren vollzogen wird, die bestehende Signalisierungsschnittstellen benutzen. Dies gilt im übrigen für alle Architekturen, die eine Kopplung über Adaptoren beinhalten (z.B. AUREUS, TINA/3.6.2, ICEBERG). Middleware-basierte Ansätze (z.B. TINA, EPFL) erfordern dagegen eine einheitliche Signalisierungsebene auf allen Infrastruktureinheiten. Dies macht Eingriffe in die bestehenden Infrastrukturkomponenten erforderlich.

Die Erweiterbarkeit der Architekturen um neue Netze wird insgesamt wenig beachtet. Oft sind die Architekturen ohnehin auf ausgewählte Netztypen beschränkt. Eine dynamische Reaktion auf eine sich ändernde Infrastruktur, d.h. **Adaptivität**, wird nur in der AUREUS-Architektur explizit definiert. Ähnliche Mechanismen findet man nur bei Parlay und JAIN. Hier können sich neue Parlay-*Service Interfaces* beim *Framework Interface* anmelden.

Die Realisierung einer **unabhängigen**, d.h. von der eigentlichen Dienststeuerung getrennten **Teilnehmerverwaltung** ist ein weiterer wichtiger Baustein für eine netzunabhängige Dienststeuerung. Sie ermöglicht es, daß die Teilnehmer auf ihre persönlichen Einstellungen und auf ihre Dienste unabhängig von Netzen und Endgeräten zugreifen können. Leider wird gerade diese Funktion in den bestehenden Architekturen noch stark vernachlässigt. Der TINA-Ansatz ist hier am weitesten ausgearbeitet, wenn auch die Unterstützung unterschiedlicher Netze oder Endgeräte nur durch die Middleware abgedeckt wird.

Insgesamt läßt sich feststellen, daß bereits eine ganze Reihe sehr guter Konzepte existiert, um eine Konvergenz der heterogenen Netze und Endgeräte auf Dienstebene zu erreichen. Leider konzentrieren sich die Ansätze entweder auf Teilaspekte, z.B. auf ausgewählte Netze, oder aber es sind starke Eingriffe in die Infrastruktur erforderlich, um z.B. eine spezielle Middleware als Konvergenzschicht einzuziehen. Daher sind die Ansätze, die offene Steuerungs- und

Programmierschnittstellen (APIs) auf heterogenen Netzen bereitstellen, für die in der vorliegenden Arbeit aufgestellten Anforderungen interessant. Darüber hinaus werden in den meisten Dienstarchitekturen wichtige Aufgaben einer Dienstarchitektur wie Adaptivität, Dienstemobilität und insbesondere die Teilnehmerverwaltung nur ansatzweise behandelt oder fehlen ganz. Dies beruht auf den unterschiedlichen Zielen, die der Standardisierung der Architekturen zugrunde liegen. Daher lohnt es sich für die vorliegende Zielsetzung, geeignete Konzepte in einer neuen Dienstarchitektur zu kombinieren und Verbesserungen einzubringen.

Dienst-architekturen	**unabh. Dienst-modell**	**horizontale Partitionie-rung**	**netzüber-greifende Steuerung**	**unabhängig von der Infrastr.**	**Adap-tivität**	**unabhängige Teiln.-Ver-waltung**
B-ISDN (3.1.1)	-	- [a]	-	-	-	-
MAGIC (3.1.2)	**x**	**x**	-	-	-	-
AUREUS (3.6.3)	**x**	**x**	-	**x**	**x**	-
IN/B-IN (3.2)	-	(x) [b]	-	-	-	-
PINT (3.6.1)	-	**x**	(x) [c]	-	-	-
SPIRITS (3.6.1)	-	**x**	(x) [c]	-	-	-
Parlay (3.5.1)	-	**x**	**x**	**x**	**x**	-
JAIN (3.5.2)	-	**x**	**x**	**x**	**x**	-
Megaco (3.4.3)	-	**x**	**x**	-	-	-
H.323 (3.4.1)	-	-	- [d]	-	-	(x)[e]
SIP (3.4.2)	(x) [f]	-	-	-	-	(x)[e]
TINA (3.3 u. 3.6.2)	**x**	**x**	**(x)** [g]	**(x)** [g]	-	**x**
EPFL (3.6.4)	k. A.	**x**	**x**	-	-	k. A.
ICEBERG (3.6.5)	(x) [h]	**x**	**x**	**x**	-	**x**

[a] Nur funktionale Dekomposition, keine offenen Schnittstellen.

[b] Der SSF ist Netz- bzw. Hersteller-spezifisch.

[c] nur PSTN/IN und Internet/VoIP

[d] Gateway zum ISDN/etc. ist zentraler Bestandteil der Spezifikation.

[e] Der Teilnehmer kann sich bei Servern im Netz (H.323-Gatekeeper, SIP-Outbound-Proxy) registrieren. Kein Dienstzugang.

[f] SDP ist VoIP-spezifisch, aber alternative Dienstbeschreibungen sind möglich, da die Dienstbeschreibung vom Signalisierungsprotokoll entkoppelt ist.

[g] Nur mit den entsprechenden, jeweils netzspezifischen Erweiterungen.

[h] Sehr eingeschränkt.

Tabelle 3.1: Vergleich der Dienstarchitekturen und Architekturansätze

Kapitel 4
Modellierung des neuen Dienststeuerservers

Die vorangegangene Analyse von Dienstarchitekturen hat gezeigt, daß die zahlreichen Anforderungen, die insbesondere aus der heterogenen Infrastruktur und dem verstärkten Wettbewerb resultieren, nur teilweise von den bestehenden Architekturen erfüllt werden. Als Herausforderungen an neue Dienstarchitekturen sind insbesondere die strikte Trennung von Dienstebene und Netzebene, ein generisches Dienstmodell und die Teilnehmerorientierung zu berücksichtigen.

Dieses Kapitel gibt einen Überblick über die in der vorliegenden Arbeit beschriebene Serverarchitektur SAMSON („Server Architecture for Network Independent Multimedia Service Control in Heterogeneous Communication Networks“), die alle genannten Anforderungen erfüllt. Wie bereits mehrfach betont, liegt der Schwerpunkt von SAMSON auf der Trennung von Dienstebene und Netzebene, um eine Steuerung von Diensten in heterogenen Kommunikationsnetzen zu ermöglichen. In diesem Kapitel werden die abstrakten Konzepte und das Systemmodell beschrieben, die den Ausgangspunkt für eine detaillierte Spezifikation der Serverarchitektur bilden. Die zum Systemmodell gehörenden Sichweisen beschreiben statische und dynamische Eigenschaften der Dienstarchitektur für SAMSON. Beispielsweise wird das Signalisierungskonzept durch die Kommunikations-Sichtweise repräsentiert. In den nachfolgenden Kapiteln 5 und 6 erfolgt die konkrete Spezifikation der Architekturkomponenten in Form von Verhaltensbeschreibungen und der Definition eines neuen Signalisierungsprotokolls.

Dieses Kapitel gliedert sich wie folgt. Zunächst werden die Grundkonzepte der Serverarchitektur SAMSON anhand der allgemeinen Architekturmerkmale aus Kapitel 2 erläutert. Anschließend wird auf die Modellierung der Dienstarchitektur eingegangen. Dafür werden in Abschnitt 4.2 sechs verschiedene Sichtweisen zur Modellierung vorgestellt. Anschließend wird die Serverarchitektur SAMSON aus diesen Sichtweisen beschrieben. Die Funktionsweise des Gesamtsystems wird abschließend an einem ausführlichen Beispiel veranschaulicht.

4.1 Grundkonzepte von SAMSON

Die Grundaufgaben des Dienstservers lassen sich am besten an einem Dienstszenario illustrieren. Als Beispiel stelle man sich eine mobile Teilnehmerin („Alex“) vor, die ihren Geschäftspartner („Bernd“) kontaktieren will (siehe Abbildung 4.1). Dabei möchte sie einen für sich vorkonfigurierten Konferenzdienst („MyBusinessCall“) nutzen. Der Dienst baut sowohl eine Sprachverbindung zum gewünschten Teilnehmer auf als auch eine Verbindung zu einem Datenserver, der beiden Teilnehmern eine Multimedia-Präsentation während des Gesprächs

zustellt. Teilnehmerin Alex kontaktiert dafür nur die Dienststeuerung und fordert ihren Dienst mit den für sie wichtigen Parametern an (Kommunikationspartner: Bernd). Die Intelligenz des Dienstservers steuert den Dienst, indem sie den Partner sucht und die Verbindungen zwischen den Teilnehmern herstellt. Der Dienstserver wählt nach einem Auswahlalgorithmus gemäß voreingestellter Parameter und der aktuellen Situation für den rufenden Teilnehmer eine GSM-Verbindung für die Sprache und eine GPRS/DVB-T Hybridverbindung für den Informationsabruf. Der gerufene Teilnehmer wird über das Internet verbunden. Gleichzeitig schaltet der Server die Informationsquelle für beide Teilnehmer frei. Die Teilnehmer brauchen sich nicht darum zu kümmern, welche ihrer Anschlüsse aktiv genutzt werden. Die Verknüpfung verschiedener Verbindungsabschnitte erfolgt durch die Dienststeuerung (vgl. Beispiel 2 aus Abschnitt 2.6). Daraus lassen sich die Grundkonzepte der Architektur des Servers aufstellen, die in den folgenden Abschnitten erläutert werden (siehe auch [KMS00, Kel00, Kel01a]).

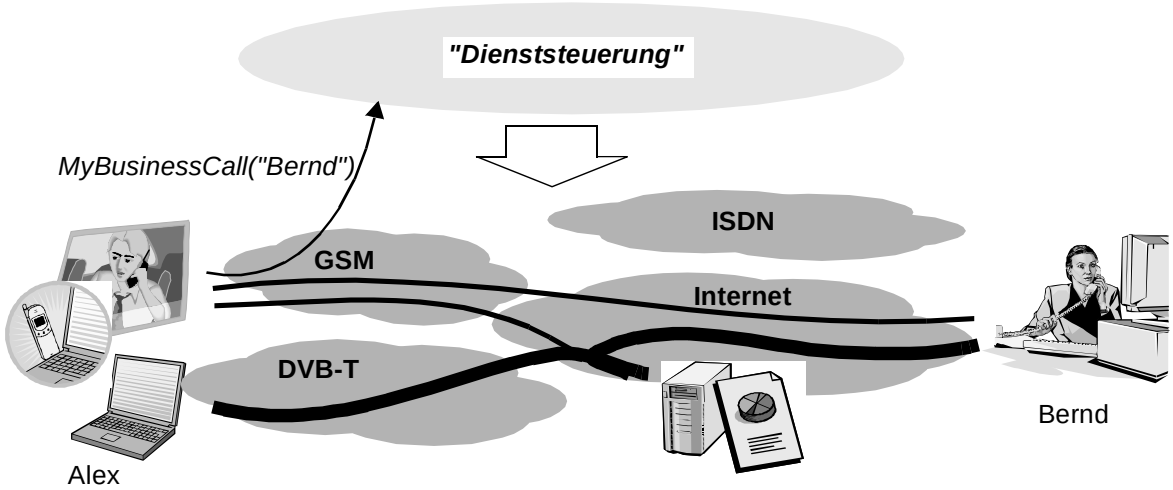


Abbildung 4.1: Dienst-Szenario „MyBusinessCall“

4.1.1 Trennung von Dienstebene und Netzebene

In Abbildung 2.3 auf Seite 14 sind die Funktionsebenen von Kommunikationssystemen dargestellt. Für das Ziel der vorliegenden Arbeit ist die Trennung der Dienstebene von der Netzebene die wichtigste funktionale Trennung. An dieser Schnittstelle sind Anforderungen aus Teilnehmer- und aus Dienstanbietersicht auf die technischen Möglichkeiten und Beschränkungen, die die bestehenden Systeme bieten, abzubilden. Während sich [SK01] schwerpunktmäßig mit einer formalisierten Modellierung dieser Abbildung im Rahmen der Requirements-Spezifikation befaßt, wird in der vorliegenden Arbeit ein technisches System in Form einer Dienstarchitektur beschrieben, das eine solche Abbildung für die Steuerung von Diensten realisiert.

Die Trennung von Dienststeuerung und Netzsteuerung bedeutet in der SAMSON-Architektur, daß die Steuerungsvorgänge für den Dienst unabhängig von netzspezifischen Details durchgeführt werden. Zu den Steuerungsvorgängen zählen die Teilnehmerverwaltung, der Dienstzugang, die Dienstbeschreibung und der Dienstablauf. Diese Entkopplung wird in SAMSON durch eine Anpassungsschicht realisiert. Die Komponenten der Anpassungsschicht (im Folgenden als **Adaptoren** bezeichnet) übersetzen die Steuerungsvorgänge in die jeweilige netzspezifische Signalisierung. Die Aufgaben der Anpassungsschicht sind somit:

- Abbildung der Signalisierung für den Teilnehmer- und Dienstzugang

- Unabhängigkeit des Teilnehmerzuganges von Endgeräte-spezifischen Darstellungsformaten
- Abbildung der Signalisierung für die Teilnehmerinteraktion
- Abbildung der Signalisierung für die Steuerung von Kommunikationsbeziehungen
- Dynamische Anpassung (Adaptivität)

Abbildung 4.2 stellt dieses Grundmodell der Serverarchitektur bestehend aus Dienstebene, Anpassungsschicht und Netz- bzw. Endgeräteebene dar.

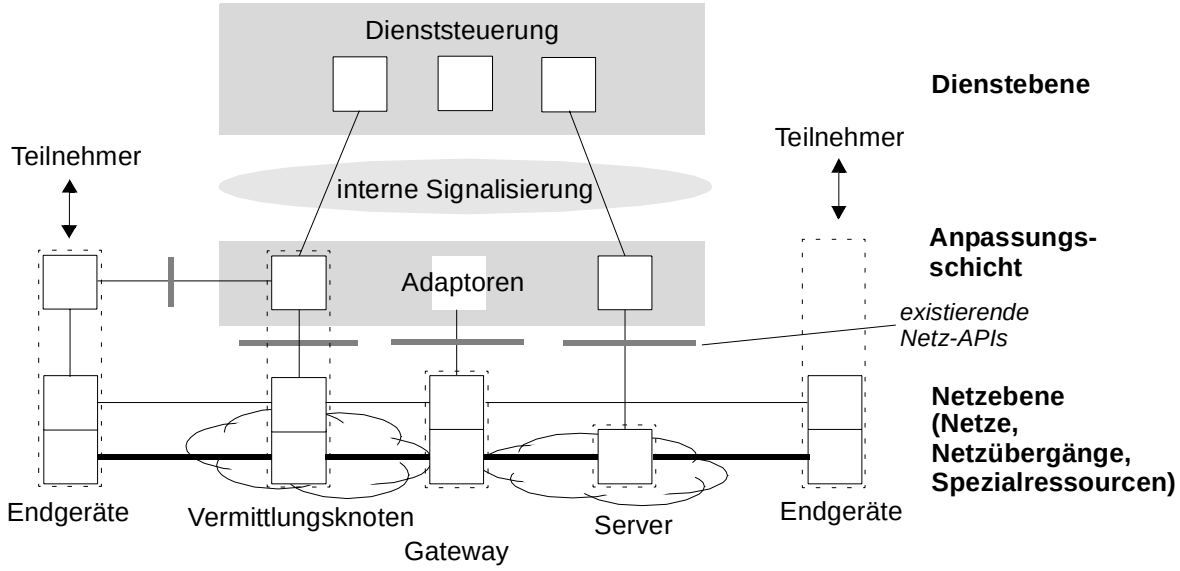


Abbildung 4.2: Grundkonzept der Serverarchitektur

Wir betrachten die Trennung von Dienst und Netz nur für die **Signalisierung**, nicht für die **Nutzdaten**. In der Dienststeuerung werden keine Nutzdaten generiert, verarbeitet oder terminiert. Die Dienststeuerung kann die Verarbeitung von Nutzdaten über entsprechende Komponenten in den Netzen (sogenannte Spezialressourcen) veranlassen und beeinflussen.

4.1.2 Intelligenzverteilung

Der „Sitz“ der Intelligenz für die Dienststeuerung wurde in Abschnitt 2.3 als weiteres Grundcharakteristikum für eine Dienstarchitektur identifiziert. SAMSON verfolgt den Ansatz, die Intelligenz **zentral** in den Netzen anzuordnen. In rein Endgeräte-basierten Systemen fehlen die Möglichkeiten, bestimmte netzübergreifende Funktionen, wie z.B. die Netzauswahl oder die Teilnehmerlokalisierung, zu erfüllen. Außerdem widerspricht ein Endgeräte-basierter Ansatz der Anforderung, beliebige Endgeräte zuzulassen. Ansätze mit verteilter Intelligenz (wie z.B. in TINA [ILM99] oder bei der EPFL-Architektur [GHP99]) haben den Nachteil, daß die Middleware-Plattform auf allen beteiligten Einheiten, d.h. Endgeräten und Netzknoten, verfügbar sein muß. Dies steht der Anforderung entgegen, keine Änderungen in der Infrastruktur notwendig zu machen. Eine zentrale Lösung hat zudem stets den nötigen Überblick über alle Steuerungsvorgänge, so daß signalisierungsintensive Koordinationsverfahren entfallen (vgl. die ICEBERG-Architektur [WRC00]).

Zentralität bedeutet in SAMSON nicht, daß die Dienststeuerung für alle Dienste von einer zentralen Komponente ausgeführt wird, wie dies im Intelligenten Netz [FGK97] der Fall ist. Vielmehr sieht das Konzept vor, für jeden aktiven Dienst eine selbständige Instanz des Dienstes zu erzeugen, welche die Dienststeuerung übernimmt. Diese Instanz kann sich an einem beliebigen physikalischen Ort des SAMSON-Systems befinden.

4.1.3 Dienste und Dienstaufruf

Es ist nicht Ziel des SAMSON-Systems, die Steuerung aller Dienste zu übernehmen. Bestehende Dienste, die bereits in gewissem Rahmen netzübergreifend funktionieren (z.B. die Standard-Sprachtelefonie), können weiterhin von den bestehenden Systemen und Netzen direkt erbracht werden. Solche Dienste bilden für uns die Basisdienste in der Netzinfrastruktur. Der Schwerpunkt bei SAMSON liegt demgegenüber auf der Steuerung von erweiterten Diensten, die

- eine Kombination von Basisdiensten darstellen (Dienste-Konvergenz), z.B. Sprachdialog mit Multimedia-Abruf,
- Basisdienste um Zusatzfunktionen erweitern (*Value Added Services*), z.B. Konferenzfähigkeit,
- Basisdienste um persönliche Funktionen erweitern (*Customization*), z.B. selektive Rufweiterschaltung und
- Dienste über unterschiedliche Netze bereitstellen (Netz-Konvergenz).

Die Dienststeuerung wird dabei entweder direkt vom Teilnehmer adressiert oder Dienste werden durch besondere Ereignisse im Netz angestoßen. Im ersteren Fall kontaktiert der Teilnehmer aktiv den Dienstserver und teilt seinen Dienstwunsch mit. Die Art der Kontaktaufnahme hängt von dem jeweiligen Netz ab, über das der Teilnehmer mit SAMSON verbunden ist (z.B. HTTP über das Internet). Der zweite Fall ist ein Spezialfall, bei dem besondere, „intelligente“ Netzelemente durch gesetzte Triggerpunkte erkennen, daß eine Aktion des Dienstservers erforderlich ist, und den Aufruf an denselben weiterleiten (z.B. IN-SSF/SCF [FGK97], Media Gateways [CGR00]). Im Folgenden liegt der Schwerpunkt auf dem ersten Fall.

4.1.4 Kosteneffizienz

Die Anforderungen aus Sicht einer kosteneffizienten Dienstentwicklung und Diensterbringung sollen in der SAMSON-Dienstarchitektur in zweierlei Hinsicht berücksichtigt werden. Zum einen bietet SAMSON eine Plattform für eine schnelle Dienstentwicklung unter Wiederverwendung der bestehenden Komponenten. Zum zweiten sind spezielle Systemmodule vorgesehen, die eine einfache Teilnehmerverwaltung und Dienstparametrisierung sowie eine adaptive Registrierung von Netzadaptoren ermöglichen, um den Aufwand für den Dienstbetrieb niedrig zu halten.

4.1.5 Software-Architektur

Wie bereits in Kapitel 2 erwähnt, wird eine Dienststeuerung fast vollständig in Software realisiert. Daher drücken sich die beschriebenen Merkmale einer Dienstarchitektur in der Spezifikation der Software-Architektur aus. Buschmann et al. [BMR96] definieren eine **Software-Architektur** als die Beschreibung der Subsysteme und Komponenten eines Softwaresystems und den Beziehungen zwischen ihnen. Diese Erläuterung ist auch Bestandteil der Definition einer Dienstarchitektur (siehe Abschnitt 2.3). Somit läßt sich eine Dienstarchitektur als eine spezielle Softwarearchitektur ansehen. Daher wird vor der weiteren Spezifikation der Dienstarchitektur näher auf die zugrunde liegende Software-Architektur und deren Beschreibung eingegangen.

Weiter heißt es in der Definition des Begriffs Software-Architektur von Buschmann et al. [BMR96], die Subsysteme und Komponenten werden aus verschiedenen Sichten betrachtet, um alle relevanten funktionalen und nicht-funktionalen Eigenschaften des Software-Systems zu zeigen. Im Software Engineering existieren dazu eine Reihe von Modellen und Beschrei-

bungstechniken für unterschiedliche Anwendungsbereiche. Im Folgenden wird daher auf die Modellierung genauer eingegangen. Die Serverarchitektur SAMSON wird aus verschiedenen Sichtweisen abstrakt beschrieben, um Systemmerkmale darzustellen, die über die Grundkonzepte, die in diesem Abschnitt beschrieben wurden, hinausgehen.

4.2 Modellbildung: Sichtweisen auf die Systemarchitektur

Dienstarchitekturen sind durch die vielfältigen Anforderungen äußerst komplexe Softwaresysteme, die aus vielen interagierenden Subsystemen bestehen. Modellierung ist ein Mittel, um mit dieser Komplexität umzugehen. Dies gilt nicht nur für die Softwareentwicklung (z.B. Informationsmodell) sondern auch für betriebswirtschaftliche (z.B. Geschäftsmodell) oder nachrichtentechnische Eigenschaften (z.B. Kanalmodell). Bei der **Modellierung** wird ein System aus verschiedenen Sichtweisen abstrakt beschrieben. Details, die für eine Sichtweise irrelevant sind, werden nicht betrachtet [BD00]. Um alle Aspekte abdecken zu können, umfaßt ein Systemmodell generell mehr als eine Sichtweise.

Leider werden Modellierungsmethoden bei der Entwicklung von Kommunikationssystemen noch zu wenig berücksichtigt. Da das System SAMSON eine komplexe Dienstarchitektur darstellt, wird im folgenden Abschnitt ein Modell entwickelt, dessen Sichtweisen auf die Spezifikation von Dienstarchitekturen abgestimmt sind. Bei der Auswahl werden die in Abschnitt 2.4 erwähnten Merkmale berücksichtigt, die die Entwicklung von Dienstsystemen (*Service Engineering*) im Software Engineering kennzeichnen.

Eine Dienstarchitektur ist dort als ein reaktives System interagierender Komponenten beschrieben. Als Ausgangspunkt werden daher geeignete, bestehende Modelle für verteilte Systeme erläutert.

4.2.1 Modellierung im ODP-Referenzmodell

Das *Open Distributed Processing*-Referenzmodell (RM-ODP) der ITU-T/ISO/IEC [X.901] stellt ein standardisiertes Rahmenwerk zur Entwicklung von Informationsdiensten in einer Umgebung heterogener Ressourcen und interagierender Unternehmenseinheiten dar. Da es aufgrund der Komplexität verteilter Systeme schwierig ist, das System in einer einzigen Beschreibung zu erfassen, definiert das RM-ODP fünf orthogonale Sichtweisen auf die unterschiedlichen Aspekte eines Systems:

- *Enterprise Viewpoint:* Beschreibt den Zweck des Systems in seiner Gesamtumgebung; Es werden u.a. die Rollen aller Einheiten, die am System beteiligt sind, und deren Aktivitäten und Schnittstellen untereinander dargestellt.
- *Computational Viewpoint:* Dekomposition des Systems in logische Funktionskomponenten, die verteilt realisiert werden können; Das System wird als eine Menge interagierender Objekte mit dedizierten Schnittstellen betrachtet.
- *Information Viewpoint:* Legt die Struktur und die Semantik der Informationen im System fest, die zwischen den Komponenten ausgetauscht und in ihren Verarbeitungsprozessen gespeichert werden;
- *Engineering Viewpoint:* Beschreibt konkrete Mechanismen, die notwendig sind, um die Verteilung im System zu realisieren; Hierzu zählt die Spezifikation einer verteilten Verarbeitungsumgebung und die Definition von Kommunikationsprotokollen.
- *Technology Viewpoint:* Legt konkrete Technologien für die Implementierung und Realisierung des Systems fest (Hardware und Software);

4.2.2 Objektorientierte Modellierung

Das RM-ODP bietet ein detailliertes Rahmenwerk für die Aufstellung einer Systemarchitektur, bietet aber keine expliziten Modelle für das Systemverhalten. Diese können zum Beispiel in der objektorientierten Softwareentwicklung gefunden werden. Rumbaugh et al. [RBP91] unterscheiden drei verschiedene Sichtweisen, von denen die Sichtweisen *Functional Model* und *Dynamic Model* das Systemverhalten fokussieren:

- Das *Object Model* beschreibt die Struktur eines Systems, bestehend aus Objekten, Attributen, Verknüpfungen und Operationen. Es wird durch ein Klassendiagramm ausgedrückt. Damit entspricht das *Object Model* in etwa den Sichtweisen *Computational Viewpoint* und *Information Viewpoint* des RM-ODP.
- Das *Functional Model* beschreibt das Systemverhalten hinsichtlich der Informationsverarbeitung. Dies geschieht in Form von Funktionen, Bedingungen und Zuordnungen. Es wird durch Datenflußdiagramme repräsentiert.
- Das *Dynamic Model* beschreibt das innere Verhalten eines Systems. Dabei kann das Verhalten eines Objektes selbst durch Zustandsautomaten dargestellt werden, während für das Verhalten der Objekte untereinander Ablaufdiagramme verwendet werden.

Brügge et al. [BD00] definiert eine weitere Sichtweise auf das Verhalten eines objektorientierten Systems, die auch als *Functional Model* bezeichnet wird:

- Das *Functional Model* nach [BD00] beschreibt die Funktionalität eines Systems oder einer Komponente aus der Sicht eines Benutzers. Hier finden Use-Case-Diagramme Anwendung. Diese Sichtweise entspricht dem Teil des *Enterprise Viewpoint* im RM-ODP, der die Rolle des Benutzers beschreibt.

4.2.3 Formale, mathematische Modelle

Die Systembeschreibung aus unterschiedlichen Sichtweisen hilft, die Systemkomplexität handhabbar zu machen und leichter verständlich darstellen zu können. Um mit den verschiedenen Beschreibungen eine präzise Spezifikation des Gesamtsystems zu erhalten, ist eine eindeutige Basis für alle Sichten notwendig. Mathematische Methoden eignen sich besonders für eine derartige logische Fundierung für die Systembeschreibung. FOCUS [BS01] ist ein Beispiel für eine mathematische Methode zur Entwicklung interaktiver, verteilter Systeme.

In der vorliegenden Arbeit liegt der Schwerpunkt auf einer funktionalen Spezifikation einer Serverarchitektur zur Dienststeuerung. Um die speziellen Anforderungen an Dienstarchitekturen ausdrücken zu können, wird ein Modell für die Beschreibung von Dienstarchitekturen in Anlehnung an bestehende Sichtweisen entwickelt. Eine mathematische Fundierung dieser Sichten ist nicht Aufgabe der vorliegenden Arbeit. Dafür kann auf existierende Konzepte wie z.B. auf die Methode FOCUS zurückgegriffen werden. Die Grundkonzepte und die Beschreibungstechniken für die Systemstruktur, das Verhalten und dynamische Abläufe, die FOCUS für unterschiedliche Systemsichten anbietet, passen grundsätzlich zu den vorliegend beschriebenen Sichten. Nähere Einzelheiten zu FOCUS können [BS01] entnommen werden. Insgesamt gesehen sind die formalen Methoden zu abstrakt, um alleiniges Mittel zur Spezifikation einer Dienstarchitektur zu sein.

4.2.4 Modellierung nicht-funktionaler Anforderungen

Die oben erläuterten Sichtweisen dienen mit Ausnahme des RM-ODP-*Technology Viewpoint* dazu, die funktionalen Eigenschaften, d.h. die Funktionsweise und das Verhalten eines

Systems zu beschreiben. Nicht-funktionale konkretisieren die funktionalen Eigenschaften, indem sie (quantitative) Bedingungen angegeben, wie eine Funktion ausgeführt wird, z.B. die maximale Dauer des Verbindungsaufbaus oder die Verfügbarkeit. Diese Bedingungen resultieren aus der Systemumgebung (der Umwelt), z.B. aus dem Nutzerkreis oder der existierenden Verkehrsbelastung. Nicht-funktionale Eigenschaften stellen daher eine weitere Sichtweise auf ein Kommunikationssystem dar.

Um die Anforderungen an Kommunikationssysteme bezüglich nicht-funktionaler Eigenschaften zu erfassen, werden in der Nachrichtentechnik verschiedene Modelle z.B. Kanalmodelle oder Mobilitätsmodelle verwendet. Sie dienen dazu, die Systemumgebung zu charakterisieren und z.B. deren statistische Eigenschaften mathematisch zu fassen. Diese Modelle haben für die Spezifikation eines Serversystems nur mittelbar eine Bedeutung, da sie nicht-funktionale Anforderungen bestimmen. Jene sind bei der Systemrealisierung notwendig, um beispielsweise Netze und Rechnerplattformen für ein Signalisierungsprotokoll in einer bestimmten Systemumgebung ausreichend zu dimensionieren.

4.2.5 Ein neuer Modellierungsansatz für Dienstarchitekturen

Obige Modellierungskonzepte für verteilte, komponentenbasierte Softwaresysteme sind nicht speziell für die hier betrachteten Kommunikationssysteme entwickelt worden. Sie lassen sich jedoch für die Modellierung von Dienstarchitekturen anpassen, da auch diesen das Grundkonzept verteilter Systeme zugrunde liegt[1].

Die Randbedingungen für die Auswahl der Sichten zur Beschreibung der SAMSON-Dienstarchitektur können aus den besonderen Kennzeichen des *Service Engineering* (siehe Abschnitt 2.4) abgeleitet werden. Aus den dort aufgestellten funktionalen Aufgaben lassen sich folgende Konzepte für die Softwarearchitektur der SAMSON-Dienststeuerung ableiten[2]:

- Modularität, um kurze Dienstlebenszyklen zu unterstützen
- äußere Schnittstellen zur Beschreibung der Interoperabilität
- Unterstützung von Teilnehmerinteraktionen für den Zugang und den Aufruf von Diensten
- Unterstützung von Teilnehmermobilität und Dienste-Mobilität
- Darstellung komplexer Datenstrukturen, um beliebige Kommunikationsbeziehungen und Zustände von Vermittlungssystemen auf Dienstebene beschreiben zu können.
- funktionelle Kommunikationsmechanismen (z.B. Kommunikationsprotokolle) zur Unterstützung der Interaktion zwischen den Modulen

1. Auch die Beschreibung der TINA-Architektur [ILM99] orientiert sich beispielsweise am RM-ODP. Sie wird in einem *Business Model* (Enterprise Viewpoint), *Computational Model*, *Information Model* und einem *Engineering Model* beschrieben.

2. Die Behandlung ungewünschter Interaktionen steht nicht im Fokus der vorliegenden Arbeit. Daher wird deren Modellierung im Folgenden nicht berücksichtigt.

Diese Konzepte lassen sich im Systemmodell für Dienstarchitekturen durch folgende Sichtweisen ausdrücken (siehe auch Abbildung 4.3). Für jede Sichtweise ist jeweils der Bezug zum RM-ODP und zu den objektorientierten Modellen angegeben.

- Das **Geschäftsmodell** (Abschnitt 4.3) beschreibt eine Sichtweise auf die Anwendung des Systems und seine Schnittstellen zur Umgebung (*Enterprise Viewpoint*). Als Notation eignen sich UML-Use-Case-Diagramme.
- Eine Betrachtung aus der **Session**-Sichtweise (Abschnitt 4.4) legt unabhängige Steuerungsbereiche fest (Teil des *Computational Viewpoint, Functional Model* nach [BD00]). Eine Session bezeichnet die Menge aller Funktionen und Daten, die für eine bestimmte Aufgabe notwendig sind. Die Unterteilung aller Aufgaben einer Dienstarchitektur in Sessions schafft eine Dekomposition der Architektur aus Sicht der Benutzer, bevor einzelne Software-Komponenten (Komponenten-Sichtweise) definiert werden. Auf diese Weise läßt sich mit Sessions auch die Dienste-Mobilität erfassen. Für die Session-Sichtweise bieten sich ebenfalls Use-Case-Diagramme als Notation an.
- Die **Informations**-Sichtweise (Abschnitt 4.5) repräsentiert die statische Struktur der Daten und gibt ihre Verknüpfung an (*Information Viewpoint, Teil des Object Model*). Die Informationsstruktur wird formal in Klassendiagrammen beschrieben.

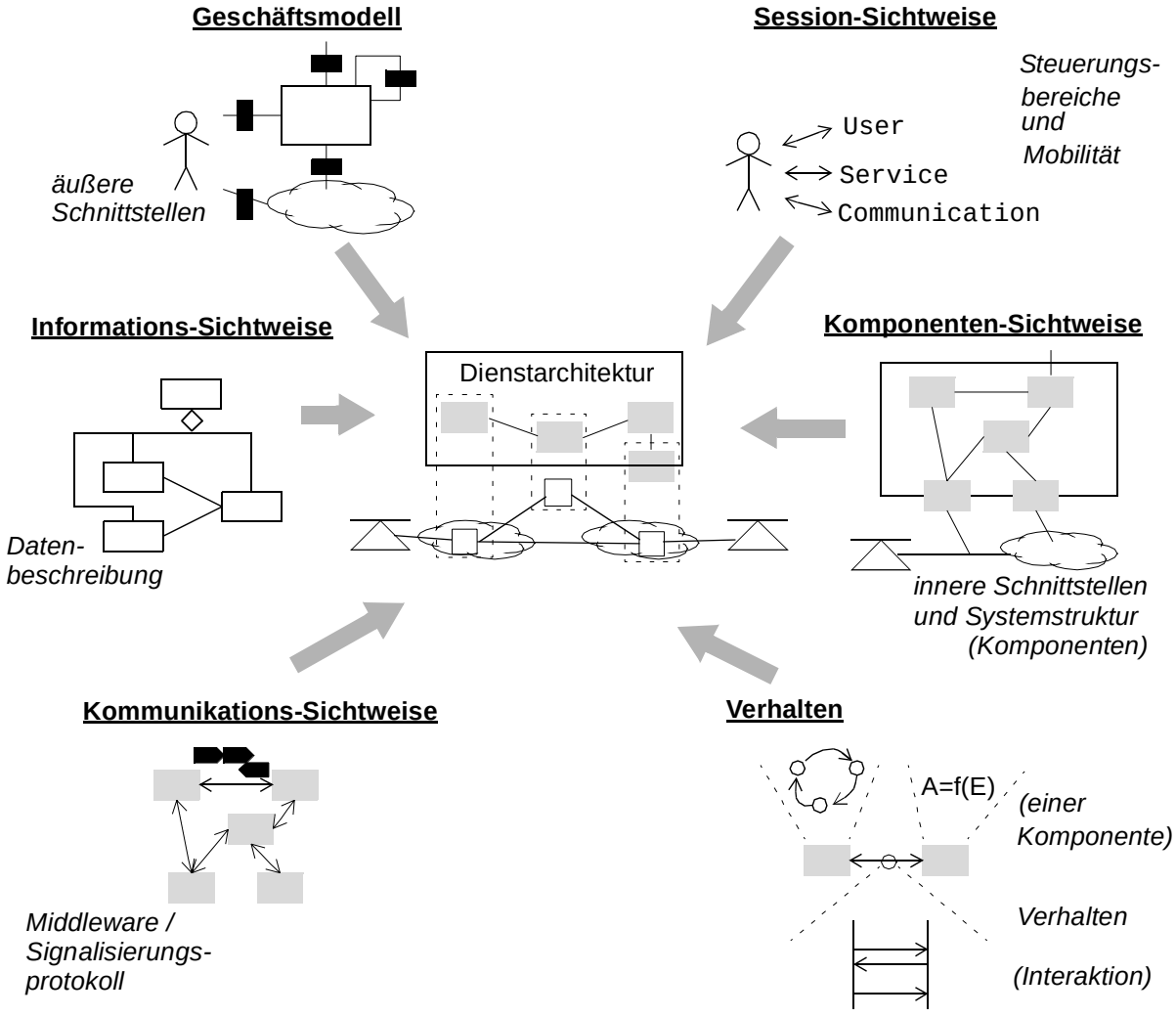


Abbildung 4.3: Sichtweisen zur Modellierung einer Dienstarchitektur

- Die **Komponenten**-Sichtweise (Abschnitt 4.6) fokussiert die Dekomposition der Architektur in interagierende Komponenten und gibt die Schnittstellen dazwischen an (*Computational Viewpoint, Teil des Object Model*). Die Komponenten können wiederum aus Sub-Komponenten zusammengesetzt werden. Als Notation eignen sich SDL-Blockdiagramme oder UML-Klassendiagramme.
- Bei der Betrachtung aus der **Kommunikations**-Sichtweise (Abschnitt 4.7) werden die Anforderungen an das Signalisierungsverfahren (z.B. Signalisierungsprotokoll) auf Dienstebene definiert (*Engineering Viewpoint*). Eine formale Beschreibung hängt von der grundlegenden Entscheidung ab, welcher Kommunikationsmechanismus (z.B. synchron, asynchron) gewählt wird. Die Anforderungen werden meist im Text formuliert.

Aus den obigen Sichtenweisen werden vor allem statische Eigenschaften der Dienstarchitektur im Sinne von verteilten Objekten (Sessions, Datenobjekte, Komponenten) beschrieben. Die Spezifikation des Verhaltens der Komponenten bezieht sich auf die Funktion, die die Komponenten aufgrund interner Vorschriften und Ereignisse oder in Abhängigkeit von externen Eingaben erfüllen. Folgende Sicht auf das dynamische Verhalten wird definiert:

- Die Sicht auf das **innere** System-**Verhalten** drückt das Verhalten der Komponenten durch Zustandsautomaten und die Informationsverarbeitung durch Algorithmen aus (*Dynamic Model* und *Functional Model* nach [RBP91]). Das Verhalten der Komponenten zueinander wird durch Signalabläufe beschrieben. Als Beschreibungstechniken eignen sich SDL-Prozeßdiagramme oder UML-Zustandsdiagramme, Flußdiagramme für Algorithmen und *Message Sequence Charts* [Z.120] oder UML-Ablaufdiagramme.

Alle diese Sichtweisen gehören zur Softwarearchitektur für die Dienststeuerung SAMSON. Sie lassen sich auch allgemein für die Modellierung von Dienstarchitekturen verwenden, da die zugrunde gelegten Konzepte allgemeine Merkmale von Dienstarchitekturen ausdrücken. Das Geschäftsmodell konzentriert sich auf die äußeren Schnittstellen und legt die Interoperabilität der Architektur fest. Interaktionen werden durch Sessions erfaßt. Die Session-Sichtweise fokussiert nicht nur die unterschiedlichen Phasen der Teilnehmerinteraktion, sondern auch die Interaktionen zwischen Steuerungsbereichen und Netzanbietern. Dies beinhaltet auch die Dienste-Mobilität. Die Informations-Sichtweise eignet sich besonders, um die komplexen Datenstrukturen zu beschreiben, die Kommunikationsbeziehungen festlegen oder die Vermittlungssysteme charakterisieren. Auf die Beschreibung der Kommunikationsmechanismen konzentriert sich die Kommunikations-Sichtweise. Die Beschreibung der Modularität der Architektur und der enthaltenen verteilten Komponenten geschieht aus zwei Sichtweisen: Die Komponenten-Sichtweise fokussiert die statische Struktur und die inneren Schnittstellen, während das Verhalten der Komponenten in einer weiteren Sicht Ausdruck findet.

Die angegebenen Notationen stellen Vorschläge für die Verwendung von formalen Beschreibungstechniken dar, deren Einsatz von der Erfahrung der Entwickler abhängt. Für die meisten Sichten kommen mehrere, größtenteils informelle, z.B. textbasierte, Darstellungsformen in Frage, die geeignet sind, die jeweilige Sichtweise zu veranschaulichen. Beschreibungstechniken wie z.B. SDL/MSC [Z.100, Z.120] oder UML [RJB98] eignen sich besonders, da sie Notationen für verschiedene Sichten beinhalten, die in enger Beziehung zueinander stehen. SDL-basierte Entwicklungswerkzeuge wie [Tel00] kombinieren diese Beschreibungstechniken, um eine integrierte Systementwicklung zu ermöglichen. In der vorliegenden Arbeit werden die Sichtweisen aus Platzgründen in einer vereinfachten Notation dargestellt.

Nicht-funktionale Eigenschaften werden nicht in einer eigenständigen Sicht beschrieben, da sie die funktionalen Eigenschaften um quantitative Angaben ergänzen. Sie werden meist in

Textform zusätzlich zu einer Funktionsbeschreibung angegeben. So sind z.B. für SDL-Systeme eine ganze Reihe von Erweiterungen entwickelt worden. Sie erlauben es, Performance-Parameter in der SDL-Spezifikation zu beschreiben, um anschließend eine Performance-Analyse durchführen zu können [HHL01]. In Anhang E wird der Signalisierungsaufwand des Signalisierungsprotokolls von SAMSON (siehe Kapitel 5) ausgehend von der Architektur-Beschreibung mit Signalablaufdiagrammen analysiert.

4.2.6 Zusammenfassung

Es wurden sechs Sichten für die Modellierung von Dienstarchitekturen vorgestellt. Diese Sichten sind eng an die des ODP-Referenzmodells und der objektorientierten Modellierung angelehnt. Die Zusammenstellung wird durch die grundlegenden Konzepte von Dienstarchitekturen motiviert. Neu an den Sichten ist eine separate Betrachtung von Interaktionen und Steuerungsbereichen (Sessions). Ziel des obigen Modells ist es, zu zeigen, wie die Komplexität eines Systems zur Dienststeuerung durch die Beschreibung aus geeigneten Sichten reduziert werden kann. Im Umfeld der Entwicklung von Kommunikationssystemen wird diese Vorgehensweise noch zu wenig beachtet.

4.3 Geschäftsmodell

Das Geschäftsmodell beschreibt die funktionalen Zusammenhänge der Architektur mit ihrer Umgebung, die sich aus betrieblichen Gesichtspunkten ergeben. Hier werden die beteiligten Rollen und deren Zusammenarbeit festgelegt. Durch die Beschreibung des Geschäftszweckes können die Funktionen der Architektur genauer abgegrenzt und auf die notwendigen Details eingeschränkt werden. Insbesondere werden die Schnittstellen der Architektur zur Umgebung festgelegt.

Das SAMSON-Geschäftsmodell unterscheidet die Rollen Dienstanbieter, Netzanbieter, Netzbetreiber und Teilnehmer. Zusätzlich existieren Dienstentwickler und Inhalte-Anbieter. SAMSON beschreibt nur das System des Dienstanbieters. Abbildung 4.4 illustriert das Zusammenspiel folgender Rollen:

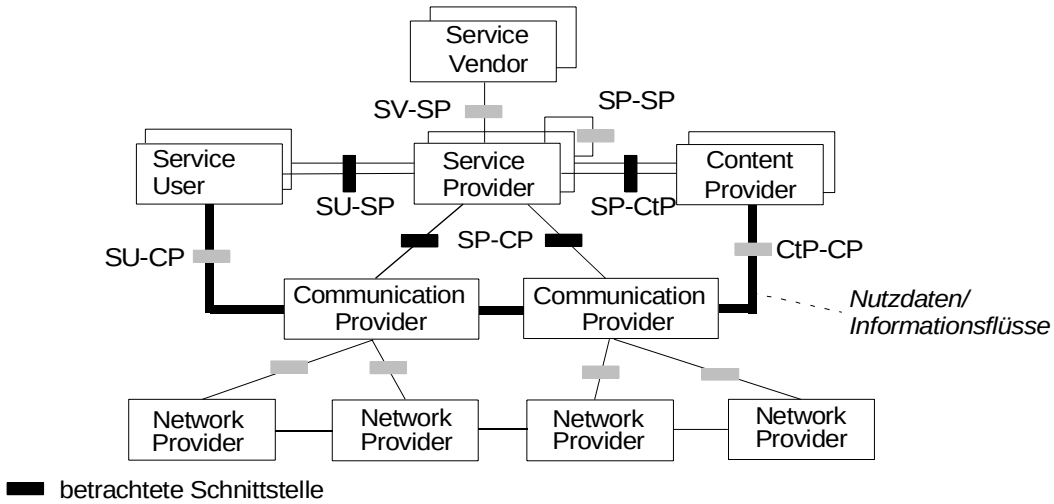


Abbildung 4.4: SAMSON-Geschäftsmodell

- Der **Dienstanbieter** (*Service Provider*) stellt den Teilnehmern Dienste über bestimmte Teilnehmerzugänge zur Verfügung, die er je nach deren Wünschen und der Situation über unterschiedliche Netze ausführt.

- Der **Netzanbieter** (*Communication Provider*) stellt dem Dienstanbieter eine Schnittstelle zur Verfügung, über die dieser Netzressourcen überwachen und steuern kann. Ein Netzanbieter charakterisiert sich durch die Netze, die er verwaltet und in denen er Verbindungen zwischen Teilnehmern und/oder Netzübergängen (Gateways) einrichten kann. Ein Netzanbieter kann verschiedene Kommunikationsnetze verwalten und ist dafür mit unterschiedlichen **Netzbetreibern** (*Network Providers*) verbunden.
- **Teilnehmer** (*Service User*) sind über unterschiedliche Netze mit dem Dienstanbieter verbunden. Eine direkte Verbindung zum Dienstanbieter besteht nur auf logischer Ebene, in der der Teilnehmer durch eine eindeutige Kennung bezeichnet wird. Auf Netzebene sind die Teilnehmer durch ihre Endgeräte und deren Netzadressen gekennzeichnet.
- Die Dienstlogik kann von einem externen Anbieter bereitgestellt werden. Dieser wird als **Dienstentwickler** (*Service Vendor*) bezeichnet.
- An verschiedenen Stellen im Netz bieten **Inhalte-Anbieter** (*Content Provider*) Daten zum (interaktiven) Abruf für den Teilnehmer an (z.B. Video-on-Demand, Spiele-Anbieter). Die Steuerung des Datenabrufs erfolgt direkt zwischen Teilnehmer und Inhalte-Anbieter. Der Dienstanbieter baut die dazu notwendige Steuerungsverbindung auf.

Folgende, logische Schnittstellen zur Umgebung sind für die Serverarchitektur, d.h. für den Dienstanbieter relevant:

- **SU-SP** (*Service User - Service Provider*): Teilnehmerzugang (Registrierung), Dienstzugang (Dienstwahl) und Teilnehmer-Dienst-Interaktion;
- **SP-CP** (*Service Provider - Communication Provider*): Steuerung von Kommunikationsbeziehungen in den Netzen über die Anpassungseinheiten und Überwachung der Netzressourcen (Statusinformationen);
- **SP-CtP** (*Service Provider - Content Provider*): Schnittstelle zur Signalisierung zwischen Dienstanbieter und Inhalte-Anbieter zur Freischaltung für den Teilnehmer und zur Auswahl der Übertragungsnetze;
- **SV-SP** (*Service Vendor - Service Provider*): Zugang zum Dienstanbieter für externe Dienstlogik, z.B. zur übergeordneten Steuerung durch spezifische Anwendungen, die auf die Dienststeuerung zurückgreifen;
- **SP-SP** (*Service Provider - Service Provider)*: Zusammenarbeit zwischen verschiedenen Dienstanbietern;

Die Teilnehmer-Netz-Schnittstelle (**SU-CP**) und die Inhalte-Anbieter-Netz-Schnittstelle (**CtP-CP**), sowie die Schnittstellen der Kommunikationsanbieter zu ihren Netzbetreibern (Network Provider) werden durch SAMSON nicht berührt und sind daher für die Spezifikation der Dienststeuerung nicht relevant.

Zur weiteren Eingrenzung betrachten wir im Folgenden einen Dienstanbieter, der für eine begrenzte Zahl von registrierten Teilnehmern seine Dienste auf einem Serversystem anbietet. Er hat über entsprechende Anpassungseinheiten Zugang zu unterschiedlichen Netzen. Eine Zusammenarbeit zwischen verschiedenen Dienstanbietern wird nicht betrachtet. Es kann sich zum Beispiel um einen neuen Wettbewerber im Zuge der in der Einleitung angesprochenen Deregulierung handeln, der registrierten Teilnehmern einen globalen Zugriff auf personalisierte Dienste ermöglicht. Insbesondere kann das Serversystem dazu dienen, neue Netze, wie

z.B. digitale Verteilnetze, in die Diensterbringung miteinzubeziehen. Ein möglicher Aufstellungsort für das Serversystem ist z.B. die Kopfstelle eines Fernsehverteilnetzes [SK99].

4.4 Modellierung der Steuerungsbereiche durch Sessions

Zur Modellierung von Diensten lassen sich Phasen im Steuerungsvorgang als sogenannte Sessions beschreiben. Die TINA-Architektur ist unter anderem besonders stark durch Sessions geprägt, daher ist dort die folgende Definition entlehnt [ILM99].

> Eine **Session** beinhaltet gemeinsame Aufgaben und Informationen, die von allen Prozessen geteilt werden, die für eine bestimmte Zeit an einem Dienst beteiligt sind.

Diese Definition unterscheidet sich insofern von der herkömmlichen Bezeichnung „Ruf“ für eine Dienstausführung[1], als in einer Session verschiedene Teilnehmer zu unterschiedlichen Zeiten mit unterschiedlichen „Ruf“-Konfigurationen beteiligt sein können, z.B. eine Multimedia-Multipoint-Konferenz. Eine Session kann somit sehr komplexe Prozesse beschreiben. Sie ist nicht auf den Aufgabenbereich Rufsteuerung begrenzt. So kann auch der Abruf des eigenen Teilnehmerprofils eine Session darstellen. Sessions legen keine Systemkomponenten fest, sondern Aktionen.

4.4.1 Session in SAMSON

Gemäß den drei Aufgabenbereichen einer Dienststeuerung (siehe auch Abschnitt 2.3) werden in SAMSON drei Arten von Sessions unterschieden, in denen die drei wichtigsten Rollen aus dem Geschäftsmodell *Service User, Service Provider* und *Communication Provider* involviert sind. Abbildung 4.5 stellt den Zusammenhang der Sessions dar. Eine formale Beschreibung mit einem Use-Case-Diagramm ist in Anhang F enthalten.

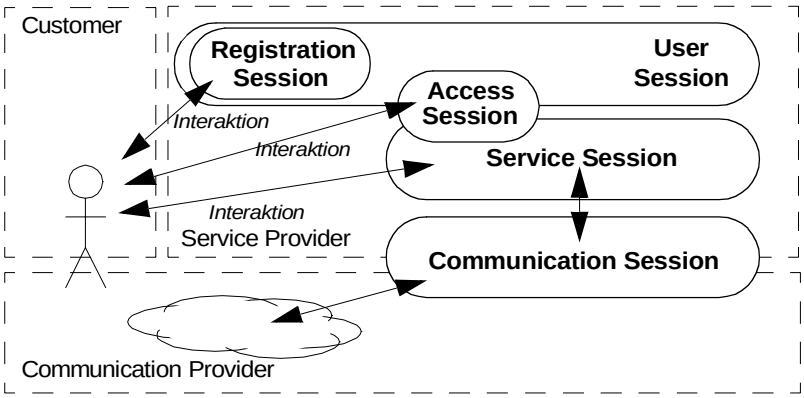


Abbildung 4.5: Modellierung durch Sessions

Die Sessions werden wie folgt definiert:

- Die ***User Session*** beinhaltet alle Funktionen, die die Teilnehmer betreffen und die die Voraussetzung für die Aktivierung eines Dienstes bilden (z.B. Autorisierung, Authentisierung). Für jeden Teilnehmer gibt es genau eine *User Session*. Sie wird weiter unterteilt in eine *Registration Session* und eine *Access Session*.

1. In der Telefonie wird die Beschreibung des Dienstzustandes als Rufmodell (*Call Model*) bezeichnet. Ein Ruf bezeichnet eine zeitlich begrenzte Kommunikationsbeziehung mit fester Zusammenstellung der Verbindungen.

(a) Die ***Registration Session*** beschreibt die Aktivitäten eines Teilnehmers zur Verwaltung seiner persönlichen Einstellungen (z.B. Endgeräte-Profile, Dienstprofile, Erreichbarkeitszeiten). Sie ist unabhängig von einer konkreten Dienstausführung.

(b) Die ***Access Session*** erlaubt (in Anlehnung an die TINA-*Access Session*) dem Teilnehmer den Start eines Dienstes (d.h. einer *Service Session*) oder die Teilnahme an einem bestehenden Dienst (Dienstzugang). Die *Access Session* ist während der Kommunikationssteuerung (*Communication Session*) aktiv, da in ihr die aktuelle Teilnehmerkonfiguration verwaltet wird.

- Die ***Service Session*** beschreibt die Dienstausführung selbst. Sie beinhaltet die zentrale Steuerung eines Dienstes, festgelegt durch die Dienstlogik und die Interaktionen mit den Teilnehmern. In ihr sind die Rollen und Rechte der einzelnen Teilnehmer an einem Dienst bestimmt (z.B. Konferenzleiter). Eine *Service Session* wird durch eine *Access Session* gestartet. Sie bleibt so lange aktiv, bis sie explizit beendet wird, oder der letzte Teilnehmer die Session verlassen hat.
- Die ***Communication Session*** beschreibt die Koordination der Netzressourcen, die an der Einrichtung von Kommunikationsbeziehungen beteiligt sind. Sie wird durch die *Service Session* gestartet und entweder von dieser vorzeitig beendet, was die Auflösung aller Kommunikationsbeziehungen zur Folge hat, oder sie endet mit der *Service Session.*

Diese Einteilung führt zu einer Dekomposition der Architektur in die drei dedizierten Steuerungsbereiche:

- ***User Control*** (Teilnehmersteuerung): Steuerung der *User Session*
- ***Service Control*** (Dienststeuerung): Steuerung der *Service Session*
- ***Communication Control*** (Kommunikationssteuerung):
  Steuerung der *Communication Session*)

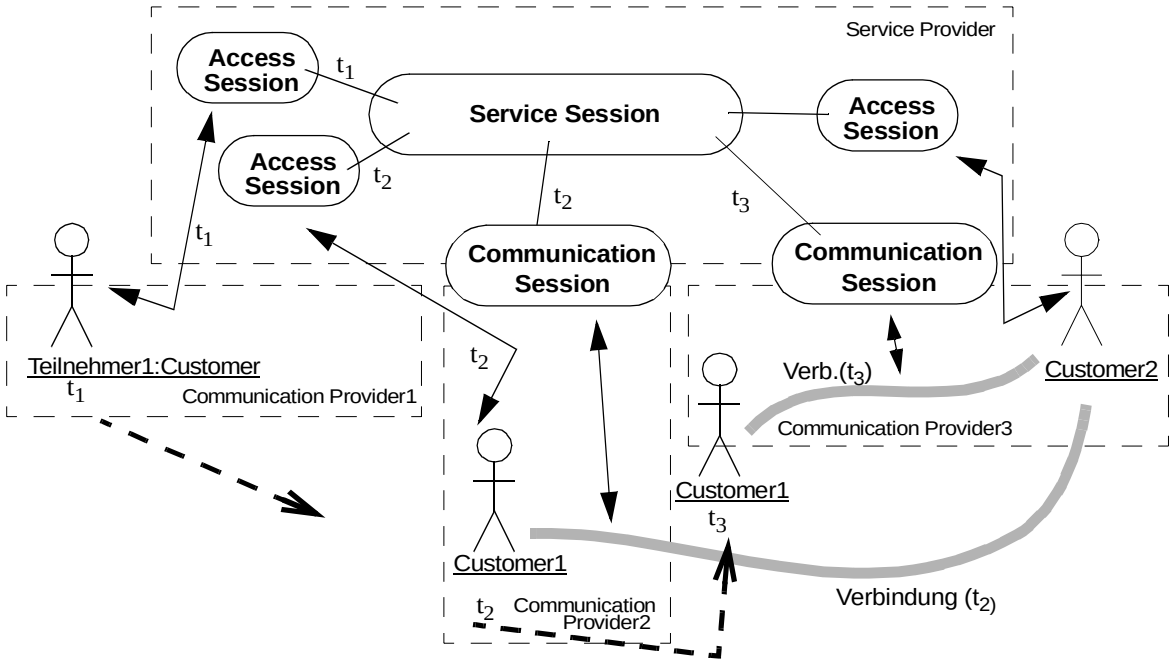
In SAMSON ist für die Steuerung jeder Session eine zentrale Steuerungskomponente vorgesehen, die für jeden Teilnehmer bzw. für jeden Dienst instanziert wird.

4.4.2 Modellierung von Dienste-Mobilität

Die Dekomposition des Dienstablaufes in Sessions bildet außerdem die Basis für Mobilitätsbetrachtungen. Im Fokus steht hier die **Dienste-Mobilität**, die durch SAMSON erbracht wird. Die separate Betrachtung einer *Access Session* erlaubt eine unabhängige Beschreibung des Dienstzuganges von der Dienststeuerung.

Erweitert man die Dienste-Mobilität über den Dienstzugang hinaus auf die Dienstausführungsphase, in der Kommunikationsverbindungen auf- und abgebaut werden, so ist festzustellen, daß in SAMSON derzeit kein Mechanismus für einen **unterbrechungsfreien Wechsel** zwischen verschiedenen Netzen/Endgeräten vorgesehen ist. Durch die Trennung von *Communication Session* und *Service Session* wird ein solcher Wechsel als Neuaufbau einer *Communication Session* modelliert, während die *Service Session* unangetastet bleibt. Dabei wird die Kommunikation zwangsläufig unterbrochen und neu aufgebaut. Soweit die Kommunikationsanbieter einen unterbrechungsfreien *Handover* mit demselben Endgerät innerhalb eines Netztyps unterstützen (Endgeräte-Mobilität), wird diese Funktion genutzt, ohne explizit in der Dienststeuerung bekannt zu sein.

Abbildung 4.6 illustriert die Modellierung der Mobilitätsmechanismen in SAMSON durch Sessions. Teilnehmer 1 greift zu verschiedenen Zeitpunkten (t_i) und aus verschiedenen Netzen



$t_1, t_2, ...$ Existenz und Teilnehmerinvolvierung in den Sessions zu verschiedenen Zeitpunkten t_x

Abbildung 4.6: Modellierung der Dienste-Mobilität durch Sessions

auf die Dienststeuerung zu. Dabei bleibt die zentrale *Service Session* bestehen. Teilnehmer 1 muß je Zugang eine eigene *Access Session* aufbauen. Gleiches gilt für die *Communication Session*, die abgebaut und zur Steuerung der Verbindungen bei einem anderen Netzanbieter wieder aufgebaut wird. Auch hier wird die Dienststeuerung nicht unterbrochen. Die Unterbrechung der *Communication Session* verursacht allerdings einen zeitweiligen Verbindungsverlust zwischen Teilnehmer 1 und Teilnehmer 2. Die formale Beschreibung dieser Darstellung durch Use-Case-Diagramme ist in Anhang F enthalten.

4.5 Modellierung der Information

Im Folgenden werden die Datenstrukturen der Sessions beschrieben. Eine Dienstinstanz wird auf unterschiedlichen Ebenen (*Levels*) modelliert. Die Ebenen sind dabei an den drei Sessions, die die drei grundsätzlichen Steuerungsbereiche (*User Control*, *Service Control*, *Communication Control)* beschreiben, ausgerichtet. Es lassen sich generell die Informationsinhalte der Sessions (bezeichnet mit ***Description***) von den Informationsinhalten der Datenstrukturen, die zwischen den Steuerungsbereichen ausgetauscht werden (bezeichnet mit ***Graph***), unterscheiden. Tabelle 4.1 zeigt die Beschreibungen der unterschiedlichen Modelle. Es sind typische Parameter und deren beispielhafte Belegung angegeben, um die Dienstbeschreibung zu illustrieren. Details werden in den folgenden Abschnitten gegeben.

Die Informationsbeschreibungen hängen nach folgendem Prinzip zusammen: Die Dienstbeschreibung wird bei der Verarbeitung durch die Steuerungsbereiche schrittweise konkretisiert. Am Anfang liegt eine sehr abstrakte Beschreibung eines Dienstes vor, die nur aus der Dienstbezeichnung besteht. Diese wird schrittweise in eine detaillierte Beschreibung umgesetzt, bis schließlich konkrete Verbindungsabschnitte in den Netzen eingerichtet werden können. In allen Beschreibungen gibt es zusätzliche Parameter für die Dienstbeschreibung, die nicht weitergegeben werden, da sie nur eine lokale Bedeutung besitzen (z.B. Rolle eines Teilnehmers in einer Konferenz).

Ebene	Beschreibung	Parameter	Beispiel	Steuerungs-bereich[a]
Access Level	*User Session Description*	Teilnehmerprofil, mögl. Dienstangebote	Alex, MyConference, MyRetrieval	*User Control*
User Level	*User Service Graph*	Dienstname, Parameter aus Teilnehmersicht	MyConference (Bernd, Christa, hohe Qualität)	UC --> SC
Service Level	*Service Session Description*	Anzahl Teiln., Rollen, Beziehung, Medien;	Konferenz (A+V; A/ Leiter, B/Tln., C/Tln.)	*Service Control*
Media Level	*Media Connection Graph*	Medientypen, QoS-Klasse, Beziehung	Audio, Realtime, Konferenz (A, B, C)	SC --> CC
Communi-cation Level	*Communication Session Description*	Explizite Adressen, Netzressourcen, QoS	a-b: H.323, G.711; b-c: ISDN, G.711	*Communic. Control*
Connectivity Level	*Connectivity Connection Graph*	Parameter eines Verbindungsabschnitts	a/Tln.A - b1/GW: H.323, G.711	CC --> NA

a. *User Control* (UC), *Service Control* (SC), *Communication Control* (CC), Adaptoren (NA).

