Universität Karlsruhe (TH) Institut für Telematik

Integriertes Management anwendungsorientierter Kommunikationsdienste

Diplomarbeit am Institut für Telematik
Prof. Dr. Dr. h.c. Gerhard Krüger