Tabelle 4.1: Modellierungsebenen einer Dienstinstanz in SAMSON

Der funktionale Zusammenhang dieser Dienstbeschreibungen wird in Abbildung 4.7 dargestellt. Der *User Service Graph* wird der zentralen Dienststeuerung (*Service Control*) übergeben. Er enthält wenige, abstrakte Parameter zur Beschreibung des angeforderten Dienstes aus Teilnehmersicht, die im Teilnehmerprofil (*User Session Description*) vorhanden sind. Die *Service Control* stellt die Kommunikationsbeziehung zusammen, die aufgrund der Dienstlogik verlangt wird, und fordert deren Einrichtung mit dem *Media Connection Graph* von der Kommunikationssteuerung (*Communication Control*) an. Für jeden Verbindungsabschnitt, der durch einen eigenen Netzadaptor verwaltet wird, bildet die *Communication Control* einen eigenen *Connectivity Connection Graph*. Dieser beschreibt die konkrete Konfiguration eines Verbindungsabschnittes. Er ist ein Ausschnitt aus der *Communication Session Description*, in der ein Überblick über alle Verbindungsabschnitte zentral verwaltet wird.

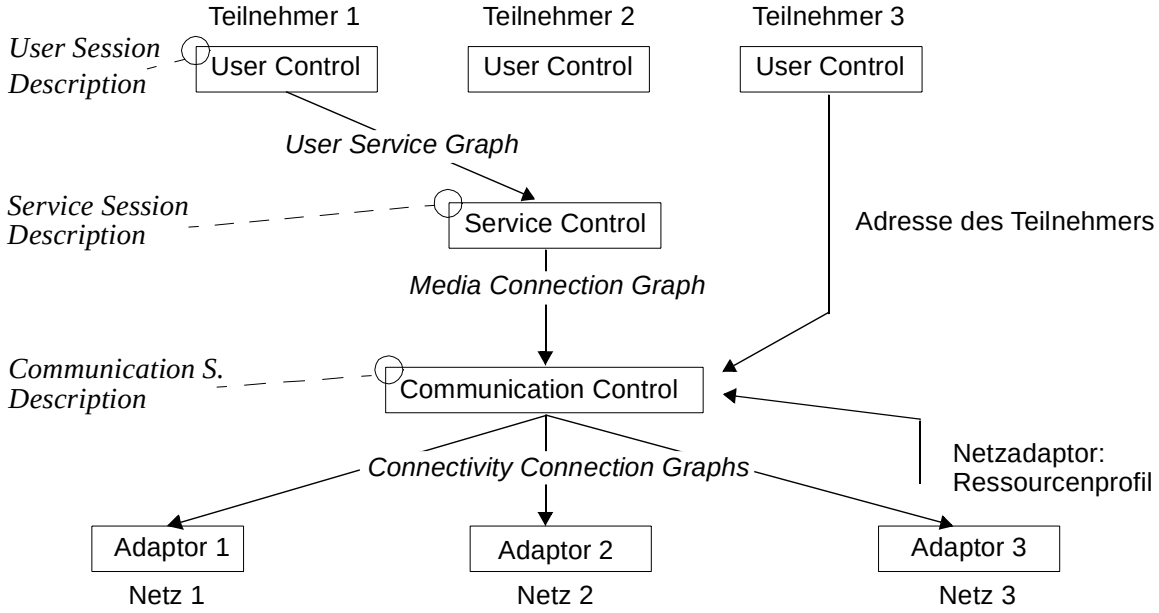


Abbildung 4.7: Funktionaler Zusammenhang der Informationsbeschreibungen

Zunächst werden einige grundlegende Begriffe definiert, um die Eindeutigkeit der folgenden Beschreibungen zu gewährleisten. Eine **Kommunikationsbeziehung** (*Communication Relationship*) kann entsprechend der enthaltenen Medien aus mehreren **Informationsflüssen** (*Information Flow*) bestehen. Ein Informationsfluß beschreibt, ebenso wie eine Kommunikationsbeziehung, eine Ende-zu-Ende-Kommunikation zweier oder mehrerer **Teilnehmer** (*Party*). Ein Teilnehmer (*User*) kann eine beliebige Person oder Maschine sein oder eine der Dienststeuerung bekannte Ressource (Server). Ein Informationsfluß kann nach Bedarf in mehrere **Informationspfade** (*Information Path*) aufgespalten werden. Die Endpunkte von Informationspfaden sind entweder Teilnehmer, vertreten durch deren **Endgeräte** (*Terminal*), oder **Spezialressourcen** (*Special Resource*).

Der *Media Connection Graph* beschreibt alle Kommunikationsbeziehungen eines Dienstes abstrakt als eine Menge von Informationsflüssen (logischer Graph), während ein *Connectivity Connection Graph* nur einen Teil einer Kommunikationsbeziehung wiedergibt. Er legt entweder eine komplette Ende-zu-Ende-Beziehung fest (Informationsfluß) oder nur einen Abschnitt davon (Informationspfad), je nachdem wie eine Kommunikationsbeziehung physikalisch umgesetzt werden kann. Abbildung 4.8 stellt den Zusammenhang der Begriffe mit den verschiedenen Informationsmodellen in SAMSON in einem UML-Klassendiagramm[1] dar.

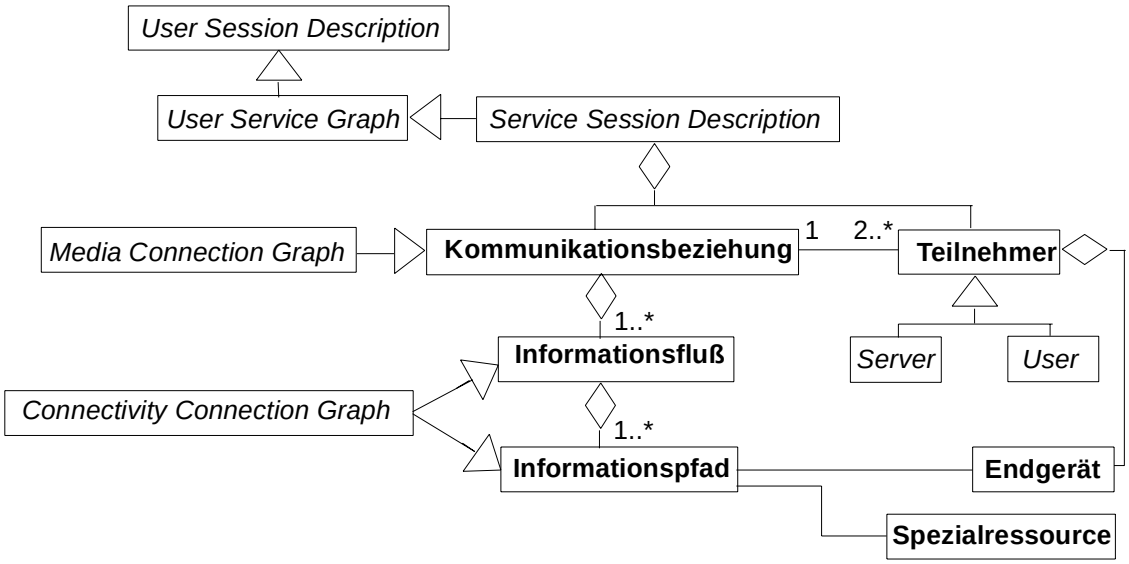


Abbildung 4.8: Begriffsdefinition (UML-Klassendiagramm)

Vor der Erläuterung der verschiedenen Informationsbeschreibungen, die den Sessions zu Grunde liegt, wird ein Modell der Dienstgüte (*Quality of Service, QoS*) aufgestellt. Ihr ist wie in jedem Kommunikationssystem eine zentrale Bedeutung beizumessen, da sie die Auswahl der Netze entscheidend mitbestimmt. Bestandteil jeder Informationsbeschreibung ist außerdem eine eindeutige Kennzeichnung der Teilnehmer. Daher wird der Aufbau der Adressierung von Teilnehmern und Ressourcen abschließend in diesem Abschnitt erläutert.

4.5.1 Modellierung der Dienstqualität

In komplexen Systemen wird die Dienstgüte meist in verschiedenen Abstraktionsgraden modelliert [Sti95, RMS97, ACH96]. Für Dienstarchitekturen typisch ist beispielsweise eine Modellierung, die nach Dienstgüte aus Teilnehmersicht, Endgerätesicht und Transportsicht

1. Eine Kurzbeschreibung der Symbole eines UML-Klassendiagrammes sind in Anhang G abgedruckt.

unterscheidet [Haf94]. Die Teilnehmerdienstqualität beschreibt die Dienstgüte durch subjektive Parameter, wie z.B. „hohe Qualität“ oder „Sprachqualität“. Aus Transportsicht ist die Übertragungsqualität (Übertragungsrate, Delay, Jitter) maßgebend. Für die Endgeräte (Ende-zu-Ende-QoS) ist neben der Übertragungsqualität auch die Darstellungsqualität wichtig. Die Parameter für die Darstellung von Informationen im Endgerät sind für Videoinformationen beispielsweise die Auflösung, die Bildwiederholrate und die Farbtiefe. Diese Parameter werden im allgemeinen durch das Kodierungsformat festgelegt.

In den bereits diskutierten Ansätzen zur erweiterten Ruf- oder Ressourcensteuerung im B-ISDN ([Kni93, Mül96, Que00]) konzentriert sich die Beschreibung der Dienstqualität auf die Endgerätesicht. Die Ansätze gehen davon aus, daß die Dienste in den Teilnehmerendgeräten durch Applikationen festgelegt werden. Eine Teilnehmerdienstqualität wird nicht beschrieben. Ebensowenig wird eine vom Dienst vorgegebene Übertragungsqualität zur Bestimmung der einzurichtenden Verbindungen verwendet. Sie ergibt sich erst aus den Endgerätesichten, die zwischen den Teilnehmern abgeglichen werden.

Die vorliegende Arbeit konzentriert sich auf die Architektur der Dienststeuerung, die durch einen separaten Betreiber übergreifend über unterschiedliche Kommunikationsnetze ausgeführt wird. Es werden die beim Dienstanbieter vordefinierten, personalisierbaren Dienste über Netzzugänge aufgerufen. Diese Dienstdefinition bildet also den Ausgangspunkt für die Gütebestimmung. Dabei werden die Endgeräte zunächst nicht berücksichtigt, sondern die Übertragungsqualität abstrakt und unabhängig von konkreten Netzen beschrieben. Ein Abgleich mit den Endgeräten findet erst kurz vor der Auswahl der Netzressourcen statt. Zusätzlich definieren wir einen Qualitäts-Parameter für die Einbeziehung der Teilnehmerdienstgüte, um die Teilnehmeransprüche berücksichtigen zu können. Vorteil dieser mehrstufigen Beschreibung der Qualität ist, daß die Dienststeuerung von irrelevanten Details frei bleibt. Konkrete Angaben werden bei der Informationsverarbeitung so spät als möglich gemacht und zwar erst dort, wo sie für eine Entscheidung benötigt werden.

Abbildung 4.9 zeigt den prinzipiellen Ablauf der Dienstgütekonkretisierung. Der Teilnehmer definiert in seinem Teilnehmerprofil, das der Dienststeuerung bekannt ist, eine individuelle Teilnehmerdienstgüte (1). Für jeden Dienst wird vom Entwickler eine Dienstgüte definiert, die die minimalen Anforderungen an die Transportdienstgüte abstrakt festlegt (2). Zusammen mit den Endgeräte-Eigenschaften (3), die der Teilnehmer ebenfalls mit seinem Teilnehmerprofil registriert, kann eine konkrete Transport-Dienstgüte (4) bestimmt werden. Diese dient zur Auswahl von Kommunikationsnetzen.

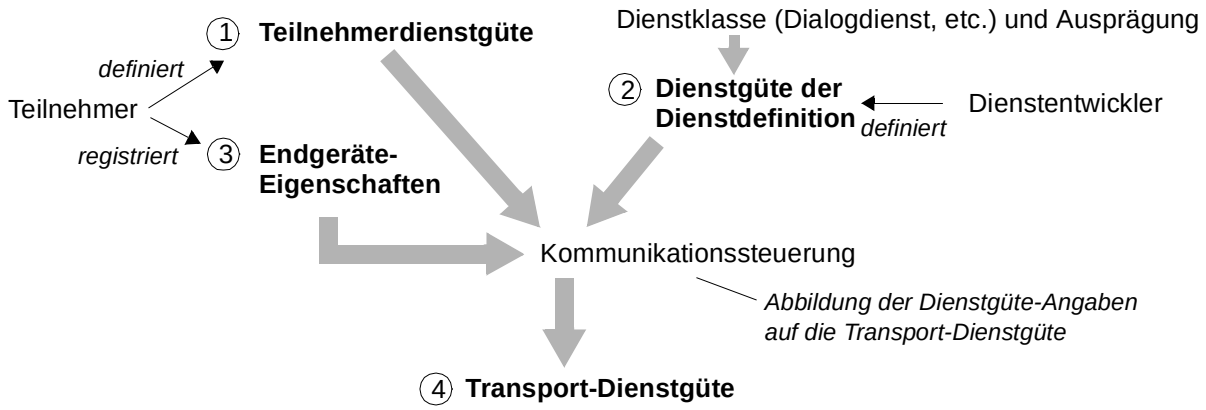


Abbildung 4.9: Ablauf der Dienstgütekonkretisierung

Modellierung der Teilnehmerdienstgüte

In der *User Access Session* können die Qualitätsansprüche der Teilnehmer global für einen Dienst durch drei abstrakte Parameterwerte beschrieben werden (*Overall Quality, OvQ*): „high“, „regular“ und „no care“. Diese Angaben werden im Dienst auf die unterschiedlichen Informationsflüsse (Übertragungsqualität) und deren Darstellung (Endgeräte) umgesetzt. „High“ bedeutet dabei, daß eine spezielle, hohe Qualität des Dienstes für seine Durchführung ausschlaggebend ist. Die Klasse „regular“ bezeichnet Dienste, deren Qualität akzeptabel ist, und „no care“ besagt, daß an die Qualität des Dienstes keine Ansprüche gestellt werden. Das Erreichen aller Teilnehmer (Realisierung aller Kommunikationsbeziehungen) ist hier wichtiger als die Qualität. Tabelle 4.2 zeigt eine Übersicht über diese drei Klassen und gibt Beispiele. Eine ähnliche Klassifikation wurde von ETSI in den TIPHON-Dokumenten speziell für die Klassifikation der Sprachqualität für Internet-Telefonie standardisiert [ETS00]. Die hier gewählten Klassen entsprechen den TIPHON-Klassen „high“, „medium“ und „best effort“.

Nr.	Teilnehmer-Qualitäts-Klasse (OvQ-Klasse)	Beschreibung	Beispiel für Sprache	Beispiel für Video (Telefon)	Beispiel für Daten (Datei-transfer)	Beispiel für Ton/ Musik
1	*high*	besondere Qualitäts-angabe hat höchste Priorität	ISDN-Qualität (G.711)	MPEG2	garantierte Übertra-gungszeit	HIFI (CD-Qualität)
2	*regular*	ausreichend akzeptable Qualität	GSM-FR[a] (LPC) (13 kbit/s)	H.261 (64 / 128 kbit/s)	innerh. ak-zeptabler Zeitspanne	mp3-Qualität
3	*no care*	Kommunikationsbe-ziehung vor Qualität	best-effort VoIP	*Thumbnail*	Eintreffen beliebig	Sprach-qualität

a. GSM-Full-Rate LPC-Code (13 kbit/s)

Tabelle 4.2: Overall Quality-Klassen (OvQ): Qualitätsklassen aus Teilnehmersicht

Modellierung der Dienstgüte in der Dienstdefinition

In der *Service Session* wird die Übertragungsqualität abstrakt aus der Sicht des Dienstes beschrieben. Hier werden - in Anlehnung an die sogenannten *Traffic Classes* bei UMTS [TS23.107] - vier Klassen unterschieden: *Realtime, Streaming, Interaction* und *Best-Effort*. Die Klassen beschreiben die Übertragungsqualität jedes einzelnen Informationsflusses in einem Dienst. Tabelle 4.3 stellt diese Dienst-Qualitäts-Klassen (*Service Quality*-Klassen, SQ) mit Beispielen dar. Die Klasse 1 (*Realtime*) bedeutet, daß in dem bezeichneten Informations-fluß eine interaktive Echtzeit-Konversation zwischen allen Quellen und Senken unterstützt wird. Ein gewisses Maß übersteigende Verzögerungen und Schwankungen in der Übertragung führen zu nicht mehr tolerierbaren Störungen der Konversation. Die zweite Klasse (*Streaming*) bezieht sich auf Dienste, bei denen ein kontinuierlicher Informationsfluß nur in eine Richtung geht. Hier kann eine Verzögerung zwischen Quelle und Senke toleriert werden, so lange die Verzögerung konstant bleibt. Die Klasse 3 (*Interactive*) beschreibt Dienste ohne kontinuierlichen Informationsfluß, bei denen Daten von einem Server interaktiv angefordert werden, wie z.B. WWW-Zugriff. Die Übertragung ist noch weniger kritisch gegenüber Verzögerungen und unkritisch gegenüber Schwankungen. Die Information sollte innerhalb einer bestimmten Zeitspanne ankommen. Für die Klassen 1, 2 und 3 muß ein minimaler Durchsatz vorhanden sein, der sich nach der Art der übertragenen Medien richtet. Die letzte Klasse 4 (*Best-Effort*) stellt

den Anspruch, daß die zu übertragende Information auch tatsächlich ankommt, ohne spezielle Anforderungen an die QoS-Parameter zu stellen.

Diese Dienstgüteklassen beziehen sich auf die Übertragungsqualität und dienen damit direkt der Auswahl von Netzressourcen. Dagegen beziehen sich die Angaben zur Dienstgüte aus Teilnehmersicht (Tabelle 4.2) auf sehr abstraktem Niveau sowohl auf die Darstellungsqualität als auch auf die Übertragungsqualität. Die Teilnehmerdienstgütebeschreibung (OvQ) wird dazu verwendet, innerhalb der vier Übertragungsqualitätsklassen weiter zu differenzieren. Das ist der Fall, wenn eine Wahlmöglichkeit zwischen Netzressourcen gleicher SQ besteht oder, um zu entscheiden, ob gegebenenfalls auch eine schlechtere Klasse verwendet werden darf. Man spricht hier von *QoS Degradation*.

Nr.	**Dienst-Qualitäts-Klasse (SQ-Class)**	**Beschreibung**	**Beispiele**	**Anforderungen an die QoS**
1	*Realtime* *(Interactive Voice /Video)*	garantierte Qualität; erhält alle Zeitbeziehungen der Quellen; realzeitig	• Telefonie • Videokonferenz	• Garantierte Übertragungsrate • Geringer Jitter • Geringe Verzögerung (Delay)
2	*Streaming*	erhält die Zeitbeziehung innerhalb des Informationsstroms; realzeitig	• Real-Time Video	• Geringer Jitter
3	*Interactive* *(Interactive Data)*	die Antwortzeit entspricht den Teilnehmererwartungen; kein Informationsverlust	• Web-Zugriff • Datenbank-abfrage	• Garantierte Gesamtlaufzeit • Geringe Bitfehlerrate
4	*Best Effort*	kein Informationsverlust	• Email • File-Transfer	• Geringe Bitfehlerrate

Tabelle 4.3: Service Quality-Klassen (SQ): Qualitätsklassen aus Dienstsicht

Anforderungen an die Transportdienstgüte

In der Kommunikationssteuerung werden die abstrakten Angaben zur Übertragungsqualität (SQ), die im *Media Connection Graph* von der *Service Session* übergeben werden, in expliziten Zahlenwerten konkretisiert. Mit diesen Angaben werden die Anforderungen an die Netze für die Einrichtung von Informationsflüssen gestellt. Diese Transportdienstgüte-Anforderungen (*Transmission Quality*) umfassen die üblichen Parameter zur Beschreibung der Verkehrs-Charakteristik und der expliziten QoS [Sti95].

- Verkehrs-Charakteristik
    - Spitzenbitrate in bit/s
    - Minimal akzeptierbare Bitrate in bit/s
    - Maximale Burstgröße
- QoS-Parameter (netzseitig)
    - Bitfehlerrate

- maximale Verlustrate (bei paketorientierten Systemen)
- maximal tolerierbare Verzögerung (Delay) in ms
- maximal tolerierbarer Jitter (Variation des Delay) in ms

Je nach Dienst-Qualitäts-Klasse werden nur manche dieser Parameter mit Werten belegt (siehe Tabelle 4.3). Bei Klasse 4 (Best Effort) ist z.B. die Angabe von Bitraten, Delay und Jitter nicht notwendig.

Die Festlegung dieser Transportdienstgüte-Parameter erfolgt in der *Kommunikationssteuerung*. Sie werden aus den Dienst-Qualitäts-Klassen (SQ), der Teilnehmerdienstgüte (OvQ) und den Fähigkeiten der Teilnehmer-Endgeräte (Terminal-Profil) gebildet (siehe Abbildung 4.9). Das Terminal-Profil ist im Dienstserver als Bestandteil des Teilnehmerprofils gespeichert und wird in der *User Registration Session* aktualisiert.

Endgeräte-Eigenschaften

Das Terminal-Profil beschreibt die Charakteristik der Datenquellen und -senken durch die Angabe von Medientypen, den zugehörigen Codecs, Verschlüsselungsverfahren, Multiplexverfahren, Anwendungsprogrammen und Protokollen. Zusätzlich werden die Abhängigkeiten zwischen diesen Parametern angegeben [RMS97]. Zum Beispiel kann der Fall auftreten, daß ein Endgerät aufgrund der verwendeten Anwendungssoftware in einer bestimmten Anwendung einen MPEG-Audio-Codec nur verarbeiten kann, wenn die Daten in einem MPEG-Transportstrom übertragen werden.

4.5.2 Informationsbeschreibung der User Session

Datenbasis der *User Session* ist der Teilnehmerdatensatz, der in der *User Session Description* verwaltet wird. Dieser wird als Sammlung verschiedener Teilnehmerprofile modelliert. Jedes Profil besteht wiederum aus einer Reihe von Datenobjekten. Diese Objekte beinhalten Informationen, die vom Teilnehmer selbst jederzeit geändert werden können (z.B. Endgerätekonfiguration, Präferenzen), oder die nur durch die Dienststeuerung selbst, d.h. durch den Dienstanbieter gesetzt werden (z.B. aktuelle Gebühren, aktive Dienste).

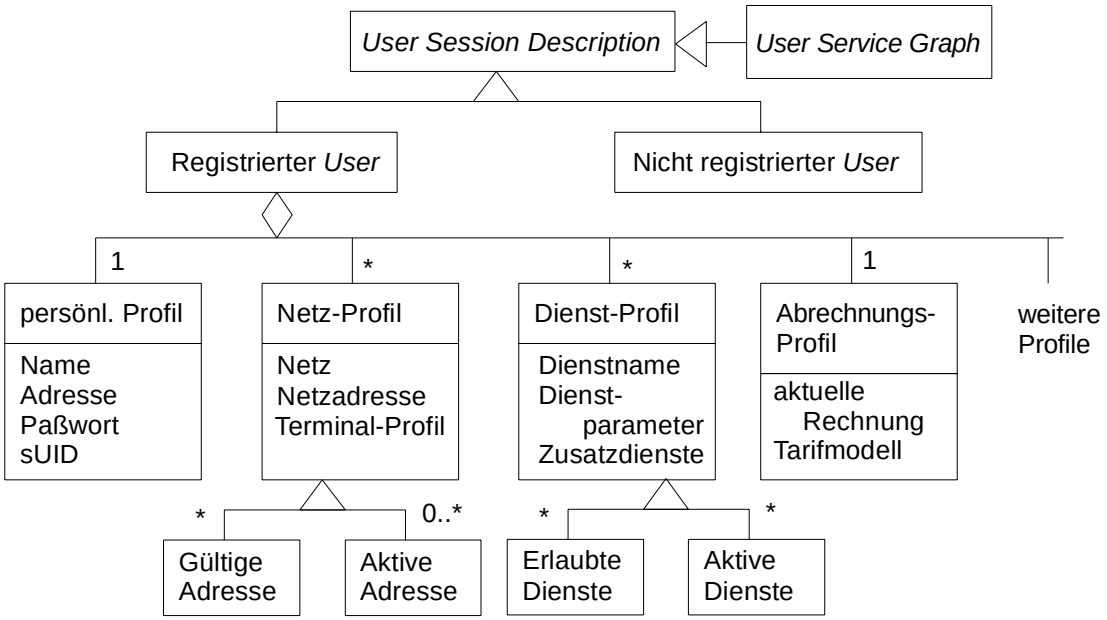


Abbildung 4.10: UML-Klassendiagramm der User Session

Die wichtigsten Objekte sind das persönliche Profil, das Netz-Profil und das Dienst-Profil (siehe Abbildung 4.10). Daneben ist ein Abrechnungs-Profil vorgesehen, das das Tarifmodell des Teilnehmers enthält, sowie je nach Tarif den aktuellen Stand der Rechnung. Beliebige weitere Profile können jederzeit angefügt werden, z.B. die Konfiguration von Applikationen oder ein Adreßbuch. Das Adreßbuch erlaubt z.B. die Umsetzung von privaten Teilnehmerbezeichnungen in interne Bezeichner des Servers oder in explizite Adressen. Die Dienstgütebeschreibung aus Teilnehmersicht ist ein Teil des Dienst-Profils.

Das persönliche Profil enthält alle Angaben des Teilnehmers, die seinen Vertrag mit dem Dienstanbieter betreffen, d.h. Name, Adresse, etc. Jeder Teilnehmer erhält zur Adressierung innerhalb des Servers einen eindeutigen Bezeichner, die SAMSON-*User-ID* (**sUID**).

Das Netz-Profil beschreibt die Eigenschaften eines Teilnehmers aus der Sicht der Netze. Ein Teilnehmer wird neben seiner eindeutigen internen sUID durch eine Reihe von gleichwertigen Adressen beschrieben, über die er in unterschiedlichen Netzen erreichbar ist. Adressen, unter denen der Teilnehmer gerade aktiv mit dem Dienstanbieter verbunden ist oder über die er einen Dienst ausführt, sind speziell gekennzeichnet (aktive Adresse). Details zur Adressierung finden sich in Abschnitt 4.5.5. Das Terminal-Profil beschreibt die Fähigkeiten der Endgeräte, die einem Netz-Profil zugeordnet sind (siehe Abschnitt 4.5.1). Die Schnittstellen zu den Netzen werden durch die Netzadressen der Endgeräte festgelegt.

Für eine zu startende *Service Session*, die von den Teilnehmer-Parametern abstrahiert, ist nur das Dienst-Profil des ausgewählten Dienstes interessant, das im *User Service Graph* während der *Access Session* an die *Service Control* übergeben wird. Es enthält vom Teilnehmer vorbelegte Dienstparameter und Zusatzdienste, die im Rahmen des Dienstes aufgerufen werden können.

4.5.3 Informationsbeschreibung der Service Session

Die *Service Session* beschreibt die Zustände einer aktiven Dienstinstanz. Die in der vorliegenden Arbeit betrachteten Multimedia-Dienste haben eine hohe Zahl an Freiheitsgraden wie Teilnehmerzahl, Zahl und Art der beteiligten Medien, etc. Diese Komplexität macht eine Modellierung der Dienste mit einem herkömmlichen Zustandsautomaten-basierten Rufmodell (wie z.B. beim Intelligenten Netz, Abbildung 3.4 auf Seite 33) nicht mehr sinnvoll möglich. Stattdessen hat sich in einigen Ansätzen (z.B. MAGIC oder TINA) ein objektorientiertes Rufmodell bzw. Session-Modell herauskristallisiert (siehe z.B. [Kni93, Kri97]). Das objektorientierte Session-Modell von SAMSON beschreibt einen Dienst aus einer Kombination von Objekten wie Teilnehmerobjekt oder Medienobjekt. Die Existenz der Objekte und deren Parameter beschreiben den momentanen Zustand einer *Service Session.* Abbildung 4.11 zeigt die objektorientierte Beschreibung der *Service Session*. Diese Beschreibung basiert auf dem in [SK01] beschriebenen generischen Dienstmodell, welches einen Dienst bestehend aus Endpunkten (z.B. Teilnehmer), Kommunikationskanälen (z.B. Medien) und Kommunikationsbeziehungen (Verbindungsgraph) beschreibt. Dieses Modell wird am Lehrstuhl für Kommunikationsnetze derzeit weiterentwickelt und bildet die Basis einer werkzeuggestützten Dienstanalyse im Hinblick auf eine schnelle Dienstrealisierung.

Die Objekte können im Rahmen der Dienststeuerung mit einfachen elementaren Operatoren angelegt, modifiziert oder gelöscht werden. Auf diese Weise läßt sich der komplexe Zustandsraum einer *Service Session* als Verknüpfung einfacher Objektzustände beschreiben.

Die wichtigsten Elemente sind die Teilnehmer-Objekte und die Objekte, die die Kommunikationsbeziehungen beschreiben. Eine *Service Session* wird global durch Attribute wie Bezeichner und Zeitdauer beschrieben. Sie setzt sich aus beliebig vielen (aber mindestens zwei)

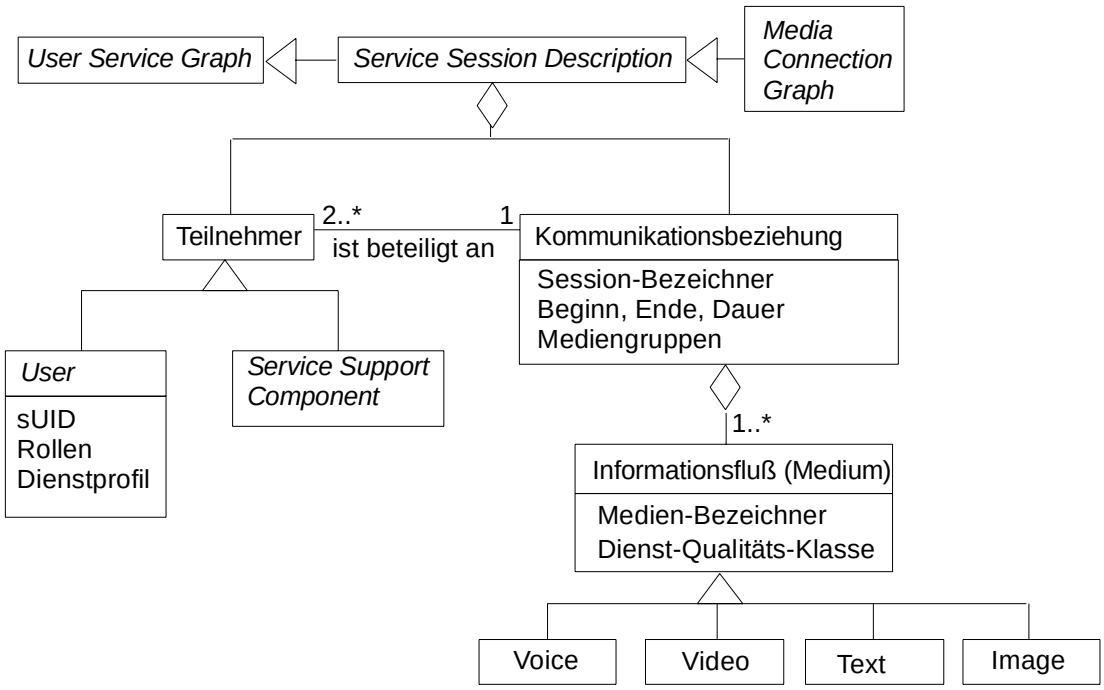


Abbildung 4.11: UML-Klassendiagramm der Service Session

Teilnehmer-Objekten und genau einem Kommunikationsbeziehungs-Objekt zusammen. Ein **Teilnehmer-Objekt** wird durch die Parameter sUID, Art (*User* oder *Service Support Component*), eingenommene Rollen (Initiator, Teilnehmer, Moderator), und damit verbundene Rechte (Änderung eigener/aller/keiner Verbindungen) beschrieben. Im Gegensatz zu einem *User* steht eine *Service Support Component* für einen der Dienststeuerung bekannten Server im Netz, der im Rahmen eines Dienstes Kommunikationsendpunkt sein kann und auf den die Dienststeuerung Einfluß nehmen kann (z.B. Videoserver, Spieleserver). Nicht bekannte Server werden als *User* modelliert. Ein **Kommunikationsbeziehungs-Objekt** besteht aus mindestens einem Informationsfluß. Diese Informationsflüsse werden durch das Medium gekennzeichnet, das sie transportieren (z.B. Voice, Video, Image, Text). Weiterhin wird die Qualität mit den oben beschriebenen abstrakten SQ-Klassen beschrieben.

4.5.4 Informationsbeschreibung der Communication Session

Während für die *Service Session* die Modellierung des dynamischen Auf- und Abbaus von Kommunikationsbeziehungen im Vordergrund steht, fokussiert die Modellierung der *Communication Session* die Art der Kommunikationsbeziehungen. Ein Grundbaustein einer Kommunikationsbeziehung, d.h. ein Informationsfluß wird als Kombination beliebig vieler Kommunikationspfade (*Information Path*) modelliert (siehe Abbildung 4.12). Die Pfade sind durch Spezialressourcen, wie z.B. Konverter und Netzübergänge, miteinander verbunden.

Die Endpunkte eines Informationsflusses sind zwei oder mehrere Teilnehmer. Ein Informationspfad hat immer zwei Endpunkte, die *User* oder Spezialressourcen sein können. Im Gegensatz zu der abstrakten Modellierung der *Service Session* beinhaltet die *Communication Session* eine konkrete Beschreibung der Informationsflüsse und der Informationspfade mit detaillierten Qualitäts-Angaben (siehe Abschnitt 4.5.1).

4.5.5 Aufbau der Teilnehmer- und Ressourcen-Adressierung

Unter dem Begriff Adressierung können generell zwei verschiedene Zwecke verstanden werden. Zum einen dient eine Adresse dazu, eine Instanz (Teilnehmer, Server) in einem verteilten

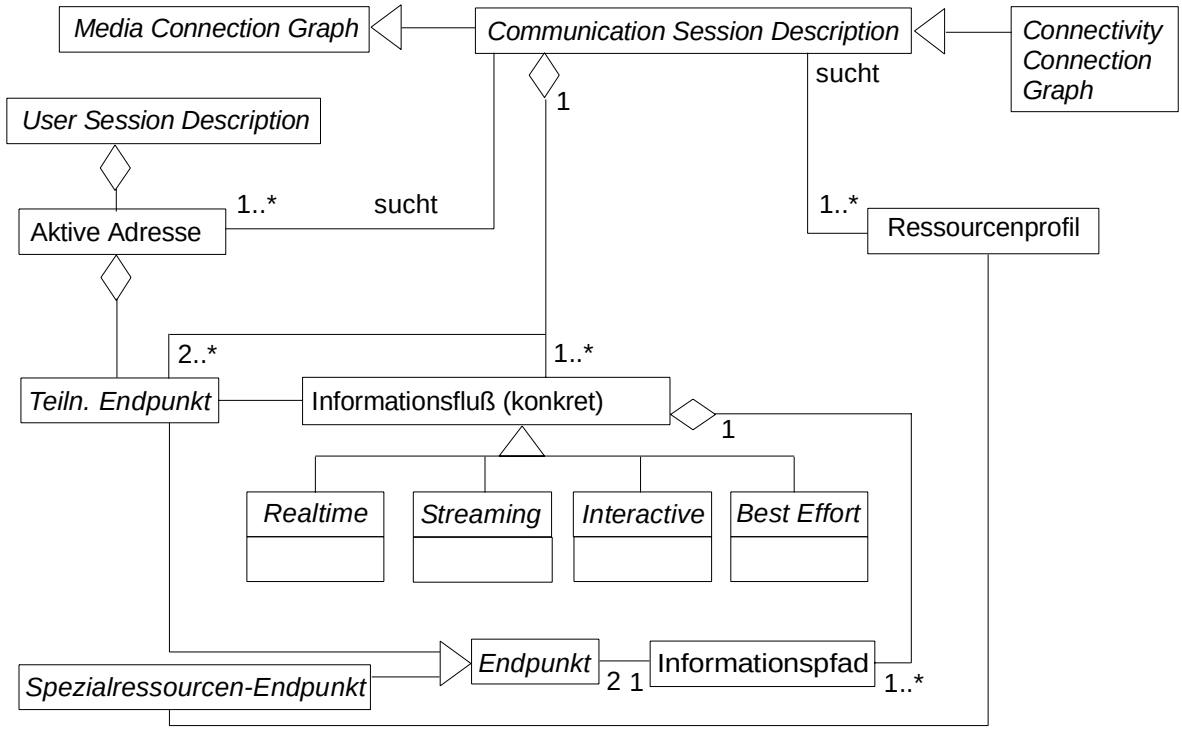


Abbildung 4.12: UML-Klassendiagramm der Communication Session

Kommunikationssystem eindeutig zu lokalisieren, um z.B. in einem Kommunikationsnetz Nachrichten austauschen zu können. Diese **Netzadresse** enthält alle Informationen, die an den verschiedenen Stellen eines Kommunikationssystems benötigt werden, um die Nachrichten zustellen zu können bzw. eine Verbindung dafür aufzubauen. Beispiele hierfür sind IP-Adressen oder E.164-Adressen, deren hierarchische Struktur die Lokalisierung eines Endgerätes (Rechner oder Telefon) erlaubt.

Andererseits kann eine Adresse auch nur ein **Bezeichner** einer Instanz sein, mit dem diese in einer bestimmten Systemumgebung eindeutig identifiziert wird. Ein derartiger Bezeichner abstrahiert von den (komplexen) Mechanismen, die der Auffindung der zugehörigen Instanz dienen. Dieser Bezeichner hat entweder nur eine lokale Bedeutung oder er kann bei Bedarf in eine Netz-Adresse umgesetzt werden. Der Vorteil ist, daß der Bezeichner unabhängig von einem Ort oder einem System vergeben werden kann. Die MAC-Adresse in der Rechnertechnik ist z.B. ein global eindeutiger Bezeichner, der nur durch zusätzliche Verfahren zur Auffindung des zugehörigen Rechners dienen kann.

Für die im vorliegenden spezifizierte Serverarchitektur sind beide Arten der „Adressierung“ wichtig. Zum einen wird ein eindeutiger Bezeichner für die Teilnehmer benötigt, der unabhängig von spezifischen Details der Netzinfrastruktur ist. Zum anderen werden eindeutige Netzadressen benötigt, um zwischen den Endgeräten der Teilnehmer die im Dienst festgelegten Informationsflüsse einzurichten.

Teilnehmeradressen

Die Teilnehmeradressierung erfolgt zweistufig. Zum einen hat der Teilnehmer eine oder mehrere **Adressen**, die ihn bzw. sein Endgerät in den verschiedenen angeschlossenen Netzen eindeutig bezeichnen. Auf jeden Fall ist dem Dienstserver eine Adresse aus dem Netz bekannt, aus dem der Teilnehmer den Diensteserver kontaktiert. Weitere (alternative) Adressen oder Adressen in anderen Netzen kommen hinzu, wenn sich der Teilnehmer z.B. über einen anderen

Adaptor verbindet oder alternative Adressen bei seiner Registrierung angibt. Alle Adressen werden im Teilnehmerprofil gespeichert, um den Teilnehmer in den verschiedenen Netzen erreichen zu können. Der Teilnehmer kann eine Adresse als die primäre Kontaktadresse konfigurieren. Für die Zuordnung der Adressen zu den entsprechenden Netzen wird das Schema <*user*>@<*network*> verwendet.

Hierbei wird für *user* die Adresse des Teilnehmers in einem speziellen Netz verwendet. Das kann eine E.164-Nummer, z.B. Telefonnummer, oder der Alias im Internet sein. *Network* steht für die Bezeichnung des Netzes bzw. der Netzdomäne, in der *user* eine eindeutige Adresse ist. Gültige Adressen sind somit z.B. 08928923505@dtag.tel oder 01721234567@d2.gsm oder kellerer@lkn.e-technik.tu-muenchen.de.net. Eine derartige Adresse wird im Folgenden als die **Netzadresse** eines Teilnehmers bezeichnet. Dieses Schema orientiert sich an der Adressierung bei der IETF Internet Telefonie, festgelegt durch das SIP-Protokoll (siehe Kapitel 5). Auf diese Weise wird nicht nur eine eindeutige Adressierung, sondern auch eine Kompatibilität zu bestehenden Adressierungsschemata erreicht.

Es wird angenommen, daß unter einer derartigen Adresse nur ein Endgerät erreicht wird. Dieses Endgerät bzw. seine Eigenschaften werden ebenfalls mit der Adresse gespeichert. Zusätzlich wird eine Klasse für das Netz angegeben, die generell die Signalisierung in diesem Netz charakterisiert, z.B. „tel“ für ISDN, „gsm“ für GSM, „net“ für Internet, etc.

Für die eindeutige, netzunabhängige Bezeichnung innerhalb der Dienststeuerung wird jedem Teilnehmer ein *Login* und ein *Paßwort* zugewiesen. Beide sind nötig, wenn sich ein registrierter Benutzer einloggt. Das *Login* entspricht der SAMSON-*User-ID* (sUID). Über diese wird das Teilnehmerprofil identifiziert. Alle Netzadressen werden unter der sUID im Teilnehmerprofil abgelegt, damit während der Kommunikationssteuerung darauf zugegriffen werden kann.

Ressourcen-Adressen

Für alle Ressourcen, die auf Dienstebene durch den Teilnehmer direkt bezeichnet werden (z.B. Datenserver), erfolgt die Bezeichnung analog zur Bezeichnung der Teilnehmer. Sie werden durch einen eindeutigen Bezeichner analog zum *Login* gekennzeichnet. Ein Passwort entfällt.

Alle anderen Ressourcen, z.B. Spezialressourcen, die nur der Steuerung durch die *Communication Session* unterliegen, werden mit ihrer Netzadresse bezeichnet, da sie im Rahmen der Diensterbringung einmalig ausgewählt werden (z.B. Mischer, Umsetzer) und nicht übergreifend bezeichnet werden müssen.

4.6 Komponenten

Die Komponenten-Sichtweise fokussiert die Dekomposition der Dienstarchitektur in einzelne Komponenten mit definiertem Funktionsumfang. Dadurch werden die internen Schnittstellen der Architektur festgelegt. Für die Serverarchitektur ergibt sich aus den bisherigen Betrachtungen das in Abbildung 4.13 dargestellte Modell, das analog zu den Sessions drei Steuerungsbereiche unterscheidet.

4.6.1 Zentralisierte Teilnehmerverwaltung (User Control)

Die Aufgaben der Teilnehmerverwaltung werden durch die *User Session* festgelegt. Analog dazu lassen sich Aufgaben, die unabhängig von einem aktiven Dienst ausgeführt werden (Teilnehmer-Zugang und Profilverwaltung) und die einen aktiven Dienst unterstützen (Dienstaufruf, Teilnehmerinteraktion, Teilnehmerprofil-Tracking) unterscheiden. Die Subskription eines

Teilnehmers, d.h. die Vertragsabwicklung mit dem Dienstanbieter bis zum Erhalt einer sUID ist nicht Bestandteil unserer Betrachtungen.

Die Teilnehmerverwaltung besteht aus einer Datenbank (UDB) für die Verwaltung der Teilnehmerprofile, einer zentralen Komponente, die den Zugang zum Dienstserver überwacht (IAM) und Komponenten, die für jeden aktiven Teilnehmer instanziert werden (UP), um diesen in SAMSON zu vertreten. Diese explizite Ausweisung einer Teilnehmerverwaltung stellt eine wesentliche Neuerung der Dienstarchitektur im Vergleich zu bisher eingesetzten Dienstarchitekturen dar.

Die Komponente für den Teilnehmerzugang (*Initial Access Manager*, IAM) ist zentrale Anlaufstelle für die Signalisierungsnachrichten jedes Teilnehmers, der neu mit dem Dienststeuerserver in Kontakt tritt. Der IAM wickelt dabei die Teilnehmeranmeldung ab und prüft die Berechtigung. Bei erfolgreicher Anmeldung erzeugt der IAM eine Instanz eines *User Proxies* (UP), der den Teilnehmer innerhalb von SAMSON vertritt. Alle Signalisierungsnachrichten werden an diesen umgeleitet. Der UP wird mit dem zugehörigen Teilnehmerprofil initialisiert.

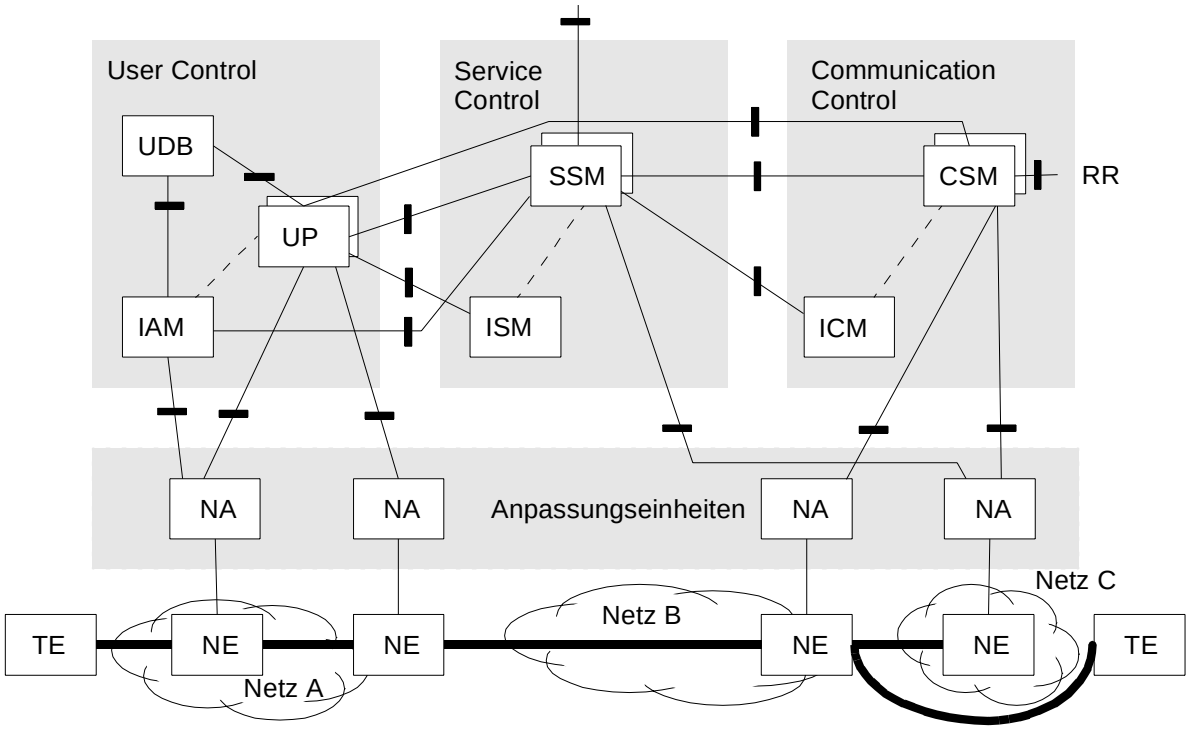


Abbildung 4.13: Komponenten und Schnittstellen der SAMSON-Dienstarchitektur

Der *User Proxy* verwaltet das Teilnehmerprofil. Er ermöglicht es dem Teilnehmer, Parameter im Teilnehmerprofil zu modifizieren und er hält die aktuellen Einstellungen des Teilnehmers (Netzzugang, Endgerät) auf dem Laufenden. Der zweite Aufgabenbereich betrifft die Dienstsignalisierung. Der UP bietet dem Teilnehmer mögliche ausführbare Dienste an (Dienstzugang) und empfängt dessen Auswahl, die er an die entsprechende Komponente der Dienststeuerung weiterleitet. Dem Dienstaufruf fügt der UP entsprechende Parameter des Teilnehmerprofils an, wie z.B. spezielle Diensteinstellungen. Er ist außerdem dafür zuständig, die Teilnehmersignali-

sierung, die während einer Dienstausführung zwischen Teilnehmer und Dienst ausgetauscht wird, an den richtigen Dienst weiterzuleiten. Der UP kann aufgrund von Teilnehmereinstellungen auch selbst Entscheidungen treffen, ohne daß der Teilnehmer involviert werden muß. Der *User Proxy* ist so lange aktiv, wie der Teilnehmer aktiv an SAMSON partizipiert.

Die Teilnehmerdatenbank (*User Data Base*, UDB) wird sowohl durch den IAM als auch durch den UP bedient. Dafür wird dieselbe Schnittstelle zur Informationsabfrage und zur Informationsübermittlung (d.h. Speicherung) verwendet. Eine weitere Schnittstelle in der Teilnehmersteuerung ist die Schnittstelle für den Teilnehmerzugang zwischen IAM/UP und den Adaptoren. Zusätzlich existiert eine Schnittstelle zwischen der Dienststeuerung und dem IAM, um vom Dienst ausgehend einen UP zu erzeugen, z.B. für die Abfrage des Teilnehmerprofils oder zur Gebührenberechnung.

4.6.2 Zentralisierte Dienststeuerung (Service Control)

Die Dienststeuerung erfüllt die Aufgaben der *Service Session*: Sie steuert die Ausführung der Dienste, indem sie die Einrichtung der im Dienst festgelegten oder vom Teilnehmer angeforderten Kommunikationsbeziehungen anstößt, die Dienstzustände verwaltet und auf Ereignisse reagiert. Ereignisse sind externe Ereignisse, wie die Teilnehmerinteraktion, oder interne Ereignisse, wie z.B. der Ablauf eines Timers.

Ähnlich wie bei der Teilnehmerverwaltung unterteilt sich die *Service Control* in eine zentrale persistente Komponente und die Dienstinstanzen. Der *Initial Session Manager* (ISM) erzeugt bei Aufruf eines Dienstes die entsprechende Dienstinstanz und initialisiert sie. Eine Dienstinstanz wird durch einen *Service Session Manager* (SSM) repräsentiert. Der *Service Session Manager* verwaltet das Informationsmodell des von ihm repräsentierten Dienstes und steuert den Dienstablauf gemäß der hinterlegten Dienstlogik.

Neben der bereits genannten Schnittstelle des SSM zur Teilnehmerverwaltung besitzt der SSM eine Schnittstelle zur Kommunikationssteuerung (SSM-ICM/CSM). Zusätzlich ist eine Schnittstelle zur Steuerung von dienstunterstützenden Komponenten (*Service Support Components*) vorgesehen (SSM-NA). Diese werden ebenfalls über einen Adaptor angesprochen.

4.6.3 Zentralisierte Kommunikationssteuerung (Communication Control)

Aufgabe der Kommunikationssteuerung ist die Koordination von Netzressourcen zu einem Dienst aufgrund von Anforderungen aus der Dienststeuerung. Es sind die Informationsflüsse aus dem *Media Connection Graph* in konkrete Verbindungen in den Netzen abzubilden. Um einen entsprechenden Überblick über alle Informationsflüsse eines Dienstes zu wahren, wird eine zentrale Instanz der Kommunikationssteuerung (CSM) für jede Dienstinstanz (SSM) erzeugt. Eine persistente Komponente (ICM) übernimmt diese Erzeugung.

Um den *Media Connection Graph* in konkrete Verbindungen abzubilden, benötigt der CSM zusätzlich Informationen über die aktuellen Einstellungen des Teilnehmers und über die momentane Verfügbarkeit von Kommunikationsressourcen. Die notwendigen Teilnehmerinformationen können über eine entsprechende Schnittstelle direkt vom *User Proxy* erhalten werden. Für die Auswahl der Netzressourcen existiert eine weitere Schnittstelle zu einer Datenbank (*Resource Registry*, RR), in der die aktuell verfügbaren Netzressourcen mit deren Parametern gespeichert sind. Aufgrund der erhaltenen Informationen trifft der CSM nach einem bestimmten Algorithmus, der in Abschnitt 5.5 detailliert beschrieben wird, seine Auswahl und fordert über eine weitere Schnittstelle (CSM-NA) von den ausgewählten Netzressourcen den Aufbau von Verbindungen (*Information Flow*) oder Teilverbindungen (*Information Path*) an.

4.6.4 Schnittstellen zu den Netzen: Anpassungseinheiten

Wie in Abbildung 4.13 zu sehen ist, bilden die Anpassungseinheiten die zu anderen Rollen des Geschäftsmodells gerichteten Schnittstellen (IAM-NA, UP-NA, SSM-NA, CSM-NA) auf die entsprechenden Signalisierungsschnittstellen der Netzinfrastruktur ab. Die Schnittstellen des SAMSON-Geschäftsmodelles SC-SP, SP-NP und SP-CP werden also durch die Anpassungseinheiten in einen internen, generischen, d.h. netzunabhängigen Teil und in einen Netz- bzw. Infrastruktur-abhängigen Teil geteilt.

4.7 Kommunikation

Die Betrachtung der Architektur aus der Kommunikations-Sichtweise beschreibt die Konzepte, die den Austausch von Informationen zwischen Komponenten der Dienststeuerung unterstützen, und bildet die Grundlage für die Signalisierung. Die Systembeschreibung aus Kommunikationssicht dient auch als Basis für das Verhalten der Komponenten. Eine detaillierte Beschreibung des Verhaltens aller SAMSON-Komponenten würde den Rahmen dieser Arbeit sprengen. Daher wird auf eine Darstellung der SAMSON Softwarearchitektur aus der Sichtweise des inneren Verhaltens in einem eigenen Abschnitt verzichtet. Die Beschreibung des Systemverhaltens mit Signalablaufdiagrammen und Zustandsautomaten wird im Gegensatz zu den anderen ausführlich dargestellten Sichtweisen bereits vielfach in der Entwicklung verwendet. Die Art der Darstellung ist somit bekannt. Kapitel 5 enthält als Verfeinerung des Kommunikationsmodells eine Spezifikation der wichtigsten dynamischen Eigenschaften der Komponenten auf der Ebene eines neu entwickelten Signalisierungsprotokolls.

Die in Abbildung 4.13 dargestellten Komponenten der Dienstarchitektur in SAMSON stellen ein System aus verteilten Einheiten dar. Die Kommunikation erfolgt asynchron über den Austausch von Nachrichten. Der Meldungsaustausch wird durch eine Folge von Transaktionen beschrieben. Eine Transaktion besteht dabei aus einer Anfrage (*Request*) und einer oder mehrerer Antworten (*Responses*), die durch die Anfrage ausgelöst werden. Das zugehörige Signalisierungsprotokoll wird als Transaktionsprotokoll bezeichnet.

Zur Modellierung der Kommunikationsabläufe wird der Nachrichtenaustausch innerhalb von SAMSON zunächst als geschlossenes Modell betrachtet. Das bedeutet, es werden im Folgenden die Eigenschaften eines internen Signalisierungsprotokolls in einem exklusiven Signalisierungsnetz beschrieben. Nach außen werden die Nachrichten über die Anpassungseinheiten umgesetzt.

4.7.1 Anforderungen an das Signalisierungsverfahren

Das Signalisierungsprotokoll unterstützt den Austausch der Informationen zwischen den SAMSON-Komponenten, daher stellen die bisher beschriebenen Sichweisen auf die Architektur Anforderungen an das Signalisierungsprotokoll dar. Das Signalisierungsprotokoll hat die Trennung der Architektur in die drei Sessions zu reflektieren und Mechanismen vorzusehen, die eine Teilnehmerverwaltung und den Dienstzugang sowie Dienststeuerung und Kommunikationssteuerung unterstützen. Dazu sind auch die entsprechenden Informationsmodelle der Sessions und deren Austausch zwischen den Sessions zu gewährleisten.

Bisherige Signalisierungsverfahren wie z.B. INAP [Q.1228] oder ISDN-Schicht-3 [Q.931] haben für jedes Ereignis, das einen Zustandswechsel hervorruft, eine eigene Meldung vorgesehen (z.B. SETUP, ALERTING,...). Dies ist durch die Mächtigkeit des objektorientierten Modells nicht mehr möglich, da die Anzahl der Meldungen zu hoch wäre. Außerdem kann eine Erweiterbarkeit für neue Dienste nicht gewährleistet werden, ohne daß der Meldungssatz um neue Ereignismeldungen ergänzt werden muß.

Um den Aufbau und die Steuerung komplexer multimedialer Dienste zu unterstützen, muß das Signalisierungsverfahren aus einem Satz einfacher, elementarer Meldungen bestehen, die den Transport beliebiger Session-Informationen erlauben. Der Meldungsaustausch muß von der transportierten Information unabhängig sein. Falls eine Signalisierungstransaktion in einer Komponente nicht erfolgreich ausgeführt werden kann (z.B. erfolglose Einrichtung eines Informationsflusses aufgrund von Überlast), ist vorzusehen, daß die anfordernde Instanz darüber informiert werden kann. Da in einer Meldung mehrere Informationselemente übertragen werden, z.B. der gesamte *Media Connection Graph* angefordert wird, ist zudem ein geeigneter Aushandlungsmechanismus zu unterstützen, der es erlaubt, einzelne Elemente separat oder in Abhängigkeit voneinander abzugleichen.

Damit ergeben sich die folgenden Grundanforderungen an die Signalisierung:

- Unterstützung von Teilnehmerverwaltung und Dienstzugang
- Aufbau und Steuerung komplexer multimedialer Dienste
- Unterstützung der Kommunikationssteuerung
- Berücksichtigung der Trennung von Funktionen in die drei Steuerungsbereiche Teilnehmer-, Dienst- und Kommunikationssteuerung

Zur Realisierung dieser Aufgaben sind folgende Detailanforderungen zu berücksichtigen:

- Unterstützung der objektorientierten Informationsmodelle
  - Begrenzter Satz an elementaren Nachrichten, die Aktionen vom Empfänger anfordern;
  - Die Aktionen ergeben sich aus der transportieren Information, d.h. aus den transportierten Informationselementen und der Funktionalität des Empfängers.
  - Die Meldungen sind unabhängig von der transportierten Information.
  - Die Einführung neuer Dienste darf keine Änderung am Meldungssatz bewirken.
  - Eine Meldung kann mehrere Aktionen auslösen.
  - Die Länge der Meldungen ist variabel, um beliebige Informationselemente transportieren zu können.
- Unterstützung eines geeigneten Aushandlungsmechanismus
  - Die anfordernde Instanz muß über den möglichen Ausgang einer Anforderung/ Aktion informiert werden, bevor die Anforderung endgültig ausgeführt wird (Rückwirkung).
  - Angeforderte Aktionen (ausgelöst durch Informationselemente) können als optional oder obligat gekennzeichnet werden.
  - Angeforderte Aktionen können gruppiert werden, um z.B. die Einrichtung eines gesamten Informationsflusses in Auftrag zu geben.
  - Eine Anforderung/Transaktion kann mehrere handelnde Instanzen umfassen, d.h. die Informationselemente können an eine andere Instanz weitergegeben werden und dabei aufgeteilt werden. Das Zusammenwirken mehrerer Instanzen muß koordiniert werden.
  - Zusammengehörige (gruppierte) Aktionen dürfen nur als Ganzes endgültig ausgeführt oder abgelehnt werden (Atomizität).
  - Eine Anforderung kann rückgängig gemacht werden.

- Unterstützung von Informationsanforderung und Informationsmitteilung zwischen den Komponenten, z.B. Datenbankabfrage
- Mechanismen zur Beendigung der Teilnahme eines Teilnehmers und zur Beendigung des gesamten Dienstes

4.7.2 Signalisierungsprinzip

Aus den obigen Anforderungen ergibt sich folgendes Signalisierungsprinzip für SAMSON: Das Signalisierungsverfahren trägt der Unterteilung der Architektur in drei Steuerungsbereiche Rechnung. Dabei ist nicht jeder Steuerungsbereich Endpunkt einer Signalisierungstransaktion (wie üblich), sondern eine Transaktion kann einen Steuerungsbereich durchlaufen. Meldungen können modifiziert werden. Abbildung 4.14 illustriert dieses Prinzip.

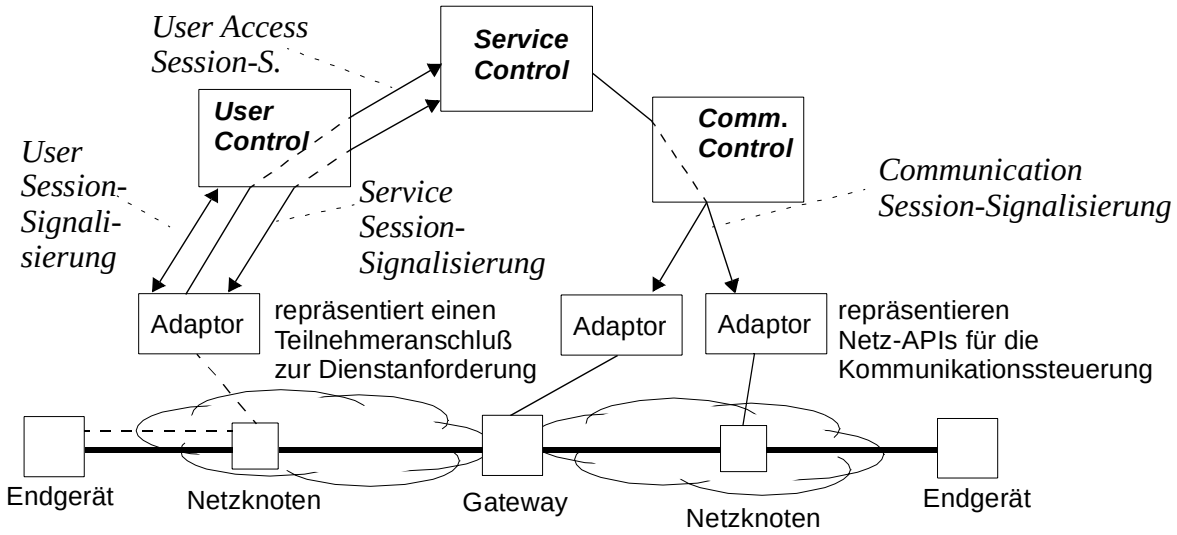


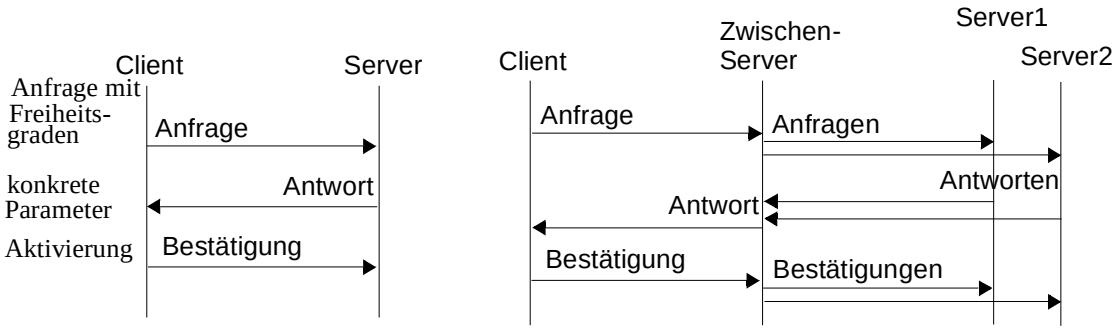
Abbildung 4.14: Signalisierungsprinzip

In der *Access Session* wird eine Dienstinstanz erzeugt. Die *Service Session* wickelt die Interaktionen der Teilnehmer mit dem Dienst ab. Die Signalisierung zur Anforderung von Informationsflüssen in den Netzen ist davon entkoppelt. Die Dienstlogik entscheidet, ob und wann eine Teilnehmeranfrage zu einem Verbindungsaufbau führt, der mittels ausgewählter Adaptoren in den Netzen durchgeführt wird. Auf diese Weise besteht keine Ende-zu-Ende Signalisierungsbeziehung zwischen den Adaptor-Prozessen, die eine Dienstanforderung stellen und den Adaptor-Prozessen, welche die Kommunikationssteuerung übernehmen. Beide Prozesse können aber in demselben Adaptor laufen.

Zur Unterstützung der Aushandlung und Abstimmung der Dienstbeschreibung werden drei Konzepte vorgesehen: ein dreiphasiger Meldungsablauf, die Verteilung von Nachrichten innerhalb einer Transaktion und die Möglichkeit, mehrere, gruppierte Aktionen durch die transportierte Information auszulösen.

Der dreiphasige Meldungsablauf besteht aus einer Anfrage, einer Antwort und einer Bestätigung (siehe Abbildung 4.15a). Eine transportierte Dienstbeschreibung löst erst dann im Empfänger (Server) endgültige Aktionen aus (z.B. Durchschalten einer Verbindung), wenn die Bestätigung angekommen ist. Der Server kann die Dienstbeschreibung im Rahmen der Freiheitsgrade der Anfrage (z.B. alternative Adressen) in der Antwort modifizieren.

Dieser Ablauf ist insbesondere dann vorteilhaft, wenn Anfragen gleichzeitig an mehrere Empfänger gestellt werden. Dies ist zum Beispiel der Fall, wenn zur Einrichtung einer Kommunikationsbeziehung mehrere Ressourcen an einem Informationsfluß beteiligt sind. Das Konzept



a) Dreiphasiger Meldungsablauf b) Dreiphasiger Meldungsablauf mit Verteilung

Abbildung 4.15: Dreiphasiger Meldungsablauf

der Verteilung erlaubt es, dieselbe Meldung an unterschiedliche Server weiterzureichen, die für die erfolgreiche Ausführung einer Anfrage-Transaktion notwendig sind (siehe Abbildung 4.15b).

Dabei ist zu gewährleisten, daß die in der Dienstbeschreibung angeforderten Aktionen alle ausgeführt werden können. Eine erfolgreiche Einrichtung einer Kommunikationsbeziehung hängt z.B. von der erfolgreichen Einrichtung aller zugehörigen Pfade ab. Kann eine einzelne Aktion nicht ausgeführt werden, so ist sicherzustellen, daß keine der zusammengehörigen Aktionen ausgeführt wird. Der Client wird erst dann eine Bestätigung ausgeben, wenn alle Server positiv geantwortet haben. Andernfalls sind die Anfragen rückgängig zu machen und modifiziert wieder auszugeben.

4.7.3 Zusammenfassung

Die Kommunikations-Sichtweise repräsentiert die prinzipiellen Eigenschaften des Signalisierungsverfahrens auf Dienstebene. Anstelle von Meldungen, die bestimmten Aktionen zugeordnet sind, verlangt die objektorientierte Dienstbeschreibung ein transaktionsbasiertes Signalisierungsprotokoll, bei dem die Informationen als Nutzlast ausgetauscht werden. Zusätzlich unterstützt das Signalisierungsverfahren Konzepte zur Verteilung und zur Aushandlung von Beschreibungen zwischen den getrennten Steuerungsbereichen der Serverarchitektur.

4.8 Funktionsweise der Dienstarchitektur

Um die Funktionsweise der Serverarchitektur insgesamt und die Zusammenarbeit der einzelnen Steuerungsbereiche zu illustrieren, wird im Folgenden ein Beispiel für einen typischen Dienstablauf beschrieben. Dabei wird auf die oben aufgestellten Sichten Bezug genommen, indem die Abläufe in der Beschreibung der Komponenten-Sichtweise dargestellt und die Informationsbeschreibungen zu verschiedenen Zeitpunkten angegeben werden. Einige Abläufe werden mit vereinfachten Signalablaufdiagrammen beschrieben, um die Kommunikationsprinzipien zu illustrieren.

Szenario „Konversation zweier Teilnehmer mit Videoabruf“

Dieses Beispiel wird an dem aus zwei Basisdiensten kombinierten Dienstszenario („MyBusinessCall“) aus Abschnitt 4.1 illustriert. Der Sprachdialogdienst dient zur Verdeutlichung der allgemeinen Funktionsweise der Dienststeuerung. Durch die Hinzunahme des neuartigen Informationsdienstes für den Multimedia-Abruf wird die Netzunabhängigkeit und Adaptivität verdeutlicht. Es wird für das folgende Beispiel angenommen, daß alle Teilnehmer beim

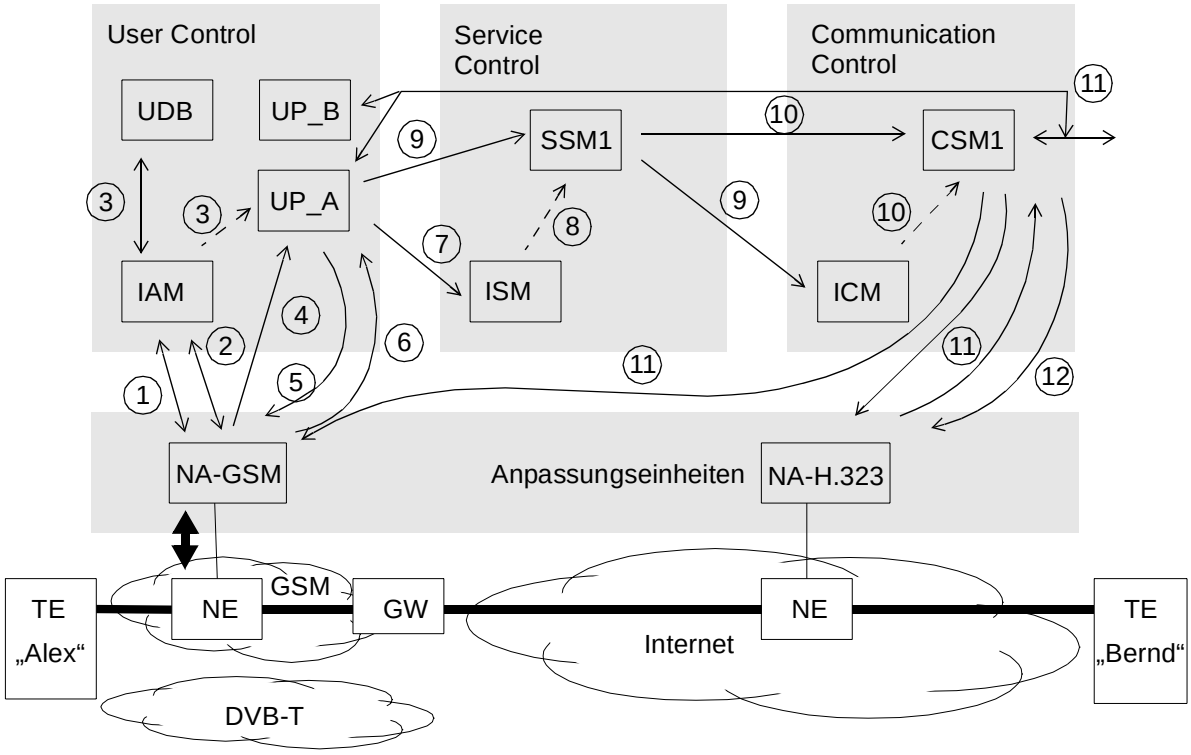


Abbildung 4.16: Beispielszenario MyBusinessCall: Dialog zweier Teilnehmer

Dienstserver bekannt sind, d.h. es handelt sich um registrierte Teilnehmer mit einer eindeutigen, internen Bezeichnung.

Den Ausgangspunkt bildet die Teilnehmerin Alex (A), die in einem Fahrzeug unterwegs ist und über GSM kommunizieren kann. Sie möchte den Teilnehmer Bernd (B) sprechen. Teilnehmer B ist an seinem Arbeitsplatz über Internet erreichbar. Es ergibt sich der folgende Ablauf, der durch Abbildung 4.16 illustriert wird.

Registrierung

1. Teilnehmer A wählt die Telefonnummer des Dienstservers über sein GSM-Mobiltelefon. Der GSM-Netz-Adaptor (NA-GSM) erkennt die ankommende Sprachverbindung und fragt das *Login* über ein Sprachmenü mit Spracherkennung gesteuert über den IAM ab.
2. Die *User Registration Session* ist damit gestartet und der Teilnehmer gibt sein Paßwort (z.B. eine PIN-Nummer) über Tonwahl ein.
3. Der IAM prüft die Daten und startet den zugehörigen User Proxy (UP_A). Er teilt dem Netzadaptor mit, wohin dieser die Registrierungsmeldung schicken muß.
4. Der UP_A empfängt die Registrierungs-Meldung. Automatisch werden außerdem die darin enthaltenen aktuellen Zugangsdaten des Teilnehmers im User Proxy und in der User Data Base gespeichert. Abbildung 4.17 zeigt einen Ausschnitt aus der aktuellen *User Session Description*.

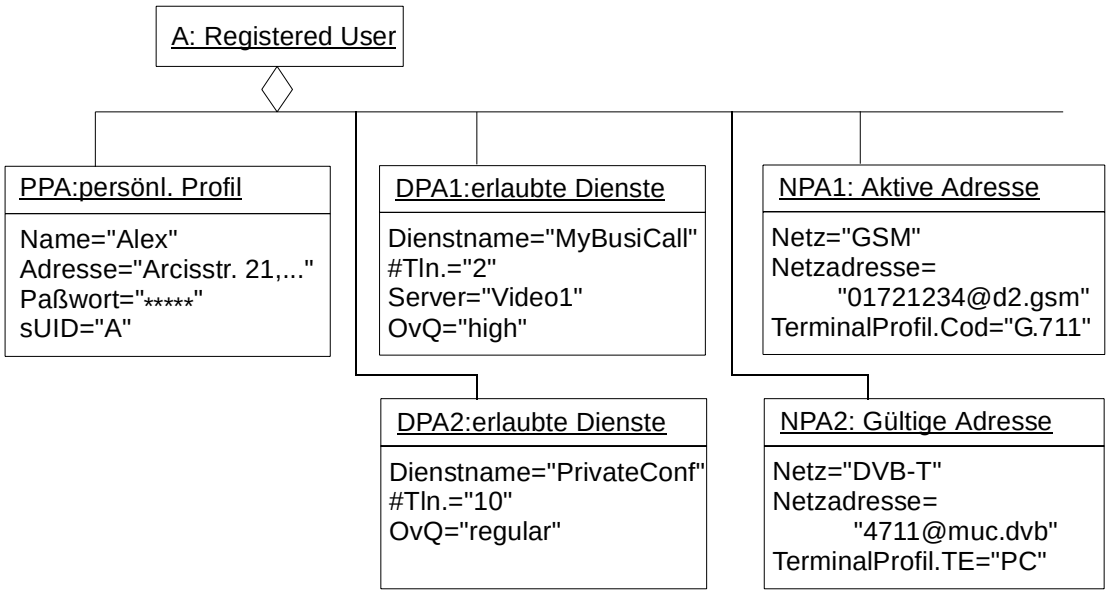


Abbildung 4.17: User Session Description für Teilnehmer A

5. Als Antwort auf die Registrierung bietet der UP_A dem Teilnehmer im Rahmen der *Access Session* verschiedene Dienste an, die dieser über seinen derzeitigen Anschluß nutzen kann. Der Teilnehmer kann auch in der *Registration Session* verbleiben und nur seine Einstellungen modifizieren.

Abbildung 4.18 stellt den Registriervorgang in einem vereinfachten Signalablaufdiagramm dar.

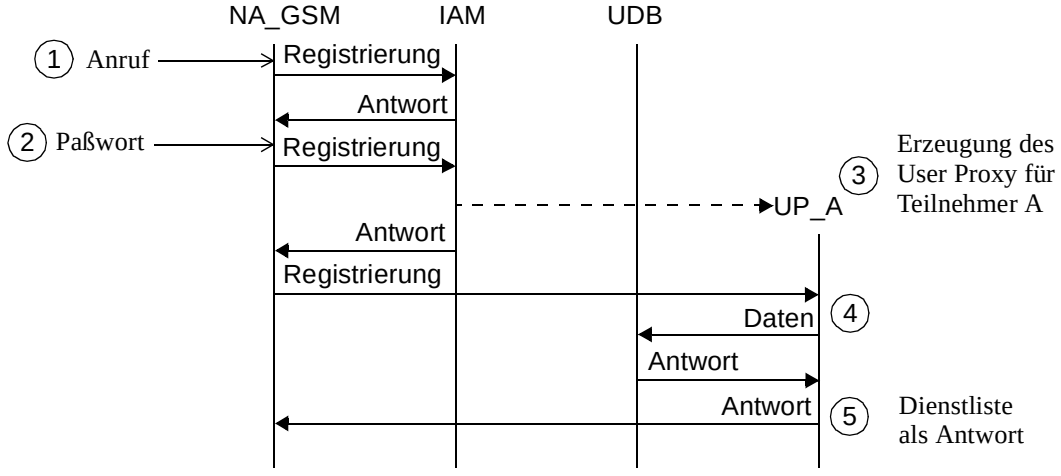


Abbildung 4.18: Beispielablauf für die Registrierung

Dienstauswahl

6. Im Rahmen der *Access Session* wählt Teilnehmerin A „MyBusinessCall with Bernd“ und fordert damit ihren Dienst an.
7. Der UP_A fügt der Anforderung das Dienst-Profil von A für diesen Dienst an und reicht die Meldung an den ISM weiter. Der UP_A setzt u.a. die persönliche Kurzwahlnummer „Bernd“ in die interne Adresse von B, dessen sUID, um.

8. Der ISM instanziert den SSM für den Dienst „MyBusinessCall“ (SSM1) und teilt dem UP_A mit, wohin dieser die Dienstanforderung von A und weitere Meldungen zu diesem Dienst senden muß.

Dienstausführung und Kommunikationssteuerung

9. Nach Erhalt der Dienstanforderung verarbeitet der SSM1 die Dienstlogik und baut den *Media Connection Graph* auf. Dieser wird in einer Kommunikationsanforderung an die *Communication Control* gerichtet. Abbildung 4.19 zeigt die nun aufgestellte *Service Session Description*.

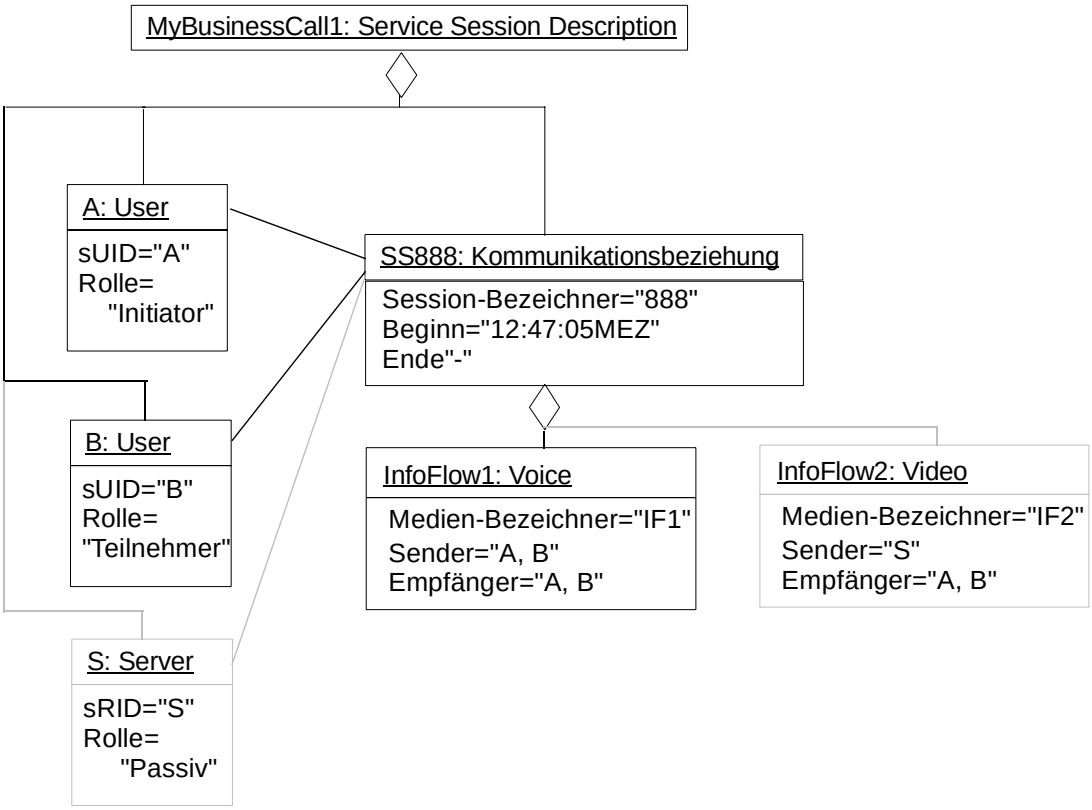


Abbildung 4.19: Service Session Description für das Dienstbeispiel (9.)

10. Über den ICM wird ein CSM instanziert (CSM1) und der SSM1 leitet die Kommunikationsanforderung an diesen um.

11. Der CSM1 holt sich Informationen über den Aufenthaltsort von A und von B und über verfügbare Netze. Die eindeutige Adresse wird dabei in der *User Data Base* aufgelöst und der aktuelle Aufenthaltsort (Netzadresse) zurückgemeldet. Für B wird ein User Proxy gestartet (UP_B), in dem die aktuelle Konfiguration und die Teilnahme am Dienst gespeichert wird. Für die realzeitige Dialogkommunikation wählt er für A GSM und für B H.323/Internet aus. Ein H.323/GSM-Gateway steht zur Verfügung. Die Steuerung erfolgt für GSM über ein Parlay-Interface (NA-Parlay-GSM) und für H.323 über eine proprietäre Schnittstelle zu einem H.323-Gatekeeper (NA-H.323). Nach erfolgter Auswahl stellt der CSM1 die komplette *Communication Session Description* zusammen (Abbildung 4.20) und verteilt die entsprechende Meldung an die ausgewählten Adaptoren. Dabei wird der *Media Connection Graph* durch die entsprechenden *Connectivity Connection Graphen* ersetzt.

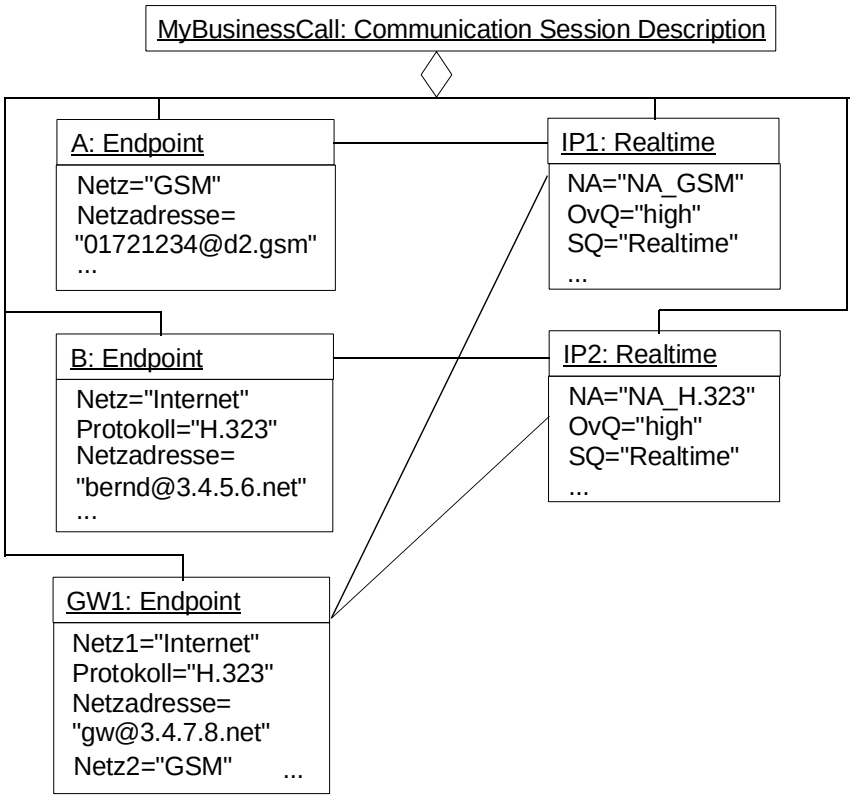


Abbildung 4.20: Communication Session Description für MyBusinessCall

12. Nachdem alle Adaptoren eine erfolgreiche Verbindungsreservierung bestätigen, werden die Informationsflüsse endgültig mit einer Bestätigung aktiviert. Die Teilnehmer werden verbunden.

Abbildung 4.21 zeigt einen Ausschnitt aus dem Signalablauf für den Dienstzugang und die Dienstausführung, um die prinzipiellen Abläufe im Beispiel zu veranschaulichen.

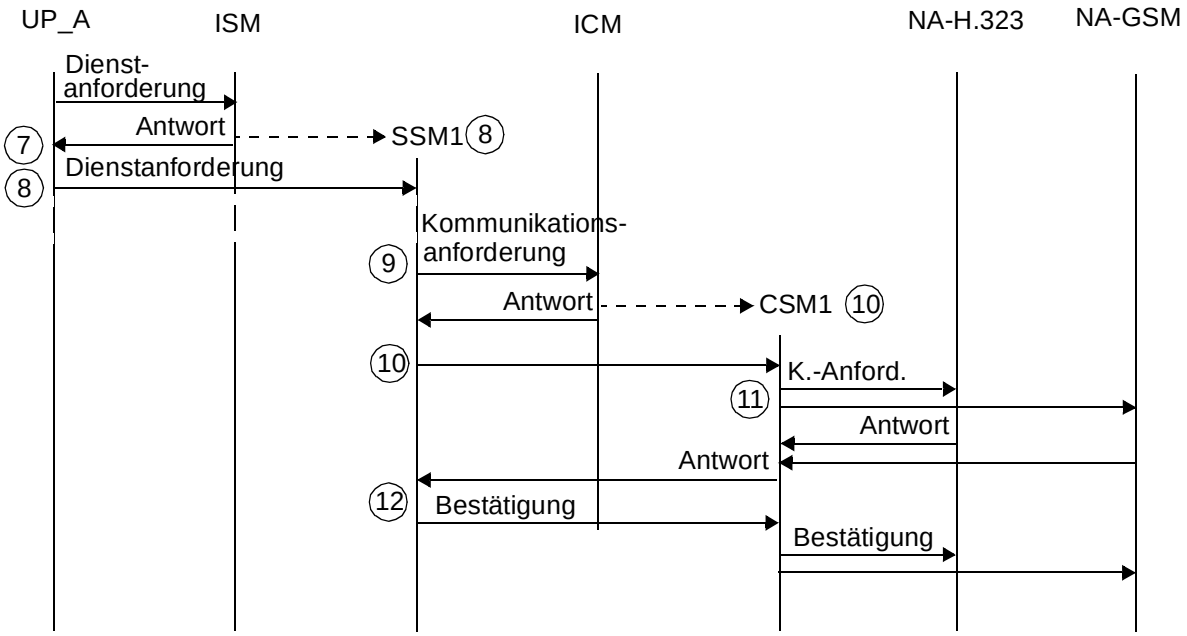


Abbildung 4.21: Beispielablauf für den Dienstzugang und die Dienstausführung

Dienstausführung: Hinzunahme einer Videoquelle

Nun möchten die Teilnehmer im Rahmen ihrer Konversation wie üblich ein benötigtes Dokument, eine Videopräsentation, zu Rate ziehen, um mehr Informationen für eine zu treffende Entscheidung zu erhalten. Da für Teilnehmer A die GSM-Verbindung zu langsam für einen interaktiven Videoabruf ist, greift er auf seinen mobilen Computer zurück, der mit seinem GSM-Anschluß im Fahrzeug verbunden ist und zudem über ein DVB-T Empfänger-Modul verfügt.

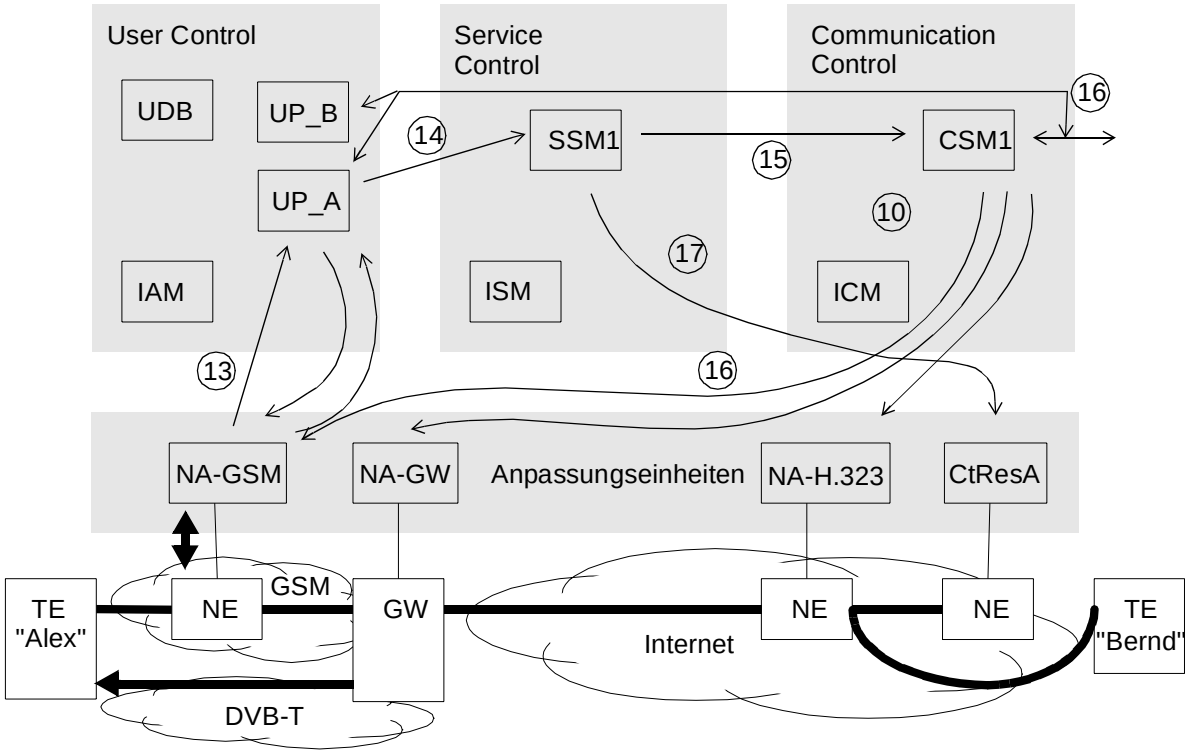


Abbildung 4.22: Beispielszenario: Hinzunahme einer Videoquelle

13. Teilnehmer A fordert im Rahmen der *Service Session* die zusätzliche Videoverbindung als eine mögliche Änderung der bestehenden Kommunikationsbeziehung an. In der Dienstlogik ist dieser Fall im Rahmen der Dienstausführung vorgesehen. Der Videoserver liegt im Einflußbereich der Dienststeuerung.
14. Der UP_A reicht die Dienstanforderung an die entsprechende SSM-Instanz (SSM1) des Dienstes „MyBusinessCall“ weiter.
15. Der SSM1 prüft die Änderungen, modifiziert die *Service Session Description* und sendet eine Kommunikationsanforderung an den CSM1, um die Änderungen in der Realisierung der Informationsflüsse in den Netzen anzufordern. Die Änderungen in der *Service Session Description* sind in Abbildung 4.19 hellgrau dargestellt.
16. Analog zu 11. sammelt der CSM1 die benötigten Informationen und verteilt die entsprechenden Kommunikationsanforderungen an die Adaptoren, die jetzt die Verbindungen in den ausgewählten Netzen (DVB-T, GSM und Internet, in dem der Video-Server steht) einrichten. Für Teilnehmer B wird weiterhin eine Internet-Verbindung gewählt, da der Videoserver selbst über IP angeschlossen ist.

17. Der Videoserver ist dem Dienstserver als Content-Ressource bekannt, so daß der SSM1 den Zugriff auf die Präsentation für A und B freischalten kann. Die Steuerung des Videoabrufs erfolgt direkt über das Protokoll des Videoservers über die vom CSM1 eingerichteten Kanäle.

4.9 Zusammenfassung

In diesem Kapitel wurden die Eigenschaften und die generelle Funktionsweise der Architektur des Dienstservers SAMSON beschrieben. Das Grundkonzept definiert für SAMSON eine zentralisierte Dienststeuerung, die über Adaptoren erweiterte Dienste in unterschiedlichen Netzen steuert. Für die Beschreibung der Software-Architektur wurden basierend auf verschiedenen Modellen für verteilte Systeme ein Systemmodell bestehend aus sechs Sichtweisen definiert, die es ermöglichen, eine Dienstarchitektur umfassend zu beschreiben. Das Geschäftsmodell, die Session-Sicht und die Komponenten-Sicht fokussieren die äußeren und inneren Schnittstellen sowie die Struktur. SAMSON unterscheidet die drei Steuerungsbereiche *User Control, Service Control* und *Communication Control*. In der Betrachtung aus der Informations-Sicht wird festgelegt, wie die Daten in SAMSON strukturiert sind und zueinander in Beziehung stehen. Die Kommunikations-Sicht beschreibt, wie die Daten zwischen den Strukturkomponenten ausgetauscht werden.

Kapitel 5
Spezifikation der Dienstebene

In Kapitel 4 wurden die Eigenschaften des Dienstservers SAMSON abstrakt modelliert. Inhalt dieses Kapitels ist die detaillierte Spezifikation der Architekturkomponenten und des Signalisierungsprotokolles. Die Spezifikation erfolgt auf der Schicht der Dienststeuerung, ohne näher auf untere Schichten einzugehen. Abbildung 5.1 gibt einen Überblick über die Komponenten der Dienstebene. Sie ist der Sitz der eigentlichen Dienststeuerung und wird gemäß dem Grundkonzept der Architektur durch die Anpassungsschicht von der Netzinfrastruktur getrennt.

Zunächst wird in Abschnitt 5.1 das auf der Beschreibung aus der Kommunikations-Sichtweise basierte, neue Signalisierungsprotokoll SesCP (*Session Control Protocol*) spezifiziert. Damit werden die Schnittstellen der Komponenten festgelegt. Anschließend erfolgt die Funktions- und die Verhaltensbeschreibung der Komponenten. Die Spezifikation der Dienstebene unterteilt sich gemäß der drei Steuerungsbereiche in die Teilnehmerverwaltung (UC), die Dienststeuerung (SC) und die Kommunikationssteuerung (CC).

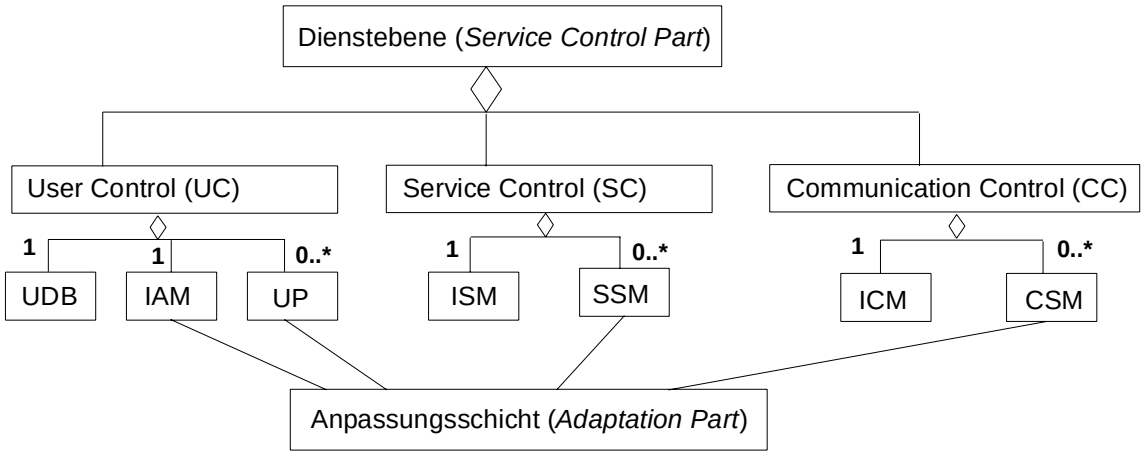


Abbildung 5.1: UML-Diagramm der Dienstebene

5.1 Ein neues Signalisierungsprotokoll für die Dienststeuerung

Zur Signalisierung von Teilnehmeranforderungen, Aktionen der Dienststeuerung und von externen sowie internen Ereignissen wird ein Verfahren benötigt, mit dem Steuerinformationen schnell, sicher und mit geringem Aufwand zwischen den verteilten Architekturkomponenten

ausgetauscht werden können. Die bereits in Abschnitt 4.7.1 aufgestellten Anforderungen an die Signalisierung lassen sich wie folgt zusammenfassen:

- Unterstützung von Teilnehmerverwaltung und Dienstzugang
- Aufbau und Steuerung komplexer multimedialer Dienste
- Unterstützung der Kommunikationssteuerung
- Berücksichtigung der Trennung von Funktionen in die drei Steuerungsbereiche Teilnehmer-, Dienst- und Kommunikationssteuerung

Folgende Merkmale soll das Signalisierungsprotokoll dabei insbesondere aufweisen:

- Unterstützung der objektorientierten Informationsmodelle
- Unterstützung eines geeigneten Aushandlungsmechanismus
- Unterstützung der Informationsabfrage zwischen den Komponenten
- Differenzierte Mechanismen zur Beendigung der Teilnahme eines Teilnehmers und zur Beendigung des gesamten Dienstes

In diesem Abschnitt wird das *Session Control Protocol* als ein neues Signalisierungsprotokoll für Dienste spezifiziert, das diese Anforderungen erfüllt. Als Basis für die Signalisierung wurde das *Session Initiation Protocol* (SIP) der IETF ausgewählt [HSS99]. Zunächst wird gezeigt, welche der gestellten Anforderungen dadurch bereits erfüllt werden und welche Änderungen notwendig sind. Anschließend werden die Protokollmeldungen und die Datenformate von SesCP spezifiziert.

Die Entscheidung für ein IP-basiertes Signalisierungsprotokoll liegt darin begründet, daß die zukünftige Signalisierungsinfrastruktur als native IP-basiert erwartet wird. Dies zeigt die Standardisierung neuer Kommunikationssysteme, wie z.B. UMTS. Hier wird bereits SIP als Signalisierungsprotokoll für Multimedia-Dienste eingesetzt [HT01]. Auf diese Weise kann man erwarten, daß sich SesCP nahtlos in die Infrastruktur einfügt.

Alternativ zu einem Protokoll bietet es sich an, die Signalisierung zwischen den SAMSON-Komponenten auf die Basis einer Middleware, wie z.B. einer CORBA-Plattform, zu stellen. Allerdings werden einige Performanceanforderungen der Dienstsignalisierung von den Middleware-Plattformen noch nicht endgültig erfüllt [MC00], wie sich z.B. bei der Umsetzung der TINA-Architektur auf Basis der CORBA-Technologie gezeigt hat. Die Spezifikation der Protokollmeldungen entspricht der Definition der Schnittstellen für die Methodenaufrufe in einer Middleware. Daher ist eine Migration beispielsweise auf CORBA einfach möglich.

5.1.1 Ausgangspunkt: Das IETF Session Initiation Protocol

Das *Session Initiation Protocol* [HSS99] ist ein an HTTP in der Version 1.1 [FGM99] angelehntes Transaktionsprotokoll für die Signalisierung von multimedialen Diensten im Internet. Signalisierungsnachrichten werden basierend auf einem einfachen Meldungssatz ausgetauscht. Eine Signalisierungsbeziehung besteht grundsätzlich Ende-zu-Ende zwischen den beteiligten Endgeräte-Anwendungen (*User Agents*) der Teilnehmer. Im Netz können die Meldungen verschiedene SIP-Server durchlaufen (siehe auch Abschnitt 3.4.2).

Jede Signalisierungstransaktion besteht bei SIP aus einer *Request*-Meldung (Anfrage) eines SIP-Clients und mindestens einer oder mehreren *Response*-Meldungen (Antwort) eines SIP-Servers. Eine Anfrage besteht aus dem SIP-Meldungsnamen, einer Empfängeradresse, Header-Feldern für Parameter und je nach Art der SIP-Meldung aus einem Message-Body, der die Session-Beschreibung enthält. Tabelle 5.1 zeigt die sechs in [HSS99] definierten Meldungen.

SIP-Meldung	Erläuterung
INVITE	zeigt dem Empfänger an, daß er eingeladen ist, an der Session teilzunehmen, deren Beschreibung im Message-Body enthalten ist.
ACK	ist im Rahmen eines dreiphasigen Meldungsablaufes, der durch INVITE ausgelöst wird, die Bestätigung des Clients (Session-Initiator).
OPTIONS	Ein Server kann mit OPTIONS nach seinen Fähigkeiten befragt werden.
BYE	Mit BYE zeigt ein Teilnehmer an, daß er eine Session verlassen möchte und seine Verbindungen darin beenden will.
CANCEL	hebt eine schwebende, noch nicht endgültig bestätigte Anfrage auf.
REGISTER	Ein Client registriert mit dieser Meldung seine Adresse bei einem Server.

Tabelle 5.1: SIP-Meldungen

Antworten können den Eingang einer Anfrage bestätigen, sie akzeptieren, ablehnen oder weitere Aktionen vom Client fordern, damit die Anfrage ausgeführt werden kann. Die Antwort-Meldungen unterscheiden sich analog zu HTTP durch einen dreistelligen Status-Code, der die Begründung für die Antwort liefert. Alle Status-Codes mit einem Wert größer oder gleich 200 kennzeichnen eine endgültige Antwort, die eine Transaktion beendet, z.B. 200 „O.K.“ oder 305 „Redirect“. 1xx-Status-Codes kennzeichnen vorläufige Antworten, z.B. *Call Proceeding* (siehe [HSS99]).

Die SIP-Meldungen erfüllen bereits einige der aufgestellten Anforderungen:

Die Meldung SIP-REGISTER eignet sich, um die geforderte Unterstützung der **Teilnehmerverwaltung** in SAMSON zu realisieren. Allerdings fehlen zusätzliche Möglichkeiten, dem Teilnehmer im Rahmen eines **Dienstzugangs** seine Dienste anzubieten. Hier sind leichte Modifikationen notwendig.

INVITE unterstützt hervorragend den **Aufbau und die Steuerung von multimedialen Diensten**. Für SAMSON kann INVITE so eingesetzt werden, daß der Empfänger eine Dienststeuerungskomponente (eine *Service Session Manager*-Instanz) ist, die eingeladen wird, die gewünschte Session aufzubauen. Diese Dienststeuerungskomponente sorgt dann für die Einrichtung der zur Session gehörenden Kommunikationsbeziehung in den Netzen. Die zugehörige Signalisierung zur **Kommunikationssteuerung** ist nicht unmittelbar an das eintreffende INVITE gekoppelt, sondern kann von anderen Ereignissen (z.B. von Timern) ausgelöst werden. Die Meldungen der Kommunikationssteuerung haben somit eine andere Bedeutung als das INVITE. Das bedeutet, daß die Ende-zu-Ende Signalisierung des INVITE nicht bis zu den Adaptoren durchgeführt wird, sondern beim SSM endet. Für die Kommunikationssteuerung ist eine neue Meldung notwendig (vgl. Abbildung 4.14).

Für die Beendigung der eigenen Teilnahme an einer Session kann die SIP-Meldung BYE für SesCP verwendet werden. Eine Meldung zur Beendigung der gesamten Session durch einen Teilnehmer fehlt.

Ergänzend zur SIP-Standardisierung in RFC 2543 definiert der RFC 2976 [Don00] die SIP-Meldung INFO für die Mitteilung von Signalisierungsinformationen während einer laufenden Session. Diese Meldung erfüllt die gestellten Anforderungen nur teilweise. Sie ist für SesCP so zu erweitern, daß auch eine explizite **Informationsabfrage** zwischen den Komponenten unterstützt wird.

Neben den Endgeräte-Anwendungen (*User Agents*) definiert SIP Server im Netz als weitere logische Einheiten, die von SIP-Meldungen durchlaufen werden können. Der SIP-Proxy-Server kann Anfragen entweder selbst beantworten oder an andere Server weiterleiten. Er interpretiert dabei eine Anfrage und kann sie modifizieren (z.B. Adreßänderung, weitere Parameter). Der SIP-Redirect-Server kann keine Anfragen aussenden. Er modifiziert die Adressfelder in einem erhaltenen Request und teilt damit dem Anfragenden durch seine Antwort die neue Adresse eines gewünschten Teilnehmers mit. Der SIP-Registration-Server akzeptiert REGISTER-Anfragen. Seine Funktion ist normalerweise in einem SIP-Proxy oder einem SIP-Redirect-Server angesiedelt, um dem Teilnehmer die Möglichkeit zu geben, seine Adressen zu registrieren.

Diese Server-Funktionen eignen sich, um die drei **Steuerungsbereiche** der SAMSON-Architektur im Einklang mit dem SAMSON zugrunde gelegten Signalisierungsprinzip (siehe Abschnitt 4.7.2) abzubilden. Die Komponenten *User Proxy* und *Communication Session Manager* können als Proxy-Server realisiert werden, da sie von den Signalisierungsnachrichten der *Service Session* bzw. der *Communication Session* durchlaufen werden. Die Aufgaben der initialen Komponenten der Steuerungsbereiche (hier: IAM und CSM) werden durch die Funktionalität eines Redirect-Servers abgedeckt. Sie leiten die Meldungen an die zuständigen Server um. Ein *User Proxy* beinhaltet gleichzeitig die Funktion eines Registration-Servers, um die Teilnehmerregistrierung abzuwickeln (siehe Abbildung 5.2).

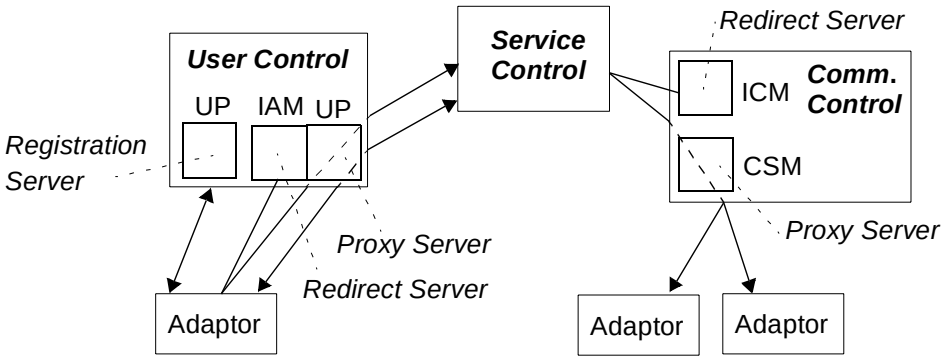


Abbildung 5.2: Realisierung des Signalisierungsprinzips mit SIP

Eine Besonderheit unter den SIP-Meldungen ist der Ablauf der INVITE-Meldung. Zwar endet die zugeordnete Transaktion auch hier mit einer Antwort, die einen endgültigen Status-Code (z.B. 200) trägt. Aber um eine Aktion im gerufenen Server auszulösen (z.B. Einrichten einer Session), muß der Client den Erhalt der Antwort mit der Meldung ACK bestätigen. Dieser sogenannte *Three-Way-Handshake* (Abbildung 5.3) dient zur Aushandlung der Session-Beschreibung. Mit dem abschließenden ACK kann der Client eine endgültige Session-Beschreibung verschicken, wenn sein INVITE mehrere Alternativen enthielt[1]. ACK wird bei SIP nur für INVITE-Anfragen verwendet.

Dieser dreiphasige Meldungsablauf bildet die Grundlage für einen geeigneten **Aushandlungsmechanismus** im *Session Control Protocol*. Eine Dienstbeschreibung kann einem Server angeboten werden und wird erst in Abhängigkeit von dessen Antwort endgültig festgelegt. Die Meldung CANCEL bietet zudem die Möglichkeit, eine noch nicht endgültig bestätigte

1. Außerdem ist die Bestätigung mit ACK für eine Telefonverbindung notwendig, da zwischen dem INVITE (Auslösen des Klingeln beim Gerufenen) und der darauffolgenden Antwort (der Gerufene hebt ab) eine unbestimmte Zeitspanne vergehen kann. ACK stellt sicher, daß der Rufende noch an der ursprünglichen Session interessiert ist, d.h. die anfangs angegebenen Ressourcen (z.B. Portnummern) bei diesem noch frei sind. Für kommerzielle Systeme legt ACK den Beginn der Vergebührung fest.

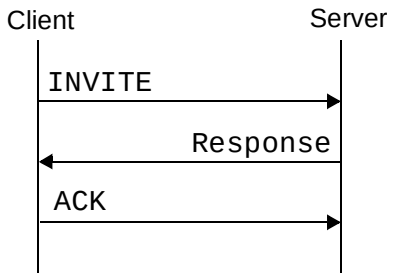


Abbildung 5.3: Three-Way-Handshake bei SIP

Anfrage rückgängig zu machen. Dies wird als besonders wichtig erachtet, wenn zusammengehörige Beschreibungen (z.B. mehrere *Connectivity Connection Graphs*) an verschiedene Adaptoren verteilt werden. Eine Verteilung einer Anfrage ist bei SIP durch den Forking-Mechanismus von Proxy-Servern vorgesehen. Ein Proxy-Server kann eine ankommende Anfrage an mehrere Server weitergeben, parallel die Antworten verarbeiten und ausgewählte Antworten an den Client zurückgeben.

Die Beschreibung einer Multimedia-Session wird in SIP unabhängig von der Art der SIP-Meldungen und deren Header im Message-Body als Nutzlast transportiert. Für SIP wird das *Session Description Protocol* (SDP, [HJ98]) empfohlen. Grundsätzlich sind beliebige Beschreibungen transportierbar. Ihr Typ, die Länge und die Kodierung werden über entsprechende Header-Felder angegeben. Eine SDP-Session-Beschreibung wird in ASCII-Text beschrieben und besteht aus einer Sequenz aufeinanderfolgender Parameter. Diese Trennung von Signalisierungsverfahren und transportierter Dienstbeschreibung entspricht den Anforderungen der SAMSON-Kommunikations-Sichtweise. Allerdings ist SDP zu stark an IP-Netzen orientiert. Daher sind die Elemente des *Session Description Protocols* nicht ausreichend, alle Informationsbeschreibungen, die für SAMSON definiert wurden, auszudrücken.

5.1.2 Überblick über das Session Control Protocol

Wie bereits gezeigt wurde, erfüllt SIP bereits viele der gestellten Anforderungen an das Signalisierungsprotokoll auf Dienstebene. Gemäß der obigen Diskussion von SIP lassen sich folgende Änderungen und Neudefinitionen zusammenfassen, die für das Signalisierungsprotokoll SesCP für SAMSON gegenüber SIP durchgeführt wurden:

- Neudefinition einer Meldung, die die Teilnehmerregistrierung in Anlehnung an SIP-REGISTER und zusätzlich den Aufbau einer *Access Session* für den Dienstzugang unterstützt (SesCP-`ACCESS`);
- Aufspaltung der Ende-zu-Ende Signalisierung von SIP-INVITE. Teilnehmer können Dienststeuerungskomponenten einladen, eine Session für sie auszuführen (SesCP-`INVITE`). Unabhängig davon kann die Dienststeuerung Kommunikationsbeziehungen mit einer neu definierten Meldung (SesCP-`SETUP`) von den Adaptoren anfordern.
- Neudefinition einer Meldung zur Beendigung einer kompletten Session (SesCP-`END`);
- Erweiterung der Meldung SIP-INFO für die Informationsanforderung (SesCP-`INFO`);
- Definition eines neuen Formats für die Dienstbeschreibung anstelle von SDP.

Im SIP-Standard sind Erweiterungen des SIP-Meldungssatzes durch die Definition neuer Meldungen oder zusätzlicher Header vorgesehen. Die SIP-Protokollautomaten ignorieren unbekannte Meldungen oder reagieren durch geeignete Antworten. Für eine Realisierung von SesCP in einer SIP-Umgebung müssen nur in den entsprechenden Clients und Servern, die zu

SAMSON gehören, die geforderten Änderungen vorgenommen werden. Die SIP-Grundprinzipien werden nicht verletzt. Somit kann SesCP einfach in einer SIP-Infrastruktur implementiert werden.

Insgesamt ergibt sich folgender Meldungssatz für SesCP (Tabelle 5.2). Details zu den Meldungen und ihren Parametern werden in Abschnitt 5.1.3 beschrieben.

SesCP-Methode	Erläuterung
ACCESS	wird für den Teilnehmerzugang verwendet, bei dem ein Teilnehmer seine aktuellen Zugangsparameter und die Diensteinstellungen dem Dienstserver mitteilt.
INVITE	dient zum Start eines Dienstes, indem der Teilnehmer eine entsprechende Dienstinstanz 'einlädt'. Nachfolgende INVITEs dienen dazu, die Dienstbeschreibung zu ändern (neuer Teilnehmer, Informationsfluß). Die Dienstinstanz benutzt INVITE, um ihrerseits Ressourcen einzuladen.
SETUP	dient dazu, die Informationsflüsse für die Dienstausführung über die Adaptoren in der Infrastruktur zu etablieren.
ACK	ist die Bestätigung einer positiven Antwort auf eine Anfrage (vgl. SIP). ACK wird bei INVITE und SETUP verwendet.
BYE	Mit BYE zeigt ein Teilnehmer an, daß er eine Session verlassen will. Eine Dienstinstanz kann mit BYE eine Kommunikationsbeziehung abbauen.
END	Mit END zeigt ein Teilnehmer an, daß er eine gesamte Session beenden will. Dies ist nur möglich, wenn der Teilnehmer die entsprechenden Rechte dazu besitzt.
CANCEL	hebt eine schwebende Anfrage auf (vgl. SIP). CANCEL wird insbesondere für die Aushandlung von Verbindungen in der Communication Session eingesetzt.
INFO	dient zur Informationsmitteilung während einer laufenden Dienstausführung zwischen den beteiligten Instanzen. INFO kann dabei eine Anfrage nach Informationen oder eine Statusmitteilung beinhalten.

Tabelle 5.2: SesCP-Meldungen

Wie bereits erläutert, läßt sich die Struktur des Dienstservers SAMSON (Abbildung 4.13 auf Seite 77) auf die typischen Elemente einer SIP-Struktur abbilden. Abbildung 5.4 zeigt eine zusammenfassende Darstellung der einzelnen Funktionen. Adaptoren (NA) arbeiten als SIP-*User Agent*-Client und initiieren Anfragen an die entsprechende Dienststeuerungsinstanz, den SSM. Dabei durchläuft die Anfrage mehrere SesCP-Komponenten. Der IAM arbeitet als Redirect-Server. Er ist bei jedem Adaptor als *Default-Proxy* eingetragen und reagiert auf die erste Anfrage (1) zu einer Session mit einem *Redirect* (2). Dabei verweist er auf den *User Proxy* des zugehörigen Teilnehmers als die nächste, korrekte Komponente zur Meldungsverarbeitung. Gleichzeitig erzeugt der IAM die UP-Instanz, falls sie noch nicht existiert. Künftige Meldungen dieser Session gehen dann direkt vom NA an den UP (3). Der UP leitet eine Dienst-Anfrage an den ISM (*Default-Proxy*) weiter, der ebenfalls als Redirect-Server arbeitet. Der ISM erzeugt und instanziert den entsprechenden SSM und teilt dessen Adresse dem UP mit. Die künftige Route für alle Transaktionen zwischen NA und SSM wird dann stets über den UP geführt. Im Gegensatz zur üblichen Verwendung von SIP, aber im Einklang mit dem Standard, wird bei SesCP nicht ein Teilnehmer, sondern eine Dienstinstanz eingeladen.

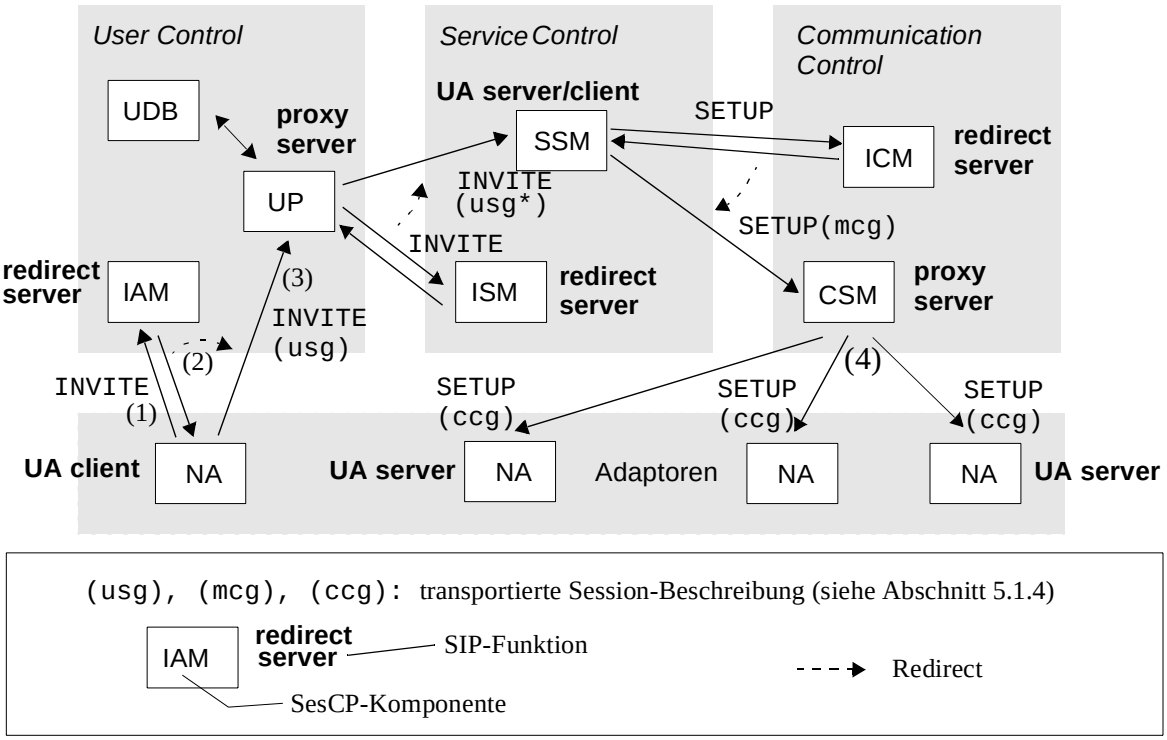


Abbildung 5.4: Zusammenhang von SesCP mit SIP-Clients und SIP-Servern

Ähnlich ist die Funktionsweise bei der Kommunikationssteuerung. Der SSM initiiert als UA-Client ein `SETUP`, das erst zum ICM (Redirect-Server) und dann über den CSM (Proxy-Server) zu den entsprechenden Netzadaptoren geleitet wird. Auch hier ist der CSM in allen künftigen Transaktionen im Pfad enthalten. Um alle notwendigen NAs zu erreichen, kann der CSM die Meldungen nach dem *Forking*-Prinzip gleichzeitig an mehrere NAs schicken (4). Die `INFO`-Meldung (nicht dargestellt) dient zur Informationsabfrage zwischen den Komponenten während einer laufenden Session (z.B. Teilnehmerprofil-Abfrage für die Adreßauflösung oder Auswahl der Adaptoren).

5.1.3 Nachrichtenformat und Parameter

SesCP-Meldungen werden in *Request*- und *Response*-Meldungen unterschieden. Ihr Aufbau entspricht dem Format der SIP-Meldungen. Alle Meldungen sind als ASCII-Text kodiert.

Request-Meldungen bestehen aus einer *Request*-Zeile, die den Meldungsnamen und die Protokollversion enthält. Die zweite Zeile beinhaltet Meldungsparameter, die durch Header-Felder gekennzeichnet sind. Optional folgt der Message-Body, der die Dienstbeschreibung enthält. Folgendes Beispiel zeigt eine Request-Meldung für einen Dienstaufruf. Weitere Beispiele können Anhang D entnommen werden.

```
INVITE sescp:MyBusiCall@ism.sc.samson SESCP/1.0
Via: SESCP/1.0/UDP NA4711.samson:5050
From: sescp: A@NA4711.samson
To: sescp:MyBusiCall@sc.samson
Call-ID: 888
CSeq: 1 INVITE
Content-Type: application/sdp+
Content-Length: ...
Message-Body (nicht dargestellt)
```

Die Header-Felder enthalten Parameter, die für die Protokollverarbeitung in den SesCP-Einheiten erforderlich sind. Die für den Protokollablauf von SesCP wichtigen Header-Felder haben folgende Aufgaben und Formate[1,2]:

Call-ID: Eindeutiger Bezeichner für eine Session. Jede Session (*User Session*, *Service Session* und *Communication Session*) hat einen eindeutigen Bezeichner, der in der ersten Meldung gesetzt wird und ab dann für alle weiteren Meldungen der Session verwendet werden muß. Die *Service Session* und die zugehörige *Communication Session* haben dieselbe Call-ID. Die Meldung `ACCESS` trägt die Call-ID der *User Registration Session*, während die Meldung `INVITE` bereits die Call-ID der *Service Session* beinhaltet. Eine eindeutige Call-ID kann man z.B. aus der Kombination von sUID, Datum und Zeit erhalten.

Format: `Call-ID:<Bezeichner>`

Contact: Dieser Header wird in der *Response*-Meldung 305 (*Redirect*) verwendet, um dem Anfordernden mitzuteilen, an welchen Empfänger die Anfrage zu wiederholen ist.

Format: `Contact:<SesCP-URI>; action=redirect`

Content-Length: Gibt die Länge des Message-Body an (Anzahl der Oktette, dezimal)

Format: `Content-Length:<Anzahl der Oktette>`

Content-Type: Gibt die Art des Message-Body an.

Format: `Content-Type: application/sdp+`

CSeq: Bezeichner für zusammengehörige Meldungen, bestehend aus einer Integerzahl und den Meldungsnamen. Alle Meldungen einer Transaktion haben dieselbe CSeq-Nummer. Darüber hinaus führen auch `ACK` und `CANCEL`-Meldungen die CSeq des zugehörigen `INVITE` bzw. `SETUP`. Die CSeq-Nummer wird vom Client vergeben und ist bei jeder weiteren Anfrage in derselben Session zu erhöhen.

Format: `CSeq:<Nummer> <Meldungsname>`

From: Bezeichnet den Initiator einer Anfrage. Das From-Feld wird von einer Anfrage in die Antwort kopiert.

Format: `From:<SesCP-URI>`

Record-Route: Wird von einem SesCP-Proxy eingefügt, um zu kennzeichnen, daß alle künftigen Anfragen über diesen Proxy gehen, z.B. UP und CSM.

Format: `Record-Route:<SesCP-URI>`

Request-URI: Gibt einen konkreten Empfänger einer Anfrage an. Im Gegensatz zum Header-Feld "To:" kann die hier angegebene Adresse von einem Proxy-Server geändert werden. Dies wird benötigt, wenn der CSM die endgültigen Adaptoren ausgewählt hat und deren Adressen in die Meldung `SETUP` einfügt.

Format: `Request-URI:<SesCP-URI>`

---

1. Darstellung: Bezeichner in spitzen Klammern „< >“ sind Platzhalter für einen Parameterwert.
2. Das Feld <SesCP-URI> bezeichnet eine Adresse eines Servers auf der Ebene des *Session Control Protocol*s. Eine Beschreibung erfolgt in Abschnitt 5.1.5

Route: Bestimmt den Pfad einer Anfrage. Jede durchlaufene Komponente entfernt den jeweils ersten Eintrag und leitet die Meldung an den nächsten Eintrag weiter.

Format: `Route:<SesCP-URI>`

To: Gibt den logischen Empfänger einer Anfrage an. Über den Tag-Parameter können verschiedene Empfänger unterschieden werden, wenn eine Anfrage beim Forking an mehrere Empfänger geschickt wird.

Format: `To:<SesCP-URI>; tag=<number>`

Die Beschreibung weiterer möglicher Header kann [HSS99] entnommen werden.

Tabelle 5.3 gibt an, in welcher *Request*-Meldung bzw. zugehörigen *Response*-Meldung die Header verwendet werden. Gegenüber SIP sind bei der Verwendung der Header in SesCP einige Änderungen notwendig. Insbesondere sind Header, die den Signalisierungsweg festlegen, verpflichtend zu verwenden. Auf diese Weise kann der Weg der Meldungen durch die Proxy-Server einer SIP-Infrastruktur explizit angegeben werden.

Header-Feld	wo[a]	SesCP proxy[b]	ACC[c]	INV[c]	ACK[c]	SET[c]	BYE/ END[c]	INF[c]	CAN[c]
Call-ID	Rr-c	r	m	m	m	m	m	m	m
Contact	r	r	m	m	-	m	-	-	-
Content-Length	Rr-*	w	m*	m	m*	m	m*	m*	-
Content-Type	Rr-*	w	*	m	*	m	m*	*	-
CSeq	Rr-c	r	m	m	m	m	m	m	m
From	Rr-c	r	m	m	m	m	m	m	m
Request-URI	R	w	o	o	o	m	o	o	o
Record-Route	Rr	r	-	m	m	m	m	o	m
Route	R	r	-	m	m	m	m	o	m
To	Rr-c	r	m	m	m	m	m	m	m

a. Gibt an, in welchen Request- oder Response-Meldungen der Header verwendet wird. „R“: im Request, „r“: im Response; Zusätze (durch „-“ getrennt): „*“ bedeutet: falls Message-Body vorhanden, „c“ bedeutet: Feld wird von Request in Response kopiert.

b. Gibt an, ob ein SesCP-Proxy das Feld nur auswertet („r“) oder auch modifizieren darf („w“).

c. Gibt an, ob das Feld enthalten sein muß (mandatory, „m“), sein kann (optional, „o“) oder nicht enthalten ist („-“). Ein „*“ bedeutet: falls Message-Body vorhanden.

Schattierte Definitionen weichen von der SIP-Baseline Definition ab.

Tabelle 5.3: Ausgewählte Header-Felder in SesCP

Eine *Response*-Meldung besteht aus einer Status-Zeile, die die Protokollversion und den Status-Code enthält, beliebig vielen Header-Feldern, die größtenteils aus den *Request*-Headern übernommen werden, und optional aus einem Message-Body. In SesCP werden alle in SIP

definierten Status-Codes unterstützt [HSS99]. Folgende Status-Codes sind für die korrekte Protokollverarbeitung in SesCP von großer Bedeutung:

200 „OK“: Die Anfrage war erfolgreich und die entsprechenden Aktionen wurden durchgeführt bzw. werden nach einer Bestätigung durch ACK durchgeführt. Je nach Art der Anfrage enthält diese Antwort einen Message-Body, der z.B. bei `INFO` die gewünschte Information enthält.

305 „Use Proxy“: Die Anfrage war nicht erfolgreich und muß an eine neue Adresse gerichtet werden. Im *Contact-Header* ist eine neue Adresse eines Proxy-Servers enthalten, an den die Anfrage erneut zu senden ist (*Redirect*).

5.1.4 Dienstbeschreibung

Ein Dienst wird in SAMSON in unterschiedlichen Ebenen beschrieben. Man unterscheidet die Information, die in den Sessions gehalten wird, und die Daten, die zwischen den Steuerungsbereichen ausgetauscht werden. Letztere sind durch das Signalisierungsprotokoll SesCP zu übertragen. In Abschnitt 4.5 wurden dazu die drei Dienst-Beschreibungen *User Service Graph* (USG), *Media Connection Graph* (MCG) und *Connectivity Connection Graph* (CCG) definiert. Zusätzlich werden mit einer `ACCESS`-Meldung Daten des Teilnehmer-Profils (*User-Profile*, UPR) an den UP übertragen.

Alle Graphen werden im Message-Body von SesCP übertragen und sind in einem einheitlichen ASCII-basierten Format kodiert. In Anlehnung an das *Session Description Protocol* bei SIP wird das neue Format **SDP+** bezeichnet. Es besteht aus unterschiedlichen, geordneten Textelementen, die die Objekte der Dienstbeschreibung definieren. Die Beschreibung ist je nach *Graph* unterschiedlich detailliert, d.h. je konkreter die Beschreibung beim Durchlauf der SesCP- Steuerungsinstanzen wird, desto mehr Elemente enthält die SDP+-Beschreibung. Da sich die Beschreibungen nur im Detaillierungsgrad unterscheiden, ist eine eindeutige Abbildung der Information aufeinander gewährleistet. Details dazu werden bei der Spezifikation der Komponenten beschrieben.

Wie Abbildung 5.4 zu entnehmen ist, kann die Dienstbeschreibung sowohl durch SesCP-Clients und Server als auch durch die als SesCP-Proxies agierenden Steuerungskomponenten UP und CSM modifiziert werden.

Alle SDP+-Beschreibungen[1] enthalten analog zu SDP zu Beginn der Beschreibung folgende Felder:

```
ver=<Protokollversion von SDP+>
ori=<Name> <Session ID> <Version>
```

Name bezeichnet den Urheber der Beschreibung (Teilnehmer oder Dienstinstanz) durch die sUID oder den Dienstnamen. Die *Session ID* entspricht dem Header-Feld Call-ID. Durch die Höhe der Versionsnummer kann die aktuellste Beschreibung als die gültige erkannt werden.

```
ses=<Session-Bezeichner>
```

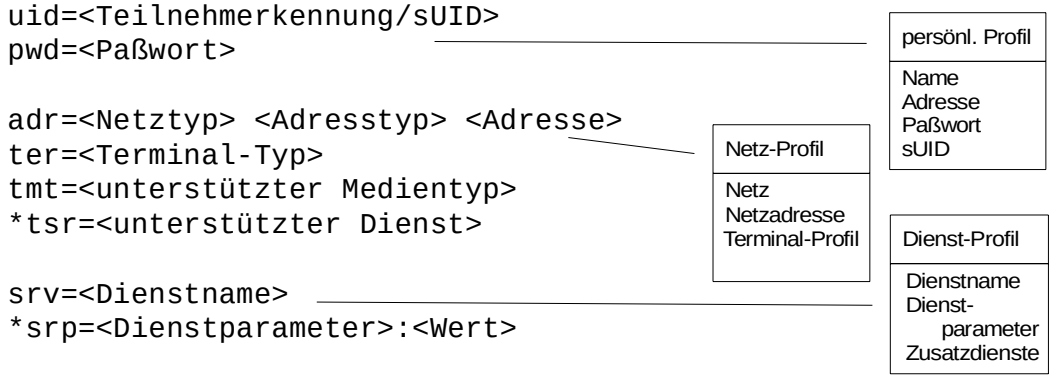
Der Session-Bezeichner dient zur Unterscheidung der verschiedenen Beschreibungen nach *User Profile* (`upr`), *User Service Graph* (`usg`), *Media Connection Graph* (`mcg`) und *Connectivity Connection Graph* (`ccg`).

1. Darstellung: Bezeichner in spitzen Klammern „< >“ sind Platzhalter für einen Parameterwert. Ein „*“ gibt an, daß ein Bezeichner mehrmals wiederholt werden kann. Mit den eckigen Klammern „[ ]“ werden optionale Parameter gekennzeichnet. Standardschrift (Times) in runden Klammern markiert Kommentare.

Eine detaillierte Aufstellung der SDP+-Parameter ist in Anhang B abgedruckt. Im Folgenden werden die wichtigsten Auszüge aus den Beschreibungen dargestellt und mit den UML-Klassendiagrammen der Session-Beschreibungen aus Kapitel 4 in Verbindung gebracht. Die SDP+-Beschreibungen stellen die Teilmenge einer Session-Beschreibung dar, die an andere Komponenten weitergereicht werden.

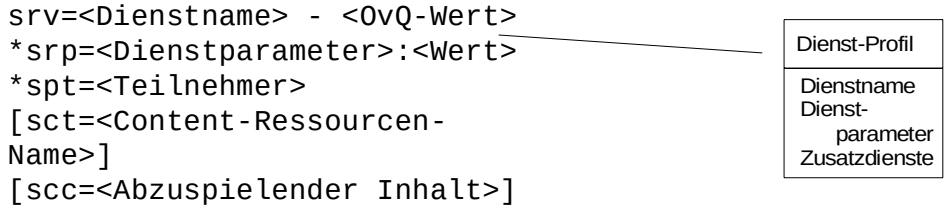
5.1.4.1 SDP+-Teilnehmerprofil

Das Teilnehmerprofil (*User Profile*) in SAMSON besteht aus den verschiedenen Teil-Profilen, die in Abbildung 4.10 auf Seite 72 dargestellt sind. SDP+ dient dazu, die Parameter dieser Profile beim Teilnehmer-Zugang mit der SesCP-Meldung `ACCESS` zu setzen oder zu modifizieren. Für jeden Parameter des Teilnehmerprofils sind in SDP+ Bezeichner definiert, z.B.



5.1.4.2 Grundelemente des User Service Graph

Der *User Service Graph* beschreibt einen Dienst aus Teilnehmersicht. Er enthält abstrakte Dienstparameter und die Bezeichner eingeladener Teilnehmer bzw. Inhalte-Server.

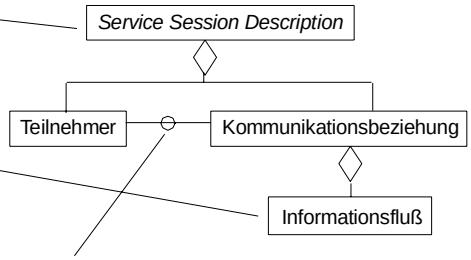


5.1.4.3 Grundelemente des Media Connection Graph

Der *Media Connection Graph* dient zur Anforderung aller Verbindungen, die für die Dienstausführung notwendig sind. Er besteht analog zu SDP aus einem allgemeinen Teil, der für alle Informationsflüsse gilt, einem Zeitfeld-Teil und der Beschreibung der einzelnen Medien. In folgendem Beispiel wird besonders deutlich, wie das objektorientierte Informationsmodell der *Service Session*, das auf einem komplexen Dienstmodell ([SK01]) beruht, in der erweiterten SDP-Beschreibung ausgedrückt werden kann. Objekte und deren Parameter werden in Bezeichner abgebildet. Die Reihenfolge der Bezeichner ist einzuhalten. Beziehungen zwischen Komponenten im Objektmodell werden in eigene Parameter abgebildet. Beispielsweise wird der Zusammenhang zwischen Teilnehmern und Informationsflüssen durch die Rollen der Teilnehmerendpunkte (Sender, Empfänger) in einem Informationsfluß ausgedrückt und in entsprechenden Parametern gruppiert.

```
srv=<Dienstname> - <OvQ-Wert>
smg=<Synchronisations-Typ>
    *<Medium-ID>

mid=<Medium-ID> <Medientyp>
sqc=<Dienst-Qualitäts-Klasse>
mex=<Existenz>
mct=<Kommunikationstyp>
mse=<sUID/sRID> <sUID/sRID> ... (Sender)
mre=<sUID/sRID> <sUID/sRID> ... (Empfänger)
```



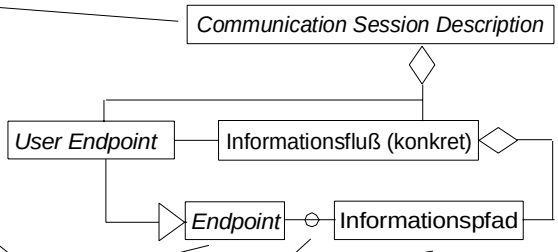
5.1.4.4 Grundelemente des Connectivity Connection Graph

Der *Connectivity Connection Graph* ist ein Ausschnitt aus dem *Media Connection Graph*. Er beschreibt die konkreten Parameter zum Aufbau *eines* Informationsflusses zwischen Teilnehmer-Endgeräten oder Ressourcen. Im Gegensatz zum MCG enthält der CCG genaue Angaben über die Teilnehmeradressen. Der CSM schickt zur Realisierung des MCG mehrere, verschiedene CCGs an die jeweiligen Adaptoren.

```
srv=<Dienstname> <OvQ-Wert>

pid=<Endpunktbezeichner>
adr=<Netztyp> <Adresstyp>
<Adresse>

mid=<Medium-ID> <Medientyp>
mct=<Kommunikationstyp>
mbw=<Spitzenbitrate>
    <Mittlere Bitrate>
    <max. Burstgröße>
mre=<Bitfehlerrate> <Verlustrate>
mde=<max. Delay> <max. Jitter>
mse=<sUID/sRID> <sUID/sRID> ... (Sender)
mre=<sUID/sRID> <sUID/sRID> ... (Empfänger)
```



5.1.4.5 SDP+-Beschreibung in der INFO-Methode

Mit der SesCP-INFO-Methode können alle oben genannten SDP+-Beschreibungen transportiert werden, um beteiligte Server über Parameterwerte während einer Session zu informieren (z.B. zur Teilnehmerverwaltung). Darüber hinaus dient die `INFO`-Meldung zur Abfrage von Parameterwerten. In diesem Fall werden die Parameter in der Form

```
<parameter>=?
```

angegeben. Auf diese Weise wird der Server aufgefordert, dem Client die aktuelle Belegung dieses Wertes in der Antwort mitzuteilen.

5.1.5 Transport der Signalisierungsnachrichten und Adressierung

Für den Transport der SesCP-Signalisierungsnachrichten können nahezu alle Transportprotokolle verwendet werden, die einen asynchronen Nachrichtenaustausch unterstützen. Betrachtet

man eine SesCP-Transaktion als Methodenaufruf beim Empfänger (SesCP-Server), so kommen auch Middleware-Systeme, wie z.B. CORBA, in Betracht. Die zusätzliche Funktionalität, die von vielen Middleware-Systemen bereitgestellt wird (z.B. *Trading*), wird nicht benötigt, da diese Mechanismen bereits im SIP-Protokoll enthalten sind und in SesCP übernommen werden. SesCP ist wie SIP primär für die IP-Protokollfamilie in einem geschlossenen Intranet gedacht. Dabei kann sowohl TCP als auch UDP als Transportprotokoll eingesetzt werden, da SesCP alle Zuverlässigkeits-Mechanismen von SIP übernimmt. Dazu zählen u.a. die geregelte Wiederholung von Meldungen nach Timerablauf sowie die Robustheit gegegenüber identischen Wiederholungen [HSS99].

Zur Adressierung auf der Ebene der Dienstsignalisierung werden in SIP die Endteilnehmer durch die sogenannte SIP-URI identifiziert. Diese ist einer EMail-Adresse ähnlich und hat den Aufbau *user@host.* Der *user*-Teil ist der Name eines Teilnehmers oder eine Telefonnummer. Der *host*-Teil bezeichnet eine IP-Adresse. Er kann um die Angabe eines Ports ergänzt werden.

z.B.: `sip:wolfgang@ei.tum.de:5060`
`sip:+498928923505@gateway.tum.de`

Bei SesCP sind die Adressaten keine Endteilnehmer, sondern Server im Pfad einer SesCP-Meldung. Daher wird die SIP-URI für eine **SesCP-URI** wie folgt verallgemeinert:

Format: `sescp:<entity>@<server>`
z.B.: `sescp:mybusinesscall@servicecontrol.samson.net`
`sescp:alex34@usercontrol.samson.net`

Eine *entity* ist entweder ein Teilnehmer, gekennzeichnet durch seine sUID, ein Dienststeuerungsmodul (SSM) oder ein Inhalte-Server (Content-Ressource, z.B. Videoserver). Ein Dienststeuerungsmodul wird durch den Dienstnamen gekennzeichnet. Instanzen desselben Dienstes werden durch das Header-Feld Call-ID unterschieden. Eine Ressource wird durch eine eindeutige Kennung (sRID) analog zur sUID bezeichnet. *server* kennzeichnet einen Server bzw. den Steuerungsbereich, in dem sich die *entity* befindet. So bezieht sich die sUID auf Adaptoren oder auf die *User Control*, ein Dienstname auf die *Service Control* und eine Ressource auf einen Adaptor.

Die jeweiligen Bezeichner können auch *Aliases* für Gruppen sein, wenn ein genauer Adressat, z.B. ein konkreter Adaptor, noch nicht feststeht und erst durch den CSM festgelegt wird (siehe 5.5.2).

Bei der Verwendung eines IP-Protokollstacks, wie hier vorgeschlagen, wird ein *server* durch eine IP-Adresse, gegebenenfalls mit Portnummer, beschrieben. Dann kann die IP-Adresse des Feldes *server* direkt oder nach Auflösung durch einen *Domain Name Service* für das Routing auf den unteren Protokollschichten benutzt werden.

5.1.6 Abgrenzung

Der IETF-Standard PINT [PC00] wurde bereits in Abschnitt 3.6.1 angesprochen. Hier wird SIP in IP-Netzen für den Transport einer Session-Beschreibung zwischen einem PINT-User (SIP-Client) und einem PINT-Gateway (SIP-Server) benutzt, um PSTN/IN-Dienste aus dem Internet heraus zu initiieren. Daneben beschreibt [EG00] einen Ansatz, wie SIP-basierte VoIP-Systeme um eine IN-Dienststeuerung (SCF) erweitert werden können. Dafür wird mit SIPIN ein Protokoll für die Interaktion zwischen SIP-Servern und IN-SCPs spezifiziert, das auf SIP basierend INAP-Operationen transportiert. Beide Vorschläge machen deutlich, wie SIP für den Transport von Session-Informationen (PINT) oder Protokollparametern (SIPIN) ähnlich wie bei SesCP eingesetzt werden kann. Der Fokus ist bei diesen Ansätzen allerdings auf bestimmte Telekommunikationsdienste und Kommunikationsnetze beschränkt.

Das ICEBERG-System (siehe Abschnitt 3.6.5) definiert eine ähnliche Architektur wie die vorliegende Arbeit mit dem Unterschied, daß das Internet als Kern-Vermittlungssystem für die Signalisierung und für Nutzdaten verwendet wird. SIP wird in diesem System nur für die initiale Einrichtung von Sessions benutzt. Die Session-Verwaltung und die Einrichtung von Kommunikationsbeziehungen differiert von SesCP. Der Fokus von ICEBERG liegt auf einer hochskalierbaren und fehlertoleranten, vollständig verteilten Session-Verwaltung. Hierfür wird ein neues Signalisierungsprotokoll definiert, bei dem keine zentrale Session-Verwaltung existiert, sondern die Teilnehmer über ihre Call-Agents periodisch ihre aktuelle Session-Beschreibung austauschen [WJK00].

Verglichen mit traditionellen Dienst-Signalisierungsmechanismen wie z.B. dem Signalisierungsprotokoll für Intelligente Netze (INAP) [Q.1228] stellt SesCP mehr Flexibilität bezüglich neuer Dienste bereit. Im Gegensatz zu INAP schränkt SIP nicht auf einen begrenzten, standardisierten Satz von INAP-Meldungen ein, sondern erlaubt, mit einer SesCP-Meldung beliebige Dienstbeschreibungen zu transportieren und außerdem mehrere Aktionen in der Dienststeuerung gleichzeitig auszulösen.

5.1.7 Zusammenfassung und Ausblick

Das *Session Control Protocol* (SesCP) wurde im Rahmen der vorliegenden Arbeit neu auf der Basis des IETF *Session Initiation Protocols* SIP entwickelt. SesCP ist ein transaktionsbasiertes Signalisierungsprotokoll für den Austausch von Nachrichten zwischen den verteilten Komponenten eines Dienstservers. Es dienst dazu, den Teilnehmerzugang sicherzustellen und Dienste über Anpassungseinheiten in heterogenen Netzen zu steuern. Die SesCP-Meldungen durchlaufen verschiedene Steuerinstanzen des Dienstservers, um schrittweise die Dienstbeschreibung bis hin zur Abbildung auf eine konkrete Infrastruktur zu detaillieren. Durch die strikte Trennung von Protokollmeldungen und Dienstbeschreibung ist SesCP für beliebige Dienste verwendbar.

Die zugrunde gelegten Kommunikationsprinzipien aus Kapitel 4 beziehen sich auf die abgeschlossene Serverplattform SAMSON (begrenzte Teilnehmerzahl, etc.). SesCP ermöglicht eine Anwendung der Serverarchitektur auch in einem größeren, weit verteilten Kontext. Die Adressen der handelnden Instanzen werden erst durch Redirect-Server und Proxy-Server von SesCP-Adressen auf Netz-Adressen umgesetzt (z.B. IP-Adressen). Das erlaubt es, nicht nur Steuerungsbereiche, sondern auch die Instanzen einer Komponente (z.B. SSM) auf mehrere physikalische Orte zu verteilen. Bei geeigneter Dimensionierung der Komponenten und der Übertragungsparameter des Signalisierungsnetzes (siehe auch in Anhang E) kann die Serverarchitektur dadurch für unterschiedlich große Teilnehmerzahlen ausgelegt werden (Skalierbarkeit).

Durch die Verwendung des SIP-Protokolls als Basis für das Signalisierungsprotokoll SesCP ist darüber hinaus eine Realisierung der Serverfunktionalität in einer zukünftigen, IP-basierten Infrastruktur möglich. Das Konzept des *Next Generation Network* (siehe Abbildung 2.1 auf Seite 10) beschreibt eine zukünftige Kommunikations-Infrastruktur als Kombination verschiedener, heterogener Zugangsnetze (z.B. PSTN, GSM, LAN), deren Signalisierung und Informationsflüsse an Gateways auf ein IP-basiertes Kernnetz (z.B. bei UMTS) umgesetzt werden. Die Verwendung von IP-basierten Signalisierungsprotokollen für multimediale Dienste, wie z.B. H.323 und SIP, ist Teil dieses Konzeptes [MU01]. SesCP kann in diesem Umfeld als Signalisierungsprotokoll für eine übergeordnete Dienststeuerung eingesetzt werden (siehe Abbildung 2.1) und Dienste über die Media-Gateways steuern.

5.2 Teilnehmerverwaltung und Dienstzugang

Aufgabe der *User Session* ist die Teilnehmerverwaltung und die Realisierung des Dienstzuganges für die Teilnehmer. Beides wird in der *Registration Session* und der *Access Session* ausgeführt. Der Teilnehmer, d.h. sein aktueller Zugang zum SAMSON-Server, wird durch eine Protokollkomponente, den sogenannten *Protocol Agent* (PA), in dem entsprechenden Adaptor vertreten. Für den Meldungsablauf in beiden Sessions hat dieser *Protocol Agent* die Rolle des Client. Der *User Proxy* ist Endpunkt der Registrierungs-Transaktionen. Gleichzeitig ist er Proxy-Server für den Dienstzugang, dessen Transaktionen in der *Service Control* enden.

5.2.1 Teilnehmerverwaltung

Eine separate Teilnehmerverwaltung unterstützt personalisierte Dienste und ermöglicht den Teilnehmern zu jeder Zeit und von jedem Netz aus den Zugang zu ihren persönlichen Informationen. Kernelement der Teilnehmerverwaltung ist das Teilnehmerprofil.

Teilnehmerprofil und Teilnehmerdatenbank

Zur Verwaltung teilnehmerspezifischer Daten enthält der Block *User Control* (UC) eine Datenbankkomponente (UDB), die für jeden registrierten Teilnehmer dessen Profile unter einer eindeutigen Teilnehmerkennung speichert. Darin enthalten sind alle Daten über den Teilnehmer, die für die Dienststeuerung relevant sind:

- Netze und zugehörige Adressen, unter denen der Teilnehmer erreichbar ist
- Endgeräte, die der Teilnehmer in den Netzen verwendet
- Konfigurationen für diese Endgeräte
- der momentane oder letzte Aufenthaltsort (Netz, Endgerät) des Teilnehmers
- Dienste, für die der Teilnehmer registriert ist
- Parameter für personalisierte Dienste, z.B. Rufumleitungsziele
- Dienstübergreifende Vorlieben und Einstellungen des Teilnehmers bezüglich Qualität, etc.

Die Struktur des Teilnehmerprofils wurde bereits in Abschnitt 4.5.2 beschrieben. Für die Realisierung der UDB wird ein *Directory Server* unter Verwendung des *Lightweight Directory Access Protocols* (LDAP) vorgeschlagen und prototypisch realisiert (siehe Kapitel 7). Durch die Verwendung eines Standard-LDAP-Servers kann der Teilnehmer sein gespeichertes Teilnehmerprofil auch für andere Anwendungen nutzen, z.B. um seine persönlichen *Bookmarks* Browser-unabhängig abzulegen.

Verhalten der Client-Komponenten in den Adaptoren

Die Teilnehmersignalisierung für SAMSON läuft stets über Adaptoren. Sie koppeln die netzspezifische Signalisierung eines Teilnehmerzugangs mit der SAMSON-internen Signalisierung (SesCP). Je angeschlossenem und aktivem Teilnehmer wird in einem Adaptor ein *Protocol Agent* (PA) gestartet, der als SesCP-Client (PAC) arbeitet.

Für das Teilnehmermanagement steht dem PAC die Meldung `ACCESS` zur Verfügung, mit der er eine *Registration Session* startet und im Message-Body die aktuellen Teilnehmerdaten mitteilt. Diese können vom Teilnehmer selbst über die netzspezifische Signalisierung eingegeben werden oder sie werden vom PA anhand des Netzanschlusses ermittelt. `ACCESS`-Meldungen mit unvollständigen Informationen im Message-Body (z.B. fehlendes Paßwort) werden abge-

wiesen und sind in einer erneuten Anfrage richtigzustellen. Änderungen in der Teilnehmerkonfiguration werden mit weiteren ACCESS-Meldungen übertragen, die dieselbe Call-ID besitzen. BYE beendet eine aktive *Registration Session* und teilt dem Server mit, daß der Teilnehmer nicht mehr über diese Konfiguration erreichbar ist. Der PAC terminiert.

Verhalten der Komponenten der Teilnehmerverwaltung

Jeder Teilnehmer wird in SAMSON durch einen *User Proxy* (UP) vertreten, der die Interessen des Teilnehmers wahrnimmt. Der UP ist der Verwalter des Teilnehmerprofils. Der UP-Prozeß ist der Teilnehmerdatenbank vorgeschaltet und verwaltet eine Kopie des Profils, um Performance-Probleme, die bei einem zentralen Datenbankzugriff auftreten können, zu vermeiden. Der UP ist Endpunkt der ACCESS-Transaktion. Er gleicht seinen Teilnehmerdatensatz regelmäßig mit der UDB ab. Für diesen Datenaustausch wird die Meldung INFO verwendet. Alle Anfragen an die UDB gehen über den UP.

Der UP ist für jeden aktiven Teilnehmer genau einmal vorhanden und seine Erzeugung wird durch das erste ACCESS, das für diesen Teilnehmer eintrifft, ausgelöst. Der PAC schickt die erste ACCESS-Meldung an den *Initial Access Manager* (IAM). Dieser unterhält eine Datenbank über alle UPs. Ist ein zugehöriger UP noch nicht vorhanden, so erzeugt der IAM eine neue UP-Instanz. Die Adresse der zu einem Teilnehmer gehörigen UP-Instanz wird dem PAC über ein *Redirect* mitgeteilt. Dieser hat an diese Adresse eine erneute Anfrage zu stellen. Ein UP holt sich bei seiner Initialisierung den in der UDB gespeicherten Teilnehmerdatensatz. Er bestätigt die erfolgreiche Anmeldung und schickt im Message-Body eine Liste mit Diensten, die vom Teilnehmer in der vorliegenden Konfiguration gestartet werden können. Diese Liste kennzeichnet außerdem Dienste, in denen der Teilnehmer gerade aktiv ist, (z.B. aus einer anderen *Access Session* gestartet). Abbildung 5.5 zeigt das zugehörige Signalablaufdiagramm.

Der *User Proxy* wird immer dann für einen Teilnehmer instanziert, wenn seine Daten innerhalb von SAMSON benötigt werden und noch kein UP existiert. Dies kann nicht nur durch den

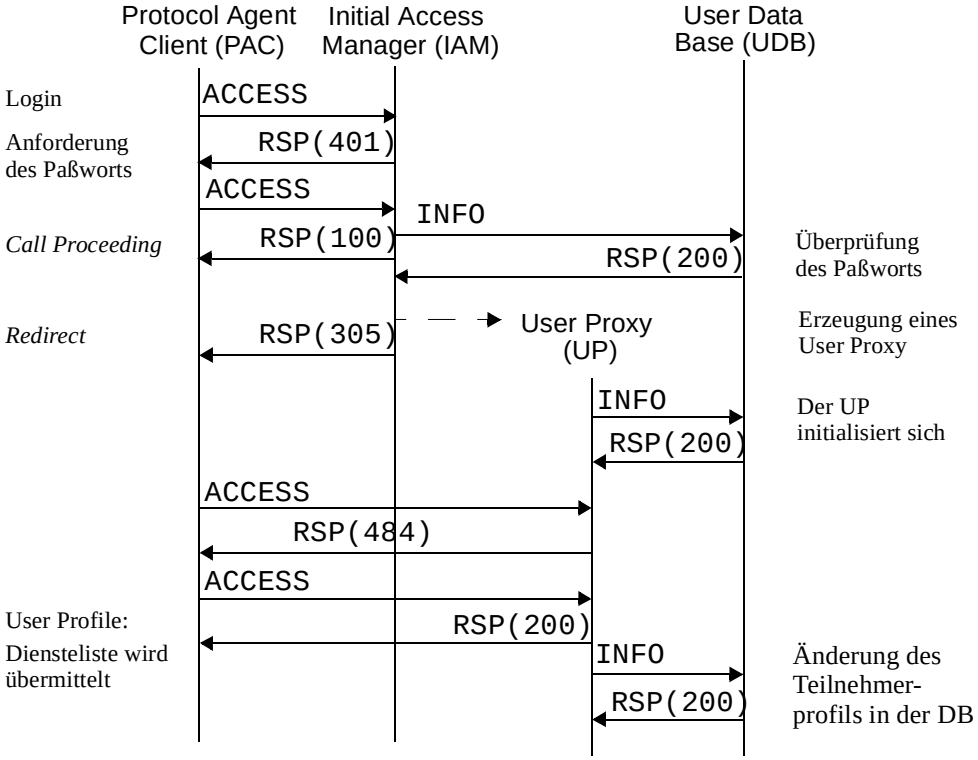


Abbildung 5.5: Signalablaufdiagramm der Registration Session

Teilnehmer, sondern auch durch eine Dienstinstanz erfolgen. Ein UP ist durch die sUID des Teilnehmers eindeutig gekennzeichnet.

Es ist außerdem vorgesehen, auch die Parameter von Teilnehmern, die keine sUID besitzen haben, aber an einem SAMSON-Dienst teilnehmen (z.B. eingeladener Konferenzteilnehmer), temporär zu speichern. Dafür wird kein Datensatz in der UDB angelegt, sondern lediglich ein sogenannter *Non-Subscribed User Proxy* erzeugt (NUP), der sich identisch zu einem UP verhält.

5.2.2 Dienstzugang und Dienstaufruf

Eine *Access Session* existiert in SAMSON für jeden Teilnehmer, der seinen aktuellen Netzzugang in der *Registration Session* registriert hat und für den ein User Proxy instanziert wurde. Sie stellt die Schnittstelle zur Dienststeuerung dar, d.h. sie erlaubt einem Teilnehmer, Dienste auszuwählen und zu starten. Für jeden aktiven Dienst eines Teilnehmers wird eine Dienstinstanz, ein *Service Session Manager,* erzeugt.

Abbildung 5.6 zeigt das Signalablaufdiagramm der Meldungen innerhalb der *Access Session*. Alle Meldungen gehen dabei über den UP. Dies gilt insbesondere auch für Nachfragen des SSM aufgrund unvollständiger Angaben in der `INVITE`-Meldung. Mit Status-Code 484 (*Address Incomplete*) in der Antwort auf das `INVITE` können weitere Angaben (über den Message-Body) angefordert werden, bevor der erfolgreiche Start des Dienstes gemeldet wird.

Der *User Proxy* agiert als Zustands-behafteter (*stateful*) Proxy-Server. Er verarbeitet nicht nur die SesCP-Parameter, sondern auch den Message-Body und ergänzt Dienstparameter, die der Teilnehmer in seinem Dienstprofil gespeichert hat, aber beim Dienstaufruf nicht explizit angibt. Bis auf die `ACCESS`-Meldung reicht der UP alle SesCP-Meldungen entsprechend weiter. Über das Header-Feld *Record-Route* sorgt der UP dafür, daß er stets im Transaktionspfad bleibt.

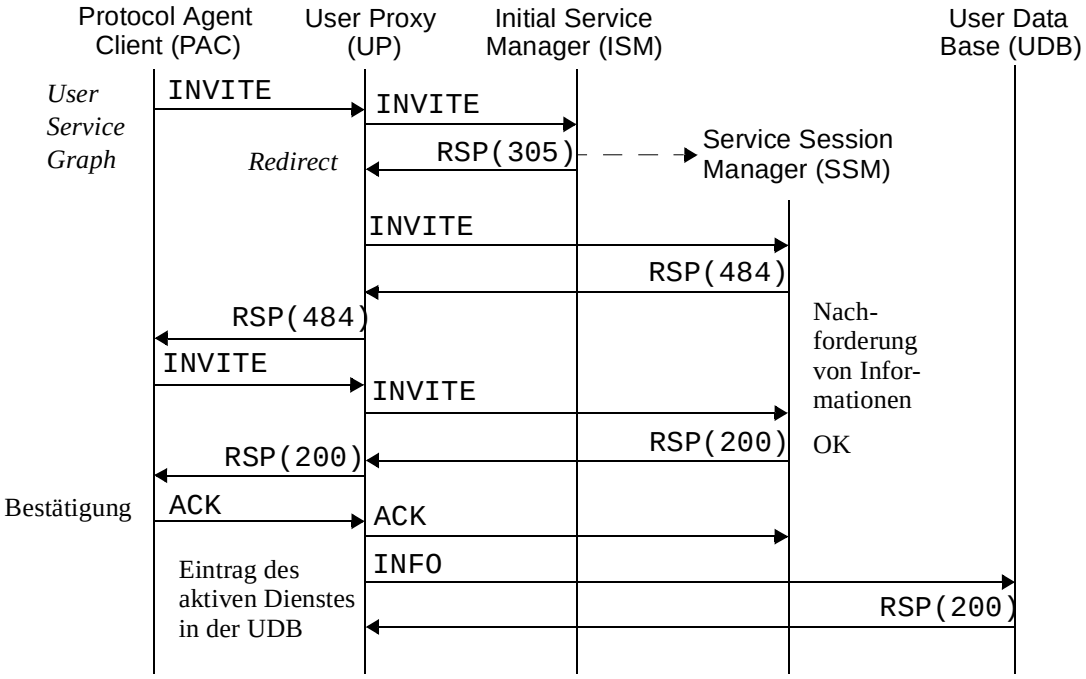


Abbildung 5.6: Signalablaufdiagramm der Access Session

5.2.3 Neues Konzept für die Netzunabhängigkeit des Dienstzuganges

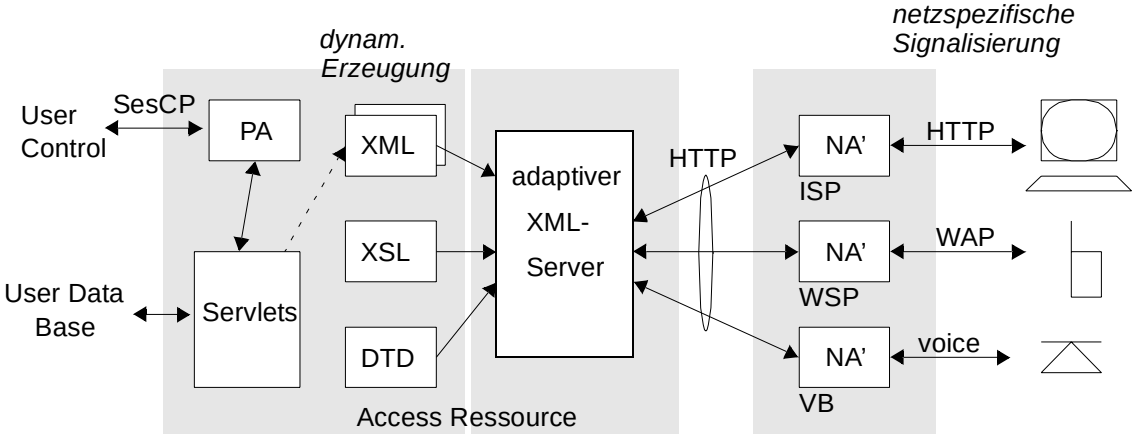
Die Adaptoren realisieren die Unabhängigkeit des Dienstservers von den Netzen, indem sie die spezifische Signalisierung in die SesCP-Signalisierung umsetzen. Die Interaktion zwischen den Teilnehmerendgeräten und den Adaptoren beschränkt sich dabei auf die Möglichkeiten der netzspezifischen Signalisierung, insbesondere der (beschränkten) Darstellungsmöglichkeiten im Endgerät (z.B. Tonwahl, kein Displayzugriff). Gerade neue Dienste, die eine besondere Interaktion des Teilnehmers erfordern (z.B. Wahl aus einer Dienstliste), können damit nicht oder nur sehr umständlich angeboten werden. Für einen flexiblen Dienstzugang sind daher Mechanismen vorzusehen, die die Inhalte von SesCP-Meldungen adaptiv an die Darstellungsmöglichkeiten des jeweiligen Endgerätes anpassen.

Zur Bereitstellung eines flexiblen Teilnehmerzugangsmoduls bietet sich der Einsatz eines Hypertextservers an, der das *Frontend* zum Benutzer darstellt und über Skripten (z.B. CGI) oder Server-seitige Programme (z.B. Servlets) die Benutzerinteraktion steuert. Ein solcher Hypertextserver kann Bestandteil eines Adaptors sein. In ihm werden dann die Interaktionen des Teilnehmers, die dieser über seinen Browser vornimmt, in die SesCP-Signalisierungsmeldungen übersetzt. Umgekehrt werden Meldungen an den Benutzer im Hypertextformat auf dessen Browser dargestellt.

Je nach Anwendungsfall existieren verschiedene Hypertextsprachen, die auf die speziellen Eigenschaften von Endgeräten (Display, Datenrate) abgestimmt sind, z.B.

- HTML (*Hypertext Markup Language*) als die Standardanwendung im Internet
- WML (*Wireless Markup Language*) für WAP-Clients in Mobilfunkgeräten und
- VXML (*Voice Extensible Markup Language*) für Sprach-*Browsing* an Telefonen.

Jede dieser verschiedenen Hypertext-Sprachen kann prinzipiell über jedes Netz, das Datenübertragung direkt oder über Modemstrecken unterstützt, übertragen werden. Es würde einen zu großen Aufwand bedeuten, jeden Adaptor mit den entsprechenden Hypertext-Servern auszustatten. In SAMSON ist eine sogenannte *Access Resource* vorgesehen, die zentral für jede Hypertextsprache und für alle möglichen Daten-Zugänge als Server fungiert. Kern ist ein Hypertext-Server, der die Daten aus der Teilnehmerdatenbank im Metadatenformat XML (*Extensible Markup Language*) bereithält. Über entsprechende *Style-Files* können diese in das gewünschte Darstellungsformat ausgegeben werden. Abbildung 5.7 zeigt den prinzipiellen



ISP Internet Service Provider WSP WAP Service Provider VB Voice Browser
XML eXtensible Markup Language XSL XML Style Sheet DTD Document Type Definition
NA' modifizierter Netzadaptor PA Protocol Agent WAP Wireless Application Protocol

Abbildung 5.7: Prinzip der Access Ressource

Aufbau der *Access Resource*. Es ist zu beachten, daß sie bereits die *Protocol Agents* enthält, die die Teilnehmerinteraktion in die SesCP-Meldungen übersetzen. Die Netzadaptoren dienen nur dem Teilnehmerzugang zum Hypertextserver. Details zur durchgeführten Realisierung werden in Kapitel 7 beschrieben.

5.2.4 Mobilitätsunterstützung

Durch die zentrale Teilnehmerverwaltung wird persönliche Mobilität unterstützt, indem der Teilnehmer unter seiner eindeutigen Kennung an jedem (registrierten) Endgerät erreichbar ist. Schon durch den einheitlichen Dienstzugang über Adaptoren und besonders durch die einheitliche Darstellung über die *Access Resource* wird darüber hinaus Dienste-Mobilität gewährleistet. Ein Teilnehmer hat jederzeit, über jedes Netz und mit jedem Endgerät Zugriff auf seine Dienste und sein Teilnehmerprofil.

Eine globale Verfügbarkeit des Teilnehmerprofils für unterschiedliche Anwendungen wird unter dem Begriff *Profile Mobility* verstanden [BKE00].

5.3 Zentrale Dienststeuereinheit

Kern der Serverarchitektur ist die Dienststeuereinheit. Ihre Aufgabe ist es, den Dienst gemäß dem in der Dienstdefinition festgelegten Verhalten zu steuern. Zur Dienststeuerung gehört die Anforderung an die Kommunikationssteuerung, Informationsflüsse einzurichten, sowie die Teilnehmerinteraktion. In SAMSON wird die Dienststeuerung von einer zentralen Einheit ausgeführt. Dabei wird für jeden aktiven Dienst eine Instanz der zentralen Dienststeuerungseinheit, der *Service Session Manager* (SSM), erzeugt.

5.3.1 Struktur und Verhalten einer Dienstinstanz

Ein Dienst definiert sich durch die Dienstlogik und den Dienstkontext. Der **Dienstkontext** beschreibt alle möglichen Zustände, die von einem Dienst eingenommen werden können, während die **Dienstlogik** die Art und Weise der Zustandsübergänge angibt. Die *Service Session Description* legt den Dienstkontext für SAMSON fest (siehe Abschnitt 4.5.3), indem sie die möglichen Parameter der Dienstbeschreibung vorgibt. Für den Nahrichtenaustausch mit den anderen Komponenten der Dienststeuerung beinhaltet ein SSM einen SesCP-Server-Prozeß und einen SesCP-Client-Prozeß, die mit dem zentralen Dienststeuerprozeß verknüpft sind (siehe Abbildung 5.8).

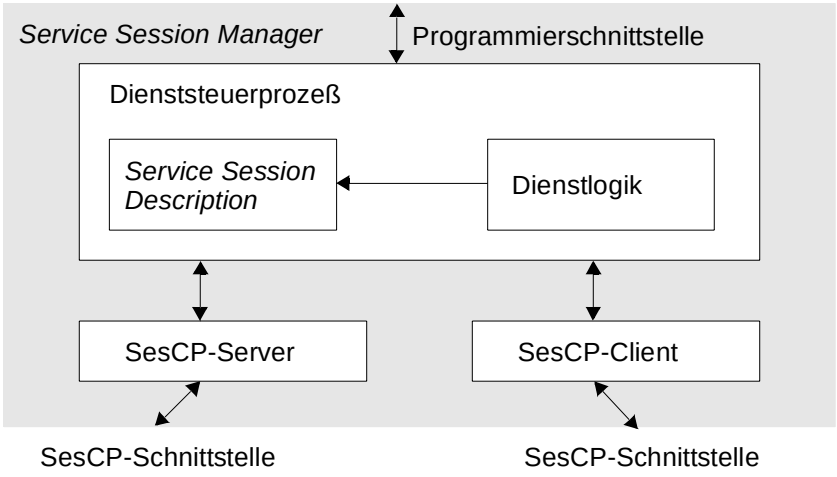


Abbildung 5.8: Struktur des Service Session Manager

Server-Prozeß des Service Session Manager

Der SesCP-Server terminiert alle Anfragen, die über den *User Proxy* von den *Protocol Agents* in den Adaptoren an ihn gerichtet werden, und ruft den Dienststeuerprozeß auf. Ein Dienst wird durch ein `INVITE` ausgelöst und wird erst aktiv, wenn die zugehörige `ACK`-Meldung empfangen wurde. Mit weiteren `INVITE`-Meldungen kann ein Teilnehmer im Rahmen der Teilnehmerinteraktion die Parameter eines Dienstes ändern, d.h. beispielsweise einen weiteren Teilnehmer hinzunehmen. Der Empfang einer `BYE`-Meldung beendet die eigene Teilnahme an einem aktiven Dienst, während `END` den gesamten Dienst terminiert.

Client-Prozeß des Service Session Managers

Der SesCP-Client fordert auf Anfrage des Dienststeuerprozesses die Einrichtung von Kommunikationsbeziehungen von der Kommunikationssteuerung an. Wichtigstes Element ist die `SETUP`-Meldung. Da die Adresse der Adaptoren, über die Verbindungen eingerichtet werden, noch nicht bekannt ist, wird die `SETUP`-Meldung mit dem *Media Connection Graph* im Message-Body an die Alias-Adresse *adaptor* geschickt, wobei im Header-Feld *Route* stets der zugehörige *Communication Session Manager* (CSM) bzw. der *Initial Communication Manager* (ICM) als Proxy eingetragen ist. Der CSM setzt den Alias in eine richtige Adresse um. Da das *Header*-Feld "To:" von Proxy-Servern nicht überschrieben werden darf, geschieht dies im *Request-URI*-Header-Feld.

Weitere Anfragen des SSM sind `INVITE` zur Einbeziehung von Content-Ressourcen (siehe Abschnitt 5.3.4) und `INFO` zur Teilnehmer-Dienst-Interaktion (siehe 5.3.3). Darüber hinaus ist der Client-Prozeß zuständig für alle anderen Teilnehmer, die neben dem Dienst-Initiator an einem Dienst partizipieren, einen *User Proxy* zu starten. Dies erfolgt über die Anfrage `ACCESS` an den *Initial Access Manager.*

5.3.2 Dienstlogik

SAMSON beschreibt eine Dienststeuerung, bei der ein Teilnehmer vorgefertigte Dienste abrufen und ausführen kann. Neben einem flexiblen Informationsmodell für den Dienstkontext (statische Dienstbeschreibung), der *Service Session Description*, wird deswegen eine **Dienstlogik** (dynamische Dienstbeschreibung) spezifiziert, die Dienste auf diesem Dienstkontext steuert. Es handelt sich dabei nicht um Zusatzdienste zu einem Basisdienst, sondern um eigenständige Dienste, die Zusatzdienstmerkmale enthalten können.

Prinzip

Jeder Dienstablauf läßt sich generell in drei Teile gliedern. Während des Dienstaufbaus (*Call Establishment*[1]) werden alle Aktionen ausgeführt, die den gewünschten Startzustand des Dienstes herstellen, z.B. eine Konferenzschaltung. Danach ist der Dienst aktiv (*Mid Call*) und reagiert entsprechend der Dienstlogik auf Ereignisse wie z.B. Teilnehmerinteraktionen. Auch bei Beendigung des Dienstes in der Abbauphase (*Call Release*) können Aktionen definiert werden.

1. In Klammern werden die entsprechenden englischen Begriffe aus der Rufverarbeitung beim IN angegeben.

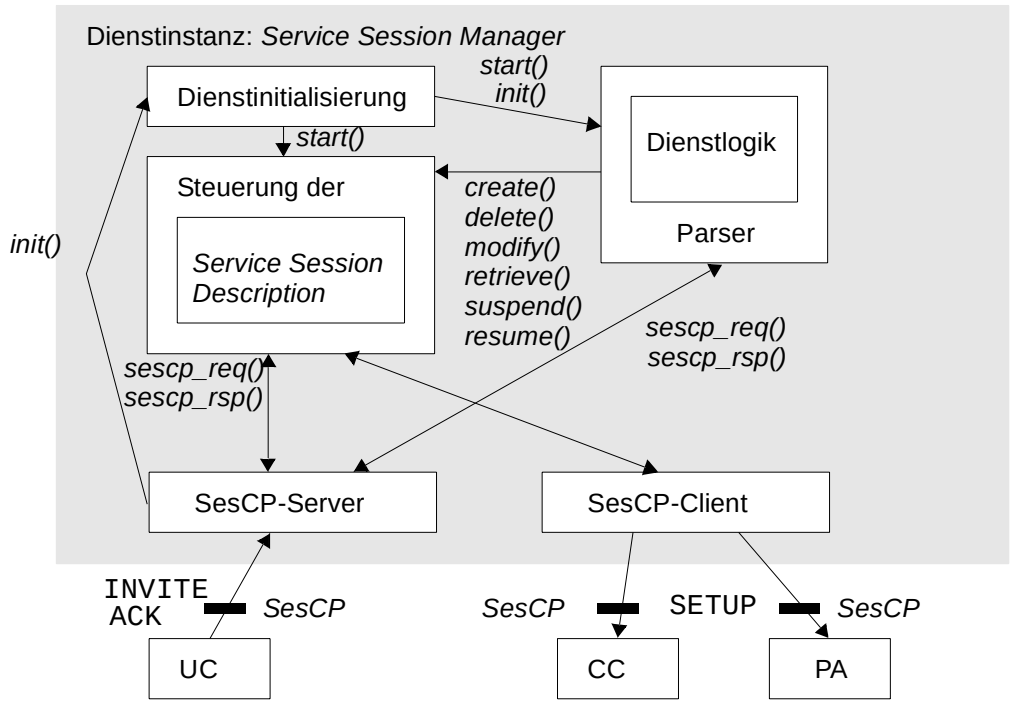


Abbildung 5.9: Innere Struktur der Dienststeuerung

Abbildung 5.9 illustriert das Prinzip und die innere Struktur einer Dienstinstanz. Das auf das erste `INVITE` folgende `ACK` stößt die Dienstinitialisierung und damit die Aktionen des Dienstaufbaus an. Dabei wird die Dienstlogik mit den Daten aus der SDP+-Beschreibung des *User Service Graph* initialisiert. Der Parser kann entsprechend der Dienstlogik mit den sechs elementaren Operationen *Create*, *Delete*, *Modify*, *Retrieve*, *Suspend* und *Resume* die *Service Session Description* aufbauen und ändern. Für jede Operation werden erst die entsprechenden Berechtigungen (z.B. anhand der Teilnehmerrollen) geprüft, bevor sie ausgeführt werden. Je nach erfolgter Änderung wird eine Aktion in der Steuerung der *Service Session Description* ausgelöst, d.h. eine SesCP-Meldung über den SSM-Client ausgegeben. Tabelle 5.4 erläutert die Operationen. Die Dienstlogik reagiert auf eine positive (*success*) oder negative (*failure*) Bestätigung einer Operation.

Operation	**Funktion**
Create	Anlegen eines Objekts/einer Objektgruppe, z.B. neuer Teilnehmer
Delete	Löschen eines Objektes/einer Objektgruppe
Modify	Ändern von Parametereinstellungen
Retrieve	Auslesen von Parametereinstellungen
Suspend	Parken eines Objektes/einer Objektgruppe
Resume	Reaktivieren eines Objektes/einer Objektgruppe

Tabelle 5.4: Operationen auf die Dienstbeschreibung

Notation

Es gibt verschiedene Möglichkeiten, wie die Dienstlogik notiert werden kann. Beim ISDN ist sie in den Zustandsautomaten des Q.931-Protokolls in einem monolithischen Programm festgelegt. Modularer ist der Aufbau der Dienstlogik aus einzelnen *Service Independent Building Blocks*, die wie beim IN durch einen SIB-Graphen verknüpft werden. Die Schwierigkeit besteht hier in der geeigneten Wahl der SIBs. Da die entsprechenden Entwicklungsumgebungen sehr proprietär aufgebaut sind und eine hohe Einarbeitung erfordern, ist man allgemein dazu übergegangen, die Dienstlogik mit Standardprogrammiersprachen zu programmieren. Objektorientierte Sprachen wie z.B. Java bieten zusätzlich die Möglichkeit, Module vorzudefinieren, die vererbbar als *high-level* SIBs kombiniert werden können.

Aufgrund der Mächtigkeit der SAMSON-Dienststeuerung und der Forderung nach Erweiterbarkeit hinsichtlich neuer Dienste kommen für die Programmierung der Dienstlogik Standardprogrammiersprachen in Frage. Nach einer eingehenden Analyse des Standes der Technik fiel in diesem Zusammenhang eine Technik auf, mit der die Dienstlogik abstrakt beschrieben werden kann. Dabei wird eine strukturierte Beschreibungssprache als High-Level Programmiersprache eingesetzt. Vorteil ist eine einfache und intuitive Programmierung. Beispiel für eine strukturierte Beschreibungssprache ist die *eXtensible Markup Language* XML [Mar00].

Ziel und Lösungsansatz

Ziel ist die Definition einer High-Level Beschreibungssprache, mit der die Dienstlogik als eine Sequenz von Aktionen dargestellt werden kann. Die Dienstlogik operiert nur mit den sechs oben beschriebenen Operationen auf der *Service Session Description*.

Als Grundlage für eine Lösung bietet sich, wie bereits oben erwähnt, die *eXtensible Markup Language* XML an. Unter XML versteht man eine standardisierte, aus mehreren Elementen bestehende Beschreibungssprache für strukturierte Daten [Mar00]. Die Struktur und die Semantik der Daten wird mit sogenannten *Tags* ähnlich wie in HTML ausgedrückt, z.B. <user> Alex </user>. Erlaubte *Tags* werden in der *Document Type Definition* (DTD) festgelegt, mit der sich die Korrektheit einer Spezifikation verifizieren läßt. Über XSL-Dokumente (*XML Style Sheet*) läßt sich, falls benötigt, das Ausgabeformat (z.B. HTML) des mit XML beschriebenen Dateninhalts festlegen.

Die Vorteile XML-basierter Dienstlogikbeschreibung wurden mit einer auf VoxML-basierten Dienststeuerung bereits praktisch erprobt [GKK00]. XML erlaubt die einfache Darstellung strukturierter Daten, z.B. Dienstlogikbäume und ist dabei sowohl für Menschen (textbasiert) als auch für Maschinen (mit DTDs verifizierbare Syntax) leicht lesbar bzw. parsbar. Mit XML können eigene *Tags* definiert werden. Damit ist die Erweiterbarkeit gewährleistet. XML ist plattformunabhängig und fehlertolerant. Nachteile liegen in einem möglicherweise hohen Umfang von Beschreibungen komplexer Dienste, einem nicht unerheblichen Standardisierungsaufwand für *Tags* und dem Umstand, daß Algorithmen und Variablenverarbeitung schwer ausgedrückt werden können. Da die Güte einer XML-Beschreibung von der Definition der *Tags* abhängt, können Informationen nicht zwangsläufig besser als mit anderen textbasierten Sprachen beschreiben werden. Der Vorteil von XML besteht vielmehr in der Tatsache, daß die Korrektheit der Syntax einer Beschreibung einfach nachgewiesen und die Beschreibung einfach weiterverarbeitet werden kann. Letzteres wird durch die hohe Verfügbarkeit von Parsern unterstützt. Damit sind die Voraussetzungen für eine Dienstlogik-Notation erfüllt.

Abgrenzung

Es existieren bereits einige auf XML-basierte Sprachen für die Spezifikation der Dienstlogik, von denen jedoch keine für die SAMSON-Dienststeuerung geeignet ist. Die in [GKK00] beschriebene und als Vorstudie zu der vorliegenden Arbeit durchgeführte Erweiterung der Hypertextsprache VoxML (Voice eXtensible Markup Language) ist auf Zusatzdienste im Bereich von interaktiven Ansagediensten abgestimmt. Auch die *Call Processing Language* (CPL) der IETF [LS00] ist auf Zusatzdienste ausgelegt. Zudem ist die CPL für die Dienstprogrammierung durch die Teilnehmer selbst gedacht, was sich in einem erheblich eingeschränkten Funktionsumfang widerspiegelt. In einigen Forschungsprojekten wird die CPL derzeit erweitert, z.B. [BCB01].

Vorgehen zur Definition einer neuen Notation

Für die Definition der *Tags* und der Struktur einer neuartigen Sprache zur Dienstlogikprogrammierung werden zunächst alle notwendigen Aufgaben eines Dienstes, die durch die Dienstlogik gesteuert werden, gesammelt, ihr Nutzen für SAMSON identifiziert und gruppiert (vgl. Abschnitt 2.3.1). Anschließend werden die verallgemeinerten Funktionen auf die drei Steuerungsbereiche von SAMSON aufgeteilt. Tabelle 5.5 zeigt, daß teilnehmerbezogene Dienste eher in den Bereich *User Control* und Routing-Aufgaben in den Bereich *Communication Control* fallen.

Da sich die Dienstlogik auf die Objekte der *Service Session Description* bezieht, werden die Funktionen auf die darin enthaltenen Objekte bezogen und auf die sechs elementaren Operatoren zurückgeführt.

User Control	**Service Control**	**Communication Control**
Teilnehmerzugang	Session-Steuerung	Adressumsetzung
Teilnehmerverwaltung	Mediensteuerung	Teilnehmerlokalisierung
Gebührenberechnung	Einladen von Teilnehmern	Ressourcenauswahl
Zusatzdienste[a] für ankommende Rufe	Teilnehmer-Interaktion	
	Ereignisverarbeitung	
	Ticketing, Logging	
	Zusatzdienste[a] für ausgehende Rufe	

a. Siehe Abschnitt 5.4

Tabelle 5.5: Dienstlogikfunktionen und ihre Verteilung auf die Steuerungsbereiche

Service Programming Language

Für die Beschreibung der Dienstlogik in SAMSON wird eine neue Notation vorgeschlagen, die im Folgenden als *Service Programming Language* (SPL) bezeichnet wird. Die Struktur der SPL richtet sich nach den drei Phasen eines Dienstes: *Session Initiation*, *Active Session* und *Session Release*. Operationen auf Objekte der *Service Session Description* werden innerhalb eines Objekttyps, z.B. Teilnehmer (*Party*) gruppiert. Dabei werden zunächst die zu ändernden oder anzulegenden Objekte mit ihren Parametern beschrieben und dann die Operationen ausgeführt. Die Reaktion auf abgelehnte Operationen wird direkt bei der Operation spezifiziert.

Operationen auf Objekte werden in der *Session Initiation*- und der *Session Release*-Phase direkt in der gewünschten Reihenfolge angegeben und in der *Active Session*-Phase durch sogenannte „actions“ beschrieben. *Actions* werden durch externe (z.B. eintreffende SesCP-Meldungen) oder interne Ereignisse (z.B. Timer) ausgelöst. Sie werden über Makros, sogenannte „*subactions*“ spezifiziert. *Subactions* stellen Module in der SPL dar, die mehrmals verwendet werden können. Abbildung 5.10 verdeutlicht den Aufbau einer Dienstlogikbeschreibung mit SPL an einem Beispiel und zeigt die typischen *Tags*. In der Initialisierungsphase wird eine Sprachverbindung zwischen zwei Teilnehmern aufgebaut, eine Datenverbindung (Text) ist optional. In der aktiven Phase kann die Kommunikation, ausgelöst durch eine INVITE-Meldung, um eine Videoverbindung erweitert werden. Das Beispiel enthält bewußt keine weiteren Details, um die Grundstruktur zu verdeutlichen. Eine Definition der *Tags* durch die DTD befindet sich in Anhang C.

Für komplizierte Dienstabläufe stößt man bei XML bald an Grenzen. Da XML nur die Struktur und die Daten der Logik beschreibt, müssen Berechnungen und Entscheidungen (*success*/*failure*) durch den Parser zur Laufzeit erfolgen. Damit besteht für die Logik komplexer Dienstmerkmale eine Abhängigkeit von dessen Fähigkeiten, die in der vorliegenden Arbeit nicht genauer betrachtet werden. Der vorgestellte, XML-basierte Ansatz ist daher nur ein Vorschlag, wie die hier systematisch herausgearbeitete Dienstlogikstruktur beschrieben werden kann.

Programmierbarkeit

Für die Programmierung der beschriebenen SPL gibt es entweder die Möglichkeit, die Dienstlogik durch damit vertraute Programmierer direkt in SPL zu spezifizieren oder zusätzliche graphische Werkzeuge zu benutzen, die eine intuitive Programmierung unterstützen. In [BCB01] wird eine graphische Oberfläche für die Programmierung von CPL-*Skripts* entwickelt, die sich auf die SPL übertragen läßt. Genereller Vorteil bei XML-basierten Dienstlogikprogrammen ist die Tatsache, daß sich die erzeugten Skripten über die DTD verifizieren lassen. Bei beiden Möglichkeiten ist es unerheblich, wo die Programmierung stattfindet. Die Dienstlogikprogramme können z.B. auf externen Arbeitsstationen entwickelt werden und dann auf die SAMSON-Dienststeuerung übertragen werden. Dort werden sie in einer Datenbank gespeichert und bei Bedarf ausgelesen. Eine Versionsverwaltung ist sicherzustellen.

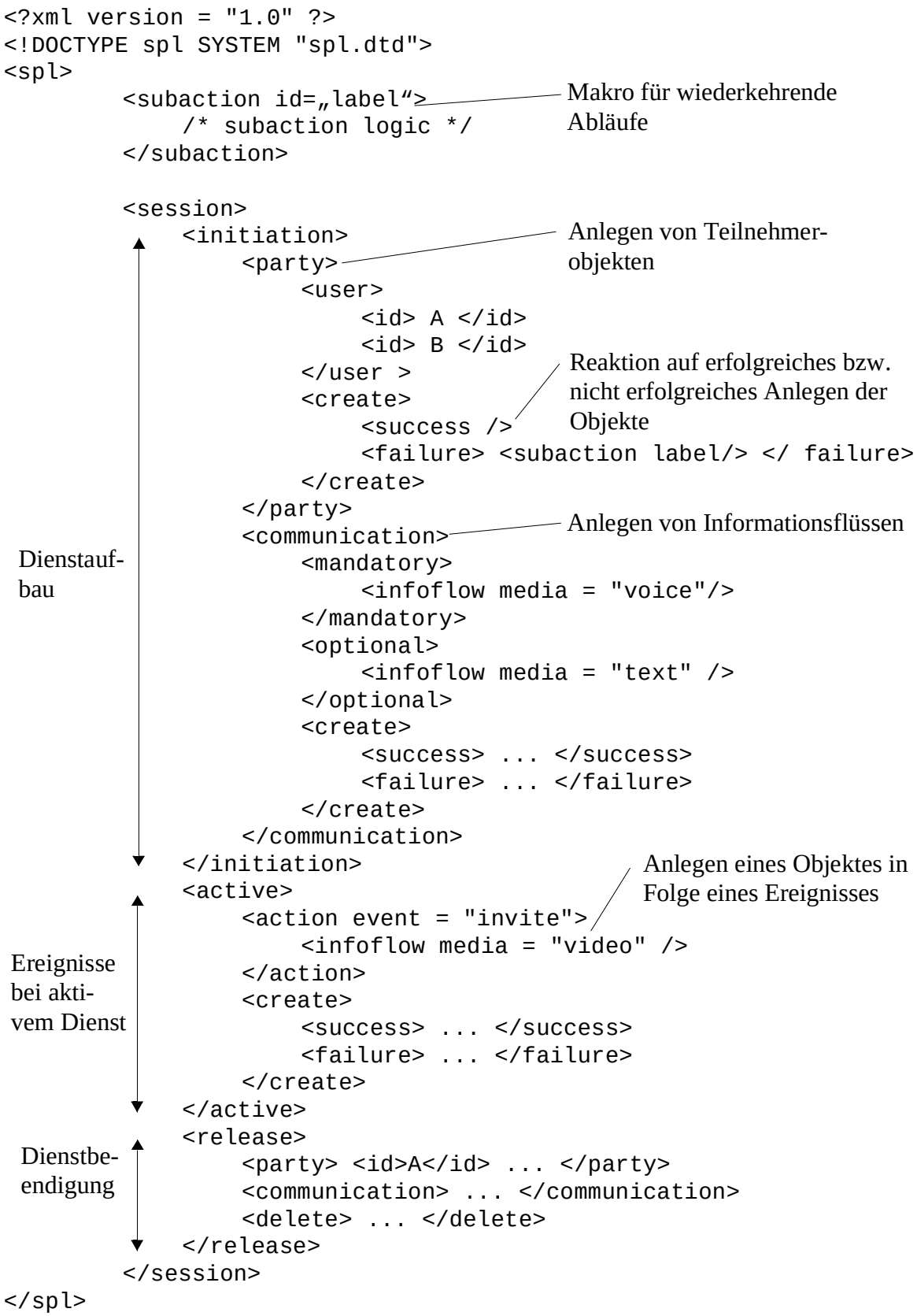


Abbildung 5.10: Beispiel für die Session Programming Language (SPL)

5.3.3 Teilnehmer-Dienst-Interaktion

Neben Registrierung und Dienstzugang kann der Teilnehmer auch zur Laufzeit eines Dienstes auf diesen einwirken. Die Interaktionen der Teilnehmer können dabei vom Teilnehmer selbst ausgelöst werden oder vom Dienst angefordert werden.

Teilnehmer-initiierte Interaktion

Ein Teilnehmer kann über einen *Protocol Agent Client* aus einer aktiven *Access Session* mit der Meldung `INVITE` nicht nur einen Dienst starten, sondern diesen mit folgenden `INVITEs` auch verändern. Im SSM funktioniert die Dienstlogik als eine Art Filter, um aus den Angaben des Teilnehmers im Message-Body die entsprechenden Änderungen am Dienst herauszulesen. Ein Teilnehmer kann z.B. neue Teilnehmer einladen oder zusätzliche Dienstmerkmale wie *Suspend* anfordern.

Teilnehmer, die nicht die Initiatoren eines Dienstes sind, die also keine *Access Session* aufgebaut haben, können über ihre *Access Session* ebenfalls Eingaben an den laufenden Dienst machen. Derselbe Mechanismus wird verwendet, wenn ein Teilnehmer während einer Dienstausführung seinen Dienstzugang wechselt (siehe Diskussion der Dienste-Mobilität in Abschnitt 4.4.2). Abbildung 5.11 stellt letzteren Fall dar: Ein Teilnehmer läßt während seiner Teilnahme an einem Dienst mit `INVITE` seine Teilnahme ruhen (Zusatzdienst *Suspend*), wechselt seinen Netzzugang (PAC1 nach PAC2) und greift dann über eine neue *Access Session* wieder über seinen UP auf denselben Dienst zu (*Resume*). Die Meldungen zur Änderung von Kommunikationsverbindungen sind nicht dargestellt.

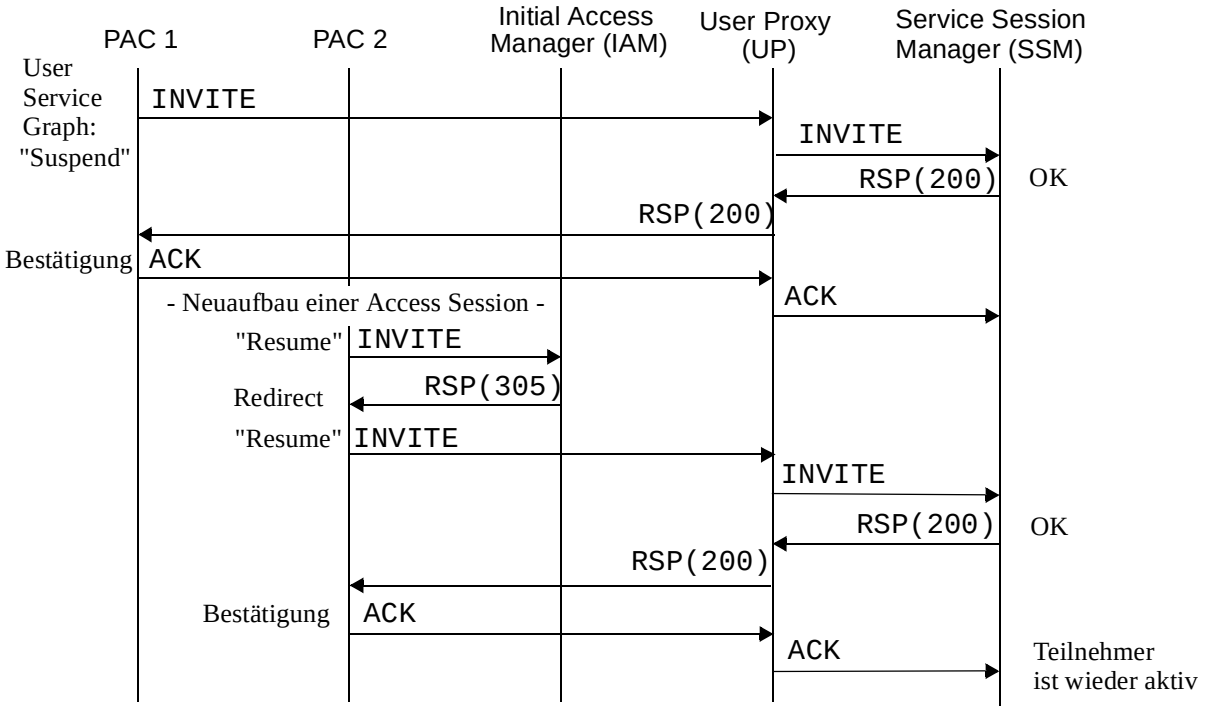


Abbildung 5.11: Signalablauf für eine Änderung der laufenden Dienstkonfiguration mit Suspend und Resume (Dienst-Mobilität)

Dienst-initiierte Interaktion

Ein laufender Dienst kann mit der Meldung `INFO` über einen entsprechenden *User Proxy* Anfragen an einen Teilnehmer stellen. Die `INFO`-Meldung wird über den UP an den *Protocol Agent* geleitet, der sie dem Teilnehmer entsprechend der vorliegenden Signalisierung anzeigt.

Außergewöhnlich an der vorliegenden Architektur ist der UP, der in vielen Fällen die `INFO`-Meldungen bereits aus dem Teilnehmerprofil heraus beantworten kann, z.B. Präferenzen.

5.3.4 Interaktionen mit Datenservern

Eine Besonderheit der SAMSON-Architektur stellen die sogenannten Content-Ressourcen dar. In vielen Diensten, insbesondere in Informationsdiensten, wird der Inhalt nicht von den Teilnehmern selbst erzeugt (z.B. Telefondialog), sondern von Media-Servern (z.B. Video Server, Ansageeinheit) im Netz oder außerhalb des Netzes bereitgestellt. Diese Ressourcen werden in der Architektur durch die Klasse der Content-Ressourcen (CtRes) repräsentiert. Content-Ressourcen können direkt von der Dienstinstanz (SSM) aus gesteuert werden. Dafür wird die Meldung `INVITE` mit einem entsprechenden *User Service Graph* im Message-Body verwendet, um z.B. Ressourcen für einen Videoabruf freizuschalten oder Inhalte auszuwählen. Die Einrichtung von Informationsflüssen zwischen Content-Ressourcen und Teilnehmerendgeräten erfolgt separat über die Kommunikationssteuerung, in der die Content-Ressourcen wie jeder andere Endpunkt behandelt werden. Die interne Adresse einer Content-Ressource wird einem SSM entweder vom Teilnehmer in einem `INVITE` mitgeteilt oder sie ist in der Dienstlogik vorkonfiguriert.

Eine separate Modellierung einer Content-Ressource ist in SAMSON insbesondere notwendig, um *Multicast-* oder *Broadcast-Push*-Server zu steuern, für die kein Rückkanal (Informationsfluß für einen Teilnehmer-Interaktionskanal) aufgebaut werden kann oder für Server, die eine spezielle, zentralisierte Authentisierung erfordern (z.B. Spiele-Server). Ansonsten kann ein Media-Server natürlich auch wie ein normaler Teilnehmer eingebunden werden, wenn keine zusätzliche Freischaltung notwendig ist.

5.4 Zusatzdienste

Der Schwerpunkt bei SAMSON liegt auf personalisierbaren, erweiterten Diensten (siehe Abschnitt 4.1.3). Die SAMSON-Architektur stellt darüber hinaus Ansatzpunkte zur Steuerung von Zusatzdiensten bereit.

> Unter dem Begriff **Zusatzdienste** werden Dienstmerkmale verstanden, die eine zusätzliche Funktionalität für Dienste bereitstellen, aber eigenständig nicht aufgerufen werden können. Zusatzdienste sind meist so angelegt, daß sie für verschiedene Dienste nutzbar sind, z.B. Anrufschutz, Rückruf bei belegt, Rufumleitung.

Man unterscheidet Zusatzdienste in erster Linie nach dem Zeitpunkt des Aufrufs in einer der drei Dienstphasen „Dienstaufbau“, „aktiver Dienst“ und „Dienstabbau“. Zusatzdienste für die Aktiv-Phase und den Dienstabbau werden in SAMSON zentral in der Dienstlogik des *Service Session Managers* spezifiziert, z.B. Halten, Rückfrage. Diese können je nachdem, wie die Rechte gesetzt sind, sowohl für den Initiator als auch für andere Teilnehmer aufrufbar sein bzw. durch Ereignisse ausgelöst werden.

Zusatzdienste der Dienstaufbauphase lassen sich in die zwei Kategorien „Ausgehend“ (*Outgoing*) und „Ankommend“ (*Incoming*) einteilen. Zur Klasse „Ausgehend“ zählen alle Zusatzdienste, die bei dem vom Teilnehmer selbst-initiierten Start eines Dienstes aufgerufen werden, z.B. Ausgehende Rufsperre, Wahlwiederholung oder Kurzwahl. Beim Teilnehmer eintreffende Anrufe können Zusatzdienste vom Typ „Ankommend“ auslösen, z.B. Rufumleitung oder Anrufschutz. Ausgehende Zusatzdienste lassen sich wie die anderen Zusatzdienste in der Dienstlogik des *Service Session Managers* spezifizieren. Die persönlichen Parameter eines Teilnehmers werden der Dienstinstanz beim Dienstaufruf im `INVITE` mitgeteilt.

Für die Realisierung von ankommenden Zusatzdiensten bietet SAMSON die Möglichkeit, diese im entsprechenden User Proxy zu spezifizieren. Adreß-basierte Zusatzdienste, z.B. eine zeitabhängige Rufumleitung, können aktiv werden, wenn der CSM die aktuelle Adresse des Teilnehmers mit `INFO` abfragt.

Eine eingehende Beschreibung der Realisierung von Zusatzdiensten in SAMSON geht über den Rahmen der vorliegenden Arbeit hinaus. Ansätze, die speziell auf die SIP-basierte Signalisierung abgestimmt sind, können den Drafts der IETF-Working-Group IPTEL[1] entnommen werden.

5.5 Kommunikationssteuerung

Der Steuerungsbereich *Communication Control* in SAMSON stellt das Bindeglied zu den Netzen dar. Es ist Aufgabe der Kommunikationssteuerung, auf Anforderung der Dienststeuerung die konkrete Einrichtung einer Kommunikationsbeziehung in den angeschlossenen Netzen vorzubereiten und durchzuführen. Dafür sind die erforderlichen Informationen zu vervollständigen und geeignete Netze auszuwählen. Die Kommunikationssteuerung unterstützt nicht nur eine netzübergreifende Steuerung von Diensten, sondern ist selbst unabhängig von bestimmten Netztypen oder Signalisierungsverfahren.

Um den Funktionsablauf der Kommunikationssteuerung einzuordnen, werden zunächst die generellen Strategien für die Einrichtung von Kommunikationsbeziehungen in den Netzen betrachtet und den Möglichkeiten, die in SAMSON gegeben sind, gegenübergestellt. Die verschiedenen Strategien unterscheiden sich durch die Reihenfolge, in der die Teilnehmer gerufen, die Endgerätekonfiguration festgestellt und die Netzressourcen belegt werden.

Bei der **Postallokation** werden zuerst die Teilnehmer gerufen und gleichzeitig deren Endgerätekonfiguration abgefragt. Erst wenn alle Teilnehmer den Ruf annehmen, werden die entsprechenden Ressourcen belegt. Umgekehrt beschreibt die **Präallokation** ein Verfahren, bei dem nach einer Endgeräteabfrage zuerst Ressourcen belegt werden und danach die Teilnehmer gerufen werden. Im ersteren Fall werden Ressourcen nicht unnötig belegt, aber es kann der Fall auftreten, daß Ressourcen bei Rufannahme nicht mehr verfügbar sind. Die Präallokation wird bei den meisten traditionellen Signalisierungsverfahren, z.B. beim ISDN, eingesetzt. Sie beinhaltet den Nachteil, daß Ressourcen unnötigerweise belegt sind, wenn der gerufene Teilnehmer nicht erreichbar ist.

In beiden Fällen wird davon ausgegangen, daß die aktuelle Verfügbarkeit der Teilnehmer (Rufannahme), die Konfiguration der Teilnehmerendgeräte und die Belegung von Ressourcen unabhängig voneinander überprüft werden können. Da SAMSON auf bestehende, heterogene Kommunikationsnetze aufsetzt, kann die Reihenfolge der Signalisierung in den Netzen nicht beeinflußt werden. Ein separater Signalisierungskanal zu den Teilnehmerendgeräten ist nicht vorhanden.

In SAMSON wird daher folgende zweistufige Strategie gewählt:

Die Teilnehmerendgeräte können zwar nicht direkt nach ihrem aktuellen Status abgefragt werden, aber die relevanten Informationen (z.B. Endgeräteeigenschaften, Kodierung) werden in der Teilnehmerverwaltung vorgehalten und können abgerufen werden, wenn die Dienststeuerung die Einrichtung einer Kommunikationsbeziehung anfordert. Dadurch können im voraus Unterschiede abgeklärt werden und die richtigen Endgeräteadressen (IP-Adresse/Port, ISDN-Nummer) festgelegt werden.

1. HTTP://www.ietf.org/html.charters/iptel-charter.html oder HTTP://www.bell-labs.org/mailing-lists/iptel/

Bei der Anforderung von Verbindungen durch einen Adaptor hängt die Art der Verbindungseinrichtung vom Netztyp ab. Beim ISDN beispielsweise wird wie bei der Präallokation vorgegangen. In anderen Systemen werden zunächst die Teilnehmer gerufen und dann die Ressourcen belegt.

5.5.1 Aufgaben der Kommunikationssteuerung

Die Kommunikationssteuerung stellt der Dienststeuerung eine einheitliche Schnittstelle bereit, an der diese die Einrichtung von Verbindungen für eine Kommunikationsbeziehung (beschrieben im *Media Connection Graph*) anfordern kann. Damit entsprechen die Aufgaben der Kommunikationssteuerung in etwa den Aufgaben der Rufebene und der Ressourcenebene in anderen modularen Signalisierungsarchitekturen, z.B. in [Mül96, Que00]. Unterschiede bestehen in der Sicht auf das Kommunikationsnetz. Während sich [Mül96] und [Que00] auf eine einheitliche, ATM-basierte Netzinfrastruktur fokussieren und darin möglichst optimale Verbindungen schalten, unterstützt SAMSON Verbindungen in unterschiedlichen Kommunikationsnetzen. Der Schwerpunkt der SAMSON-Kommunikationssteuerung liegt daher auf der Auswahl von Netzressourcen und der Bestimmung der Quell- und Zieladressen. Die Wegeführung in einem Teilabschnitt wird der Verbindungssteuerung in den einzelnen Netzen überlassen.

In SAMSON wird jedes Kommunikationsnetz unterschiedlichen Typs oder unterschiedlicher Betreiber durch einen **Netzadaptor** (NA) repräsentiert, der einen Verbindungsaufbau durch Dritte unterstützt. Das Netzmodell ist so angelegt, daß nicht nur der Aufbau einfacher Punkt-zu-Punkt Verbindungen, sondern möglichst viele Funktionen (z.B. Konferenzaufbau) der zum Teil mächtigen Schnittstellen, die von den Netzen angeboten werden, verwendet werden können. Gateways, die zwei Netze verbinden, werden ebenfalls durch ihre Netzadaptoren in SAMSON repräsentiert. Zusätzlich wird die Existenz von **Spezialressourcen** (z.B. Kodierungs-Konverter oder Konferenzbrücken) angenommen, die ebenfalls über Adaptoren, sogenannte Spezialressourcen-Adaptoren (SpResA), steuerbar sind.

In der Kommunikationssteuerung wird die Beschreibung einer Kommunikationsbeziehung, deren Einrichtung von der Dienststeuerung angefordert wird, in mehreren Schritten konkretisiert, bis die entsprechenden Verbindungen über die Adaptoren in den Netzen geschaltet werden können. Folgende Schritte und Teilaufgaben sind dabei durch den *Communication Session Manager* auszuführen, wenn eine Anforderung (`SETUP`) mit einem *Media Connection Graph* vom *Service Session Manager* eintrifft:

1. Abfrage der konkreten Teilnehmerprofile
   - Adressumsetzung, d.h. Ermittlung der aktuellen Teilnehmeradresse(n)
   - Abfrage der aktuellen Endgerätekonfiguration
2. Abstimmung der Teilnehmerprofile zur Auswahl von Netzen
3. Unterteilung der Informationsflüsse in Informationspfade durch Einfügen von Platzhaltern für Gateways und Spezialressourcen
4. Abbildung der Dienst-Qualitätsmerkmale auf die Transportdienstgüte für jeden Informationspfad

5. Auswahl konkreter Netzressourcen für Informationspfade sowie von Gateways und Spezialressourcen
   - Prüfung der Verfügbarkeit
   - Auswahl der Günstigsten bei Alternativen
   - Integration weiterer Spezialressourcen und weitere Unterteilung in Pfade bei Nichtverfügbarkeit
6. Bildung der *Connectivity Connection Graphs*
7. Anforderung der Netzressourcen

Bei Unvereinbarkeit von Parametern oder bei Nichtverfügbarkeit von Ressourcen können einzelne Schritte rückgängig gemacht und nochmals durchlaufen werden. Gegebenenfalls ist eine neue, alternative Anfrage zu stellen.

Im Folgenden werden die einzelnen Schritte detailliert beschrieben. Der *Media Connection Graph* sowie die einzelnen Zwischenschritte der damit angeforderten Kommunikationsbeziehung werden graphisch dargestellt. Als Beispiel dient die Dienstkonfiguration des Szenarios aus Kapitel 4. Abbildung 5.18 faßt die Aufgaben des *Communication Session Managers* abschließend in einem Flußdiagramm zusammen.

5.5.2 Anforderung durch die Dienststeuerung

Der *Service Session Manager* fordert zu Beginn jedes Dienstes und für jede Erweiterung eines Dienstes um Informationsflüsse mit der SesCP-Meldung `SETUP` die Einrichtung einer Kommunikationsbeziehung an. Die zentrale Komponente der Kommunikationssteuerung ist der *Communication Session Manager* (CSM), der für jede *Communication Session* instanziert wird.

Wie bereits in 5.3.1 beschrieben, fordert der Client-Prozeß des *Service Session Managers* mit der Meldung `SETUP` und dem *Media Connection Graph* im Message-Body (siehe Abbildung 5.12) die Einrichtung einer Kommunikationsbeziehung von der Kommunikationssteuerung an. Das erste `SETUP` einer *Service Session* wird dabei über den *Initial Communication Manager* (Redirect-Server) an die *Alias*-Adresse *adaptor* gerichtet („To:“- und „Request-URI:“-Header-Feld). Alle weiteren `SETUP`-Meldungen werden über den instanzierten CSM geroutet, dessen

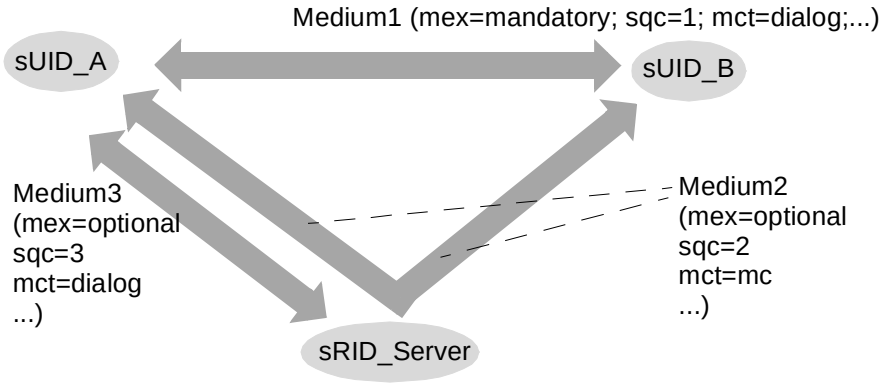


Abbildung 5.12: (Graphische) Beschreibung einer Kommunikationsbeziehung durch den Media Connection Graph

Aufgabe es u.a. ist, den *Alias* aufzulösen und die Meldung an den richtigen bzw. die richtigen Netzadaptoren weiterzuleiten. Wie auch die `INVITE`-Meldung wird `SETUP` im Erfolgsfall positiv beantwortet und mit `ACK` bestätigt.

`SETUP` dient sowohl zum ersten Aufbau einer Kommunikationsbeziehung als auch zur Anforderung von Erweiterungen mit nachfolgenden `SETUP`-Meldungen. In Erweiterungen sind neue Informationsflüsse durch neue Mediennummern gekennzeichnet, so daß nur dieser Teil der Kommunikationsbeziehung neu hinzugenommen werden muß, nicht aber alles neu aufgebaut wird. Die Löschung eines Informationsflusses erfolgt mit `BYE`, das in diesem Fall den zu löschenden Informationsfluß im Message-Body enthält. Mit `CANCEL` können noch nicht bestätigte Anforderungen gelöscht werden. Dies ist notwendig, wenn z.B. einzelne Pfade eines Informationsflusses nicht eingerichtet werden können oder in der Dienstlogik Timer für den Verbindungsaufbau gesetzt sind.

Die Reaktion auf den Status-Code 380 (*Alternative Service)* und 503 (*Service Unavailable)* stellen zwei wichtige Fälle der Sonderfallbearbeitung durch den *Service Session Manager* dar. Ersterer zeigt an, daß die Anfrage (z.B. `SETUP`) nicht erfolgreich war und mit geänderter Session-Beschreibung wiederholt werden muß. Dem SSM wird dabei Gelegenheit gegeben, aufgrund des Vorschlags in der Antwort eine neue Anfrage zu stellen. Die Bearbeitung der 380-Antwort muß entsprechend in der Dienstlogik definiert werden. 503 meldet einen vollständigen Fehlschlag und führt zum Abbruch des Dienstes. Belegte Endteilnehmer können z.B. mit 486 *Busy Here* angezeigt werden. In vielen Fällen kann der CSM bereits die Sonderfälle bearbeiten, so daß nicht jedesmal bis zum SSM signalisiert werden muß.

5.5.3 Abfrage und Abstimmung der Teilnehmerprofile

Nach Eintreffen der `SETUP`-Meldung wird der *Communication Session Manager* aktiv:

Schritt 1: Abfrage

Zur Vervollständigung der Information, die für die Auswahl der Netzressourcen benötigt wird, ist es die Aufgabe des *Communication Session Managers*, den *Media Connection Graph*, den er vom SSM erhalten hat, zu detaillieren. Dazu fordert er mit der Meldung `INFO` die Profile aller Dienst-Teilnehmer, von denen bisher nur die sUID bekannt sind, von den entsprechenden *User Proxies* an (siehe Abbildung 5.13). Es werden sowohl die aktuellen Teilnehmeradressen, d.h. die Netze, über die der Teilnehmer erreichbar ist, als auch die Endgerätekonfiguration aller Teilnehmer ermittelt. Auch für externe Teilnehmer, die nicht in SAMSON bekannt sind, werden diese Informationen aus den entsprechenden *User Proxies* gewonnen. Der Dienst-initiierende Teilnehmer hat beim Dienstaufruf die Adreß-Angaben mitgeschickt. Diese wurden durch die *Service Session* in einem *Non Subscribed User Proxy* gespeichert.

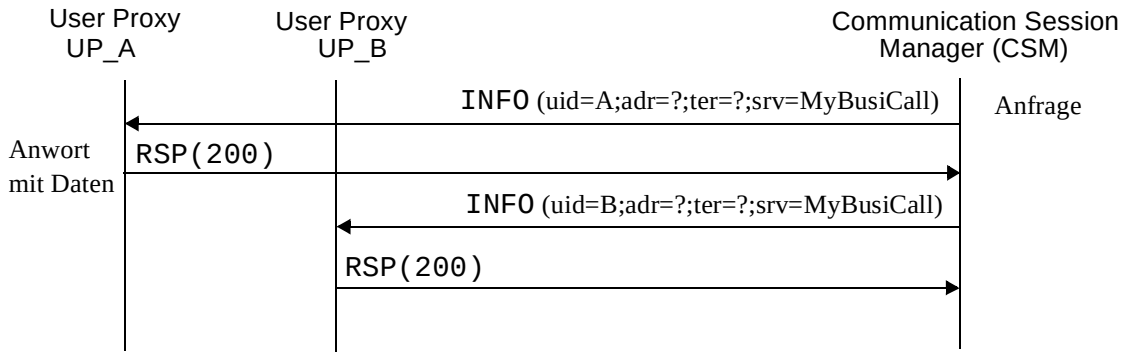


Abbildung 5.13: Meldungsablauf zur Abfrage der Teilnehmerprofile

Die Adresse von Ressourcen wie z.B. Videoservern, die in SAMSON über die sRID referenziert werden, wird entweder bereits vom SSM mitgeliefert oder aus einer zusätzlichen Dienst-Ressourcen-Datenbank in der Kommunikationssteuerung gewonnen.

Schritt 2: Abstimmung der Teilnehmerprofile zur Netzwahl

Der zweite Schritt besteht aus dem Abgleich der Teilnehmerprofile. Nacheinander wird die Vereinbarkeit der Teilnehmeranschlüsse für die aufzubauende Kommunikationsbeziehung im Dienst ausgehend vom *Media Connection Graph* und dem Teilnehmerprofil des Dienstinitiators überprüft. Es werden zu den Endgeräteanschlüssen passende Netze ausgewählt und die zugehörigen Netzadressen der Endgeräte gespeichert. Damit sind die entsprechenden Netze festgelegt. Die Auswahl richtet sich dabei nach Anforderungen, die vom Dienst her gesetzt werden, und nach dem Ziel, unter allen Teilnehmern einheitlich zu sein. Die Auswahl der Netze und Endgeräteanschlüsse richtet sich in erster Linien danach, ob die gewünschten Medien mit der definierten Dienst-Qualitäts-Klasse (SQ) unterstützt werden. Zusätzlich wird darauf geachtet, daß einheitliche Codecs und einheitliche Netzprotokolle oder zumindest zueinander ähnliche unterstützt werden. Codecs und Netzprotokolle gelten als ähnlich, wenn eine Konversion bzw. ein Übergang einfach zu realisieren ist (z.B. H.323 und ISDN).

Es entsteht eine implizite Beschreibung der gewünschten Kommunikationsbeziehung, wie in Abbildung 5.14 graphisch dargestellt. Sie kann die Verwendung von Gateways für die Überbrückung von Netzübergängen und Spezialressourcen für die Konvertierung von Kodierungen beinhalten.

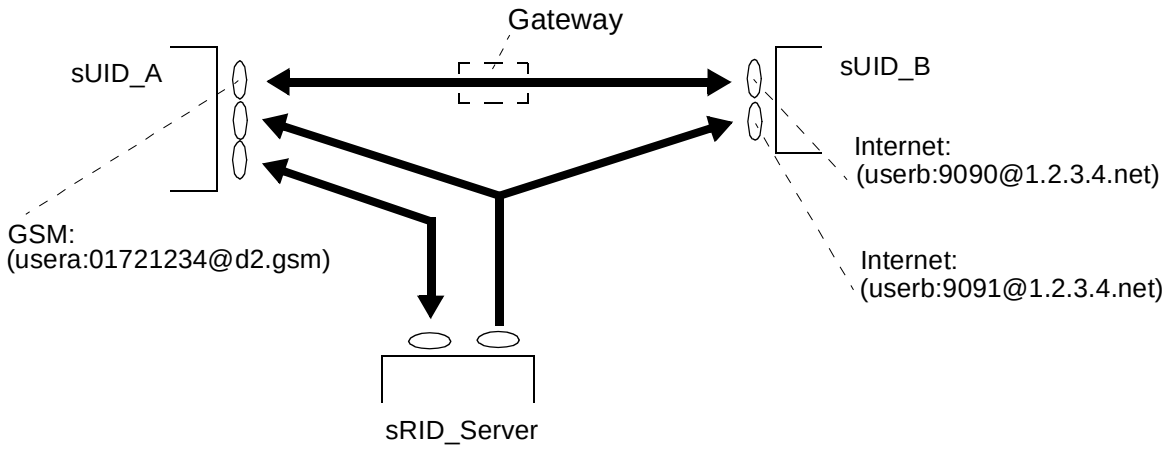


Abbildung 5.14: Implizite Beschreibung einer Kommunikationsbeziehung

Schritt 3: Erweiterung um Gateways und Spezialressourcen

Vor der Auswahl der Netzressourcen werden in einem dritten Schritt notwendige Gateways und Spezialressourcen explizit eingefügt. Gateways werden an allen Stellen benötigt, wo die Anschlüsse kommunizierender Teilnehmer in unterschiedlichen Netzen liegen. Als Spezialressourcen werden in diesem Schritt Konverter gesetzt, um unterschiedliche Kodierverfahren der Endgeräte auszugleichen. Ein Beispiel für die entstehende explizite Beschreibung ist in Abbildung 5.15 dargestellt. Die Informationsflüsse sind darin in Informationspfade unterteilt.

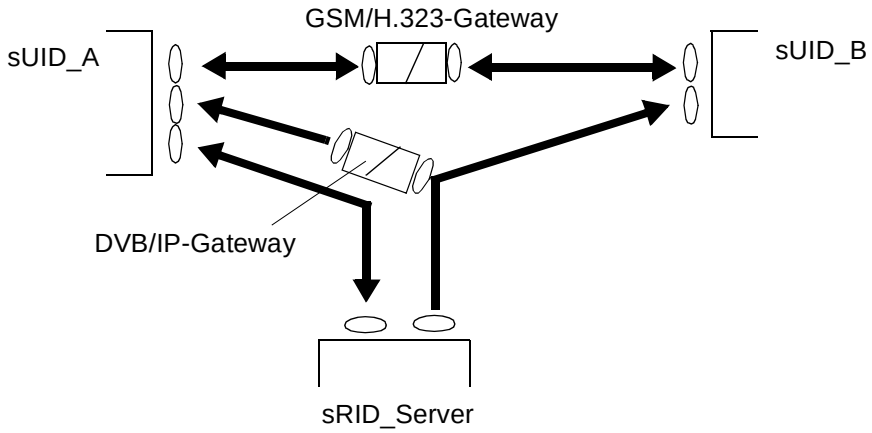


Abbildung 5.15: Explizite Beschreibung einer Kommunikationsbeziehung

5.5.4 Abbildung der Informationspfade auf Netzressourcen

Steht eine Konfiguration fest, dann werden in den folgenden Schritten für jeden Informationspfad die Netzressourcen ermittelt und belegt.

Schritt 4: Ermittlung der geforderten Transportdienstgüte

Im vierten Schritt werden die abstrakten Qualitätsmerkmale der Dienstebene (Teilnehmerdienstgüte: OvQ; Übertragungsqualität: SQ) auf eine Beschreibung der Transportdienstgüte abgebildet. Jene dient dazu, Netzressourcen auszuwählen. Tabelle 5.6 zeigt den Zusammenhang der Teilnehmer-Qualitäts-Klassen, der Dienst-Qualitäts-Klassen (Tabellen 4.2 und 4.3) und der Transportdienstgüte für Sprachdienste. Die Werte wurden in Anlehnung an die Qualitätsparameter des ETSI-Projektes TIPHON [ETS00] gewählt.

Medientyp: "Voice"				
Service-Quality	**Transport-dienstgüte**	**OvQ=1 (high)**	**OvQ=2 (regular)**	**OvQ=3 (no care)**
SQ=1 (Realtime)	Bitrate[a]	> R1	> R2	-
	Paketverlust	< 0,5 %	< 1 %	< 2 %
	EtE-Delay[b]	< 100 ms	< 150 ms	< 400 ms[c]
	Jitter	< 10 ms	< 20 ms	< 40 ms
SQ=2	...	...	...	...

a. Abhängig vom Codec; Es gilt: R1 > R2

b. Ein *End-to-End-Delay* setzt sich aus der Verzögerung in den Endgeräten (Kodierung/Dekodierung, Paketierung, etc.), und der Verzögerung der Übertragungsstrecke zusammen, vgl. [ETS00].

c. Als Richtwert

Tabelle 5.6: Beispiel für die Ermittlung der Transportdienstgüte für Sprachdienste

Die Bitrate kann nur als Richtwert angegeben werden, da sie von der endgültigen Wahl der Codecs abhängt. Diese werden normalerweise erst beim Verbindungsaufbau zwischen den Endgeräten ausgehandelt. Für eine Abschätzung der Bitrate zur Auswahl von Netzressourcen wird vorausschauend die Bitrate angenommen, die sich für den qualitativ besten Codec (höchste Bitrate) ergeben kann.

Diese vorausschauende Abschätzung bezüglich der Kodierformate war auch bereits die Grundlage für die Einfügung von Konvertern als Spezialressourcen in Schritt 3.

Schritt 5: Auswahl der Netzressourcen

Schritt fünf wählt nun geeignete Netzressourcen anhand der in Schritt vier ermittelten Parameter aus. Als Netzressource wird ein Informationsfluß und zugehörige Netzknoten bezeichnet, dessen Kommunikationsverbindung durch einen Adaptor gesteuert werden kann. Außerdem werden Gateways und Spezialressourcen zu den Netzressourcen gezählt. Die Auswahl erfolgt über die Anfrage bei einer weiteren Komponente der Serverarchitektur, die die Aufgabe einer zentralen Registration für alle verfügbaren Netzressourcen übernimmt (siehe Abschnitt 6.4). Diese Komponente liefert bei Übereinstimmung der Parameter und bei Verfügbarkeit die Beschreibung einer geeigneten Netzressource für jeden Informationspfad.

Bei Alternativen werden die günstigsten Netzressourcen ausgewählt. Ein Kostenparameter gibt die Kosten an, die für einen Verbindungsaufbau und für den Nutzdatentransfer entstehen. Die einzelnen Parameter werden bei der Beschreibung der Adaptoren in Kapitel 6 näher erläutert.

Der **zentrale Ansatz** für die Ressourcen-Datenbank hat sich bei SAMSON als sinnvoll erwiesen, da die Serverarchitektur als zentraler Server konzipiert ist, so daß aufwendige Signalisierungs- und Koordinationsverfahren entfallen. Außerdem sind mit den *Service Discovery*-Mechanismen bereits Verfahren und Systeme standardisiert, die eine zentrale Datenbank unterstützen. Bei einem hochskalierbaren System kann die Konzentration auf eine Datenbank zu einem Performance-Problem führen. Für kleine Lösungen mit einer überschaubaren Teilnehmeranzahl, wie bei SAMSON angenommen, stellt die zentrale Datenbank die beste Lösung dar. Im Hinblick auf die Skalierbarkeit können Konzepte der verteilten Datenbanken verwendet werden, für die zahlreiche, kommerzielle Lösungen erhältlich sind.

Wird für die Einrichtung eines komplexen Informationspfades, wie z.B. einer Konferenzschaltung oder einer Point-to-Multipoint-Verbindung, kein Netzadaptor gefunden, der diesen als Ganzes etablieren kann, so wird der Informationspfad in weitere Pfade unterteilt, die durch Spezialressourcen, wie z.B. Konferenzbrücken verbunden werden. Für diese Teilpfade wird erneut bei der Ressourcendatenbank um Netzressourcen angefragt.

Werden keine geeigneten Netzressourcen gefunden, so wird auf Schritt 2 zurückgegangen und versucht, durch alternative Netzanschlüsse eine geänderte Konfiguration zu finden. Dabei kann man so weit gehen, optionale Informationsflüsse oder andere Objekte im *Media Connection Graph* wegzulassen. Falls keine einheitliche Konfiguration erzielt werden kann, wird dem SSM die bestmögliche Alternative mitgeteilt.

Schritt 6: Bildung der Teilgraphen

Sind für alle Informationspfade geeignete Netzressourcen und Spezialressourcen gefunden und die Adressen der zugehörigen Adaptoren dem CSM bekannt, so existiert eine konkrete Beschreibung der angeforderten Kommunikationsbeziehung (siehe Abbildung 5.16). Nun ist für jeden Informationspfad oder jeden Netzübergang ein zuständiger Adaptor festgelegt. Im sechsten Schritt folgt dann die Abbildung dieser globalen Beschreibung in einzelne *Communi-*

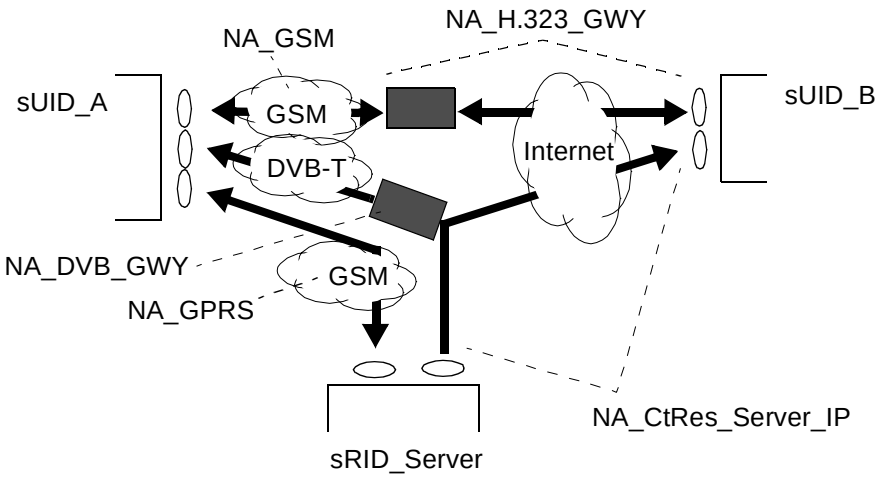


Abbildung 5.16: Konkrete Beschreibung einer Kommunikationsbeziehung

cation Connection Graphs (CCG). Für jeden Adaptor wird ein CCG ausgebildet. In Anhang D wird ein Beispiel für einen in SDP+ beschriebenen CCG gegeben (Meldung 25).

5.5.5 Signalisierung zur Einrichtung von Verbindungen

Schritt 7: Anforderung von Netzressourcen

Wenn die Ressourcenauswahl abgeschlossen ist, wird im letzten Schnitt die Einrichtung der einzelnen *Connectivity Connection Graphs* von den Adaptoren angefordert.

Dazu schickt die Client-Seite des als Proxy-Server arbeitenden *Communication Session Managers* die vom SSM erhaltene `SETUP`-Meldung mit folgenden Veränderungen weiter (siehe Abbildung 5.17).

- Die *Alias*-Adresse *adaptor* im "Request-URI:"-Header-Feld wird durch die tatsächliche Adresse des Adaptors ersetzt.
- Zusätzlich wird im "To:"-Header-Feld der "Tag"-Parameter gesetzt, um verschiedene Requests zu verschiedenen Adaptoren unterscheiden zu können.
- Im Message-Body wird der *Media Connection Graph* durch den entsprechenden Teilgraph (*Connectivity Connection Graph*) ersetzt.
- Durch Einfügen des Header-Feldes "Route" wird sichergestellt, daß alle Antworten über den CSM laufen.

Unter Benutzung des SIP-*Forking*-Mechanismus werden die Anfragen (`SETUP`) mit den entsprechenden CCGs an alle ausgewählten Adaptoren gleichzeitig versandt. Der Signalisierungsablauf entspricht dabei den folgenden drei Schritten: Anfrage mit `SETUP`, positive Antwort im Erfolgsfall, d.h. bei Verfügbarkeit, und endgültige Aktivierung mit `ACK`. Die Verfügbarkeitsprüfung umfaßt dabei nicht nur die Ressourcen, die der Netzadaptor direkt kontrolliert, sondern insbesondere das Teilnehmerendgerät und den Teilnehmer selbst. So kann der Fall „belegt“ oder „Teilnehmer hebt nicht ab“ als Nichtverfügbarkeit mit entsprechendem Status-Code gemeldet werden. Wie oben bereits diskutiert, werden dabei je nach Signalisierungsverfahren, das der Adaptor in seinem Netz unterstützt, Ressourcen möglicherweise unnötig reserviert oder belegt.

In Kombination mit dem *Forking*-Mechanismus wird dieser dreiphasige Ablauf dazu verwendet, die Adaptoren im Sinne einer *Atomic Action* zu koordinieren. Das heißt, nur wenn alle Adaptoren tatsächlich bereit sind, den jeweils gewünschten *Connectivity Connection Graph* aufzubauen, wird die Kommunikationsbeziehung dem Initiator bestätigt und kann als Ganzes eingerichtet werden. Ein Beispiel wird in Abbildung 5.19 dargestellt.

Fällt die Antwort eines Adaptors negativ aus, so kann bereits der CSM handeln, ohne daß ein Fehlschlagen bis an den SSM zurückgemeldet werden muß. Wird eine alternative Beschreibung vom Adaptor in der Antwort vorgeschlagen, so prüft der CSM, ob die Alternative mit dem Dienst vereinbar ist. Falls durch eine Änderung mehrere Adaptoren betroffen sind, können noch nicht bestätigte Anfragen mit `CANCEL` rückgängig gemacht werden. Abbildung 5.17 zeigt ein typisches Ablaufbeispiel. Ein Netzadaptor für GSM (NA_GSM) kann keine Verbindung zu einem ausgewählten Gateway (NA_H.323_GW1) schalten und teilt dies dem CSM mit Status-Code 380 mit. In dieser Antwort gibt er als Alternative das Gateway H.323_GW2 an. Der CSM widerruft das `SETUP` an den zuerst gewählten Gateway mit `CANCEL`, wählt den vorgeschlagenen Gateway als Alternative und signalisiert die Änderungen mit einem neuen `SETUP` an die Adaptoren.

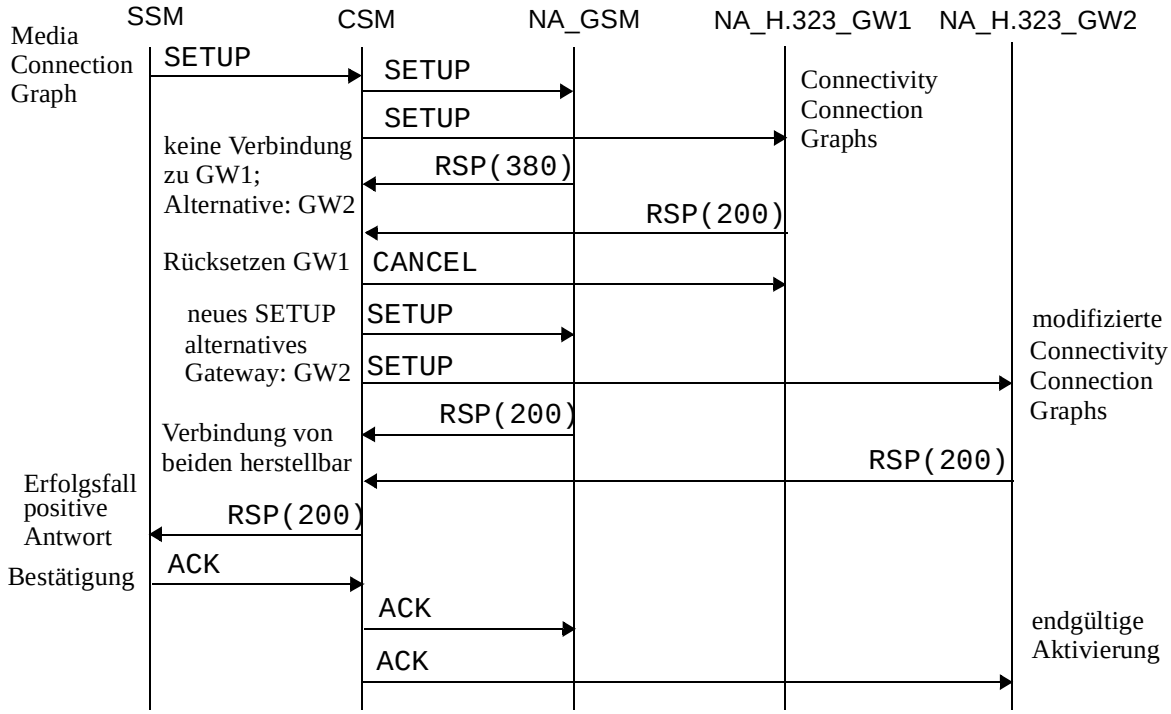


Abbildung 5.17: Signalablaufdiagramm für die Kommunikationssteuerung

In einem Adaptor wird je `SETUP`-Anfrage ein *Protocol Agent Server* (PAS) gestartet. Dieser wickelt das SesCP-Protokoll für den Adaptor ab und trifft entsprechende Maßnahmen, um die angeforderten Aktionen netzseitig auszuführen, d.h. sie in die netzspezifische Signalisierung zu übersetzen. Die Struktur und Funktionsweise der Adaptoren wird ausführlich in Kapitel 6 behandelt.

Änderung einer bestehenden Kommunikationsbeziehung

Die Kommunikationssteuerung dient nicht nur zum einmaligen Aufbau einer Kommunikationsbeziehung in den Netzen, sondern auch zu deren Änderung im Rahmen der Dienstausfüh-

rung. Dazu werden die SesCP-Meldungen `SETUP`, `BYE` und `END` verwendet. Auf das erste `SETUP` folgende `SETUP`-Meldungen enthalten Erweiterungen der bereits etablierten Kommunikationsbeziehung. Sollen einzelne Objekte (z.B. Informationsflüsse) entfernt werden, so wird dies mit `BYE` signalisiert. `END` signalisiert den Abbau der kompletten Kommunikationsbeziehung. Die Meldungen werden, wie oben im Text beschrieben, verarbeitet. Der CSM hat die Modelle der verschiedenen Schritte gespeichert und braucht lediglich Änderungen vorzunehmen. Er richtet die Anfrage nur an die Adaptoren, bei denen sich eine Änderung ergeben hat. Die **Mobilität** eines Teilnehmers während der Dienstausführung (d.h. ein Teilnehmer wechselt sein Endgerät) kann somit durch eine `BYE`-Meldung, gefolgt von einer `SETUP`-Meldung, realisiert werden (vgl. Abbildung 5.11 für die zugehörige Darstellung der Teilnehmerinteraktion).

5.5.6 Zusammenfassende Betrachtung des Communication Session Managers

Die Kommunikationssteuerung bildet eine Kommunikationsbeziehung eines Dienstes auf die heterogene Netzinfrastruktur ab. Alle dazu nötigen Aktionen sind so angelegt, daß sie unabhängig von netzspezifischen Details ausgeführt werden können. Das Kernstück der Kommunikationssteuerung ist der *Communication Session Manager*, der als Proxy-Server arbeitet und die Dienstbeschreibung vom *Media Connection Graph* auf *Connectivity Connection Graphen* konkretisiert, die im *Forking*-Verfahren auf die Adaptoren verteilt werden. Abbildung 5.18 faßt das Verhalten des CSM in einem Flußdiagramm zusammen.

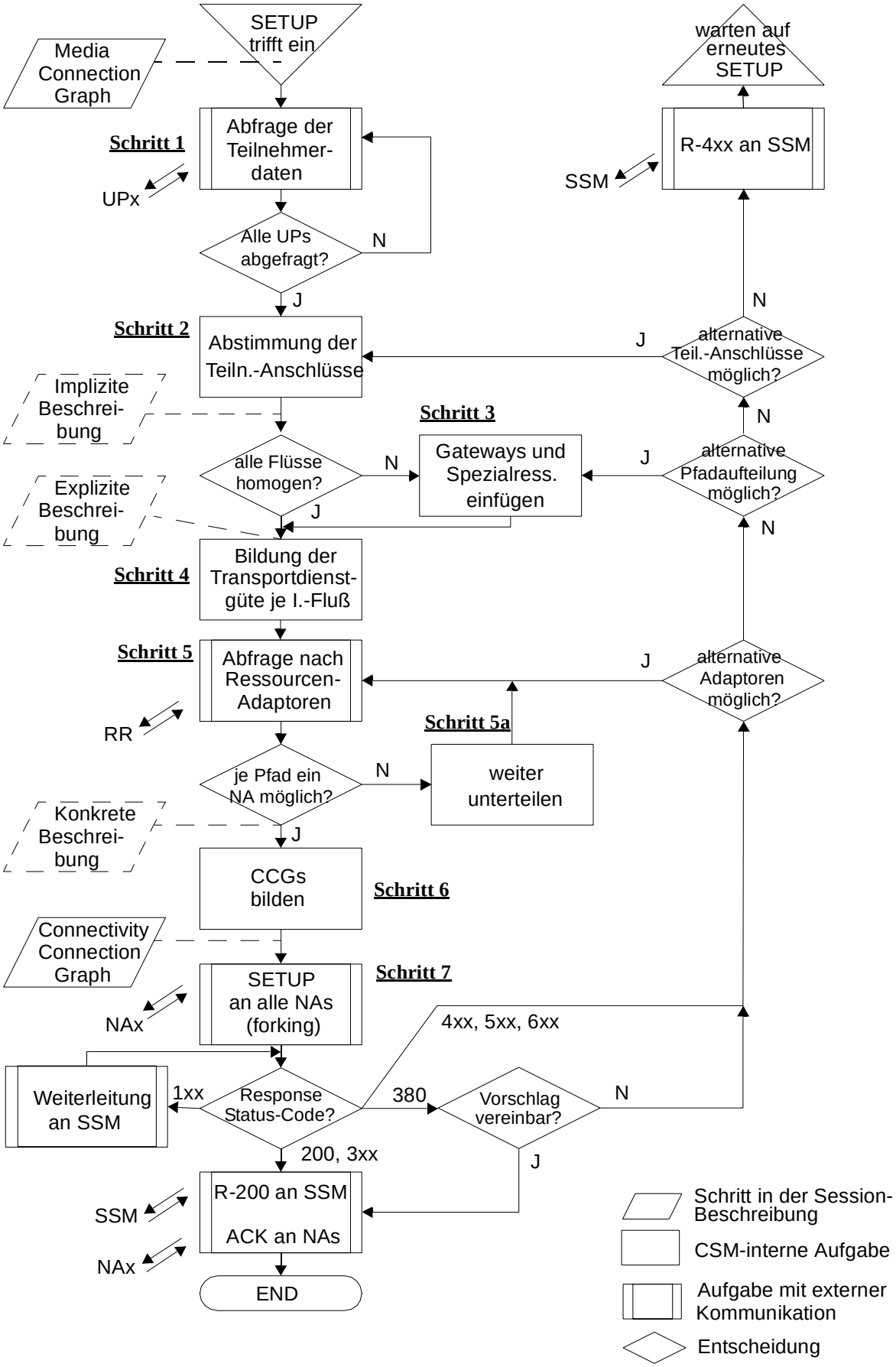


Abbildung 5.18: Verhalten des CSM

5.6 Zusammenfassendes Signalisierungsbeispiel

In diesem Kapitel wurde das Verhalten der einzelnen Komponenten der Serverarchitektur SAMSON auf der Ebene des Signalisierungsprotokolles SesCP spezifiziert. Ausgehend von der Definition und der Beschreibung des neuen Signalisierungsprotokolls auf Dienstebene (SesCP) wurden nacheinander die Komponenten der *User Session*, der *Service Session* und der *Communication Session* erläutert.

Um den Gesamtzusammenhang zu illustrieren, ist in Abbildung 5.19 der SesCP-Signalisierungsablauf des Beispielszenarios aus Kapitel 4 dargestellt, bei dem die beiden Teilnehmer A und B einen Sprachdialog und den Abruf von Multimedia-Daten anfordern. Wir nehmen an, daß die *Access Session* für Teilnehmer A bereits aufgebaut ist. Sonderfälle und Ausnahmebehandlungen sind nicht dargestellt. Die Meldungen sind nach ihrer zeitlichen Abfolge durchnumeriert. Zu den Meldungen, deren Zahlen dargestellt sind, sind im Anhang D die Protokollmeldungen mit den zugehörigen Parametern und der transportierten Session-Beschreibung abgedruckt.

Teilnehmer A ruft über seinen *Protocol Agent Client* (PAC) den Dienst „MyBusinessCall“ mit der Meldung `INVITE` auf (1). Nach Initiierung einer Dienstinstanz (SSM) erhält er eine positive Antwort (6), die er mit `ACK` bestätigt (7). Sein *User Proxy* teilt der Datenbank (UDB) mit, daß Teilnehmer A aktiv in einen Dienst involviert ist (9, 10). Bevor der SSM die Einrichtung der Kommunikationsbeziehung anstößt, startet er den *User Proxy* für Teilnehmer B (11-14).

Über den *Initial Communication Manager* wird durch das SETUP zur Kommunikationssteuerung ein *Communication Session Manager* erzeugt, der die weitere Steuerung übernimmt. Mit einer vorläufigen Meldung (18) zeigt er dem SSM an, daß der Rufaufbau in Bearbeitung ist. Zunächst werden die Teilnehmerprofile abgefragt (19-22). Im Rahmen des ablaufenden Algorithmus werden die Netzressourcen aus der *Resource Registry* ausgewählt (23, 24). Ist eine Konfiguration gefunden, so wird die `SETUP`-Meldung an die ausgewählten Netzadaptoren weitergeleitet und eine positive Antwort dem SSM mitgeteilt (25-32).

Im zweiten Teil des Diagramms (ab Schritt 33) ist dargestellt, wie Teilnehmer A im Laufe des Dienstes die Zustellung einer Präsentation anstößt. Der Ablauf ist identisch zu oben, mit dem Unterschied, daß die Einrichtung von *User Proxies* und die Abfrage der Teilnehmerprofile entfällt.

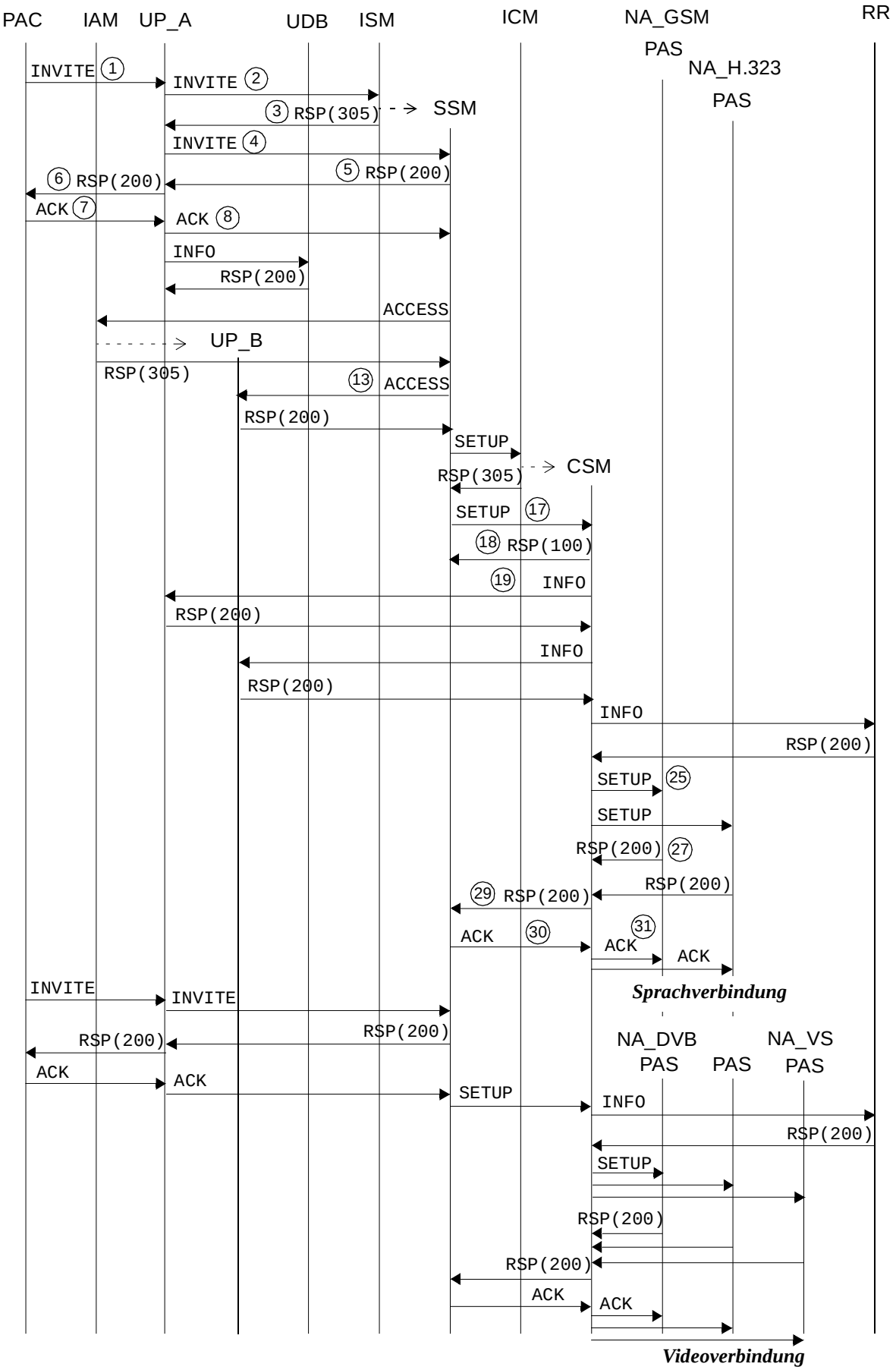


Abbildung 5.19: Signalisierungsbeispiel

Kapitel 6
Adaptoren zur Anpassung an unterschiedliche Netze

In den vorangegangenen Kapiteln 4 und 5 wurde die Struktur und die Funktionsweise der netzunabhängigen Dienstarchitektur beschrieben. Im Folgenden wird das Konzept der Adaptoren dargestellt, welche die Kopplung der Dienststeuerung an die netzspezifischen Signalisierungssysteme der Kommunikationsanbieter realisieren. Man unterscheidet Adaptoren für Kommunikationsnetze (NA), Adaptoren für Server, die Dateninhalte bereitstellen (CtResA), und Adaptoren für Spezialressourcen (SpResA), sowie eine Komponente zur Endgeräte-Adaption (AR) (Abbildung 6.1, siehe auch [KS01]). Eine neu entwickelte Komponente (RR) verwaltet die verschiedenen Adaptoren nach dem Prinzip der *Service Discovery.*

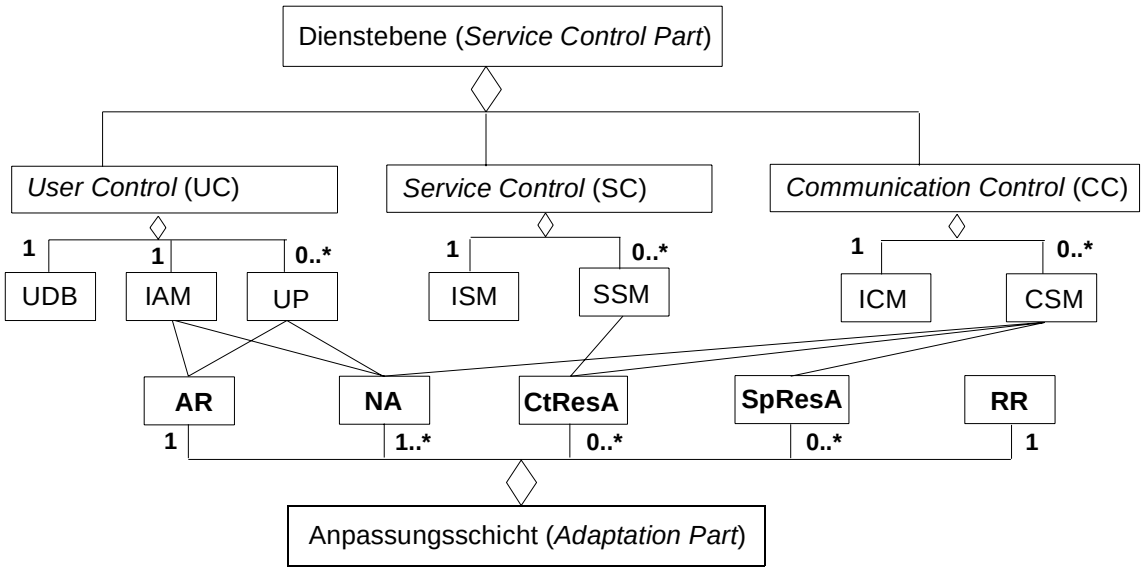


Abbildung 6.1: UML-Klassendiagramm der Anpassungsschicht mit Bezug zu den Komponenten der Dienstebene

6.1 Netzadaptoren zur Steuerung der Dienstausführung

Die **Netzadaptoren** (NA) stellen die Schnittstellen der Dienststeuerung zu den unterschiedlichen Kommunikationsnetzen mit den daran angeschlossenen Teilnehmern dar. Sie vertreten

einen Netzzugang oder eine netzspezifische Komponente in der Dienststeuerung. Die einheitliche Signalisierung auf Dienstebene, die durch das *Session Control Protocol* festgelegt ist, wird durch sie auf die im jeweils angeschlossenen Netz verwendete Signalisierung abgebildet. Somit stellen die Netzadaptoren Signalisierungsgateways dar.

Netzadaptoren dienen dabei nicht nur zur Umsetzung der Kommunikationssteuerung im angeschlossenen Netz, sondern auch zur Interaktion der Teilnehmer mit der Dienststeuerung, indem sie die netzspezifische Teilnehmersignalisierung (z.B. ISDN-UNI, DTMF) in SesCP-Meldungen umsetzen und umgekehrt. Es lassen sich daher zwei Schnittstellen zur Dienststeuerung unterscheiden, wie in Abbildung 6.2 dargestellt: Teilnehmerinteraktion (NA-UI) und Netzsteuerung (NA-CC).

Der Schwerpunkt dieses Abschnitts liegt auf der Steuerung der Dienstausführung, d.h. der Einrichtung von Informationsflüssen und Informationspfaden in den Netzen, die durch je einen Netzadaptor mit der Dienststeuerung verbunden sind. Abschnitt 6.2 geht genauer auf die Teilnehmerinteraktion ein. Für die Anpassung der Darstellung von Meldungen der Dienststeuerung an unterschiedliche Endgeräte wird die Access-Ressource (siehe auch Abschnitt 5.2.3) zwischen den Netzadaptoren und der Dienststeuerung eingefügt. Diese Verknüpfung wird in Abschnitt 6.2 näher beschrieben.

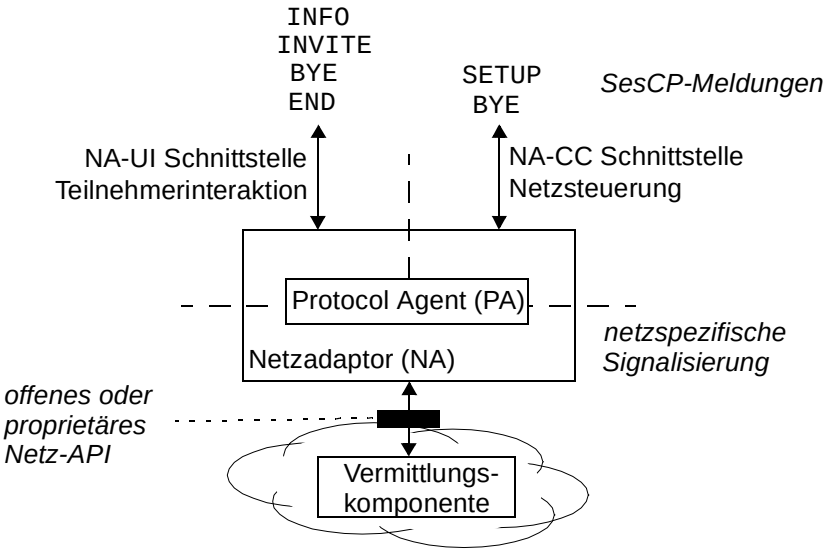


Abbildung 6.2: Modell eines Netzadaptors

Darüber hinaus werden neben den Netzadaptoren weitere Anpassungseinheiten als Schnittstellen zu zwei speziellen Arten von Netzressourcen unterschieden. **Content-Ressource-Adaptoren** bieten eine Schnittstelle zu Servern, die Dateninhalte (z.B. Videoserver, Gameserver) bereitstellen. Werden Spezialressourcen für den Aufbau von Informationspfaden benötigt (siehe Abschnitt 5.5.4), so können diese direkt über die **Spezialressourcen-Adaptoren** angesprochen werden. Diese beiden Spezialfälle werden in Abschnitt 6.3 beschrieben. Abbildung 6.3 gibt einen Überblick über die verschiedenen Typen von Adaptoren.

Die Netzadaptoren (NA) selbst werden nach der Art und Weise klassifiziert, wie der Verbindungsaufbau im angeschlossenen Netz erfolgt. Man unterscheidet parallelen (P-NA), sequentiellen (S-NA) und Netzrand-initiierten (G-NA) Verbindungsaufbau. Für die Dienststeuerung erscheinen alle Netzadaptoren als Schnittstellen für einen stellvertretenden (sogenannten *3rd Party*) Verbindungsaufbau, der durch die Meldung `SETUP` ausgelöst werden kann.

Beim **parallelen** Verbindungsaufbau beinhaltet der Netzadaptor eine echte Schnittstelle für einen stellvertretenden Verbindungsaufbau, wie z.B. das Parlay-API. Beim **sequentiellen** Ver-

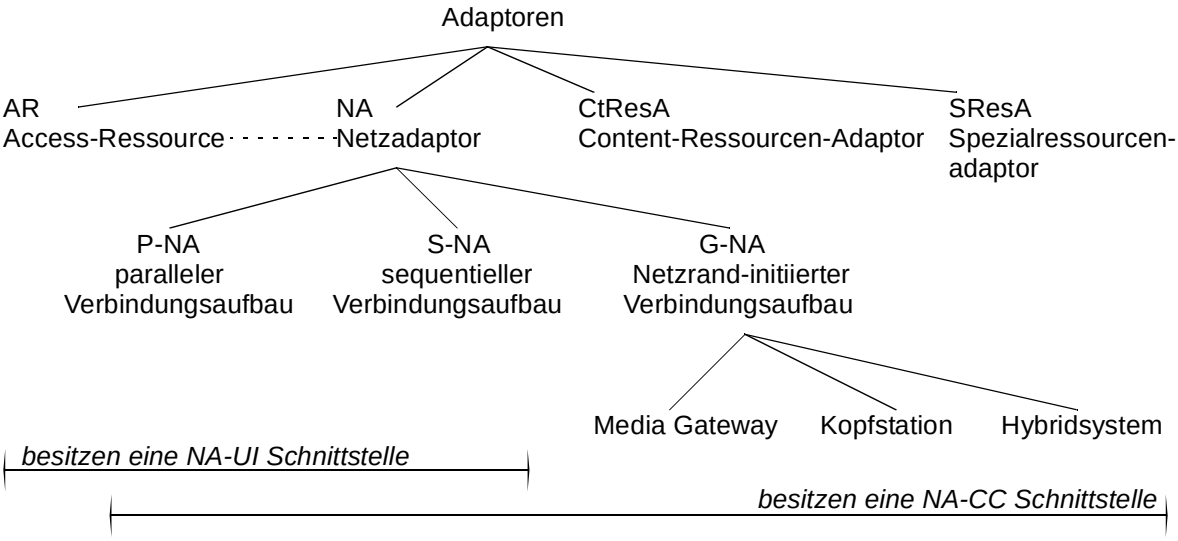


Abbildung 6.3: Klassifikation der (Netz-)adaptoren

bindungsaufbau werden Ende-zu-Ende Informationspfade durch die Verknüpfung von im Netzadaptor terminierenden Einzelverbindungen erreicht. Netzadaptoren der letzten Gruppe bilden eine Schnittstelle zu Gateways, die den Verbindungsaufbau (vom **Netzrand** her) in den angeschlossenen Netzen erlauben. Neben den Media Gateways kann es sich um Gateways in Hybridsystemen oder Kopfstellen in Verteilnetzen handeln.

Je nach der Art und Ausprägung der Netzschnittstelle, auf die ein Netzadaptor zugreift, werden Funktionen zur Einrichtung von Informationsflüssen angeboten, die über einfache Punkt-zu-Punkt-Verbindungen hinausgehen. So werden allein beim Parlay-*Generic Call Control Service* die Schnittstellen *Generic Call Control*, *Multiparty Call Control*, *Multimedia Call Control* und *Conference Call Control* unterschieden [Par00]. Die Kommunikationssteuerung von SAMSON berücksichtigt diese Funktionalität, um die erweiterten Funktionen auszunutzen und damit einen Overhead zu vermeiden. Dieser entsteht, wenn stets alle Informationsflüsse aus Einzelpfaden zusammengesetzt werden, obwohl die Netzschnittstelle beispielsweise bereits eine Konferenzfunktion anbietet. Die Unterscheidung der Netzadaptoren geschieht durch Parameter, mit denen die Netzadaptoren in SAMSON registriert sind.

6.1.1 Paralleler Verbindungsaufbau durch Netz-APIs

Besitzt ein Kommunikationsnetz die Möglichkeit, Verbindungen nicht nur über die Teilnehmer-Netz-Schnittstelle, sondern stellvertretend für die Teilnehmer durch eine Netzkomponente aufzubauen, so stellt diese Netzkomponente einen geeigneten Zugangspunkt für einen SAMSON-Netzadaptor dar. Voraussetzung ist, daß dafür eine Programmierschnittstelle (API) existiert. Folgende Schnittstellen sind für einen P-NA geeignet:

- Parlay/JAIN-API [Par00, BBD00]
- INAP-basierte Schnittstelle zu einer IN-SSF-Komponente [Q.1228]
- CTI/TAPI-Schnittstelle [CL00]
- PINT/SPIRIT-Schnittstelle [PC00, SFL00]
- ISDN-ISUP-Schnittstelle [Boc97] bzw. GSM-MAP-Schnittstelle [EVB01]
- Proprietäre Schnittstelle zu einer IN-SCF-Komponente [BCL00]

- Proprietäre Schnittstelle zu einem H.323 Gatekeeper [P916] oder einem SIP-Proxy-Server

Bei proprietären Schnittstellen können Netzadaptoren nur realisiert werden, wenn der Hersteller die Schnittstelle in geeigneter Weise offenlegt. Vorteilhafter sind standardisierte Schnittstellen, bei denen der Netzadaptor über eigene Signalisierungsnetze (z.B. SS#7 bei INAP oder Internet bei PINT) auf die Netzkomponente zugreifen kann. Es besteht derzeit eine große Anzahl an standardisierten Schnittstellen, die einen steuernden Zugriff auf ein Kommunikationsnetz zum stellvertretenden Verbindungsaufbau erlauben (siehe auch Kapitel 3), so daß es möglich ist, die meisten Netze auf diese Weise mit der Dienststeuerung zu erfassen. Allerdings sind nahezu alle Schnittstellen auf die Definition der Steuerungs-Funktionalität beschränkt und unterstützen keine gesicherte Trennung zu externen Anbietern. Ein Zugriff Dritter muß vorab über erweiterte *Service-Level-Agreements* abgesprochen werden, wobei ein Mißbrauch nur schwer kontrolliert werden kann.

Einzig die Parlay-Schnittstelle (und damit auch die JAIN-Schnittstelle) realisiert einen gesicherten Zugriff Dritter auf die Netzressourcen. Das Parlay *Framework-Interface* regelt den Zugang durch geeignete Autorisierungs-, Authentisierungs- und Managementverfahren. Damit kann durch die Parlay-Schnittstelle eine vollständige Entkopplung der Dienststeuerung von der heterogenen Netzinfrastruktur im Sinne des SAMSON-Geschäftsmodells erreicht werden. Im Folgenden wird daher näher auf die Realisierung eines Netzadaptors mit Parlay eingegangen.

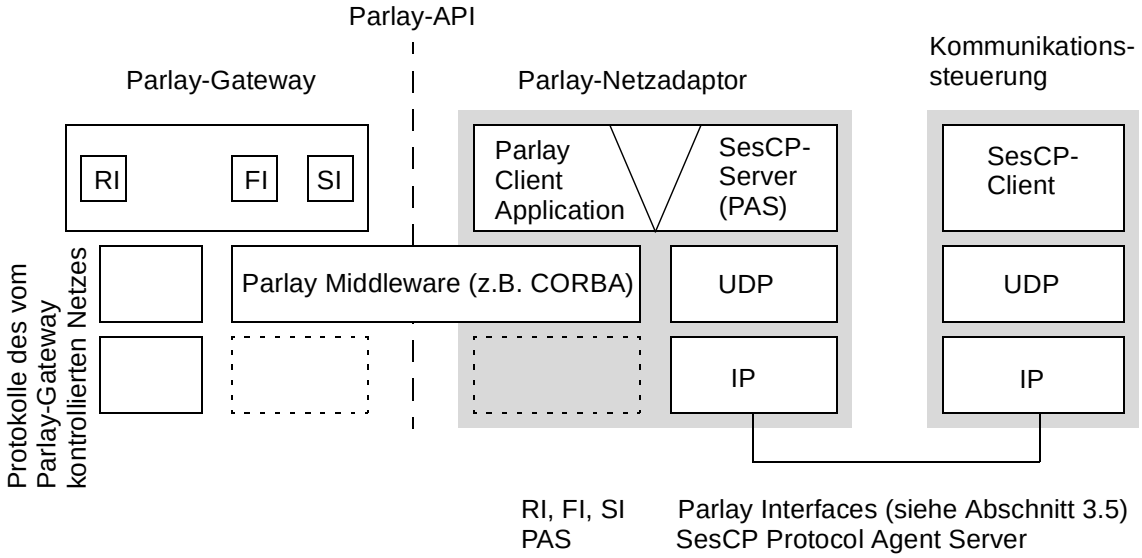


Abbildung 6.4: Protokollarchitektur eines P-Netzadaptors am Beispiel Parlay

Abbildung 6.4 zeigt die Protokollarchitektur eines Parlay-Netzadaptors, der als Signalisierungsgateway zwischen dem Parlay-Gateway und dem SesCP-Protokollstapel arbeitet. Ein Aufruf einer SesCP-SETUP-Meldung löst den Aufruf von verschiedenen Funktionen des Parlay-Framework-Interfaces und eines Parlay-Service-Interfaces aus. Für das folgende Beispiel nehmen wir einen zu Parlay 2.1 [Par00] konformen Parlay-Gateway an, der über ein Parlay *Conference Call Control Interface* der Klasse *Generic Call Control Service Interface* verfügt. Das zugehörige Signalablaufdiagramm für ein SETUP ist in Abbildung 6.5 dargestellt. Der Parlay-Authentisierungs- und *Service Discovery*-Vorgang ist bereits bei der Registrierung des Netzadaptors erfolgt und erscheint nicht im Signalablaufdiagramm.

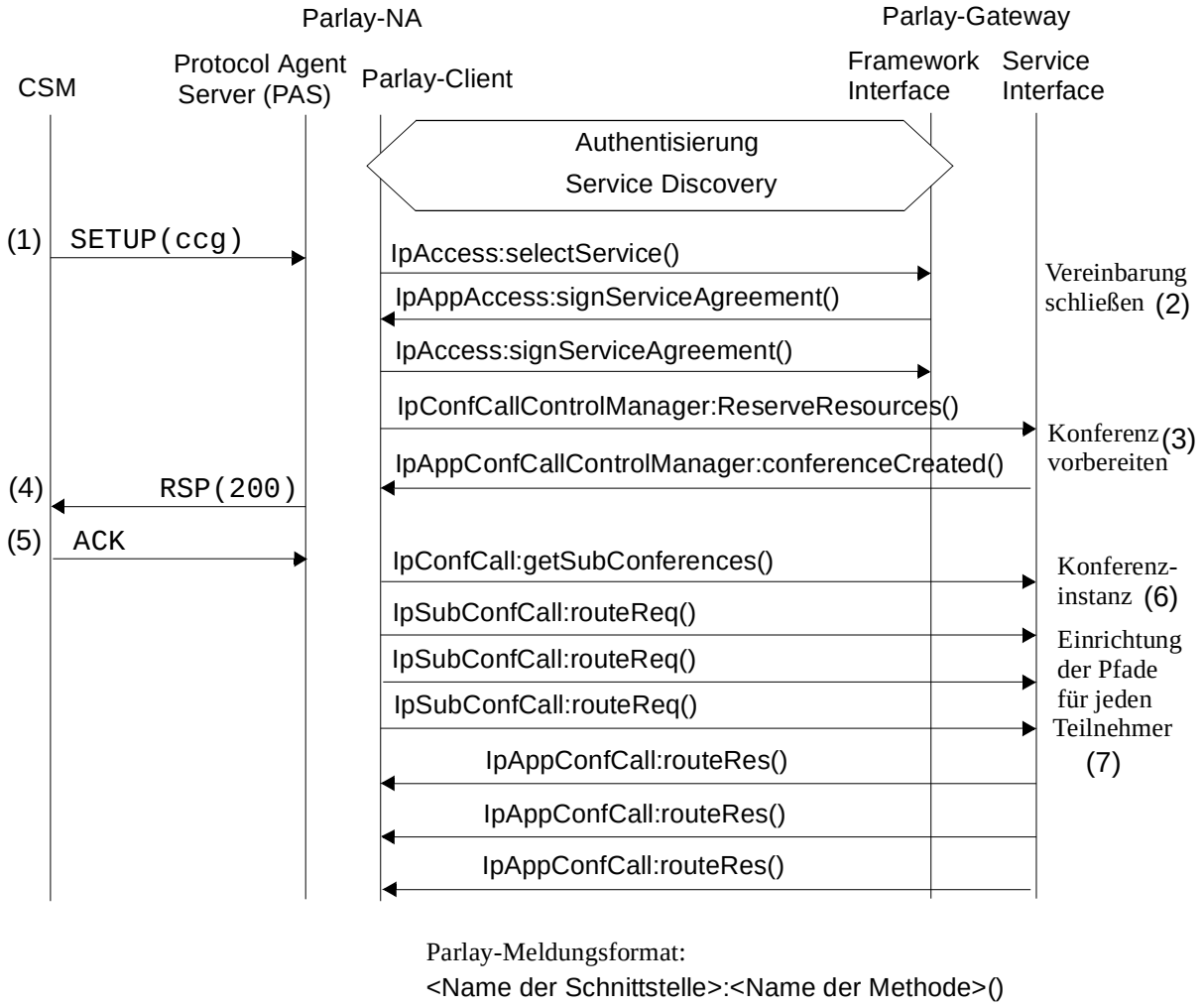


Abbildung 6.5: Signalablauf für eine SETUP-Nachricht in einem Parlay-NA

Das SETUP, das den *Connectivity Connection Graph* enthält, löst am Parlay-Interface die Auswahl des gewünschten Dienstes aus (1). Bevor die benötigten Ressourcen für die Konferenz reserviert werden, wird die Vereinbarung zur Nutzung dieses Parlay-Dienstes geschlossen (2). Erhält der Parlay-Client die Bestätigung, daß alle nötigen Steuerprozesse für die Konferenz gestartet wurden (3), so wird eine positive Antwort an den CSM geschickt (4). Das darauffolgende ACK (5) löst die Einrichtung der Informationspfade für die Konferenzteilnehmer aus (hier drei Teilnehmer) (7). Die Einrichtung der Pfade wird mit routeRes() bestätigt, wenn der Teilnehmer den Ruf annimmt. Mit getSubConferences() wird das Interface zur Konferenzinstanz (*SubConference*) geholt (6).

6.1.2 Sequentieller Aufbau von Kommunikationsverbindungen

Ist ein Kommunikationsnetz nicht in der Lage, einen stellvertretenden Verbindungsaufbau direkt zu unterstützen, sondern kann es nur Verbindungen vom Teilnehmeranschluß her aufbauen, so kann dennoch ein netzseitiger Verbindungsaufbau über einen Netzadaptor bereitgestellt werden, indem der Netzadaptor eine ausreichende Zahl von Teilnehmer-Netz-Anschlüssen bündelt. Bei einem SETUP baut der Netzadaptor für jeden Informationspfad eine Einzelverbindung auf und verknüpft diese intern gemäß der gewünschten Konfiguration (siehe Abbildung 6.6). Der Netzadaptor empfängt in diesem Fall nicht nur die Signalisierungsnach-

richten des angeschlossenen Netzes, sondern auch die Nutzdaten. Diese werden nur durchgeschleift und nicht verarbeitet.

Da für jede Ende-zu-Ende Verbindung mindestens zwei Einzelverbindungen aufgebaut werden müssen, eignet sich ein derartiger S-NA eher für kleine Teilnehmerzahlen[1]. Zudem schlägt sich der doppelte Verbindungsaufbau möglicherweise in hohen Verbindungskosten nieder. Ein Vorteil besteht darin, daß durch die Verwendung von standardisierten Teilnehmer-Netz-Schnittstellen eine vollständige und sichere Trennung zwischen Dienstanbieter und Kommunikationsanbieter gegeben ist.

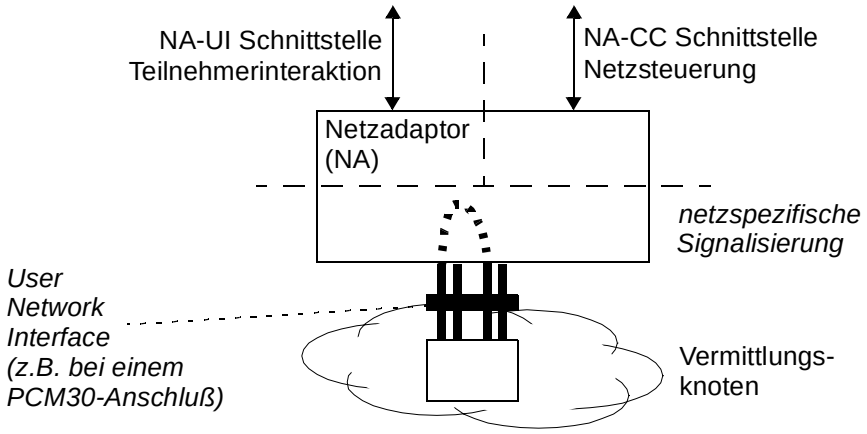


Abbildung 6.6: S-Netzadaptor

6.1.3 Aufbau von Kommunikationsverbindungen durch Gateways

Die dritte Möglichkeit, Informationsflüsse stellvertretend in Kommunikationsnetzen aufzubauen, bieten Gateways, die unterschiedliche Netze koppeln. Sie stellen Komponenten am Rande von Netzen dar. Man spricht daher von einem *Network Edge Initiated*-Verbindungsaufbau durch Gateways.

Media Gateways

Bereits in Abschnitt 3.4.3 wurde das Konzept der *Media Gateways* vorgestellt und deren Eignung für den Verbindungsaufbau in einer heterogenen Infrastruktur betont. Mit Megaco/H.248 [CGR00] ist ein Protokoll standardisiert, mit dem ein *Media Gateway Controller* einen *Media Gateway* steuern und überwachen kann. Ansatzpunkt für den Netzadaptor (G-NA) ist der *Media Gateway Controller.* Entweder ist er direkt Bestandteil des Netzadaptors (wie in Abbildung 6.7 dargestellt), oder der Netzadaptor setzt die SesCP-Signalisierung in die Steuerungsmeldungen für den *Media Gateway Controller* auf ein weiteres Signalisierungssystem um. Die erste Variante, die weniger Aufwand durch das Zusammenfassen von Komponenten bedeutet, wird dadurch vereinfacht, daß die Verarbeitung von SIP-Meldungen durch *Media Gateway Controller* bereits angedacht ist [Tay00]. Eine Erweiterung auf das SIP-basierte SesCP ist dadurch auf einfache Weise möglich.

1. Für das ISDN kann ein solcher Netzadaptor einfach, z.B. mit einer PCM30-(S1-Anschluß) ISDN-Karte, realisiert werden. Beispielsweise wurde am Lehrstuhl für Kommunikationsnetze der Technischen Universität München im Rahmen dieser Arbeit der Service Node I@C der Firma Siemens prototypisch installiert und getestet [Fre99]. Er verwirklicht nach diesem Prinzip IN-ähnliche Zusatzdienste für eine beschränkte Benutzergruppe.

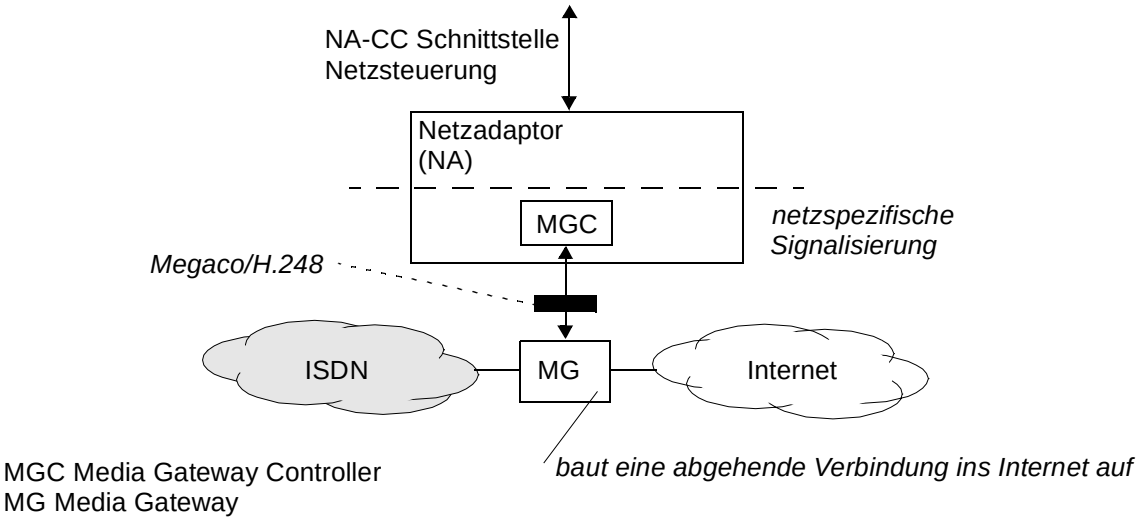


Abbildung 6.7: G-NA am Beispiel eines Media Gateways

Kopfstation eines Verteilnetzes

Einen Spezialfall eines *Media Gateways* stellt die Kopfstation eines Verteilnetzes dar, da hier nur unidirektionale Verbindungen abgehen können. Solange die transportierten Daten für alle Empfänger (Rundfunk) bestimmt sind, ist kein Zugriff auf die Kopfstation notwendig[1], d.h. ein Netzadaptor wird nicht benötigt. Sobald geschlossene Benutzergruppen oder einzelne Benutzer adressiert werden, kann es notwendig sein, für die Adressierung oder vielmehr die Adreßumsetzung einen Netzadaptor zu verwenden.

Hybridsysteme

Um Daten im Zusammenhang mit einem Verteilnetz bidirektional zu übertragen (siehe das Beispiel in Abschnitt 4.8 auf Seite 82) ist eine zusätzliche Infrastruktur für den Rückkanal notwendig. Die Kopfstation wird über einen Netzadaptor angesteuert, um zu entscheiden, wie die Daten auf die angeschlossenen Netze verteilt werden (siehe Abbildung 6.8). Auch hier sind bislang keine Schnittstellen standardisiert. Vielmehr steht die intelligente Verteilung von Daten in Hybridnetzen im Fokus der derzeitigen Forschung, z.B. für drahtlose Netze [Zit01].

6.1.4 Fazit

Zusammenfassend läßt sich feststellen, daß es eine ganze Reihe von standardisierten Schnittstellen gibt, die einen stellvertretenden Verbindungsaufbau ermöglichen. So können Netzadaptoren für die meisten Kommunikationsnetze auf einfache Weise bereitgestellt werden. Auch für Spezialfälle wie Verteilnetze gibt es Lösungen. Das jüngst entwickelte Parlay-API ermöglicht zusätzlich eine vollständige Entkopplung des Dienstanbieters von den Kommunikationsanbietern. Darüber hinaus zeigt Parlay deutlich den Trend zur Öffnung der Kommunikationsnetze für externe Dienstanbieter durch die Bereitstellung von Schnittstellen und bestätigt das Geschäftsmodell, das der vorliegenden Arbeit zugrunde liegt.

1. Eine Möglichkeit, durch die Dienststeuerung Einfluß auf einen Datenserver zu nehmen, der an die Kopfstation gekoppelt ist, wird in Abschnitt 6.3 beschrieben.

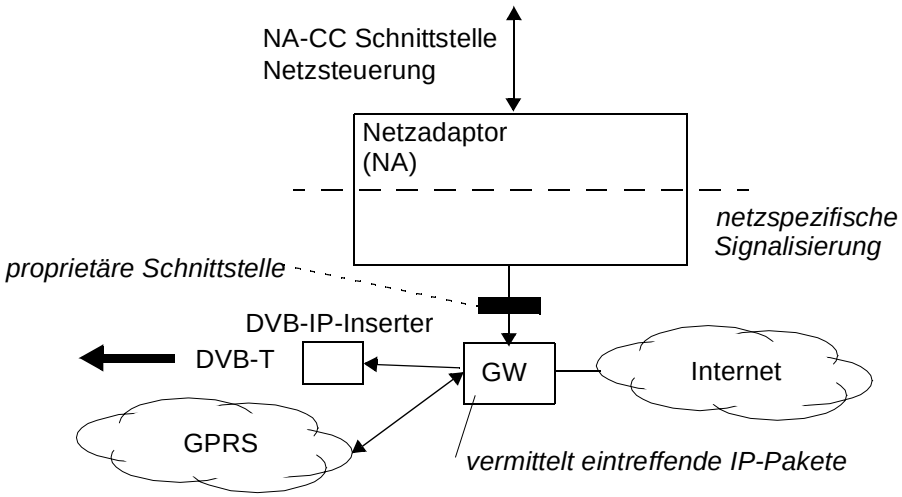


Abbildung 6.8: G-NA am Beispiel eines Hybridsystems

6.2 Teilnehmerinteraktion

Die Netzadaptoren stellen die einzige Verbindung der Dienstarchitektur zur heterogenen Netzinfrastruktur dar und realisieren damit auch den Teilnehmerzugang und die Interaktion der Teilnehmer mit der Dienststeuerung. Dies wird durch die Schnittstelle NA-UI in Abbildung 6.2 ausgedrückt. Angeschlossene Netze können die Teilnehmerinteraktion (NA-UI) und die Steuerung der Dienstausführung (NA-CC) unterstützen (z.B. Parlay-API) oder reine Netze für die Teilnehmerinteraktion sein.

Zur Realisierung der Teilnehmerinteraktion sind, abhängig von den registrierten und aktiven Endgeräten eines Teilnehmers und dem Teilnehmer-Status (z.B. Initiator, eingeladener Teilnehmer), verschiedene Fälle zu betrachten. Grundsätzlich lassen sich drei Interaktions-Typen unterscheiden: Teilnehmerzugang, die Teilnehmer-initiierte Interaktion und die Dienst-initiierte Interaktion.

Teilnehmerzugang

Gemäß dem zugrunde gelegten Geschäftsmodell (siehe Abschnitt 4.3) adressiert ein Teilnehmer von sich aus den Dienststeuerserver, indem er die Adresse eines Netzadaptors benutzt, um sich zu registrieren oder einen Dienst zu starten. Darüber hinaus können intelligente Netzkomponenten von sich aus die Dienststeuerung aufrufen, d.h. Ereignisse an einen Netzadaptor signalisieren. Eine SSF des IN ist ein Beispiel für eine derartige Netzkomponente.

Die Signalisierung für den Teilnehmerzugang ist grundsätzlich auf die Mächtigkeit der netzspezifischen Signalisierung des Zugangsnetzes und die Darstellungsfähigkeit des Endgerätes beschränkt (z.B. ISDN-Q.931 im D-Kanal).

Integration des Geräte-/Darstellungs-unabhängigen Dienstzugangsmoduls

Um die Darstellung der SesCP-Signalisierungsinhalte beim Teilnehmer unabhängig von einem spezifischen Signalisierungssystem zu machen, wurde in Abschnitt 5.2.3 das Konzept der Access-Ressource neu eingeführt. Dabei wird die Teilnehmerschnittstelle durch einen XML-Server gebildet, der die Signalisierungsmeldungen angepaßt an das Endgerät darstellen kann (z.B. HTML, WML, VXML) [KS01].

Da hier die Access-Ressource die Schnittstelle zur Dienststeuerung darstellt, beinhaltet sie den *Protocol Agent Client*-Prozeß für das SesCP. Die Netzadaptoren beinhalten als Gegenstück zum XML-Server ein WAP- oder WWW-Portal, das gegebenenfalls eine *Internet Service Provider*-Funktion (z.B. Modem-Pool) umfaßt. Für VXML besteht der Netzadaptor aus einem *Voice Browser* [GKK00]. Abbildung 6.9 zeigt die Verknüpfung der Access-Ressource mit Netzadaptoren.

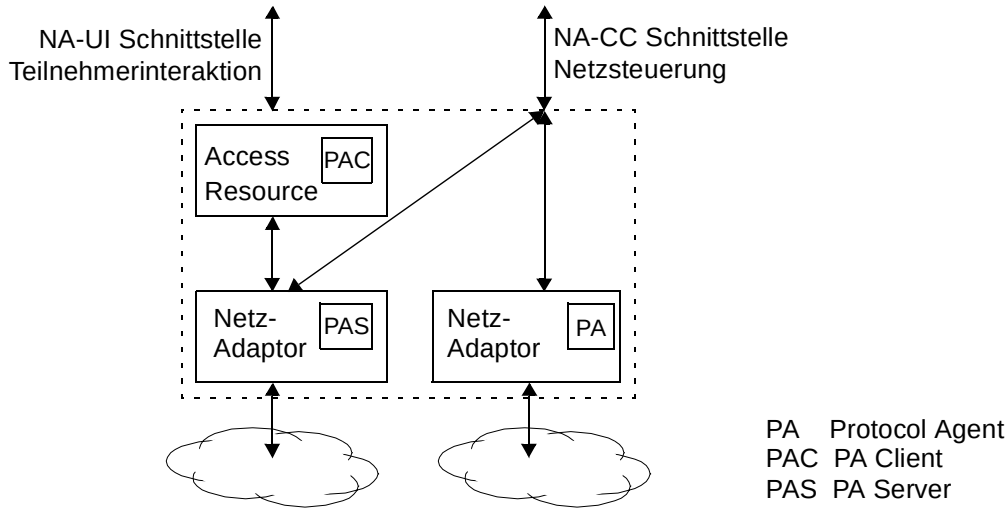


Abbildung 6.9: Verknüpfung der Access Resource mit Netzadaptoren

Teilnehmer-initiierte Interaktion

Die Interaktion eines Teilnehmers mit einem aktiven Dienst (`INVITE`) geschieht im Regelfall über den Netzadaptor, mit dem der Dienst gestartet wurde. Dies schließt auch die Verwendung der Access-Ressource mit ein. Falls der bestehende Zugang ungeeignet ist, hat der Teilnehmer einen geeigneten Zugang selbständig aufzubauen. Da die gesamte Teilnehmer-Dienst-Signalisierung über den *User Proxy* abgewickelt wird, ist die Konsistenz fortwährend gewährleistet.

Dasselbe gilt für einen (registrierten) Teilnehmer, der zu einem Dienst eingeladen ist. Dieser hat eine geeignete *Access Session* aufzubauen, wenn er in die Dienststeuerung eingreifen will. Der zugehörige *User Proxy* wurde bereits bei Start der *Communication Session* angelegt, um das Teilnehmerprofil abzufragen. Für nicht registrierte Teilnehmer ist eine Interaktion mit dem Dienst nicht vorgesehen.

Teilnehmer-Abfrage durch die Dienststeuerung

Bei bestehender *Access Session* wird auch die Interaktion eines aktiven Dienstes mit einem Teilnehmer (SesCP-`INFO`), z.B. um diesem eine Mitteilung zu machen oder Parameter abzufragen, über den Netzadaptor der *Access Session* ausgeführt.

Für den Fall, daß kein geeigneter Zugang existiert, enthält der *User Proxy* Informationen, über welchen Netzadaptor ein Interaktionskanal aufgebaut werden kann. Ein Interaktionskanal kann für den Fall einer einfachen Mitteilung beispielsweise aus einer Email bestehen. Einige Netz-Schnittstellen beinhalten Mechanismen, die es erlauben, mit einem Endteilnehmer Informationen auszutauschen. Hierzu gehören z.B. die Parlay *Generic User Interaction Service Interfaces* [Par01].

6.3 Steuerung zusätzlicher Ressourcen

Wie bereits erwähnt, unterscheiden wir zwei Sonderfälle von Adaptoren, die der Steuerung von speziellen Netzressourcen dienen: Adaptoren für Server, die Dateninhalte bereitstellen (hier als Content-Ressourcen bezeichnet) und Adaptoren für Spezialressourcen.

Content-Ressourcen-Adaptoren

Adaptoren für Content-Ressourcen dienen der Integration von Applikationsservern oder Datenservern in die Dienststeuerung. Applikationsserver können prinzipiell als Teilnehmer behandelt und so in eine Dienstausführung eingebunden werden. Für alle Applikationsserver, die keinen direkten Teilnehmerzugang erlauben, weil ihnen z.B. Sicherheitsmechanismen fehlen, werden Content-Ressource-Adaptoren eingeführt. Über diese kann eine Content-Ressource für Teilnehmer freigeschaltet werden (SesCP-`INVITE`) und darüber hinaus auch eine Verbindung zum Teilnehmer aufgebaut werden (SesCP-`SETUP`). Content-Ressource-Adaptoren sind somit die einzigen Adaptoren, die direkt auf Endgeräte wirken. Sie sind bei allen Applikationsservern notwendig, die keinen Rückkanal bzw. Interaktionskanal besitzen, wie zum Beispiel *Push*-Server, *Broadcast*-Server oder *Multicast*-Server.

Content-Ressource-Adaptoren besitzen zwei Schnittstellen zur Dienststeuerung (siehe Abbildung 6.10). Die CT-CC Schnittstelle entspricht der NA-CC Schnittstelle der Netzadaptoren. Es entfällt die NA-UI Schnittstelle für die Teilnehmerinteraktion. Zur Dienststeuerung, z.B. Freischaltung des Applikationsservers, kommt die Schnittstelle CT-SC neu hinzu. Über diese Schnittstelle kann der SSM mit SesCP-`INVITE` der Content-Ressource eine bestehende Dienstkonfiguration mit dem *User Service Graph* mitteilen.

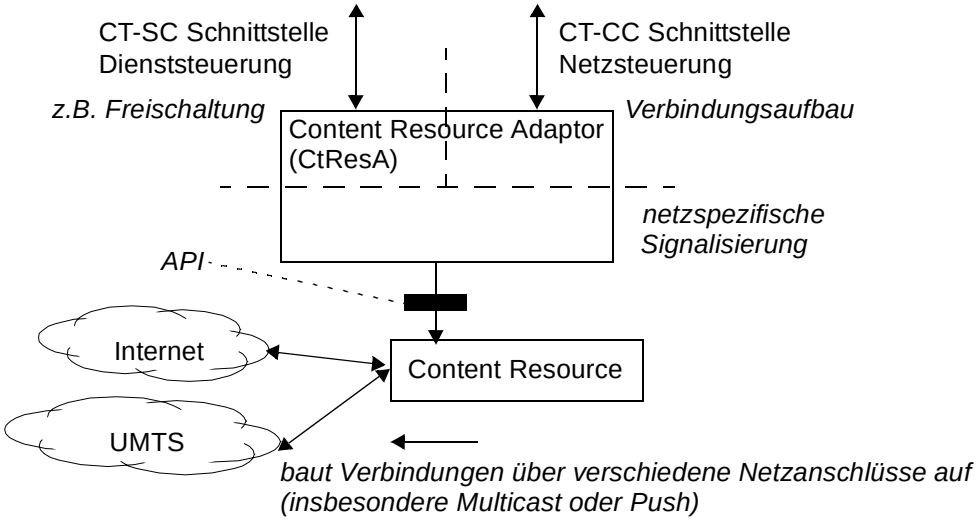


Abbildung 6.10: Modell eines Content Resource Adaptors

Spezialressourcen-Adaptoren

Spezialressourcen sind Funktionen eines Netzelements, die zur Verarbeitung der transportierten Information dienen, um einen durchgehenden Ende-zu-Ende Informationsfluß zu ermöglichen, wie z.B. Umsetzer für unterschiedliche Kodierverfahren, Konferenzbrücken oder Mischer für reine Audiokonferenzen. Diese Funktionen gehen über die Vermittlungsfunktionen zur Einrichtung von Informationsflüssen hinaus. Um die gewünschte Funktion anzusteuern, sieht die SAMSON-Architektur vor, daß Spezialressourcen eine eigene Schnittstelle besitzen. Auf diese Weise können Spezialressourcen auch von externen Betreibern angeboten

werden. Spezialressourcen-Adaptoren verfügen nur über die S-CC Schnittstelle (SesCP-SETUP und SesCP-BYE) zur Dienststeuerung.

6.4 Neuartige Lösung für die Ressourcenverwaltung

Wie bereits bei der Kommunikationssteuerung in Kapitel 5 erwähnt, wird die Verwaltung der Netzadaptoren durch eine zentrale Datenbank realisiert. Aufgabe dieser Datenbank und deren Steuerungskomponenten ist es, die aktuelle Verfügbarkeit von Netzressourcen durch deren Adaptoren zu ermitteln, zu speichern und auf Basis dieses Wissens die Auswahl von Adaptoren zu unterstützen.

6.4.1 Problemstellung

Die Ressourcenverwaltung hat folgende Aufgaben und Merkmale zu erfüllen:

- Automatische Erkennung und Registrierung von Adaptoren, um einen menschlichen Verwaltungsaufwand so gering wie möglich zu halten;
- Beschreibung der Fähigkeiten der Netzressourcen mit geeigneten Attributen, um die Suche nach bestimmten Eigenschaften zu ermöglichen;
- Speicherung der Datensätze in einer Datenbank;
- Dynamisches Update der Datensätze, um stets den aktuellen Zustand von Netzressourcen zu erhalten und nichtverfügbare Netzressourcen aus dem Speicher zu entfernen;
- Einfache Implementierung mit Standard-Software-Techniken und Portabilität auf unterschiedliche Plattformen, um unterschiedliche Netzadaptoren zu unterstützen.

6.4.2 Attribute der Adaptoren

Die Eigenschaften der Adaptoren drücken sich in Parameterwerten aus, die der Kommunikationssteuerung, d.h. dem *Communication Session Manager*, dazu dienen, geeignete Adaptoren auszuwählen. Folgende Attribute (Tabelle 6.1) werden gespeichert[1]:

Unter dem Attribut **Bezeichner** ist die Art des Adaptors referenziert (Netzadaptoren: P-, S-, G-NA; CtResA, SpResA). Die SAMSON-Adresse kennzeichnet alle Schnittstellen eines Adaptors, die innerhalb von SAMSON für SesCP-Meldungen zur Verfügung stehen.

Die erreichbaren Teilnehmer-Adressen werden durch den **Netztyp** (z.B. ISDN oder Internet) und die Angabe des Netzanbieters festgelegt. Letzterer Parameter dient zur weiteren Eingrenzung des Adressraumes, wenn eine hierarchische, Domänen-bezogene Adressierung vorliegt.

Über den **Kommunikationstyp** wird die Funktion des Adaptors beschrieben, Informationsflüsse aufzubauen. Man unterscheidet zwischen Konferenzfähigkeit (conf), Dialogfähigkeit (dialog), Multicastfähigkeit (mc), der Fähigkeit bidirektionale, aber asymmetrische Verbindungen aufzubauen (asym) und Unicastfähigkeit (uc). Diese Parameter werden in einigen Fällen durch die Angabe der maximalen Anzahl erreichbarer Teilnehmer ergänzt.

1. Einige Attribute (z.B. Kodierformate) sind nur bei speziellen Adaptoren (für Content-Ressourcen oder Spezialressourcen) zutreffend.

Attribut	**Felder**	**Beispiel: Netzadaptor für ein DVB-T/GPRS-Hybrid-Gateway für IP**
Bezeichner	Art des Adaptors; SAMSON-Adresse	G; NA1234.samson
Netztyp	Typ; Netzanbieter	dvbt_gprs; irt.muc.de
Kommunikationstyp	Typ; max. Anzahl von Teilnehmern	asym; -
Medientyp	Medium	data
Qualitätstyp	SQ, Leitungskapazität, Zuverlässigkeit, Verzögerung	streaming, 2mbps, 10^{-11}, 5 ms
Kodierung	unterstützte Kodierformate	-
Kodierungskonversion	Ausgangsformate	-
Interaktionsfähigkeit	Boolean-Parameter	0
Auslastung	Auslastung in Prozent	50 %
Kosten	Kostenniveau	low

Tabelle 6.1: Attribute der (Netz-)Adaptoren in der Ressourcenverwaltung

Analog zur SDP+-Beschreibung gibt der **Medientyp** an, welche Informationsflüsse eingerichtet werden können. Man unterscheidet zwischen *Voice*, *Video*, *Image*, *Data* und *Text*.

Der **Qualitätstyp** beschreibt die konkreten Parameter der unterstützten Übertragungsqualität bezüglich Kapazität (Bitrate) und QoS-Parameter (Bitfehlerrate, Delay und Jitter). Dieser Beschreibung geht die allgemeine Klassifizierung der Qualität aus Dienstsicht, die sogenannte Dienst-Qualitäts-Klasse (SQ), voraus, da abhängig davon nur bestimmte Werte der Übertragungsqualitätsparameter belegt sind.

Die unterstützten Kodierformate werden bei der **Kodierung** angegeben und beschreiben die Darstellungsqualität. Bei Umsetzern (Spezialressourcen) ist zudem angegeben, in welche Kodierformate die Eingangsformate konvertiert werden können. Es wird angenommen, daß die Umsetzung von jedem Eingangsformat in jedes Ausgangsformat und umgekehrt möglich ist. Ist dies nicht der Fall, sind für einen Umsetzer mehrere Datensätze in der Ressourcenverwaltung anzulegen.

Das Attribut **Interaktionsfähigkeit** gibt in einem Boolean-Parameter an, ob eine Netzressource Teilnehmerinteraktion unterstützt. Damit ist insbesondere die Fähigkeit bezeichnet, SesCP-`INFO`-Meldungen zu verarbeiten.

Zusätzlich zu den obigen Attributen, die statische Eigenschaften beschreiben, werden zwei Attribute eingeführt, die dynamische Parameter beinhalten. Die **Auslastung** gibt eine qualitative Angabe (0% .. 99%; 100% entspricht nicht verfügbar) über die Belegung einer Netzressource. Dieser Parameter ist insbesondere bei Netzen wichtig, die nur statistische Qualitätsangaben zulassen, wie z.B. das Internet. Über das Attribut **Kosten** kann die Auswahl von Netzressourcen in Abhängigkeit vom Verbindungspreis oder vom Nutzungspreis getroffen werden. Wir nehmen an, daß in Zukunft der Preis hoch dynamisch ist. Bereits jetzt hat jeder Netzbetreiber eine nach Tageszeit differenzierte Preistabelle.

6.4.3 Lösung durch das Service Discovery-Konzept

Klassische Verfahren zur Adreßfindung aus dem Internet-Bereich, wie zum Beispiel der *Internet Domain Name Service* [McG00], erfüllen die oben aufgestellten Anforderungen an eine zentrale Ressourcenverwaltung nicht, da sie statische Datenbanken verwenden, die zudem von autorisierten Personen verwaltet werden müssen. Sie garantieren weder die Verfügbarkeit der eingetragenen Ressourcen, noch haben sie eine ausreichende Attributbeschreibung für die Ressourcenauswahl.

Um diese Unzulänglichkeiten auszuräumen, sind in der jüngsten Zeit verschiedene Konzepte entstanden, die die Registrierung von Serverfunktionen und die explizite Suche danach ermöglichen. Diese Verfahren werden unter dem Begriff *Service Discovery* zusammengefaßt. ***Service Discovery***-Systeme bestehen im allgemeinen aus **Dienstkomponenten**, die ihre Funktionen für andere Nutzer anbieten, Client-Komponenten, die in einer bestimmten Umgebung nach Diensten (Funktionen) suchen, um die Ausführung von Anwendungen zu unterstützen, und einem **Register**, in dem die Funktionen aufgelistet und für die Suche aufbereitet werden (Abbildung 6.11). Das Register kann aus einer zentralen Instanz bestehen oder verteilt angeordnet sein. In den *Service Discovery*-Protokollen ist festgelegt, wie ein Client die *Service Discovery*-Infrastruktur findet (*Discovery*) und eine Anfrage an das Register richtet (*Lookup*). Ebenso ist dadurch definiert, wie eine Dienstkomponente das Register findet (*Discovery*) und seine Funkionen einträgt (*Registration*). Zusätzliche Mechanismen stellen sicher, daß alle Einträge aktuell sind (*Update*) und obsolete Einträge entfernt werden (*Cleanup*).

Ist eine gewünschte Funktion gefunden, so muß der Zugriff ausgehandelt werden (*Access*) und schließlich auf die Funktion zugegriffen werden (*Usage*). Dies bedeutet, daß eine Schnittstelle (und das zugehörige Protokoll) bekannt gemacht oder übergeben werden muß, über die die Funktion aufgerufen werden kann. Letztere Mechanismen sind nicht Bestandteil der meisten *Service Discovery*-Konzepte und bleiben dem Entwickler überlassen.

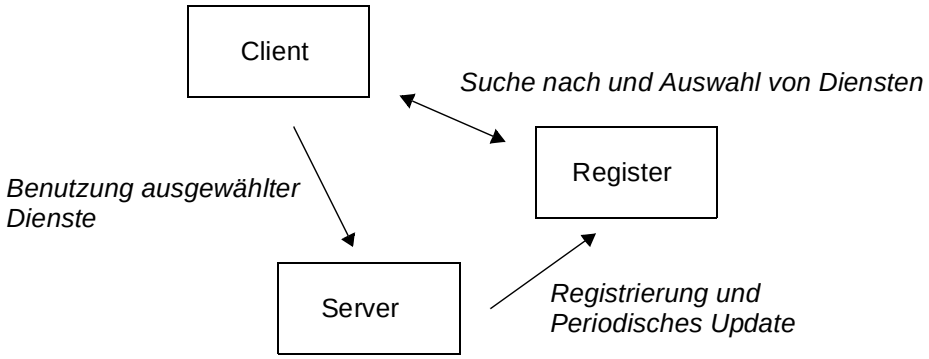


Abbildung 6.11: Service Discovery-Prinzip

Für die vorliegende Arbeit wurde aus den Verfahren *Jini, SLP, Salutation* und *UPnP* das Jini-Konzept[1] ausgewählt [Leg00]. Jini unterstützt alle Phasen des *Service Discovery* (*Discovery, Registration, Lookup*) und beinhaltet zudem Mechanismen für den Funktionsaufruf (*Usage*). Die Suche nach und die Auswahl von Funktionen wird durch einen Vergleichsprozeß auf Basis von Attributen bereitgestellt. Jini ist das einzige System, bei dem ein für den Funktionsaufruf lauffähiger Code an den Client übergeben werden kann, der die Schnittstelle zu einer Funktion beinhaltet. Dieser Vorteil bringt zusätzliche Vereinfachungen bei der Systemrealisierung. Ermöglicht wird dieser Download durch die Software-Basis Java. Java-Code ist unabhängig

1. HTTP://www.sun.com/jini/

vom Betriebssystem und die *Service Discovery*-Funktionen sind damit auf nahezu jedem Netzadaptor realisierbar, ohne daß die Software angepaßt werden muß. Dies ist ein entscheidender Vorteil, wenn Netzadaptoren von den Netzbetreibern bereitgestellt werden. Aber auch bei der Verwendung von Netzadaptoren, die von den Kommunikationsanbietern entkoppelt sind, zeigen sich Vorteile in der Verwendung von Java, da mit dem ebenfalls Java-basierten JAIN-Parlay-System leicht die Jini/Java-Komponenten integriert werden können.

Im Folgenden werden die generellen Komponenten und Mechanismen beschrieben, mit denen die Ressourcenverwaltung nach dem *Service Discovery*-Konzept basierend auf Jini realisiert wird. Eine prototypische Realisierung ist in Kapitel 7 beschrieben.

6.4.4 Aufbau und Funktionsweise der Resource Registry

Für die Ressourcenverwaltung ist in SAMSON ein Register in Form einer Datenbank zu ergänzen, in dem die Attribute der Netzadaptoren gespeichert und abgefragt werden können. Diese Datenbank wird mit *Resource Registry* (RR) bezeichnet (Abbildung 6.12). Sie entspricht dem Jini-*Lookup Service*. Die Client-Funktionalität wird im *Communication Session Manager* implementiert. Jede CSM-Instanz weiß, wie sie die *Resource Registry* kontaktieren kann. Der *Discovery*-Prozeß entfällt hier. Die einzelnen Netzadaptoren agieren als Dienstserver, die ihre Funktionen, d.h. ihre Schnittstellen und deren Attribute, gemäß dem oben beschriebenen Parametersatz in der *Resource Registry* registrieren.

In SAMSON ist analog zur Teilnehmerdatenbank nur eine, zentrale *Resource Registry* vorgesehen. Kommerzielle Lösungen zur verteilten Realisierung der Datenbank können auch hier Verwendung finden, werden aber nicht weiter betrachtet. Außerdem wird angenommen, daß alle Komponenten über ein IP-basiertes Netz verbunden sind.

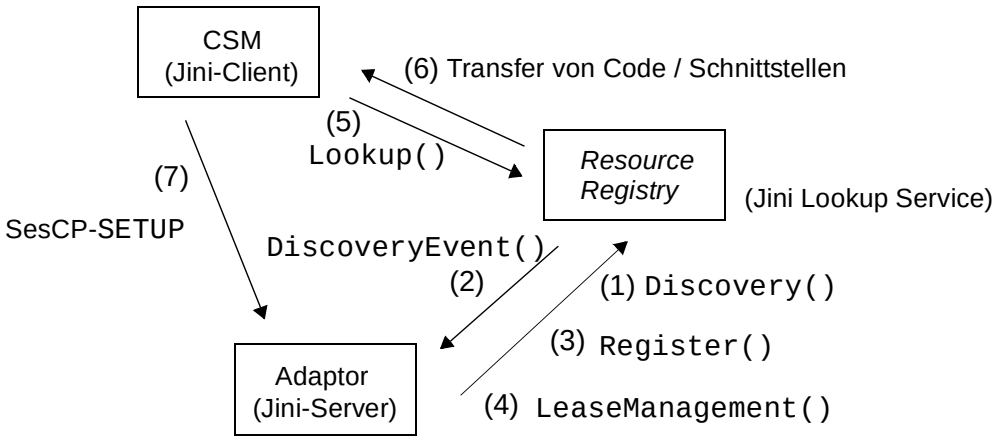


Abbildung 6.12: Verwendung von Jini in SAMSON

Die Adaptoren beinhalten eine Softwarekomponente, die einen Jini-Server implementiert. Wird ein Adaptor neu an das SAMSON-Intranet angeschlossen, so wird zunächst eine Jini-*Multicast Discovery* durchgeführt, da der Server den *Lookup Service* noch nicht kennt. Dabei wird eine Broadcast-Anfrage im Netz gestartet (1). Bei positiver Antwort (2) wird ein *Service Registrar* in der Adaptor-Software lokal erzeugt, über den der sich anschließende Protokollablauf für die Registrierung der Attribute beim *Lookup Service* läuft. Mit der Jini-Methode `register()` (3) werden die Attribute übermittelt.

Die Aktualität der Einträge im *Lookup Service* werden durch den *Lease*-Mechanismus bestimmt. Bei jeder Registrierung wird durch den *Lookup Service* eine Zeitspanne festgelegt, in der der Eintrag seine Gültigkeit behält. Der Server hat seinen *Lease* nach Ablauf der Zeit zu

erneuern (4), andernfalls wird der Eintrag entfernt. Das hat den Vorteil, daß bei geeigneter Wahl der *Lease*-Zeit nicht mehr verfügbare Adaptoren schnell aus der *Resource Registry* (*Lookup Service*) entfernt werden und durch die periodische Erneuerung dynamische Attribute schnell aktualisiert werden.

Jede Instanz des *Communication Session Managers* enthält eine Software-Komponente, die einen Jini-Client implementiert, um die für die Einrichtung von Informationsflüssen notwendigen Netzressourcen auszuwählen und deren Adaptoren mit SesCP-`SETUP` (7) die gewünschte Konfiguration mitzuteilen. Da sowohl der CSM als auch die *Resource Registry* (*Jini Lookup Service*) fester Bestandteil der Serverarchitektur sind, ist der *Lookup Service* dem Client bereits bekannt. Der Vorgang der *Unicast-Discovery* umfaßt daher nur die Erzeugung des *Service Registrars,* über den der weitere Protokollablauf abgewickelt wird.

Beim Lookup (5) wird dem *Resource Registry* ein sogenanntes *Service Template* übergeben, das eine Attributliste mit einer gewünschten Parameterbelegung enthält. Diese Parameter werden mit den in der *Resource Registry* enthaltenen Datensätzen abgeglichen. Findet sich eine Übereinstimmung, so wird eine Schnittstelle oder ein Objekt übergeben (6), woraus der CSM die benötigten Informationen für die Anforderung von Informationsflüssen von dem gefundenen Adaptor entnehmen kann. Bei mehreren Alternativen wird der zuerst gefundene Adaptor ausgewählt.

Grundsätzlich genügt es dem CSM, die Adresse eines ausgewählten Adaptors auf der Ebene des Signalisierungsprotokolls SesCP zu kennen, um mit diesem SesCP-Meldungen austauschen zu können (*Usage*), da die SesCP-Schnittstellen bekannt sind. Die Adresse wird entweder aus den Attributen entnommen, die dem CSM von der *Resource Registry* bei erfolgreicher Anfrage zurückgeliefert werden, oder die *Resource Registry* übergibt ein Java-Objekt, dessen Funktionen eine Abfrage der Adresse und weiterer Parameter beim ausgewählten Adaptor erlauben. Dieses Objekt wird von dem Adaptor bei der Registrierung mit den Attributen an die *Resource Registry* übergeben.

Die Fähigkeit von Jini, nicht nur Attribute, sondern auch Objekte an den Client zu übergeben, die beim Client eingebunden werden und die Schnittstelle zum Server implementieren, erlaubt eine weitergehende Vereinfachung bei der Entwicklung und Benutzung von Adaptoren. Der Client ruft in den übergebenen Objekten lokale Methoden auf, deren Schnittstellen vorher bekannt sein müssen und die ihrerseits die Signalisierung mit dem Server abwickeln. Auf diese Weise können den Entwicklern von Adaptoren erweiterte Möglichkeiten eingeräumt werden, wie die Kommunikation zwischen CSM und NA mit `SETUP` abläuft. Falls die Adaptoren von externen Netzbetreibern bereitgestellt werden, ist es beispielsweise möglich, in der Signalisierung zwischen CSM und Adaptor Sicherheitsmechanismen zu integrieren, die im normalen SesCP-Protokollablauf an dieser Stelle nicht vorgesehen sind.

Die Übergabe der kompletten Software eines Adaptors ist allerdings nicht möglich, da Adaptoren als Gateway zwischen zwei verschiedenen Signalisierungsnetzen arbeiten. Jini basiert jedoch auf einem einheitlichen Kommunikationsmechanismus (Java-*Remote Method Invocations,* RMI), der nur innerhalb einer Java-Umgebung funktioniert. Es können also nur die Teile von Adaptoren übergeben werden, die nicht auf eine spezielle Infrastruktur (z.B. Vermittlungsknoten) angewiesen sind. Bei Parlay/JAIN kann z.B. die Parlay-Client-Application direkt als Objekt an den CSM übergeben werden. Der Parlay-NA-Gateway (siehe Abbildung 6.4) entfällt dann. Voraussetzung ist, daß der CSM in einer IP/RMI-Umgebung betrieben wird.

6.5 Zusammenfassung

In diesem Kapitel wurden die Elemente der Anpassungsschicht und ihre Funktionsweise innerhalb der Serverarchitektur SAMSON erläutert. Je nach den Netzen und deren Schnittstellen, die der Kommunikationsanbieter zur Verfügung stellt, lassen sich verschiedene Adaptoren nach der Art des Verbindungsaufbaus klassifizieren. Weiter unterscheiden sich die Adaptoren durch Attribute, die dazu dienen, passende Adaptoren für die Einrichtung von Kommunikationspfaden auszuwählen. Eine neu entwickelte Komponente zur *Service Discovery* nach Jini verwaltet die Adaptoren und realisiert den Auswahlmechanismus. Verschiedene Fallstudien belegen die beschriebene Funktionsweise der unterschiedlichen Adaptor-Klassen.

Betrachtet man abschließend das Eingangsszenario („MyBusinessCall“, Abbildung 4.1), das schon bisher als Grundlage für die Beispielabläufe diente, dann kann die Funktionsweise der Anpassungsschicht der Serverarchitektur folgendermaßen illustriert werden (Abbildung 6.13):

Teilnehmerin Alex kontaktiert über ihren GSM-Modemanschluß den Server (1) und interagiert über WAP mit der Teilnehmersteuerung (2). Durch die Dienstauswahl wird ein *Service Session Manager* gestartet (3). Dieser stößt die Kommunikationssteuerung an (4) und teilt dem Datenserver mit, welche Präsentation abgespielt werden soll (5). Der *Communication Session Manager* kontaktiert die *User Proxies* (6). So erhält er die Daten, mit denen er anschließend mittels der *Resource Registry* die notwendigen Adaptoren auswählt (7). Die GSM-Strecke bis zum Gateway wird über ein Parlay-API angefordert (8). Die weitere Verbindung zum Partner Bernd im Internet erfolgt durch ein Media-Gateway (9). Die Datenzuspielung wird über ein Hybrid-Gateway (10) vorgenommen, das vom Datenserver gespeist wird (11).

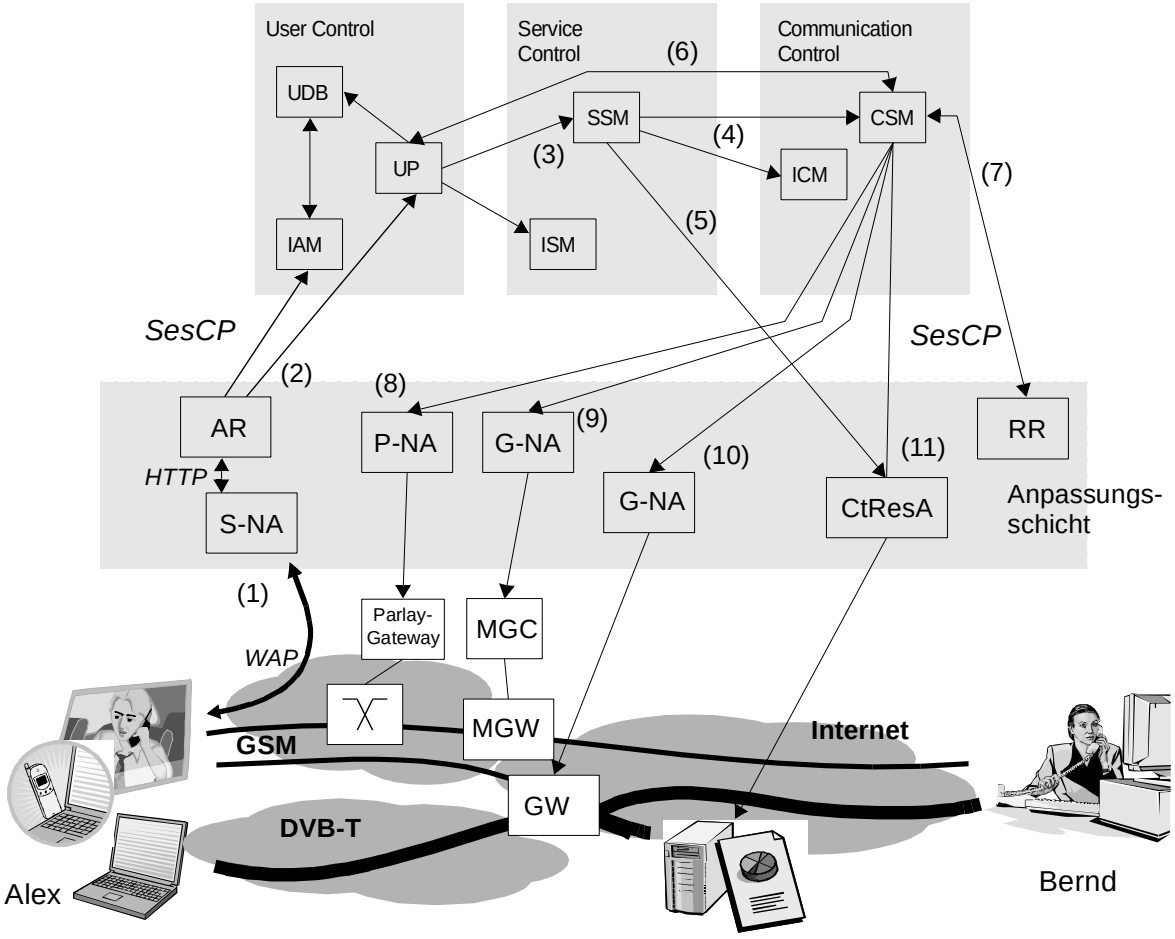


Abbildung 6.13: Funktion der Anpassungsschicht im Dienst-Szenario „MyBusinessCall“

Kapitel 7
Prototypische Realisierung

Ausgewählte Teile der Serverarchitektur SAMSON wurden im Rahmen der vorliegenden Arbeit prototypisch implementiert. Ziel war es, die erarbeiteten Signalisierungsabläufe zu verifizieren und den Nachweis über die prinzipielle Realisierbarkeit zu erbringen. Der Schwerpunkt der prototypischen Realisierung lag vor allem auf den Konzepten, die neue System-Ansätze und Software-Techniken beinhalten, deren Eignung für SAMSON und deren korrekte Funktionsweise aus Vorarbeiten nicht ausreichend abgeleitet werden konnte. Abbildung 7.1 gibt einen Überblick über die durchgeführten Arbeiten.

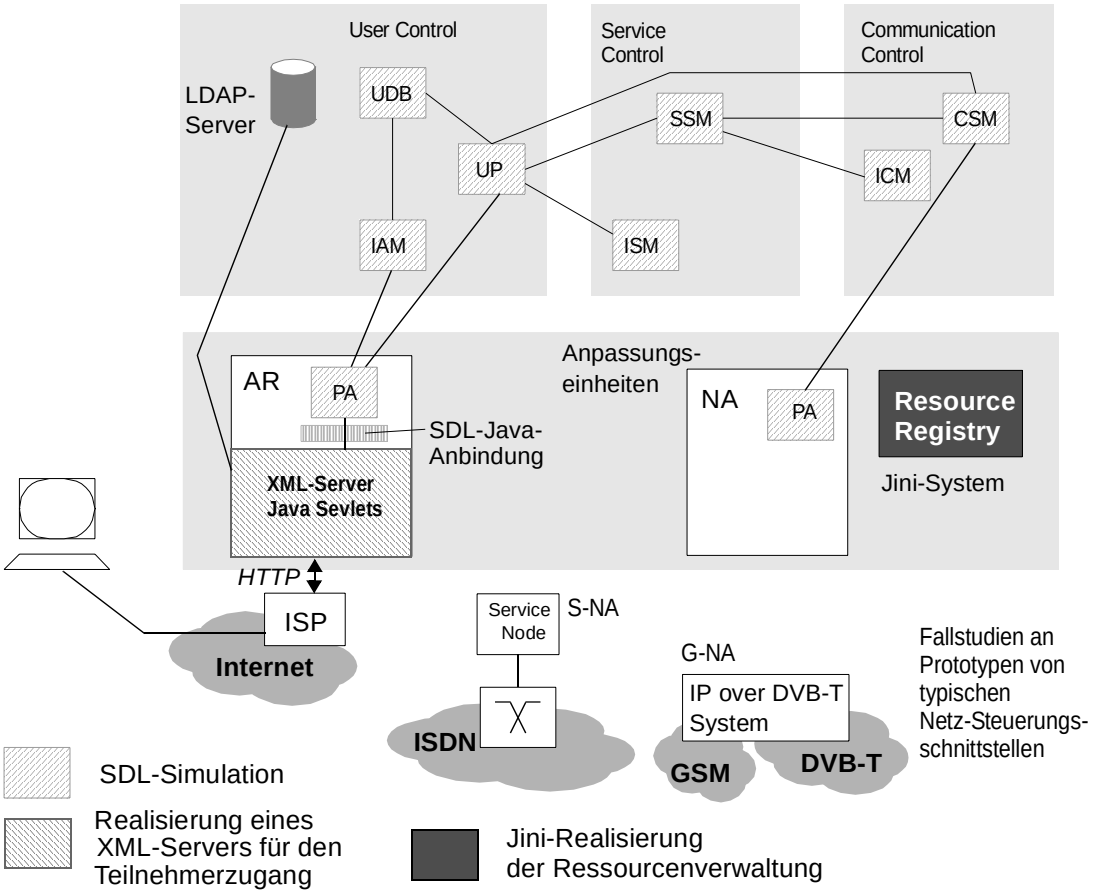


Abbildung 7.1: Übersicht über die prototypische Realisierung

Für die prototypische Entwicklung wurden verschiedene Techniken eingesetzt, um die Konzepte in einer möglichst realen Umgebung testen zu können. Die Dienstzugangsfunktion (Access Ressource) wurde auf einem Standard-Web-Server in Java implementiert. Die Funktionen der Ressourcenverwaltung wurden ebenfalls in Java unter Verwendung einer Jini-Plattform realisiert. Der Prototyp für das Signalisierungssystem (Protokoll SesCP) wurde hingegen nicht in einer realen Umgebung, sondern als SDL-Simulation ausgeführt. Da es Ziel war, die korrekte Funktion des Protokolls zu überprüfen, wurde von unteren Protokollschichten und einer Hardwareplattform abstrahiert.

7.1 XML-basierter Teilnehmerzugang

Die grundlegende Idee des netzunabhängigen Teilnehmerzugangs (siehe Abschnitt 5.2.3) ist es, die Information innerhalb der Teilnehmerverwaltung mit dem Metadatenformat XML zu beschreiben. Somit wird Unabhängigkeit von einem speziellen Darstellungsformat des Endgerätes erreicht. Der Schwerpunkt der Arbeiten lag auf der Umsetzung des Konzeptes in Software. So konnte die Realisierbarkeit nachgewiesen werden. Insbesondere galt es, das Zusammenspiel der ausgewählten Beschreibungs- und Programmiertechniken XML, Java und LDAP zu zeigen.

Abbildung 7.2 stellt die implementierten Funktionen zur Realisierung der Access-Ressource dar. Kernstück ist ein Standard-Web-Server, auf dem die einzelnen Funktionen als Java-Servlet-Bibliotheken (*COCOON* und *XML Processor*) erstellt wurden. Die XML-Dateien werden in einem eigenen Verzeichnis, dem XML-*Repository* abgelegt. Es wurde eine DTD-Datei erstellt, welche die XML-Syntax festlegt. Verschiedene *Stylesheets* (XSL-Dateien) regeln das Ausgabeformat der mit XML-beschriebenen Inhalte. Da die XML-Dateien dynamisch erzeugt werden, wurde eine LDAP-Datenbank implementiert, die die Benutzerdaten speichert. Sie entspricht der SAMSON *User Data Base*.

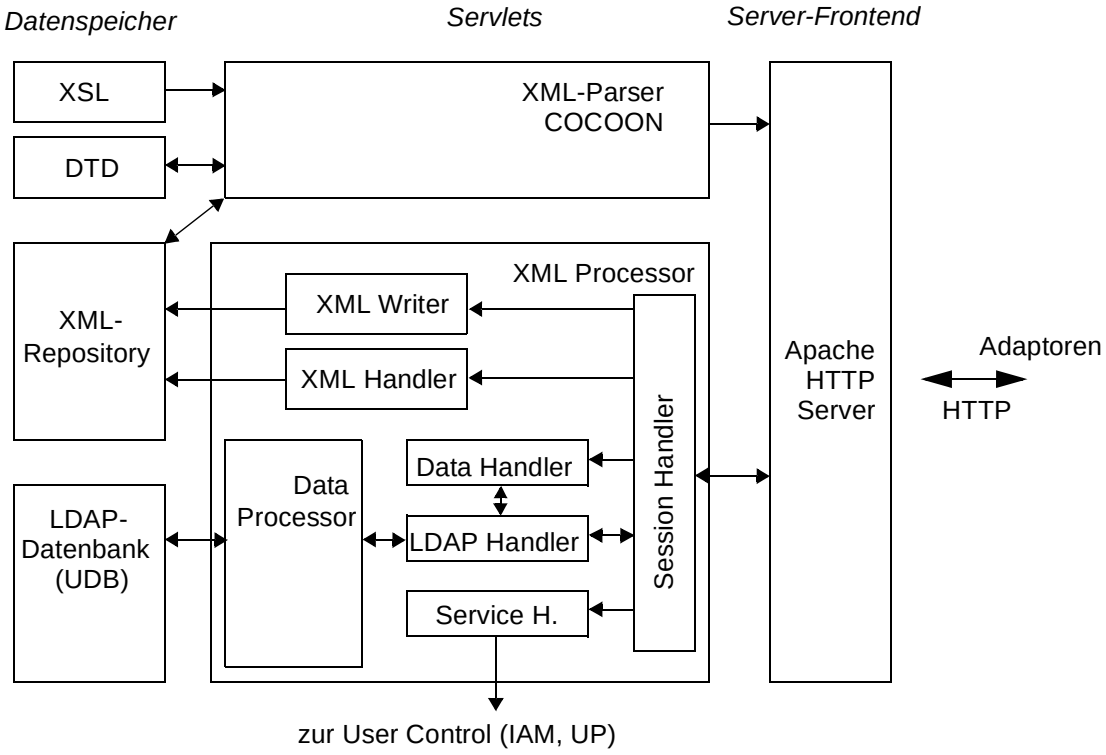


Abbildung 7.2: Programmstruktur der Access-Ressource

Das *Lightweight Directory Access Protocol* (LDAP, [WHK97]) ist ein standardisierter, hierarchisch strukturierter Verzeichnisdienst für IP-Netze, der es erlaubt, Benutzerprofile beliebiger Art zu speichern und im Netz verfügbar zu machen. Dagegen erlaubt XML in Kombination mit *Stylesheets* (XSL) zwar eine strukturierte Dokumentenbeschreibung und Datenausgabe, besitzt aber keine Mechanismen zur Verwaltung und Änderung der Datensätze wie LDAP. Java-Servlets realisieren die Anbindung der LDAP-Datenbank an den XML-Server.

Die Ausgabe und die Formatierung von XML-Dokumenten erfolgt durch das Servlet COCOON. Dieses wurde dem Apache-XML-Projekt[1] entnommen. Es wird beim Abruf einer XML-Datei (Endung .xml) vom Web-Server automatisch aufgerufen. Das gewünschte Ausgabeformat wird anhand der MIME-Parameter der HTTP-Anfrage erkannt und die Datei durch das Servlet mit Hilfe der entsprechenden XSL-Datei in das entsprechende Ausgabeformat umgewandelt (z.B. HTML, WML, VXML). Die Umwandlung der Dokumente erfolgt somit im Server. Mit dieser Realisierung ist man unabhängig davon, ob der verwendete Browser XML-Dokumente darstellen kann.

Die Funktionen des neu erstellten Java-Packages *XML Processor* steuern die Teilnehmerinteraktion und die dynamische Erzeugung der XML-Dateien aus der LDAP-Datenbank, sowie die Anbindung an die Dienststeuerung. Der *Session Handler* ruft für jede Teilnehmerinteraktion (z.B. Klicken auf einer HTML-Seite) die entsprechenden Klassen auf. Die Klassen *Data Handler*, *LDAP Handler* und *Data Processor* bedienen die LDAP-Datenbank und bereiten die Daten auf. *XML Handler* und *XML Writer* erzeugen die gewünschten XML-Dateien für die Ausgabe.

Da die Kommunikation zwischen Teilnehmer (Client) und Dienstserver auf einer HTTP-Transaktion beruht, ist es nicht möglich, Meldungen der Dienststeuerung (Server) beim Teilnehmer (Client) darzustellen. Diese können nur indirekt als Antwort auf eine HTTP-Anfrage des Clients erfolgen. Für direkte Anfragen sind zusätzliche Mechanismen notwendig. Java-Applets können z.B. für den Empfang von Server-Meldungen auf den Client geladen werden. Dies erfordert allerdings eine erweiterte Browser-Plattform.

Insgesamt wurde gezeigt, wie das Konzept der Access Ressource mit den genannten Techniken realisierbar ist. Dabei ist eine umfangreiche Bibliothek an Java-Servlets entstanden, die die gewünschte Zusammenarbeit zwischen dem Web-Server, den XML/XSL-Dateien und der LDAP-Datenbank ermöglichen. Weitere Details können [Ver00] entnommen werden.

7.2 Spezifikation und Simulation des Signalisierungssystems

Um die Funktionsweise des in Kapitel 5 spezifizierten Signalisierungsprotokolls zu überprüfen und zu demonstrieren, wurden die Steuerprozesse der SAMSON-Komponenten auf der Ebene des Signalisierungsprotokolls prototypisch als SDL-Simulation implementiert. Der Schwerpunkt der Simulation lag auf den Signalabläufen und der Interaktion zwischen den Komponenten. Daher wurden die Algorithmen stark vereinfacht und auf eine Spezifikation der transportierten Dienstbeschreibung verzichtet. Bei der Spezifikation mit SDL konnte auf den Erfahrungen eines vorangegangenen Projektes aufgebaut werden, in dem Teile der TINA-Architektur mit SDL spezifiziert und simuliert wurden, um die Funktionsweise mit Beispieldiensten zu validieren [Nis98].

SDL ist eine objektorientierte, formale Spezifikations- und Beschreibungssprache (*Specification and Description Language*) der ITU-T [Z.100]. Sie erlaubt die Systembeschreibung aus

1. Apache-XML-Projekt: HTTP://xml.apache.org/.

verschiedenen Sichten: durch Zustandsautomaten, Blockdiagramme, Signalaustausch und Abstrakten Datentypen. Zusammen mit der formalen Beschreibungssprache *Message Sequence Charts* (MSC, [Z.120]) ist außerdem die Dynamik durch Signalablaufdiagramme darstellbar. Damit kann das SAMSON-Systemmodell einfach realisiert werden. Für den Prototyp wurde das Entwicklungswerkzeug *Tau* eingesetzt [Tel00]. Es erlaubt, *Message Sequence Charts* für die Validierung der Spezifikation zu erzeugen.

SDL-basiertes *Prototyping* spielt für den Funktionsnachweis von komplexen IuK-Systemen eine wichtige Rolle [Kel99]. Durch die Komplexität der Systeme ist es oftmals nicht möglich, in einer realen Umgebung zu testen. Eine SDL-Simulation bietet hier den Vorteil, daß die formale Beschreibung als Grundlage für eine spätere Implementierung dienen kann. Außerdem stellen die graphischen Dokumente eine detaillierte Spezifikation und Dokumentation eines Systems dar. SDL-basiertes *Prototyping* wurde nicht nur in dieser Arbeit, sondern im Rahmen vieler Vorarbeiten an entscheidenden Stellen eingesetzt (z.B. [Kel96, VKK98, Nis98, KAI98, KAI00]).

SDL-Spezifikation der Signalisierungsarchitektur

Das System teilt sich in zwei SDL-Blöcke, die Dienstebene (*service_control_part*) und die Anpassungsebene (*adaptation_part*). Abbildung 7.3 zeigt die Spezifikation der Dienstebene. Die drei Steuerungsbereiche *User Control*, *Service Control* und *Communication Control* sind als SDL-Blöcke spezifiziert. Die jeweils zugehörigen Komponenten sind als SDL-Prozesse beschrieben, wie z.B. im Block user_contol zu sehen. Damit ist bereits die unterste Ebene der Strukturierung bei SDL erreicht. Für eine detaillierte Spezifikation der Komponenten (hinausgehend über die Zustandsautomaten des Signalisierungsprotokolls) ist dies zwar ungünstig und unübersichtlich, aber für den Zweck der funktionalen Spezifikation ausreichend. Nach dem derzeitigen SDL-Standard [Z.100] können SDL-Prozesse nicht hierarchisch strukturiert werden. Um das Signalisierungsverfahren in SesCP richtig abbilden zu können, kommt trotzdem

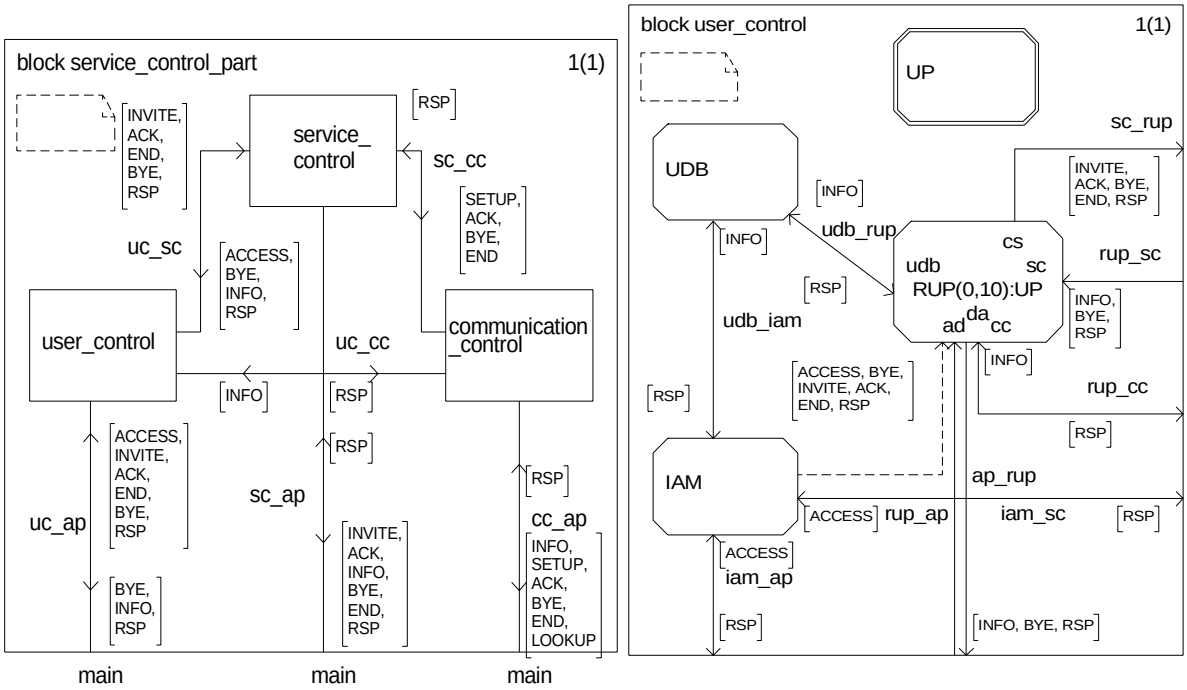


Abbildung 7.3: Spezifikation der Dienstebene des Signalisierungssystems

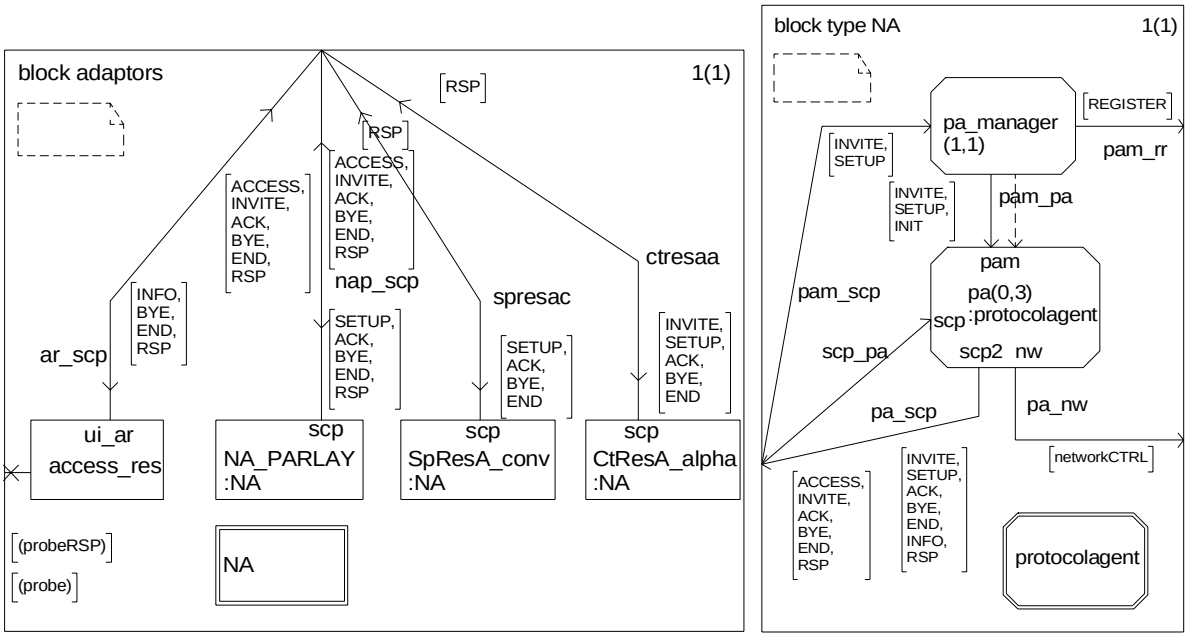


Abbildung 7.4: Spezifikation der Anpassungsebene mit SDL

nur die Spezifikation dieser Komponenten als Prozesse und nicht als Blöcke in Frage, da in SDL keine Instanzierung von Blöcken zur Laufzeit vorgesehen ist (siehe auch [IA97]).

Die Anpassungsebene teilt sich in einen SDL-Block für die Adaptoren (adaptors) und einen SDL-Block für die Ressourcenverwaltung. Der Block adaptors wird durch weitere SDL-Blöcke analog zu den verschiedenen Typen von Adaptoren in vier Bereiche unterteilt (Abbildung 7.4): den Block access_resource und Blocktypen für Netzadaptoren (NA), Spezialressourcenadaptoren (SpResA) und Content-Ressourcen-Adaptoren (CtResA). Hier wurde für jeden Adaptor ein Block statisch instanziert, da eine Simulation der dynamischen Abläufe der Ressourcen-Registrierung nicht Ziel der SDL-Simulation war.

Jeder Adaptor-Block (NA) enthält eine Management-Komponente und dynamisch instanzierbare SDL-Prozesse für die Kommunikation mit der Dienstebene (*protocolagent*). Eine Anbindung an reale Netzressourcen ist vorgesehen, wurde aber nicht durchgeführt. Um den Teilnehmerzugang, der separat von der SDL-Spezifikation implementiert wurde, an das SDL-System anzubinden, wurden im Block access_res zusätzliche Ein-/Ausgabe-Meldungen spezifiziert (Schnittstelle AR).

Kopplung des SDL-Systems mit dem Teilnehmerzugang

Um die Teilnehmerverwaltung an den SDL-Simulator der Signalisierungsarchitektur zu koppeln, waren zusätzliche Komponenten erforderlich. Das verwendete Entwicklungswerkzeug für SDL beinhaltet die Kommunikationsplattform *Postmaster* für die Verknüpfung der einzelnen Werkzeuge wie Editor oder Simulator. Da vom Hersteller des SDL-Werkzeugs, das unter dem Betriebssystem Solaris läuft, keine Bibliothek für eine externe *Postmaster*-Schnittstelle (PM-I) für Linux erhältlich ist, mußte eine Kopplungs-Komponente unter Solaris eingefügt werden. An diese werden die Meldungen aus der Teilnehmerverwaltung über eine TCP/IP-Socketverbindung übergeben.

Für diese Komponente konnte auf eine bereits in vorangegangenen Projekten erstellte Software zurückgegriffen werden [Wei97]. Das Programm sdttcl, das ursprünglich für die Anbindung von Tcl/Tk-Benutzeroberflächen an SDL-Simulatoren entwickelt wurde, konnte auch für

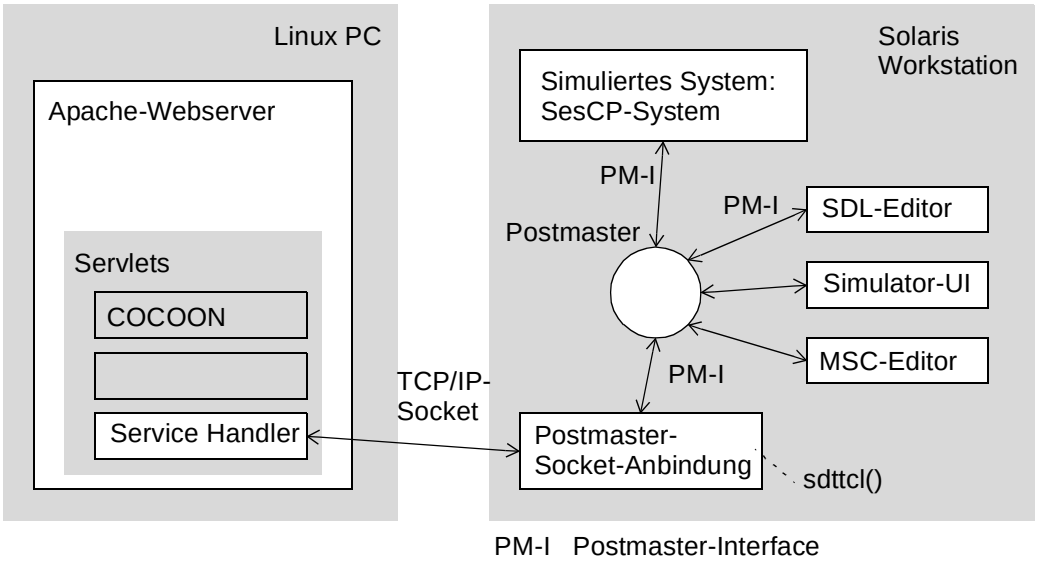


Abbildung 7.5: Anbindung des Teilnehmerzugangs an den SDL-Simulator

das vorliegende Prototypsystem eingesetzt werden. Die Java-Servlets-Bibliothek wurde um die Klasse *Service Handler* erweitert. Abbildung 7.5 illustriert die Kopplung der Teilnehmerverwaltung mit der SDL-Simulation.

7.3 Realisierung der Ressourcenverwaltung mit Jini

Um den in Abschnitt 6.4 beschriebenen Ansatz zur Verwendung der *Service Discovery*-Konzepte für die Ressourcenverwaltung und insbesondere die Verwendung von Jini zu validieren, wurde im Rahmen der vorliegenden Arbeit eine prototypische Implementierung durchgeführt. Grundlage bildete das Objekt *Reggie* (*Jini Lookup Service*) des Jini-Starter-Kits[1]. Diese Software wurde mit den entsprechenden Modifikationen direkt als Basis für die Implementierung der *Resource Registry* verwendet. Der benötigte Web-Server, der das Laden von Java-Klassen ermöglicht, wurde ebenfalls der Jini-Distribution entnommen. Der *Daemon* für die Jini zugrundeliegende *Remote Method Invocation* entstammt der Java2-Distribution.

Implementiert und getestet wurde das System mit drei Adaptoren unterschiedlicher Art: einem Netzadaptor, einem Spezialressourcenadaptor und einem Content-Ressourcen-Adaptor. Kosten und Auslastung wurden während des Testablaufes dynamisch geändert. Es konnte sowohl die einfache Entnahme der Komponentenadresse aus den Attributen als auch die Übergabe von Objekten erfolgreich demonstriert werden. Nähere Einzelheiten können [Leg00] entnommen werden.

Obwohl Jini aus dem Bereich der Bürokommunikation kommt und bisher weder für Telekommunikationsdienste verwendet wurde noch dafür vorgesehen war, konnten nahezu alle funktionalen Anforderungen durch die Implementierung erfüllt werden. Als problematisch erwies sich lediglich die Umsetzung einiger Attribute, da Jini keine Verarbeitung von Parameterlisten (z.B. für unterschiedliche Kodierverfahren) unterstützt. Bei der Dienstauswahl kann nur bei exakter Übereinstimmung der Liste im *Service Template* und in der *Resource Registry* eine

1. Jini Technology Specifications. HTTP://developer.java.sun.com/developer/products/jini/

positive Auswahl getroffen werden. Um diese Einschränkung zu umgehen, wurde für jeden Wert einer Liste ein eigenes Attribut eingefügt.

7.4 Anbindung an reale Netze

Die prinzipielle Realisierbarkeit und der Aufbau von Adaptoren wurden in Kapitel 6 in Fall-Studien gezeigt. Diese Studien beruhen unter anderem auf den Ergebnissen einiger Projekte, die im Umfeld der vorliegenden Arbeit durchgeführt wurden und die sich intensiv mit Schnittstellen zur Steuerung von Informationsflüssen in Kommunikationsnetzen beschäftigten. Eine Anbindung an diese Systeme oder an andere reale Netze wurde im Rahmen dieser Arbeit nicht durchgeführt.

In [GKK00] wurde eine Dienststeuerung für Telekommunikationsdienste ausgehend von einem Sprachbrowser-System realisiert. Der Aufbau von Telefonverbindungen erfolgte über die JTAPI-Schnittstelle einer Siemens HiCom-Telekommunikationsanlage. Die TAPI-Schnittstelle ist ein offener Industriestandard für die Hardware-unabhängige Entwicklung von Applikationen zur Steuerung von Telefonverbindungen. Dieses Projekt ist ein anschauliches Beispiel für die Anbindung von P-Netzadaptoren.

Ein Beispiel für einen S-Netzadaptor ist die Kommunikationssteuerung im Service Node I@C der Firma Siemens, der im Projekt FORSOFT C2 evaluiert und erweitert wurde [Fre99]. Einzelne ISDN-Telekommunikationsverbindungen werden in einem ISDN-S1-Board gekoppelt, um die Konfiguration für einen bestimmten Dienst, z.B. eine Anrufumleitung, herzustellen.

Die Überlegungen zur Klasse der G-Netzadaptoren beruhen auf umfangreichen Implementierungsarbeiten an einem Gateway zur Datenkommunikation über Broadcastnetze [KSZ99, SEK99, SK99, RKS01]. In [KSE00] ist die Architektur und der Prototyp eines Hybridsystems beschrieben, mit dem Informationsflüsse über DVB-T und GSM aus der Kopfstation eines Verteilnetzes gesteuert werden können.

7.5 Fazit

Mit der prototypischen Realisierung wurden verschiedene Ziele verfolgt. Zum einen wurde die Realisierbarkeit ausgewählter Konzepte verifiziert und demonstriert, indem die Spezifikation in Prototypen umgesetzt wurde. Um ferner die korrekte Funktion des Signalisierungsverfahrens SesCP nachzuweisen, wurden die Basiselemente des Protokolls mit SDL spezifiziert und simuliert. Zusätzlich wurde die SDL-Simulation mit dem Teilnehmerzugang gekoppelt, um das Zusammenspiel zu demonstrieren.

Die durchgeführten Arbeiten belegen die prinzipielle Umsetzbarkeit der Serverarchitektur in einer IP-basierten Serverumgebung. Da die prototypische Realisierung parallel zur Spezifikation erfolgte, flossen Ergebnisse direkt in die Spezifikation ein. Durch die SDL-Spezifikation des Signalisierungsprotokolles SesCP konnte in Simulationen nachgewiesen werden, daß das in Kapitel 5 spezifizierte Verhalten tatsächlich zu den gewünschten Signalisierungsabläufen führt.

Die einzelnen Komponenten können als Grundlage für die Implementierung des Gesamtsystems verwendet werden. Auch die SDL-Simulation stellt die Basis für eine Implementierung dar, da das verwendete Werkzeug die automatische Generierung einer Applikation unterstützt. Ein Prototyp des Gesamtsystems erlaubt es dann, konkrete Untersuchungen hinsichtlich Performance und Skalierbarkeit anzustellen.

Zusammenfassung und Ausblick

Die steigende Heterogenität von Kommunikationsnetzen und die Vielfalt an Teilnehmerendgeräten stellen hohe Anforderungen an neue Dienstarchitekturen. Insbesondere für eigenständige Dienstanbieter ist entscheidend, wie existierende Systeme integriert werden können, um möglichst vielen Teilnehmern Informations- und Kommunikationsdienste unabhängig von deren Netzzugang zur Verfügung zu stellen. Bisherige Dienstarchitekturen unterstützen die geforderte Netzunabhängigkeit nur ansatzweise. In der vorliegenden Arbeit wurde mit der Spezifikation der Serverarchitektur SAMSON ein neuer Ansatz für eine Dienstarchitektur beschrieben, die es ermöglicht, Dienste netzunabhängig zu definieren und in heterogenen Kommunikationsnetzen zu steuern.

Da die Entwicklung von Dienstsystemen eine noch junge Ingenieur-Disziplin ist, war es zunächst notwendig, die Begriffe zu definieren und das Umfeld sowie wichtige Systemaspekte zu beschreiben. Die Anforderungen, die sich daraus für netzunabhängige Dienstarchitekturen ergeben, fokussieren sich in der Realisierung der horizontalen Partitionierung zwischen Dienst- und Netzebene und in der Fähigkeit, Dienste netzunabhängig zu beschreiben. Ein wichtiges Kriterium aus Teilnehmersicht ist darüber hinaus ein unabhängiger Teilnehmer- und Dienstzugang. Bestehende und in Forschung befindliche Dienstarchitekturen erfüllen bereits einige der Anforderungen. Sie konzentrieren sich allerdings in ihrer Zielsetzung nur auf Teilaspekte, z.B. programmierbare Schnittstellen, Kopplung ausgewählter Netze, oder sie befriedigen einige Ansprüche durch Modifikationen in der Infrastruktur, wie zum Beispiel bei der CORBA-Middleware-Plattform.

Dienststeuerung in heterogenen Netzen

Die Serverarchitektur SAMSON beschreibt eine Dienstarchitektur für ein Serversystem, das für die unterschiedlichen Kommunikationsnetze eine zentral angeordnete Intelligenz darstellt. Der Schwerpunkt bei den Diensten liegt auf der Steuerung von erweiterten Informations- und Kommunikationsdiensten, die bestehende Basisdienste (z.B. Telefonie) verknüpfen oder um zusätzliche, personalisierbare Merkmale ergänzen.

Adaptoren entkoppeln den Server von den Netzen. Sie übernehmen den Teilnehmerzugang und den Verbindungsaufbau, indem sie als Signalisierungsgateways zwischen dem internen Signalisierungsprotokoll und der netzspezifischen Signalisierung arbeiten. Durch diese Trennung in eine dienstspezifische und eine netzspezifische Signalisierung können unterschiedliche Kommunikationssysteme für die Ausführung eines Dienstes ausgewählt und kombiniert werden.

Für die Kopplung der Adaptoren an die jeweiligen Netzelemente (Vermittlungsknoten, Gateways) wird auf existierende Schnittstellen für einen stellvertretenden Verbindungsaufbau im Netz zurückgegriffen. So kann die Architektur einfach umgesetzt werden, ohne daß aufwendige Änderungen in den Netzen erforderlich sind.

Modellierung der Dienstarchitektur

Die Teilaspekte eines Systems werden aus verschiedenen Sichtweisen (z.B. Komponenten-Sichtweise, Informations-Sichtweise) abstrakt beschreiben. Um alle Systemeigenschaften des Servers spezifizieren zu können, wurden in dieser Arbeit sechs Sichten zur Modellierung von Dienstarchitekturen vorgeschlagen und angewendet. Diese Sichten umfassen statische Systemeigenschaften wie die inneren und die äußeren Schnittstellen und die Informationsstruktur, sowie dynamische Eigenschaften zur Verhaltensbeschreibung. Für nicht-funktionale Merkmale wird ein Verfahren zur Performance-Modellierung des Signalisierungsprotokolls erläutert. Anhand dessen werden konkrete Daten für die Dimensionierung des internen Signalisierungsnetzes aufbereitet.

Getrennte Steuerungsbereiche für Teilnehmer, Dienste und Kommunikationsressourcen

Zur Reduzierung der Komplexität wird die Dienstarchitektur in Anlehnung an das TINA-Konzept in die drei getrennt verwalteten Bereiche Teilnehmersteuerung, Dienststeuerung und Kommunikationssteuerung unterteilt.

Die Teilnehmersteuerung realisiert den Zugang der Teilnehmer zum Server und übernimmt die Verwaltung persönlicher Profile. Sie schirmt damit die Dienststeuerung von teilnehmerspezifischen Details ab, wie z.B. der Autorisierung oder der Endgerätekonfiguration. Damit kann umgekehrt jeder Dienst unabhängig vom Serverzugang aufgerufen werden, wie es für die Realisierung von Dienste-Mobilität gefordert wird. In bestehenden Architekturansätzen ist die Teilnehmerverwaltung zu wenig berücksichtigt.

Die Dienststeuerung selbst ist zudem unabhängig von netzspezifischen Daten und steuert die Zusammensetzung einer abstrakten Kommunikationsbeziehung anhand der Dienstlogik. Die Parameter sind die Rollen und Rechte der Teilnehmer und deren Verknüpfung mit unterschiedlichen Medien. Für die Programmierung der Dienstlogik wurde eine XML-basierte Notation vorgeschlagen.

Die Kommunikationssteuerung bildet das abstrakte Beziehungsmodell der Dienststeuerung für die Adaptoren auf konkrete Netze ab. Dazu bedient sie sich der Daten aus der Teilnehmersteuerung für die Endgeräteanschlüsse und zerlegt eine Kommunikationsbeziehung schrittweise in einzelne Informationspfade für unterschiedliche Netze. Soweit die Kommunikationsnetze an ihren Schnittstellen zu Adaptoren mehr als Punkt-zu-Punkt-Verbindungen zulassen, wie zum Beispiel die Einrichtung von Konferenzen, wird dies bei der Auswahl der Adaptoren berücksichtigt. Dies vermeidet eine Zergliederung einer Kommunikationsbeziehung in kleine, aufwendig zu verwaltende Verbindungsabschnitte.

Dynamisch verfeinerbare Dienstbeschreibung

Multimedia-Dienste mit mehreren Teilnehmern können aufgrund ihrer Komplexität mit einem herkömmlichen, auf Zustandsautomaten basierten Rufmodell (wie z.B. beim Intelligenten Netz) nicht mehr sinnvoll dargestellt werden. Daher wird in allen drei Steuerungsbereichen des Servers ein objekt-orientiertes Modell als Grundlage verwendet. Dies ermöglicht die Beschreibung eines umfassenden Dienstspektrums und ist nicht auf bestimmte Dienstklassen beschränkt. Diese Beschreibung wird bei ihrer Verarbeitung durch die Steuerungskomponen-

ten ausgehend vom Dienstaufruf des Teilnehmers, der nur wenige qualitative Parameter umfaßt, bis zu einer detaillierten Beschreibung auf Kommunikationsebene schrittweise erweitert. Durch das einheitliche Modell vereinfacht sich der Austausch von Parametern zwischen den Steuerungsbereichen.

Signalisierungsprotokoll

Für die Signalisierung innerhalb der Dienststeuerung wurde ein neues Signalisierungsprotokoll spezifiziert, das auf dem *Session Initiation Protocol* der IETF (SIP) basiert. Daher wird zum einen durch die Ausrichtung auf das Internet eine einfache Integration der Architekturkomponenten in zukünftige, Internet-basierte Signalisierungsnetze ermöglicht, wie z.B. beim UMTS. Zum zweiten können bestehende SIP-Server mit geringer Modifikation für die Signalisierung eingesetzt werden. Das neue Signalisierungsprotokoll SesCP unterstützt die komponentenorientierte Struktur des Servers hinsichtlich der drei oben genannten Steuerungsbereiche. Die Komponenten werden dabei als Redirect-Server und Proxy-Server realisiert, die nicht nur Signalisierungsnachrichten, sondern auch die Dienstbeschreibung verarbeiten. Letztere wird in einer neu entwickelten Erweiterung des IETF *Session Description Protocols* als Signalisierungsnutzlast transportiert.

Adaptoren zur Anpassung an heterogene Netze und Endgeräte

Zur Umsetzung der Server-internen Signalisierung auf die Signalisierungsprotokolle oder Programmierschnittstellen der Netzelemente werden Adaptoren eingesetzt. Sie werden in einer neu aufgestellten Klassifikation in drei Gruppen eingeteilt. P-Adaptoren koppeln den Server an Netzelemente, die einen stellvertretenden Verbindungsaufbau erlauben, wie z.B. das Parlay-API. Netzelemente, wie z.B. Media-Gateways, die einen Verbindungsaufbau vom Netzrand her unterstützen, werden über G-Adaptoren angesprochen. Zu ihnen zählen auch die Kopfstellen von Verteilnetzen. Die dritte Klasse (S-Adaptoren) realisiert einen stellvertretenden Verbindungsaufbau durch die Kopplung von Teilnehmeranschlüssen im Adaptor. Darüber hinaus werden Adaptoren für Spezialressourcen und für Inhalte-Server spezifiziert. Protokollarchitekturen und Ablaufdiagramme verdeutlichen die Funktionsweise der Adaptoren.

Für den Teilnehmer- und den Dienstzugang werden die Adaptoren durch eine neuartige Komponente unterstützt, in der ein XML-Server die Ausgabe von Signalisierungsnachrichten (z.B. Paßwortabfrage, Dienstangebot) an Endgeräte mit unterschiedlichen Darstellungsfähigkeiten, z.B. als HTML, WML oder VXML, ermöglicht.

Die Auswahl der Adaptoren durch die Kommunikationssteuerung erfolgt nach dem Service-Discovery-Konzept über eine zentrale Komponente zur Ressourcenverwaltung. Die Adaptoren registrieren dort ihre Fähigkeiten (z.B. Netzart, Qualität). Dabei werden auch dynamische Parameter, wie Auslastung und Preise berücksichtigt. Die Serverarchitektur kann sich dadurch automatisch an eine veränderte Umgebung anpassen.

Prototypische Realisierung

Wesentliche Teile der Serverarchitektur wurden prototypisch realisiert, um die Funktionalität und die Realisierbarkeit praktisch zu evaluieren. Der Kern des Signalisierungsprotokolls wurde mit SDL spezifiziert und simuliert. Die XML-basierte Dienstzugangskomponente wurde mit Java-Servlets auf einem Web-Server implementiert und an den SDL-Simulator angebunden. Ein Prototyp der Ressourcenverwaltung entstand auf Basis von Jini. Für die Anbindung des Servers an reale Netze stützt sich die Arbeit auf umfangreiche Vorarbeiten, in denen Prototypen zur Dienststeuerung in IN-Systemen und für IP-basierte Verteilnetze entwikkelt wurden.

Wirtschaftliche Bedeutung

Die in dieser Arbeit beschriebene Serverarchitektur unterstützt das Geschäftsmodell eines eigenständigen Dienstanbieters, der den Teilnehmern verschiedener Kommunikationsnetze seine erweiterten Dienste zur Verfügung stellt. Im Gegensatz zu traditionellen Dienstarchitekturen, die Dienste nur über ihre spezifische Netzinfrastruktur bereitstellen, erlaubt der vorliegende Ansatz nahezu beliebige Dienstzugangs- und Ausführungsnetze. Die dadurch erzielte globale Verfügbarkeit des Teilnehmerprofils ist für Dienstanbieter ein weiteres vorteilhaftes Merkmal der Serverarchitektur. Die fortschreitende Standardisierung von sicheren APIs, die den Zugriff externer Dienstanbieter auf Kommunikationsnetze erlauben, z.B. Parlay, beweist die praktische Umsetzbarkeit der vorliegenden Architektur.

Ausblick

Aus dieser Arbeit ergibt sich in verschiedener Hinsicht weiterer Forschungsbedarf. Für einen Einsatz des Servers in öffentlichen Netzen sind in der Teilnehmersteuerung geeignete Mechanismen für die Autorisierung und Authentisierung zu integrieren. Hierbei kann teilweise auf die Ansätze im *Session Initiation Protocol* zurückgegriffen werden. Der Registrierungsvorgang ist aber auch dort noch nicht festgelegt. Darüber hinaus ist für ein kommerzielles System die Frage der Vergebührung zu klären. Ein Abrechnungsmechanismus betrifft nicht nur die Teilnehmersteuerung, sondern auch die Dienststeuerung. Berechenbare Ereignisse im Dienstablauf müssen dort erfaßt und z.B. mit Tickets weiterverarbeitet werden. Weitere Sicherheits- und Autorisierungsmechanismen sind für die Zusammenarbeit verschiedener Dienstanbieter (vertikale Partitionierung) zu entwickeln.

Das vorgeschlagene Konzept für die XML-basierte Dienstprogrammierung ist noch zu verfeinern. Dabei ist genau abzuwägen, welche Dienste durch autorisierte Programmierer und welche durch die Teilnehmer selbst erstellt werden. Danach richtet sich die Ausgestaltung der Dienstentwicklungsumgebung, die nicht Bestandteil dieser Arbeit war. Aufbauend auf [SK01] werden solche Fragestellungen in einem weiteren Projekt am Lehrstuhl für Kommunikationsnetze bearbeitet. Ein Ziel ist es, die Anforderungsbeschreibung für die Dienstentwicklung zu formalisieren und zu automatisieren.

Ebenfalls nicht näher behandelt wurde das Problem der Interaktionen zwischen gleichzeitig aktiven Diensten, z.B. [Fri96]. Insbesondere in dem hier zu Grunde gelegten multimedialen Umfeld ergeben sich neue Ursachen für die Entstehung von ungewünschten Interaktionen [Kel99a]. Es ist zu untersuchen, in welchem Maße bereits die gewählte Struktur der Dienstarchitektur das Entstehen ungewünschter Interaktionen vermeidet [KSM00]. Darüber hinaus sind Mechanismen zur Aufdeckung und Behandlung von Interaktionen zu integrieren.

Weitere offene Fragen betreffen die Mobilität der Teilnehmer. Während ein Dienst in der Serverarchitektur unabhängig vom Zugangsnetz oder dem Endgerät aufgerufen werden kann, ist ein Wechsel der Zugangsart nur durch Unterbrechung des Dienstes möglich. Notwendig ist ein solcher Übergang vor allem für mobile Teilnehmer, die einen Dienst unterbrechungsfrei über unterschiedliche Funknetze nutzen wollen. Innerhalb eines Mobilfunknetzes sind unterbrechungsfreie Handover bereits realisiert. Ein sogenannter vertikaler Handover zwischen verschiedenen Netzen stellt eine große Herausforderung an die Forschung dar.

Um in einem mobilen Endgerät, das z.B. ein Fahrzeug sein kann, verschiedene Zugangsnetze nutzen zu können, ist das Endsystem von den Netztechnologien zu entkoppeln. Auch hier bietet sich ein Konzept an, das die in der vorliegenden Arbeit beschriebenen Adaptoren verwendet. In Zusammenarbeit mit einem Automobilhersteller wurden die Grundprinzipien der vorliegenden Architektur bereits in ein Konzept für ein Kommunikationsgateway in einem Fahrzeug umgesetzt [KBS01, KVS01, KSV99a, KSV99b]. Dieses kann Endgeräte über aktuell verfügbare Funknetze mit einem Dienstserver im Festnetz verbinden.

Literaturverzeichnis

Hinweis: Literaturangaben, deren Verweis fettgedruckt ist, sind Veröffentlichungen des Verfassers, die als Autor oder Co-Autor erstellt wurden. Vorveröffentlichungen zu dieser Arbeit entsprechend §4, Satz (5) der Promotionsordnung der Technischen Universität München sind zudem mit * gekennzeichnet.

[AC227] ACTS 227 SCREEN. *Service Creation Engineering Environment.* Final Report, 1999.

[ACG99] R. Arsaut, J.-M. Chenevat, S. Gorse, J.-M. Pageot. *Multi-Networks Service Platform*. ICIN'00, Bordeaux, Januar 2000, S. 347-351.

[ACH96] C. Aurrecoechea, A. Campbell, L. Hauw. *A Review of QoS Architectures*. 4th IFIP International Workshop on Quality of Service, Paris, März 1996.

[Asa99] K. Asatani. *IP and Telecommunication Integration: De Jure and De Facto Standards Have Entered a New Era*. IEEE Communications Magazine, Vol. 37, No. 7, Juli 1999, S. 140-147.

[ATMF] ATM Forum. HTTP://www.atmforum.com/.

[BBD00] S. Beddus, G. Bruce, S. Davis. *Opening Up Networks with JAIN Parlay.* IEEE Communications Magazine, Vol. 38, No. 4, April 2000.

[BCB01] G. Benini, S. Cetiner, O. Becker. *Extended Service Creation with CPL*. Unveröffentlichter Bericht, Siemens AG, ICN WN CC AK, 2001.

[BCC00] A. Brusilovski, J. Buller, L. Conroy, V. Gurbani, L. Slutsman. *PSTN Internet Notification (PIN) Proposed Architecture, Services and Protocols*. In Proceedings ICIN'00, Bordeaux, Januar 2000, S. 106-110.

[BCL00] U. Bittroff, C. Capellmann, C. Lorang, J. Pageot, F. Panken. *ITAU: the IN - TINA Adaptation Unit*. IEEE IN'2000, Kapstadt, Mai 2000.

[BD00] B. Brügge, A. Dutoit. *Object-Oriented Software Engineering - Conquering Complex and Changing Systems*. Prentice Hall, London, 2000.

[BHG00] H. Berndt, T. Hamada, P. Graubmann. *TINA: Its Achievements and its Future Directions*. IEEE Communications Surveys, Vol. 3, No. 1, 2000.

[BJM00] R. Brennan, B. Jennings, C. McArdle, T. Curran. *Evolutionary Trends in Intelligent Networks*. IEEE Communications Magazine, Vol. 38, No. 6, Juni 2000, S. 86-93.

[BKE00] C. Bettstetter, W. Kellerer, J. Eberspächer. *Personal Profile Mobility for Ubiquitous Service Usage.* Book of Visions 2000. IST Wireless Strategic Initiative (WSI), Version 1.0, November 2000.

[BMR96] F. Buschmann, et al. *Pattern-oriented Software Architecture*. Wiley, 1996.

[Boc97] P. Bocker. *ISDN - Digitale Netze für Sprach-, Text-, Daten-, Video- und Multimediakommunikation*. Springer-Verlag, Berlin, 1997.

[Bro97] M. Broy. *Mathematical Methods in System and Software Engineering.* In: Marktoberdorf Summer School 1996. Springer Verlag, NATO ASI Series F, 1997.

[BS01] M. Broy, K. Stolen. *Specification and Development of Interactive Systems.* Springer, Berlin, 2001

[BS96] M. Bafutto, M. Schopp. *Network Performance and Capacity Figures of Intelligent Networks based on the ITU-TS IN CS 1*. AIN'96, Passau, 1996, S. 15-29.

[BVE99] C. Bettstetter, H.-J. Vögel, J. Eberspächer. *GSM Phase 2+ General Packet Radio Service GPRS - Architecture, Protocols, and Air Interface.* IEEE Communications Surveys, Vol. 2, No. 3, 1999.

[CA99] A. Couturier, L. Anquetil. *A TINA unified service layer for IN and VoIP.* TINA'99, Hawaii, April 1999, S. 287-294.

[Cam00] B. Campbell. *Framework for SIP Call Control Extensions*. draft-ietf-sip-cc-framework-00, März 2000. - work in progress -

[CDG00] T.-C. Chiang, et al. *IN Services for Converged (Internet) Telephony.* IEEE Communications Magazine, Vol. 38, No. 6, Juni 2000, S. 108-115.

[CGR00] F. Cuervo, et al. *Megaco Protocol Version 1.0*. IETF RFC 3015, Nov. 2000.

[CL00] S. Chou, Y. Lin. *Computer Telephony Integration and its Application*. In IEEE Communications Surveys, Vol. 3, No. 1, 2000.

[Don00] S. Donovan. *The SIP INFO Method*. IETF RFC 2976, Oktober 2000.

[Ebe96] J. Eberspächer, et al. *Architekturen und Verfahren der Vermittlungstechnik*. ITG-Fachausschuß 5.2, VDE-Verlag, Berlin, 1996.

[EG00] E. Evloguieva, R. Glitho. *Plugging IN SCPs in SIP Networks*. IEEE Intelligent Network Workshop, Kapstadt, Mai 2000.

[EP00] J. Eberspächer, A. Picot (Hrsg.). *Vision 21 - Perspektiven für die Informations- und Kommunikationstechnik im 21. Jahrhundert*. Münchner Kreis, 2000.

[ETS00] ETSI Technical Specification ETSI TS 101 329-2 v.1.1.1. *Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON); End to End Quality of Service in TIPHON Systems; Part 2*. ETSI, Juli 2000.

[EVB01] J. Eberspächer, H.-J. Vögel, C. Bettstetter. *GSM Global System for Mobile Communication*. Teubner, Stuttgart, 3. Auflage, 2001.

[FGK97] I. Faynberg, L. Gabuzda, M. Kaplan, N. Shah. *The Intelligent Network Standards*. McGraw-Hill, 1997.

[FGM99] R. Fielding, et al. *Hypertext Transfer Protocol -- HTTP/1.1*. IETF Network Working Group, Request For Comments, RFC 2616, June 1999.

[Fre99] M. Fresson. *IN based Video Conference Services for CSCW-Systems*. Diplomabeit, Lehrstuhl für Kommunikationsnetze, Technische Universität München, 1999.

[Fri96] N. Fritsche. *Vermittlungsarchitektur mit getrennter Ruf- und Leistungsmerkmalsteuerung*. Dissertation, Technische Universität München, Lehrstuhl für Kommunikationsnetze. Herbert Utz Verlag, München, 1996.

[GHH99] C. Gbaguidi, J.-P. Hubaux, M. Hamdi, A. Tantawi. *A Programmable Architecture for the Provision of Hybrid Services*. IEEE Communications Magazine, Vol. 37, No. 7, Juli 1999, S. 110-116.

[GHP99] C. Gbaguidi, J.-P. Hubaux, G. Pacifici, A. Tantawi. *Integration of Internet and Telecommunications: An Architecture for Hybrid Services.* In IEEE JSAC, Vol. 17, No. 9, September 1999, S. 1563-1579.

[GKK00] B. Guedhami, C. Klein, W. Kellerer: *Web Enabled Telecommunication Service Control Using VoxML*. SmartNet2000, Sixth IFIP International Conference on Intelligence in Networks, Wien, September 2000, S. 597-621.

[GKM01a] J. Glasmann, W. Kellerer, H. Müller. *Service Development and Deployment in H.323 and SIP.* ISCC2001, IEEE International Symposium on Computers and Communication, Hammamet, Juli 2001.

[GKM01b] J. Glasmann, W. Kellerer, H. Müller. *Service Architectures in H.323 and SIP - A Comparison*. Submitted to IEEE Communications Surveys, 2001.

[GP01] M. Griss, G. Pour. *Accelerating Development with Agent Components*. IEEE Computer, Vol. 34, No. 5, May 2001.

[GRR00] N. Greene, M. Ramalho, B. Rosen. *Media Gateway Protocol Architecture and Requirements*. IETF RFC 2805, April 2000.

[H.323] ITU-T. Recommendation H.323: *Packet-Based Multimedia Communications System*. 1998.

[H.450] ITU-T. Recommendation H.450: *Generic Functional Protocol for the Support of Supplementary Services in H.323*. 1998.

[Haf94] A. Hafid. *On JVTOS QoS Experiments*. GMD-Studie Nr. 236, Sankt Augustin, 1994.

[HHL01] J. Hintelmann, R. Hofmann, F. Lemmen, A. Mitschele-Thiel, B. Müller-Clostermann. *Applying techniques and tools for the performmance engineering of SDL systems*. Computer Networks, Elsevier, Vol. 35, No. 6, 2001, S. 647-665.

[Hin98a] U. Hinkel. *Home Shopping - Die Spezifikation einer Kommunikationsanwendung in FOCUS*. Technical Report TUM-I9808, 1998.

[Hin98b] U. Hinkel. *Formale, semantische Fundierung und eine darauf abgestützte Verifikationsmethode für SDL*. Dissertation, Fakultät für Informatik, Technische Universität München, 1998.

[HJ98] M. Handley, V. Jacobson. *SDP: Session Description Protocol*. IETF RFC 2327, April 1998.

[HKG97] E. Holz, O. Kath, M. Geipl, G. Lin, V. Vogel. *The CAMOUFLAGE Project - Introduction of TINA into Telecommunication Legacy Systems*. TINA'97, Santiago, November 1997, S. 206-215.

[HKS97] U. Hinkel, W. Kellerer, P. Sties. *Multimediale Anwendungen in der Telekommunikation -Szenarien und Abläufe-*. Technical Report TUM-LKN-TR-9702, 1997.

[HSS99] M. Handley, H. Schulzrinne, E. Schooler, J. Rosenberg. *SIP: Session Initiation Protocol.* IETF RFC 2543, März 1999.

[HT01] H. Holma, A. Toskala (Hrsg.). *WCDMA for UMTS*. Wiley, Chichester, 2001.

[Hun95] J. Hunt. *Service Architecture*. In A. Clarke, et al. (Hrsg.): Bringing Telecommunication Services to the People (IS&N'95), LNCS 998, S. 445-446, Springer, 1995.

[HZ98] J.-P. Hubaux, S. Znaty. *Guest Editorial: Telecommunication Services as a new discipline*. ICON Journal, Special Issue on Telecommunication Service Engineering, Vol. 1, Nr. 1, Baltzer Science Publishers, 1998.

[I.112] ITU-T. Recommendation I.112: *Vocabulary of Terms for ISDNs*. 1993.

[I.130] ITU-T. Recommendation I.130: *Method for the Characterization of Telecommunication Services Supported by ISDN and Network Capabilities of an ISDN*. 1988.

[IA97] A. Iselt. A. Autenrieth. *An SDL-based Platform for the Simulation of Communication Networks using Dynamic Block Instantiations*. SDL'97, Evry, Sept. 1997.

[ILM99] Y. Inoue, M. Lapierre, C. Mossotto. *The TINA Book - A Co-operative Solution for a Competitive World*. Prentice Hall Europe, London, 1999.

[JAIN] HTTP://java.sun.com/products/jain/.

[Jaj99] A. Jajszczyk. *What is the Future of Telecommunications Networking?* IEEE Communications Magazine, Vol. 37, No. 6, Juni 1999, S. 12-20.

[JK00] C. Jacobi, W. Kellerer. *Total Cost of Ownership von IN Diensten*. Interner Projektbericht (FORSOFT), Technische Universität München, Februar 2000.

[JK99] C. Jacobi, W. Kellerer. *Kostenmanagement für den Lebenszyklus dienstespezifischer Software in der Telekommunikation*. Technical Report TUM-LKN-TR-9902, 1999.

[KAI00] W. Kellerer, A. Autenrieth, and A. Iselt. *Experiences with SDL based protocol engineering in education*. Journal of Computer Science Education, Swets & Zeitlinger Publishers, Volume 10, Number 3, 2000, S. 225-241.

[KAI98] W. Kellerer, A. Autenrieth, A. Iselt. *SDL based protocol engineering and visualization for education: ISDN Q.931 case study.* Proceedings FORTE/PSTV'98 - ECASP Session, Paris, France, November 1998, S. 137-145.

[KBS01] W. Kellerer, C. Bettstetter, C. Schwingenschlögl, P. Sties, K. Steinberg, H.-J. Vögel. *(Auto-)Mobile Communication in a Heterogeneous and Converged World.* IEEE Personal Communications Magazine, Vol.8, No.6, Dezember 2001, S. 41-47.

[Kel00]* W. Kellerer. *A Versatile Network Independent Server Architecture For Multimedia Information and Communication Services*. SmartNet2000, Sixth IFIP International Conference on Intelligence in Networks, Wien, September 2000, S. 331-350.

[Kel01a]* W. Kellerer. *Server Architecture for Network Independent Multimedia Service Control*. ITG Workshop IP in Telekommunikationsnetzen, Bremen, Januar 2001.

[Kel01b]* W. Kellerer. *Intelligence on Top of the Networks: SIP based Service Control Layer Signaling*. IN2001, IEEE Intelligent Network Workshop, Boston, Mai 2001.

[Kel98] W. Kellerer. *Dienstarchitekturen in der Telekommunikation -Evolution, Methoden und Vergleich-*. Technical Report TUM-LKN-TR-9801, 1998.

[Kel99a] W. Kellerer. S*ervice und Feature Interactions - Neue Aspekte durch Multimedia und Dienstekonvergenz*. Technical Report TUM-LKN-TR-9901, 1999.

[Kel99b] W. Kellerer. *Prototyping von Telekommunikationssystemen mit SDL*. it + ti - Informationstechnik und Technische Informatik, 41 (1999) Heft 3, Oldenbourg Verlag, 1999, S. 21-28.

[KIR96] W. Kellerer, A. Iselt, R. Riek. *Specification, Simulation and Implementation of an Advanced OSI Data-Link Protocol on an embedded Microcontroller System.* FORTE/PSTV'96, Kaiserslautern, Oktober 1998, S. 419-434.

[KK99] M. Korpi, V. Kumar. *Supplementary Services in the H.323 IP Telephony Network.* IEEE Communications Magazine, Vol. 37, No. 7, 1999, S. 118-125.

[KMS00]* W. Kellerer, P. Moritz, P. Sties. *System zur netzunabhängigen Steuerung von Diensten*. Patentanmeldung, September 2000, Deutsches Patentamt.

[Kni93] R. Knight (Ed.). *Service Description Framework and B-ISDN Service Descriptions*. Deliverable RACE Project R2044 MAGIC, 1993.

[Kni00] D. Knight. Broadband Signaling Explained. Wiley, 2000.

[KQ98] W. Kellerer, B. Quendt. *Multimedia Service Architectures - An Overview.* EUNICE'98, Open European Summer School on Network Management and Operation, München, Aug. /Sept. 1998, S. 67-76.

[Kri97] L. Kristiansen. *TINA-C Service Architecture*, Version 5.0, TINA-C, 1997.

[Kru99] P. Kruchten. *The Rational Unified Process*. Addison-Wesley, 1999.

[KS01]* W. Kellerer, P. Sties. S*ignalisierungsplattform zur netzübergreifenden Dienststeuerung in heterogener TIME Infrastruktur*. KIVS'01, GI/VDE/ITG-Konferenz Kommunikation in verteilten Systemen, Hamburg, Februar 2001, S. 391-401.

[KSE00] W. Kellerer, P. Sties, J. Eberspächer. *IP based Enhanced Data Casting Services over Radio Broadcast Networks.* ECUMN'00, IEEE European Conference on Universal Multiservice Networks, Colmar, Oktober 2000, S. 195-203.

[KSM00] W. Kellerer, P. Sties, P. Moritz. *Service Interactions beyond IN: The new Challenge for Multimedia and Convergence*. ICIN00, International Conference on Intelligence in Networks, Bordeaux, Januar 2000, S. 277-282.

[KSN97] H. Kim, F. Steegmans, N. Natarajan, J. Rajahalme. *Managing TINA Streams that use Internet Transport*. TINA'97, Santiago, November 1997, S. 111-118.

[KSS00] W. Kellerer, A. Schmidt, P. Sties. *Strukturierte Software Entwicklung von Informations- und Kommunikationsdiensten*. Technical Report TUM-LKN-TR-0001, 2001.

[KSV99a] W. Kellerer, K. E. Steinberg, H.-J. Vögel. *Verfahren zur Übertragung von Informationen an einen mobilen Empfänger.* Patentanmeldung, Februar 2000, Deutsches Patentamt.

[KSV99b] W. Kellerer, K. E. Steinberg, H.-J. Vögel. *System zur adaptiven Auswahl von Übertragungssystemen für mobile Kommunikation*. Patentanmeldung, Februar 2000, Deutsches Patentamt.

[KSZ99] W. Kellerer, P. Sties, G. Zurek-Terhardt. *System zur Datenübertragung von einem Anbieter zu einem Benutzer.* Patentanmeldung, März 1999. Deutsches Patentamt.

[KTG00] J. Keijzer, D. Tait, R. Goedman. *JAIN: A New Approach to Services in Communication Networks*. IEEE Communications Magazine, Vol. 38, No. 1, Januar 2000, S. 94-99.

[KVS01] W. Kellerer, H.-J. Vögel, K.-E. Steinberg. *A Communication Gateway for Infrastructure Independent Wireless Access*. 3Gwireless2001, IEEE Conference on Third Generation Wireless and Beyond, San Francisco, Mai/Juni 2001.

[KW95] H.-U. Küpper, J. Weber. *Grundbegriffe des Controlling*. Schäffer-Poeschel, Stuttgart, 1995.

[Leg00] M. J. Martin Leguey. *Service Discovery for an Advanced Communication Server Architecture*. Diplomarbeit, Lehrstuhl für Kommunikationsnetze, Technische Universität München, 2000.

[Lin93] R. van der Linden. *An Overview of ANSA*. Architecture Report, Architecture Projects Management Limited, Poseidon House, Cambridge, UK, 1995.

[LKH99] F. Lodge, K. Kimbler, M. Hubert. *Alignment of the TOSCA and SCREEN Approaches to Service Creation*. IS&N'99, Barcelona, April 1999, S. 277-290.

[LLF98] Yu Lu et al. *Universal Service Platform: Implementing TINA-based service platform over the existing infrastructure*. ICIN'98, Bordeaux, Juni 1998, S. 144-148.

[Low97] C. Low. *Integrating Communication Services.* IEEE Communications Magazine, Vol. 35, No. 6, 1997, S. 164-169.

[LS00] J. Lennox, H. Schulzrinne. *CPL A Language For User Control Of Internet Telephony Services*. IETF Internet Draft, draft-ietf-iptel-cpl-04.txt, November 2000. - work in progress-

[M.3010] ITU-T. Recommendation M.3010: *Principles for a Telecommunications Management Network.* 1992.

[MC00] M. Mampaey, A. Couturier. *Using TINA Concepts for IN Evolution*. IEEE Communications Magazine, Vol. 38 No. 6, Juni 2000, S. 94-99.

[McG00] R. McGrath. *Discovery and Its Discontents: Discovery Protocols for Ubiquitous Computing*. Report, NCSA, University of Illinois, 2000.

[MU01] S. Moyer, A. Umar. *The Impact of Network Convergence on Telecommunications Software*. IEEE Communications Magazine, Vol. 39, No. 1, Januar 2001, S. 78-84.

[Mül00] Frank Müller-Veerse. *Mobile Commerce Report*. Durlacher Research Ltd., London, Bonn, August 2000, HTTP://www.durlacher.com/.

[Mül96] H. Müller. *Flexible Signalisierungsarchitektur für Multimediadienste mit heterogenen Endgeräten*. Dissertation, Technische Universität München, Lehrstuhl für Kommunikationsnetze. Herbert Utz Verlag, München, 1996.

[Nis98] A. Nispel. *SDL-Spezifikation einer TINA-Dienststeuerung und Validierung mit Beispieldiensten*. Diplomarbeit am Lehrstuhl für Kommunikationsnetze, Technische Universität München, 1998.

[OMG] Object Management Group. HTTP://www.omg.org/.

[OT99] J. Ott, J. Toga. *ITU-T Standardization Activities for Interactive Multimedia Communications on Packet-Based Networks: H.323 and Related Recommendations*. Computer Networks, Vol. 31, 1999, S. 205-224.

[P103] EURESCOM Project P103. *Framework for Service Description with Supporting Architecture*. P103 (Evolution of the Intelligent Network) Final Report, 1994.

[P508] EURESCOM Project P508. *Evolution, Migration Paths and Interworking to TINA*. Deliverable 2, Volumes 1-8, Final Report. April 1997.

[P909] EURESCOM P909-GI. *Enabling Technologies for IN Evolution and IN-Internet Integration*. Deliverable 1/2, März 2000.

[P916] EURESCOM P 916-PF. *Supporting of H.323 by IN*. Deliverable 1/2, 2000.

[Par00] The Parlay Group. HTTP://www.parlay.org/, 2000.

[PC00] S. Petrack, L. Conroy. *The PINT Service Protocol: Extensions to SIP and SDP for IP Access to Telephone Call Services*. IETF RFC 2848, Juni 2000.

[Q.1228] ITU-T. Recommendation Q.1228: *Interface Recommendation for Intelligent Network Capability Set 2*. 1997.

[Q.12xx] ITU-T. Recommendation Series Q.12xx: *Intelligent Network.*

[Q.2931] ITU-T. Recommendation Q.2931: *B-ISDN - DSS 2 - UNI Layer 3 Specification for Basic Call/Connection Control*. 1995.

[Q.2982] ITU-T. Recommendation Q.2982: *B-ISDN - DSS 2 - Q.9231-based Separated Call Control Protocol*. 1999.

[Q.2983] ITU-T. Recommendation Q.2983: *B-ISDN - DSS 2 - Bearer Control Protocol.* 1999.

[Q.931] ITU-T. Recommendation Q.931: *Digital Subscriber Signaling System No.1 (DSS1) - ISDN UNI Layer 3 Specification for Basic Call Control*. 1993.

[Que00] B. Quendt. *Agentenunterstützte Steuerung von Multimediadiensten.* Dissertation, Technische Universität München, Lehrstuhl für Kommunikationsnetze. Herbert Utz Verlag, München, 2000.

[RBP91] J. Rumbaugh, et al. *Object-Oriented Modeling and Design*. Prentice-Hall, 1991.

[Ree95] R. Reed. *Technologies for Service Engineering*. In A. Clarke, et al. (Hrsg.): Bringing Telecommunication Services to the People (IS&N'95), LNCS 998, S. 290-291, Springer, Berlin, 1995.

[Rei95] U. Reimers (Hrsg.). *Digitale Fernsehtechnik*. Springer Verlag, 1995.

[RJB98] J. Rumbaugh, I. Jacobson, G. Booch. *Unified Modeling Language Reference Manual*. Addison-Wesley, 1998.

[RKS01] C. Rauch, W. Kellerer, P. Sties. *Hybrid Mobile Interactive Services Combining DVB-T and GPRS*. EPMCC 2001, 4th European Personal Mobile Communication Conference, Wien, Februar 2001.

[RLS99] J. Rosenberg, J. Lennox, H. Schulzrinne. *Programming Internet Telephony Services*. IEEE Network, Vol. 13, No. 3, Mai/Juni 1999.

[RMS97] J. Rajahalme, T. Mota, F. Stegmanns, P. Hansen, F. Fonseca. *Quality of Service Negotiation in TINA*. TINA'97, Santiago, November 1997, S. 278-286.

[SEK99] P. Sties, J. Eberspächer, W. Kellerer, B. Kreutzer, H. Reichel, G. Zurek-Terhardt: *Broadband Internet Access Over Digital Video Broadcast (DVB)*. NOC'99, European Conference on Networks and Optical Communication, Delft, Juni 1999, S. 257-264.

[SFL00] L. Slutsman, I. Faynberg, H. Lu, M. Weissman. *The SPIRITS Architecture*. IETF Draft, draft-ietf-spirits-architecture-00.txt, 2000. - work in progress-

[SK01] P. Sties, W. Kellerer. *A Generic and Implementation Independent Service Description Model.* ICDCS-21W 2001, 21st IEEE International Conference on Distributed Computing Systems Workshops, Phoenix, April 2001.

[SK99] P. Sties, W. Kellerer. *Radio Broadcast Networks enable Broadband Internet Access for Mobile Users*. EUNICE'99, Open European Summer School on Access to Internet in the Next Century, Barcelona, September 1999, S. 171-176.

[SN95] R. Steinmetz, K. Nahrstedt. *Multimedia: Computing, Communications, and Applications.* Prentice-Hall, 1995.

[Som87] I. Sommerville. *Software Engineering*. Addison-Wesley, Bonn, 1987.

[SR99] H. Schulzrinne, J. Rosenberg. *The IETF Internet Telephony Architecture and Protocols.* IEEE Network, Vol. 13, No. 3, Mai/Juni 1999.

[Sti95] B. Stiller. *Quality-of-Service - Dienstgüte in Hochleistungsnetzen.* Intl. Thomson Publishing, 1995.

[Tay00] T. Taylor. *Megaco/H.248: A New Standard for Media Gateway Control*. IEEE Communications Magazine, Vol. 38, No. 10, Oktober, 2000, S. 124-132.

[Tel00] *Telelogic Tau, Version 4.1*. Telelogic AB, Schweden, 2000.

[TINA-C] *TINA - Telecommunications Information Networking Architecture*. TINA-Consortium. http://www.tinac.com./

[Tri95] S. Triglia. *Open Services Architectural Framework for Integrated Service Engineering*. RACE R2049 CASSIOPEIA Deliverable, 1995.

[TS23.107] 3GPP. *QoS Concept and Architecture (Release 4)*. 3rd Generation Partnertship Project, Technical Specification Group Services and System Aspects. 3GPP TS 23.107 v4.0.0 (2000-12), 2000.

[TWG99] TINA-IN Work Group RFP: *IN access to TINA services and Connection management (IN-TINA Adaptation Unit)*. Alcatel, Deutsche Telekom, France Telecom, Lucent Technologies, November 1999.

[V01] *Das V-Modell*. HTTP://www.v-modell.iabg.de/index.htm. 2001.

[Ver00] G. F. Vergara Ramirez. *Spezifikation und Implementierung der XML-Server basierten Zugangsfunktion für eine IuK-Dienstarchitektur.* Diplomarbeit, Lehrstuhl für Kommunikationsnetze, Technische Universität München, Dezember 2000.

[VH98] I. Venieris, H. Hussmann (Hrsg.). *Intelligent Broadband Networks*. Wiley, New York, 1998.

[VKK98] H.-J. Vögel, W. Kellerer, S. Karg, M. Kober, A. Beckert, G. Einfalt: *SDL based prototyping of ISDN-DECT-PBX switching software*. SAM'98, 1st Workshop of the SDL Forum Society on SDL and MSC, Berlin, Juni/Juli 1998, S. 191-200.

[Wei97] P. Weikert. *Entwicklung einer offenen Simulations- und Visualisierungsplattform für Signalisierungsabläufe in Kommunikationsnetzen*. Diplomarbeit, Lehrstuhl für Kommunikationsnetze, Technische Universität München, 1997.

[WHK97] M. Wahl, T. Howes, S. Kille. *Lightweight Directory Access Protocol*. IETF RFC 2251, Dezember 1997.

[Wil00] I. Willimowski (Hrsg.). *FMC - Konvergenz von Fest- und Mobilfunknetzen*. ITG-Fachgruppe 5.2.4, VDE-Verlag, Berlin, 2000.

[Wit98] E. Witte. *Regulierungspolitik*. In V. Jung und H. Warnecke (Hrsg.): Handbuch für die Telekommunikation, Springer, 1998.

[WJK00] H. Wang, A. Joseph, R. Katz. *A Signaling System Using Lightweight Call Sessions*. In Proceedings InfoCom 2000, Tel Aviv, März 2000.

[WK90] G. Willmann, P. Kühn. *Performance Modeling of Signaling System No. 7*. IEEE Communications Magazine, Vol. 28, No. 7, Juli 1990, S. 44-56.

[WRC00] H. Wang et al. *ICEBERG: An Internet Core Network Architecture for Integrated Communications*. IEEE Personal Communications, Vol. 7, No. 4, August 2000, S. 10-19.

[X.901] ITU-T. Recommendation X.901: *Information technology - Open Distributed Processing - Reference Model: Overview.* 1997. (ISO/IEC 10746-1).

[Mar00] D. Martin, et al. *Professional XML*. Wrox Press Ltd. 2000.

[Z.100] ITU-T. Recommendation Z.100: *SDL Specification and Description Language*. 1993.

[Z.120] ITU-T. Recommendation Z.120: *MSC Message Sequence Chart*. 1996.

[Zit01] M. Zitterbart, et al. *IPonAir - Nahtlose Integration Selbstkonfigurierender Drahtloser Umgebungen*. Projektantrag im BMB+F Förderschwerpunkt HyperNet.

Anhang

A Erläuterungen zu den Informationsmodellen der Sessions

Im Folgenden werden die Klassen der Informationselemente der drei Sessions und deren Attribute beschrieben.

A.1 User Session Description

(siehe Abschnitt 4.5.2)

- Registrierter User — Teilnehmer, für den in SAMSON eine sUID existiert
  - sUID — eindeutige SAMSON-Teilnehmer-Kennung
- Nicht registrierter User — Teilnehmer, für den keine sUID existiert
  - t_sUID — temporärer Bezeichner
- persönliches Profil
  - Vertragsdaten — Name, Adresse
  - Paßwort
- Netz-Profil — für jeden Netzzugang eines Teilnehmers
  - Netz — Netztyp
  - Netzadresse — Adreßtyp, Adresse, Art, Bedeutung
Adreßtyp: E.164, IP,...
Art: gültige Adresse oder aktiv genutzte Adresse
Bedeutung: Vorzugs-, Privat-, Geschäftsadresse
  - Terminalprofil — Typ, Mobilität, unterstützter Medientyp, Richtung, Port, unterstützte Codecs, Delay, Kapazitätsbeschränkung, unterstützte Dienste, unterstützte Applikationsprogramme
- Dienst-Profil — für jeden subskribierten Dienst
  - Dienstname — eindeutige Kennung des Dienstes
  - Dienstparameter — Liste mit Parametern, z.B. Dauer
  - Zusatzdienste — Liste mit möglichen Zusatzdiensten
- Abrechnungsprofil
  - Tarifmodell — z.B. Business, Prepaid
  - Aktuelle Rechnung — Summe der aufgelaufenen Beträger oder Stand des Prepaid-Kontos

A.2 Service Session Description

(siehe Abschnitt 4.5.3)

- Teilnehmer — Allgemeiner Kommunikationsendpunkt in einem Dienst
- Teilnehmer: User
  - sUID — eindeutige SAMSON-Teilnehmer-Kennung
  - Rollen — z.B. Initiator, eingeladener Teilnehmer, Konferenzleiter
  - Dienst-Profil — (siehe A.1)
- Teilnehmer: Service Support Component
  - sRID — eindeutige Kennung für *Service Support Server*
- Kommunikationsbeziehung — enthält Informationsflüsse
  - Session-Bezeichner — eindeutige Kennung

  - Beginn — Zeitpunkt
  - Ende — Zeitpunkt
  - Dauer — maximale Dauer
  - Mediengruppen — Medien-Bezeichner von synchronisierten Medien

- Informationsfluß
  - Medien-Bezeichner — eindeutige Kennung
  - Medientyp — Voice, Video, Text, Image
  - Existenz — optional, obligat
  - OvQ — Teilnehmer-Qualitäts-Klasse (siehe Tabelle 4.2)
  - SQ — Dienst-Qualitäts-Klasse (siehe Tabelle 4.3)

A.3 Communication Session Description

(siehe Abschnitt 4.5.4)

- Endpoint — terminiert einen Informationspfad; kann entweder eine Informationsquelle oder -Senke sein, z.B. ein Endgerät, ein Server (*User Endpoint*) oder eine Schnittstelle zu einer Spezialressource (*Special Resource Endpoint*)
- User Endpoint — ist gekennzeichnet durch eine aktive Adresse aus dem Teilnehmerprofil
  - sUID — eindeutige Kennung
  - Netz — Netztyp
  - Netzadresse — Adreßtyp, Adresse
  - Anwendung — Applikationsprogramm, Port
  - QoS-Parameter — Angabe konkreter QoS-Parameter des Endgerätes (z.B. Delay)
  - Kodierformat — Angabe des verwendeten Kodierformates mit den entsprechenden Parametern
- Special Resource Endpoint — ist gekennzeichnet durch sein Ressoucenprofil
  - Bezeichner — eindeutige Kennung
  - Netz — Netztyp
  - Netzadresse — Adreßtyp, Adresse
  - Anwendung — Applikationsprogramm, Port
  - QoS-Parameter — Angabe konkreter QoS-Parameter der Spezialressource
  - Kodierformat — Angabe des verwendeten Kodierformates mit den entsprechenden Parametern
- Informationsfluß — Kommunikationsbeziehung eines Informationstyps zwischen beliebig vielen User Endpoints, z.B. Konferenz, Abruf
  - Medien-Bezeichner — eindeutige Kennung
  - Medien-Typ — Voice, Video, Text, Image
  - OvQ — Teilnehmer-Qualitäts-Klasse (siehe Tabelle 4.2)
  - SQ — Dienst-Qualitäts-Klasse (siehe Tabelle 4.3)
  - Kommunikationstyp — Konferenz, Dialog, Multicast, Asymmetrisch, Unicast
  - QoS-Parameter — Angabe konkreter QoS-Parameter (z.B. Spitzenbitrate, Minimal akzeptierbare Bitrate, Bitfehlerrate, Verlustrate, Delay, Jitter)
- Informationspfad — Kommunikationsverbindung zwischen zwei Endpunkten; Teil eines Informationsflusses
  - Medien-Bezeichner — eindeutige Kennung
  - Kommunikationstyp — Dialog, Unicast
  - QoS-Parameter — siehe oben

B Details zur Dienstbeschreibung mit SDP+

Nachfolgend werden die Bezeichner und die zugehörigen Parameter der SesCP-Dienstbeschreibung im Format SDP+ erläutert. Die Reihenfolge der Bezeichner (z.B. „ver=“) und die Reihenfolge der Parameter („<para>“) ist einzuhalten. Nicht-belegte Parameter werden mit „-“ gekennzeichnet. Parameter, deren Belegung von der Gegenseite erfragt wird, sind mit „?“ markiert. Ein „*“ kennzeichnet beliebige Wiederholungen des nachfolgenden Bezeichners, Parameters oder einer Gruppierung (runde Klammern). Eckige Klammern („[]“) kennzeichnen optionale Bezeichner. Kommentare sind in der Schriftart *TimesKursiv* angefügt.

B.1 SDP+-Teilnehmerprofil

Die SDP+-Elemente des Teilnehmerprofils (*User Profile*, UPR) werden zwischen den Komponenten der *Registration Session* ausgetauscht (siehe 5.1.4.1).

```
ver=<Protokollversion von SDP+>
ori=<Name> <Session ID> <Version>
ses=upr
uid=<Teilnehmerkennung/sUID>
[pwd=<Paßwort>]
*(
adr=<Netztyp> <Adreßtyp> <Adresse> <Aktiv> <Bedeutung>
          Aktiv = 0/generell gültige Adresse; Aktiv = 1/momentan genutzte Adresse;
          Bedeutung = Vorzugs-Adr. | Privatadr. | Geschäftsadr)
ter=<Terminal-Typ> <Mobilität>  Telefon | PC | PDA | ...;  mobile | stationary
   *(
tmt=<unterstützter Medientyp> <Richtung> [<Port>]
[*tmm=<unterstützter Codec/Protocol> *<Codec Parameter>]
[tmb=<Kapazität>]
    )
*tsr=<unterstützter Dienst>
*tap=<unterstütztes Applikationsprogramm> <Protokollstack>
 )
*(
srv=<Dienstname> <Aktiv>  =0: erlaubter  Dienst/ =1: aktiver Dienst
*srp=<Dienstparameter>:<Wert>  statischer Dienstparameter
ssr=<Zusatzdienstname> <Ausprägung>  Zusatzdienst, z.B. Call Forwarding erlaubt
*ssp=<Dienstparameter Zusatzdienst>:<Wert>
 )
```

B.2 User Service Graph

Die SDP+-Elemente des *User Service Graph* (USG) werden im Rahmen der *Access Session* zum Start eines Dienstes transportiert (siehe 5.1.4.2) .

```
ver=<Protokollversion von SDP+>
ori=<Name> <Session ID> <Version>
ses=usg
srv=<Dienstname> - <OvQ-Wert>  angeforderter Dienst
*srp=<Dienstparameter>:<Wert>  dynamische Dienstparameter, z.B. eingeladener
Teilnehmer; statischer Dienstparameter aus dem Teilnehmerprofil
[sti=<Startzeit> <Stopzeit>]
[stz=<Zeitzone>]
[str=<Dauer> <Wiederholintervall> <Liste der Abstände von der
Startzeit>]
*(
ssr=<Zusatzdienstname> <Ausprägung>
*ssp=<Dienstparameter Zusatzdienst>:<Wert>
)
[*spt=<Teilnehmer>]
[sct=<Content-Ressourcen-Name>]
[sad=<Netztyp> <Adresstyp> <Adresse>]
[scc=<Abzuspielender Inhalt>]
```

B.3 Media Connection Graph

Die SDP+-Elemente des *Media Connection Graph* (MCG) werden vom SSM zum CSM signalisiert. Sie beschreiben eine einzurichtende Kommunikationsbeziehung (siehe 5.1.4.3).

```
ver=<Protokollversion von SDP+>
ori=<Name> <Session ID> <Version>
ses=mcg
srv=<Dienstname> - <OvQ-Wert>
smg=<Synchronisations-Typ> *<Medium-ID>
                Mediengruppe, z.B. für Synchronisation
*(
mid=<Medium-ID> <Medientyp>
mex=<Existenz>          optional oder obligat
sqc=<Dienst-Qualitäts-Klasse>
mcc=<Kodekonvertierung erlaubt?>
mct=<Kommunikationstyp>  Konferenz | Dialog | Multicast | Asymmetrisch | Unicast
mse=<sUID/sRID> <sUID/sRID> ...  Sender
mre=<sUID/sRID> <sUID/sRID> ...  Empfänger
 )
```

B.4 Connectivity Connection Graph

Mit den SDP+-Elementen des *Connectivity Connection Graph* (CCG) wird die Einrichtung eines Informationsflusses oder eines Informationspfades von einem Netzadaptor angefordert (siehe 5.1.4.4).

```
ver=<Protokollversion von SDP+>
ori=<Name> <Session ID> <Version>
ses=ccg
srv=<Dienstname> <OvQ-Wert>
*(
pid=<Endpunktbezeichner>      sUID | sRID | Nummer
pty=<Teilnehmer-Typ>          Mensch | Ressource | Spezialressource | Gateway
adr=<Netztyp> <Adresstyp> <Adresse>          Terminal-Adresse
tap=<Applikationsprogramm> <Protokollstack>  Term.-Interface: Protokolle, etc.
 )
*(
mid=<Medium-ID> <Medientyp>
mct=<Kommunikationstyp>  Konferenz | Dialog | Multicast | Asymmetrisch | Unicast
mbw=<Spitzenbitrate> <minimal akzeptierbare Bitrate>
     <max. Burstgröße>
mre=<Bitfehlerrate> <Verlustrate>
mde=<max. Delay> <max. Jitter>
mco=<Kodierformat> <Protokollstack>
mse=<sUID/sRID> <sUID/sRID> ...  Sender
mre=<sUID/sRID> <sUID/sRID> ...  Empfänger
 )
```

B.5 Verwendung von SDP+ bei der INFO-Methode

Inhalt des Message-Bodies der SesCP-INFO-Methode können beliebige Bezeichner aus allen obigen Beschreibungen sein (siehe 5.1.4.5). Durch ein „?“ als Parameterwert wird angegeben, daß dieser Wert in der Antwort auf die INFO-Anfrage erwartet wird. Ein einziges „?“ bei einem Bezeichner fordert alle Parameterwerte zu einem Bezeichner an. Um die Eindeutigkeit zu gewährleisten ist der Session-Typ (upr, usg, mcg, ccg) anzugeben, aus dessen Bereich der Beizeichner kommt und zusätzlich weitere Bezeichner und Parameterwerte. Im folgenden sind Beispiele dargestellt.

Mögliche Rückfrage des SSM an den UP zu einem gerade gestarteten Dienst:

```
ver=<Protokollversion von SDP+>
ori=<Name> <Session ID> <Version>
ses=info
rss=usg (Session Bezeichner der abgefragten Session-Beschreibung)
srv=MyBusinessCall (Dienstname,
                    damit die folgende Anfrage eindeutig zugeordnet werden kann)
sep=invited-party:? (der SSM fordert die sUID des eingeladenen Teilnehmers an)
sct=videoserver-alpha net ip alpha@3.5.6.7.net
   (Name und Adresse der Content-Ressource)
scc=? (der SSM fordert die genaue Bezeichnung des abzuspielenden Inhalts an)
```

Teilnehmername und die sUID zur Kennzeichnung des *User Proxies* sind im SesCP-Teil der Nachricht enthalten.

C Document Type Definition der Session Programming Language (SPL)

Eine *Document Type Definition* (DTD) definiert die Struktur eines XML-Dokumenttyps und die verwendbaren XML-Konstrukte. Im folgenden wird die Struktur der *Session Programming Language* SPL, die in Abschnitt 5.3.2 vorgestellt wurde, mit der zugehörigen DTD beschrieben. Die DTD ist nicht vollständig, da die SPL im Rahmen dieser Arbeit als Vorschlag für eine XML-basierte Dienstlogikbeschreibung verstanden wird, der die Möglichkeiten des Einsatzes von XML aufzeigt. Die nachstehend beschriebene DTD definiert daher nur die Hauptstruktur und die wichtigsten Elemente.

```
<!-- Session Programming Language DTD Version 1.0 -->
<!ELEMENT spl (subaction* | session)>

  <!ELEMENT subaction (...)>
    <ATTLIST subaction id CDATA #REQUIRED>

  <!ELEMENT session (initiation | active | release)>
    <!ELEMENT initiation (party , create, communication , create)>
      <!ELEMENT party (user+ | server*)>
        <!ELEMENT user (id | role)>
          <!ELEMENT id (#PCDATA)>
          <!ELEMENT role (#PCDATA)>
        <!ELEMENT server (id)>
      <!ELEMENT communication (mandatory+ , optional?)>
        <!ELEMENT mandatory (infoflow+)>
        <!ELEMENT optional (infoflow+)>
          <!ELEMENT infoflow >
            <ATTLIST infoflow
                     media (voice | video | text | image) #REQUIRED
                     ...>
      <!ELEMENT create (success , failure)>
        <!ELEMENT success (subaction*)>
        <!ELEMENT failure (subaction*)>
    <!ELEMENT active (action)*>
      <!ELEMENT> action ((user | server | infoflow ) , (create | delete |
modify | retrieve | suspend | resume))>
        <ATTLIST action
          event (timer | invite | ack | info | end | bye | response)>
        <!ELEMENT delete (success , failure)>
        <!ELEMENT modify (success , failure)>
        <!ELEMENT retrieve (success , failure)>
        <!ELEMENT suspend (success , failure)>
        <!ELEMENT resume (success , failure)>
    <!ELEMENT release (party , communication , delete)>
```

Erläuterungen:

Es werden erst die Objekte: `party` (Teilnehmer), `communication` (Kommunikationsbeziehung) beschrieben. Anschließend wird die Aktion aufgerufen, die mit diesen ausgeführt wird. Dadurch können gebündelte Aktionen in einer Anforderung aufgerufen werden, z.B. durch ein SesCP-SETUP.

Mit der DTD wird nur das Format der XML-Tags und die Struktur eines XML-Dokumentes beschrieben. Eine Ersetzung des Makros <subaction> erfolgt durch den Parser.

D SesCP-Signalisierungsbeispiel

Beschreibung ausgewählter Meldungen des Signalablaufdiagrammes aus Kapitel 5.

(1) INVITE PAC -> UP_A

```
INVITE sescp:MyBusiCall@ism.sc.samson SESCP/1.0
Via: SESCP/1.0/UDP NA4711.samson:5050
From: sescp: A@NA4711.samson
To: sescp:MyBusiCall@sc.samson
Route: UP_A@uc.samson
Call-ID: 888
CSeq: 1 INVITE
Content-Type: application/sdp+
Content-Length: ...

ver=0
ori=A 888 0
ses=usg
srv=MyBusiCall - 2
spt=B
```

(2) INVITE UP_A -> ISM

```
INVITE sescp:MyBusiCall@ism.sc.samson SESCP/1.0
Via: SESCP/1.0/UDP UP_A.uc.samson:6060
Via: SESCP/1.0/UDP NA4711.samson:5050
From: sescp: A@NA4711.samson
To: sescp:MyBusiCall@ism.sc.samson
Record-Route: UP_A@uc.samson
Call-ID: 888
CSeq: 1 INVITE
Content-Type: application/sdp+
Content-Length: ...

ver=0
ori=A 888 1
ses=usg
srv=MyBusiCall - 2
srp=party:B
srp=type:voiceonly
```

(3) 305 Redirect ISM -> UP_A

```
SESCP 305 Redirect
Via: SESCP/1.0/UDP UP_A.uc.samson:6060
Via: SESCP/1.0/UDP NA4711.samson:5050
From: sescp: A@NA4711.samson
To: sescp:MyBusiCall@ism.sc.samson
Call-ID: 888
CSeq: 1 INVITE
Contact: sescp:SSM_mbc5@sc.samson; action=redirect
Content-Length: 0
```

(4) INVITE UP_A -> SSM_mbc5

```
INVITE sescp:MyBusiCall@ism.sc.samson SESCP/1.0
Via: SESCP/1.0/UDP UP_A.uc.samson:6060
Via: SESCP/1.0/UDP NA4711.samson:5050
From: sescp: A@NA4711.samson
To: sescp:SSM_mbc5@sc.samson
Record-Route: UP_A@uc.samson
Call-ID: 888
CSeq: 1 INVITE
Content-Type: application/sdp+
Content-Length: ...
```

(5) 200 OK SSM_mbc5 -> UP_A

```
SESCP 200 OK
Via: SESCP/1.0/UDP UP_A.uc.samson:6060
Via: SESCP/1.0/UDP NA4711.samson:5050
From: sescp: A@NA4711.samson
To: sescp:SSM_mbc5@sc.samson
Call-ID: 888
CSeq: 1 INVITE
Content-Length: 0
```

(6) 200 OK UP_A -> PAC

```
SESCP 200 OK
Via: SESCP/1.0/UDP NA4711.samson:5050
From: sescp: A@NA4711.samson
To: sescp:SSM_mbc5@sc.samson
Call-ID: 888
CSeq: 1 INVITE
Content-Length: 0
```

(7) ACK PAC -> UP_A

```
INVITE sescp:SSM_mbc5@sc.samson SESCP/1.0
Via: SESCP/1.0/UDP NA4711.samson:5050
From: sescp: A@NA4711.samson
To: sescp:SSM_mbc5@sc.samson
Route: UP_A@uc.samson
Call-ID: 888
CSeq: 1 INVITE
Content-Length: 0
```

(8) ACK UP_A -> SSM_mbc5

```
INVITE sescp:SSM_mbc5@sc.samson SESCP/1.0
Via: SESCP/1.0/UDP UP_A.uc.samson:6060
Via: SESCP/1.0/UDP NA4711.samson:5050
From: sescp: A@NA4711.samson
To: sescp:SSM_mbc5@sc.samson
Record_Route: UP_A@uc.samson
Call-ID: 888
CSeq: 1 INVITE
Content-Length: 0
```

(13) ACCESS SSM_mbc5 -> UP_B

```
ACCESS sescp:UP_B@uc.samson SESCP/1.0
Via: SESCP/1.0/UDP sc.samson:9090
From: sescp: SSM_mbc5@sc.samson
To: sescp:UP_B@uc.samson
Call-ID: 2222
CSeq: 1 ACCESS
Content-Type: application/sdp+
Content-Length: ...

ver=0
ori=SSM_mbc5 2222 0
ses=upr
uid=B
srv=MyBusiCall 1
srp=party:A
```

(17) SETUP SSM_mbc5 -> CSM_mbc5

```
SETUP sescp:adaptor@cc.samson SESCP/1.0
Via: SESCP/1.0/UDP sc.samson:9090
From: sescp: SSM_mbc5@sc.samson
To: sescp:adaptor@cc.samson
Route: CSM_mbc5@uc.samson
Call-ID: 888
CSeq: 1 SETUP
Content-Type: application/sdp+
Content-Length: ...

ver=0
ori=SSM_mbc5 888 0
ses=mcg
srv=MyBusinessCall - 2
smg=0
mid=if1 voice
sqc=1
mex=mandatory
mcc=1
mct=dialog
mse=A B
mre=A B
```

(18) 100 Trying CSM_mbc5 -> SSM_mbc5

```
SESCP 100 Trying
Via: SESCP/1.0/UDP sc.samson:9090
From: sescp: SSM_mbc5@sc.samson
To: sescp:adaptor@cc.samson
Call-ID: 888
CSeq: 1 SETUP
Content-Length: 0
```

(19) INFO CSM_mbc5 -> UP_A

```
INFO sescp:UP_A@uc.samson SESCP/1.0
Via: SESCP/1.0/UDP cc.samson:9999
From: sescp: CSM_mbc5@cc.samson
To: sescp:UP_A@uc.samson
Call-ID: 444
CSeq: 1 INFO
Content-Type: application/sdp+
Content-Length: ...

ver=0
ori=CSM_mbc5 444 0
ses=info
rss=upr
adr=?
ter=?
srv=MyBusinessCall 1
srp=type:voiceonly
```

(25) SETUP CSM_mbc5 -> PAS(NA_GSM)

```
SETUP sescp:adaptor@cc.samson SESCP/1.0
Via: SESCP/1.0/UDP cc.samson:9999
Via: SESCP/1.0/UDP sc.samson:9090
From: sescp: SSM_mbc5@sc.samson
To: sescp:adaptor@cc.samson; tag=876
Request-URI: NA_GSM@na_gsm.samson
Call-ID: 888
CSeq: 1 SETUP
Content-Type: application/sdp+
Content-Length: ...

ver=0
ori=CSM_mbc5 888 0
ses=ccg
srv=MyBusinessCall 1
pid=A
pty=user
adr=gsm E164 01721234@d2.gsm
tap=GSM GSM
pid=gw1
pty=gateway
adr=gsm/in E164 01729876@d2.gsm
tap=GSM GSM
mid=if1 voice
sqc=1
mct=dialog
mbw=13kbps 13kbps -
mre= - 0.5%
mde=100ms 10ms
mco=GSM-FR -
mse=A gw1
mre=A gw1
```

(27) 200 OK PAS(NA_GSM) -> CSM_mbc

```
SESCP/1.0 200 OK
Via: SESCP/1.0/UDP sc.samson:9090
From: sescp: SSM_mbc5@sc.samson
To: sescp:adaptor@cc.samson; tag=876
Call-ID: 888
CSeq: 1 SETUP
Content-Length: 0
```

(29) 200 OK CSM_mbc5 -> SSM_mbc5

```
SESCP/1.0 200 OK
From: sescp: SSM_mbc5@sc.samson
To: sescp:adaptor@cc.samson
Call-ID: 888
CSeq: 1 SETUP
Content-Length: 0
```

(30) ACK SSM_mbc5 -> CSM_mbc5

```
ACK sescp:adaptor@cc.samson SESCP/1.0
Via: SESCP/1.0/UDP sc.samson:9090
From: sescp: SSM_mbc5@sc.samson
To: sescp:adaptor@cc.samson
Route: CSM_mbc5@uc.samson
Call-ID: 888
CSeq: 1 SETUP
Content-Length: 0
```

(31) ACK CSM_mbc5 -> PAS(NA_GSM)

```
ACK sescp:adaptor@cc.samson SESCP/1.0
Via: SESCP/1.0/UDP cc.samson:9999
Via: SESCP/1.0/UDP sc.samson:9090
From: sescp: SSM_mbc5@sc.samson
To: sescp:adaptor@cc.samson; tag=876
Request-URI: NA_GSM@na_gsm.samson
Call-ID: 888
CSeq: 1 SETUP
Content-Length: 0
```

E Performance-Analyse des Signalisierungsprotokolls SesCP

Wenngleich der Signalisierungsverkehr im Vergleich zu den Nutzdaten, die in einem Kommunikationsnetz transportiert werden, recht gering ausfällt, so beeinflussen Verzögerungen im Signalisierungsablauf doch direkt die Antwortzeiten und damit die QoS eines Systems. Insbesondere für komplexe Dienstarchitekturen, bei denen die Signalisierung über separate Signalisierungsnetze (z.B. SS#7) geführt wird, kann der Aufbau und die Dimensionierung des Signalisierungsnetzes entscheidend für die Systemperformance sein. Daher ist es sinnvoll schon aus der Protokollspezifikation analytisch mögliche Rückschlüsse auf die Performance ziehen zu können und nicht erst durch Simulationen oder gar Messungen an der fertigen Implementierung.

In der Literatur existieren nur wenige Ansätze, die eine Performance-Analyse von (Signalisierungs-)Protokollen auf Basis deren Spezifikation beschreiben. In [HHL01] wird ein Verfahren beschrieben, wie mit Hilfe einer Erweiterung der Spezifikationssprache SDL um zusätzliche Konstrukte (*Queueing SDL*) nicht-funktionale Bedingungen, wie z.B. Prozessorleistung, in einem SDL-System beschrieben und werkzeuggestützt analysiert werden können. Als Beispiel wird ein Performance-Modell des TCP/IP-Protokoll-Stacks analysiert. [WK90] beschreibt ein Verfahren für die Performance-Analyse des Signalisierungssystems No. 7 auf Basis dessen funktionaler Spezifikation. Ausgangspunkt ist das Signalablaufdiagramm eines zu analysierenden Signalisierungsszenarios, z.B. *Service Session Setup*. Das Performance-Modell reflektiert insbesondere den Schichten-Aufbau des SS#7-Protokoll-Stacks, um ein möglichst generisches Modell für verschiedene Implementierungsvarianten zu erhalten. Ein ähnlicher, einfacherer Ansatz, der nur eine Protokollschicht betrachtet, wird in [Kni00] für B-ISDN beschrieben. In [BS96] wird aufbauend auf dem Verfahren von [WK90] ein Werkzeugkonzept beschrieben, mit dem SS#7-Systeme in Abhängigkeit von verschiedenen IN-Diensten dimensioniert werden können. Der dazu benötigte Signalisierungsablauf, der die Basis für die Performance-Analyse bildet, wird aus der IN-Dienstlogik (SIB-Graph) gewonnen.

Im folgenden wird das Signalisierungsprotokoll SesCP hinsichtlich seiner Performance in Anlehnung an die Grundprinzipien aus [Kni00] und [WK90] ansatzweise analysiert. Ziel ist es dabei, eine allgemeine Vorgehensweise aufzuzeigen. Als typische Szenarien werden die Teilnehmer-Registrierung, der Dienstaufruf, die Dienstmodifikation (z.B. Hinzunahme eines Mediums) und die Dienstbeendigung betrachtet. Grundlage sind die Signalablaufdiagramme in Kapitel 5 (Registrierung: Abbildungen 5.5 und Dienstaufruf/Modifikation: Abbildung 5.19 Meldungen 1-32 bzw. 33-52).

Belastung der Signalisierungsknoten

Eine erste Analyse kann bereits aus der Anzahl an *Visits* pro Komponente gewonnen werden. Es werden die am meisten belasteten Komponenten identifiziert. Jede Signalankunft und jede Aussendung wird dabei als ein *Visit* gezählt [Kni00]. Es ergibt sich folgende Tabelle A.1.

Kompo-nente	NA PAC	IAM	UP_ A	UP_ B	UDB	ISM	SSM	ICM	CSM	NA_ GSM	NA_ H.323	RR
Registrierung	9	8	8	-	6	-	-	-	-	-	-	-
Aufruf	3	3	12	4	2	3	13	3	16	3	3	2
Modifikation	3	-	6	-	-	-	6	-	9	3	3	2
Beendigung	2	-	4	2	4	-	4	-	6	2	2	-
Summe	17	11	30	6	12	3	23	3	31	8	8	4

Tabelle A.1: Visits pro SesCP-Komponente

Der *User Proxy* und die *Session Manager* SSM und CSM sind mit Abstand am meisten belastet. Dies wurde bereits in der Architektur berücksichtigt, indem diese Komponenten für jeden Teilnehmer bzw. für jeden Dienst instanziert werden.

Für die weitere Analyse wird angenommen, daß das Protokoll auf einem Internet-Protokoll-Stack läuft. Die Komponenten NA, IAM/UDB, ISM und ICM sind in separaten Servern untergebracht zwischen denen die Protokollnachrichten über Socket-Verbindungen ausgetauscht werden. UP, SSM und CSM, die zur Laufzeit erzeugt werden, laufen auf den Servern ihrer Initial-Komponenten (IAM, ISM, ICM). Erneut werden die *Visits* gezählt. Ohne Einbeziehung der Registrierung, die unabhängig von einer Dienstnutzung erfolgt, ergibt sich eine relativ gleichmäßige Auslastung der Server UC, SC und CC (siehe Tabelle A.2).

Server	**NA** PAC	**UC** (IAM, UP, UDB)	**SC** (ISM, SSM)	**CC** (ICM, CSM)	**NA_** GSM	**NA_** H.323	**RR**
Registrierung	9	9	-	-	-	-	-
Dienstaufruf	3	16	15	18	3	3	2
Modifikation	3	6	6	9	3	3	2
Beendigung	2	6	4	6	2	2	-
Summe	7+9	27+9	25	33	8	8	4

Tabelle A.2: Visits für SesCP pro Server

Ermittlung der Netzbelastung

Obige Betrachtungen beziehen sich auf die Belastung der Rechnerplattformen. Die Performance-Analyse des Signalisierungsprotokolls umfaßt auch eine Analyse der Netzbelastung, d.h. des Meldungstransports im Signalisierungsnetz. Dafür wird nun genauer auf die Signalisierungsmeldungen selbst eingegangen, um die transportierten Datenmengen zu ermitteln. Anhand der ausgetauschten Meldungen (siehe Anhand D) ergeben sich die folgenden, aus Tabelle A.3 ersichtlichen Durchschnittswerte für die transportierten Datenmengen.

Meldung	**typische Länge in Bytes**[a]	**UDP-Paket in Bytes**	**TCP-Paket in Bytes**	**IP-Pakete (bei UDP)** (ohne Fragmentierung)
INVITE mit SDP+ (USG)	360	368	396	444
SETUP mit SDP+ (CCG)	570	578	606	654
INFO, ACCESS, Request ohne Message Body (ACK),	200 - 250	258	286	334
Response ohne Message Body	200 - 250	258	286	334

a. Ohne Kompression

Tabelle A.3: Datenmengen pro SesCP-Meldung

Mit diesen Daten und den Signalablaufdiagrammen kann für jedes der eingangs betrachteten Szenarien die transportierte Datenmenge ermittelt werden (Tabelle A.4). Wir betrachten auch hier nur die Schnittstellen zwischen den drei Servern UC, SC und CC sowie der Anpassungsschicht (als ein Bereich).

Szenario	Daten in Byte
Registrierung	3 006
Dienstaufruf	10 962
Modifikation	7 846
Beendigung	4 008
Summe	25 822

Tabelle A.4: Transportierte Signalisierungsdaten je Szenario

Nimmt man an, daß pro Dienst durchschnittlich mindestens ein Megabyte an Nutzdaten transportiert wird, so bestätigt sich die Aussage, daß der Signalisierungsverkehr einen recht geringen Anteil am Datenverkehr hat.

Interessant sind die obigen Daten für den Vergleich mit anderen Signalisierungsprotokollen oder, um ein Signalisierungsnetz zu dimensionieren. Einen Richtwert für die Dimensionierung erhält man aus dem Nutzerverhalten.

Typisches Nutzerverhalten

Auf Basis der SAMSON-Dienstbeispiele wird angenommen, daß ein typischer Teilnehmer tagsüber alle zwei Stunden einen Dienstaufruf initiiert und bei jedem Dienst zwei Zusatzmerkmale aufruft (Modifikation). Im Durchschnitt ist ein Dienst 60 Minuten aktiv. Ein Wechsel des Netzzuganges (Registrierung) kommt nur alle drei Stunden vor. Damit ergibt sich eine insgesamte Signalisierungsbelastung von 16 kByte pro Teilnehmer in einer Stunde. Viel wichtiger als ein Gesamtwert ist für eine detaillierte Dimensionierung die Verteilung der Daten (siehe Abbildung A.1).

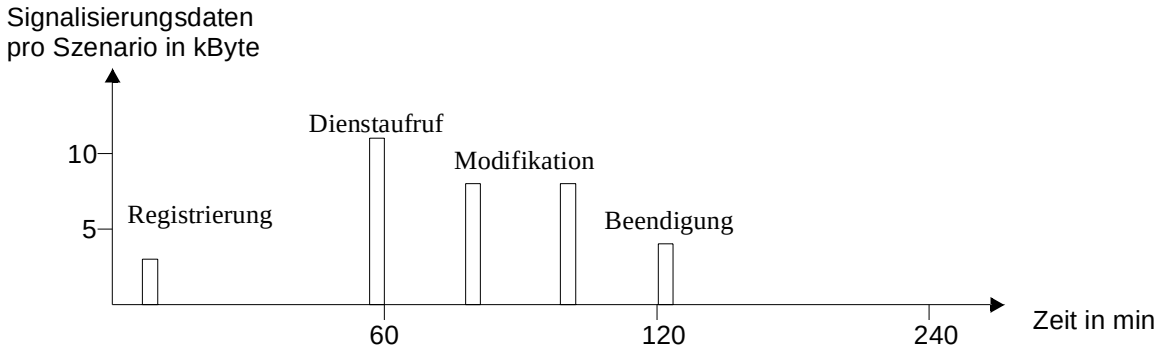


Abbildung A.1: Typische Verteilung des Signalisierungsverkehrs eines Teilnehmers

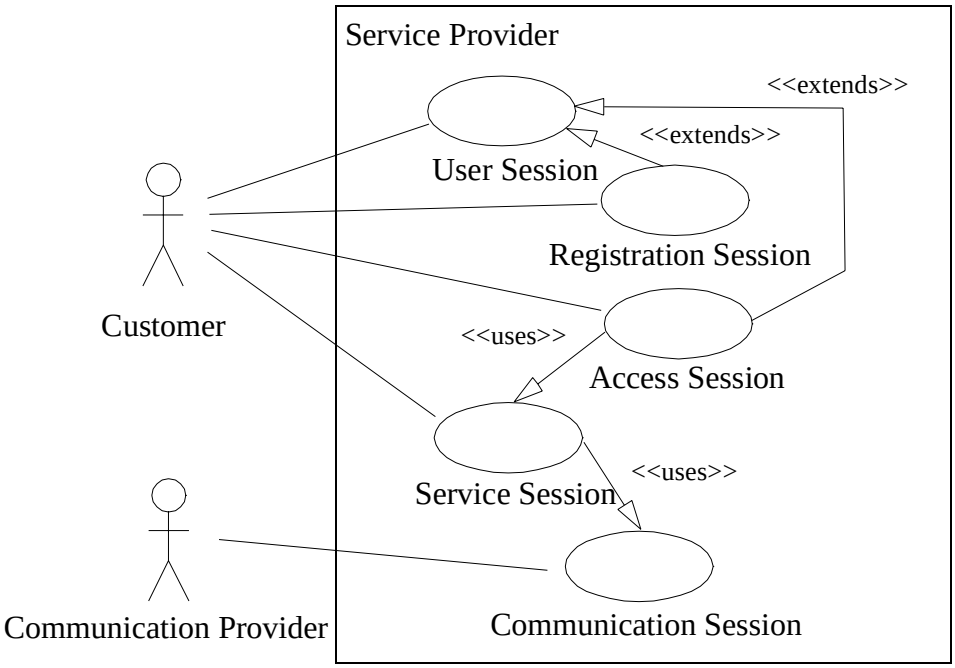
Dimensionierung

Da die Datenraten linear mit der Teilnehmeranzahl skalieren, kann ein UDP/IP-basiertes Signalisierungsnetz auf diese Weise anhand der Funktionsbeschreibung des Protokolls dimensioniert werden. Für 1000 Teilnehmer ist z.B. eine mittlere Datenrate von 36 kBit/s erforderlich.

Weitergehende Fragestellungen der Performance-Analyse von Signalisierungssystemen beziehen sich auf die Einhaltung konkreter Teilnehmeranforderungen. In Anlehnung an die Anforderungen aus der Telefonie [ETS01] ist auch bei der vorliegenden Dienststeuerung beispielsweise die Zeit zwischen Dienstaufruf und Ausführung (Verbindungsaufbauzeit) kleiner als fünf Sekunden zu halten. Für eine diesbezügliche Analyse ist die Rechenzeit der Server für jede Verarbeitungsinstanz des Protokollstacks (SesCP, UDP, IP) zu berücksichtigen. Für eine detaillierte Analyse bedient man sich hier der Warteschlangenmodelle, z.B. [WK90].

F Beschreibung des Sessionmodells als Use-Case

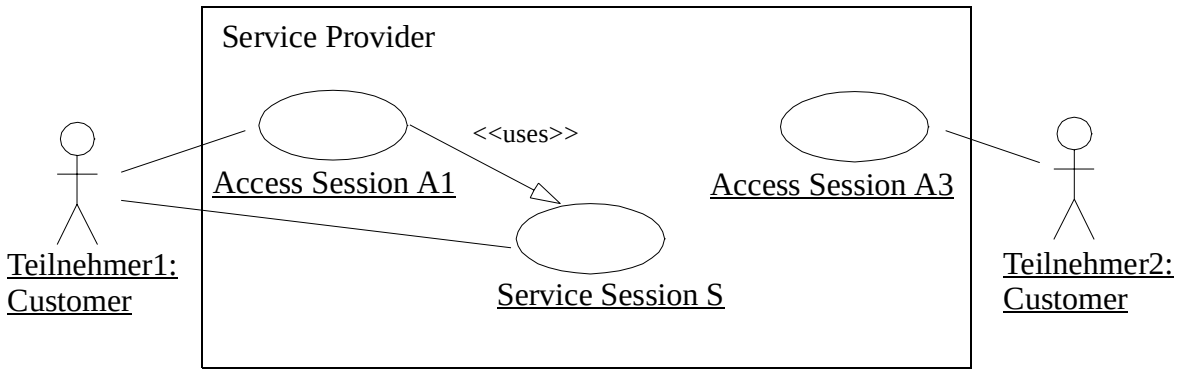
Das Sessionmodell für die SAMSON-Dienstarchitektur (Abbildung 4.5) läßt sich folgendermaßen in einem Use-Case-Diagramm beschreiben. Die verwendete Syntax ist in Anhang G erläutert.



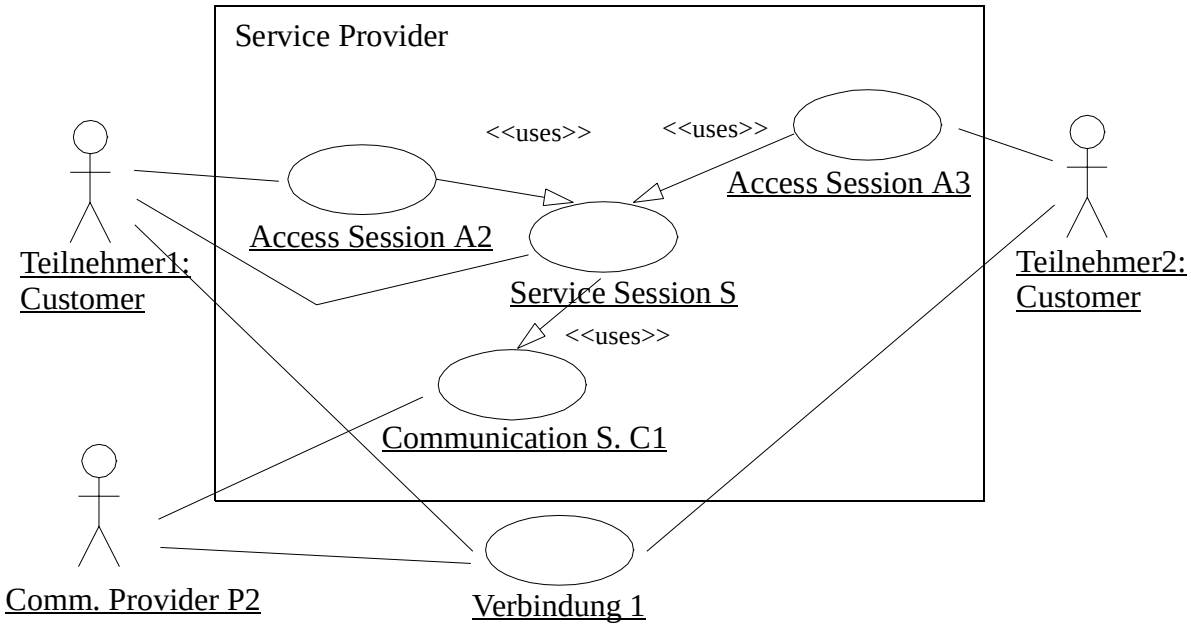
Dienste-Mobilität

Damit ergibt sich folgende Beschreibung der Dienste-Mobilität aus Abbildung 4.6 ausgedrückt durch die Interaktion mit verschiedenen Instanzen der *Use Cases*. Für jeden Zeitpunkt wird ein eigenes Use-Case-Diagramm verwendet.

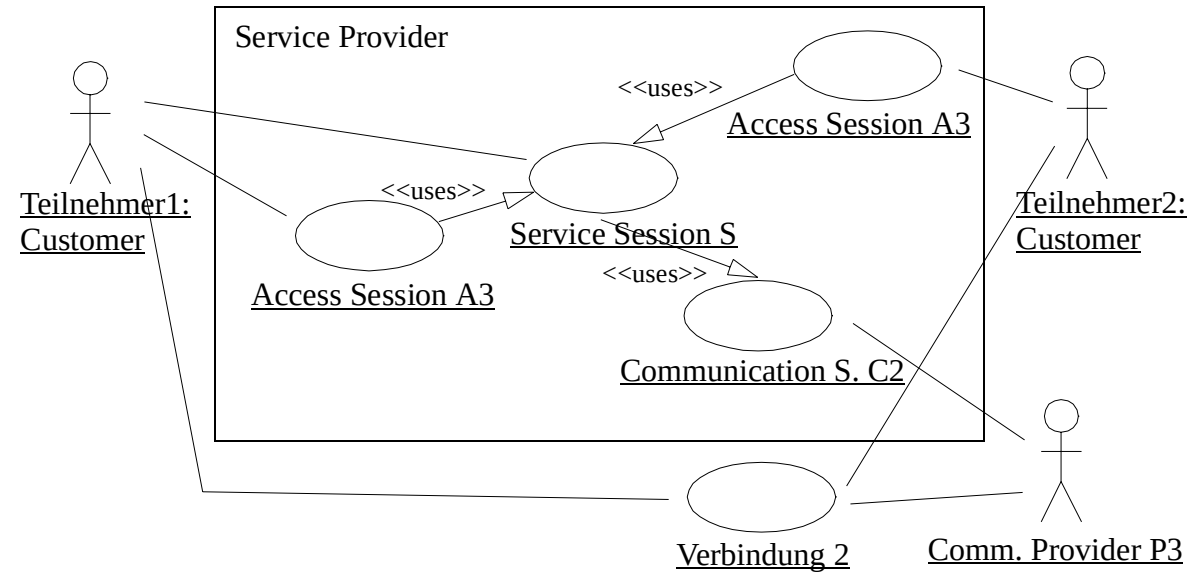
Zeitpunkt T1: Teilnehmer 1 und Teilnehmer 2 haben je eine aktive *Access Session*. Teilnehmer 1 hat einen Dienst gestartet.



Zeitpunkt T2: Teilnehmer 1 wechselt seinen Dienstzugang (z.B. sein Endgerät) noch bevor eine Kommunikationsbeziehung durch den Dienst aufgebaut wird. Die Service Session startet eine *Communication Session* und die Teilnehmer werden verbunden.



Zeitpunkt T3: Teilnehmer 1 wechselt erneut seinen Netzanschluß. Er teilt dies der *Service Session* über eine neue *Access Session* mit. Die ursprüngliche *Communication Session* wird abgebaut, da sich Teilnehmer 1 jetzt im Bereich des Kommunikationsanbieters P3 befindet. Über eine neue *Communication Session* (C2) wird dann die Verbindung zu Teilnehmer 2 wieder hergestellt. Die Informationsflüsse befinden sich nun im Bereich des Kommunikationsanbieters P3.



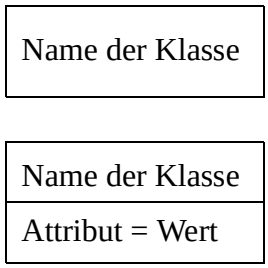
G Erläuterungen zur Unified Modeling Language

Die *Unified Modeling Language* (UML) ist eine graphische Sprache zur Visualisierung, Spezifikation und Dokumentation von komplexen Softwaresystemen [RJB98]. Sie besitzt eine wohldefinierte Syntax und Semantik. Die Sprache UML besteht aus verschiedenen Notationen wie dem Use-Case-Diagramm, dem Sequenzdiagramm oder dem Klassendiagramm. Mit jeder Notation läßt sich eine andere Sichtweise auf das System zu beschreiben.

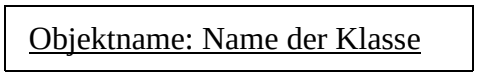
Kurzerläuterung der Notation des UML-Klassendiagramms

Das Klassendiagramm dient dazu logische Zusammenhänge zwischen Dateninhalten und zwischen Systemkomponenten darzustellen.

Klassen

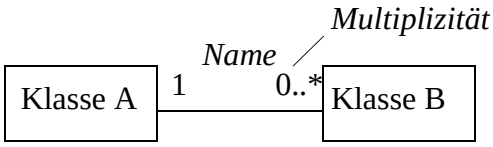


Objekte

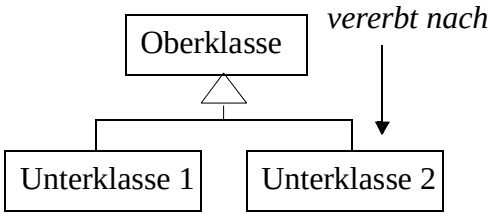


Ein Objekt ist eine Instanz einer Klasse.

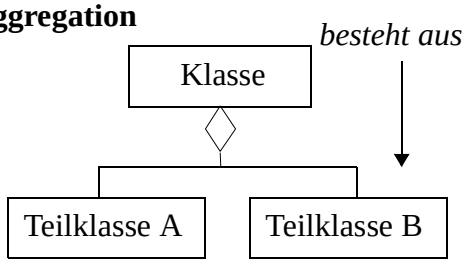
Assoziation



Generalisierung



Aggregation



Kurzerläuterung der Notation des Use-Case-Diagramms

Ein *Use Case* beschreibt eine typische Interaktion eines Systems mit der Umgebung, die durch unterschiedliche Aktoren repräsentiert wird. Ein *Use Case* kann einen anderen *Use Case* auslösen (<<uses>>). Mit <<extends>> wird angegeben, daß ein *Use Case* eine Verfeinerung eines anderen ist. Instanzen (Objekte) können ebenso wie beim Klassendiagramm bezeichnet werden (nicht dargestellt).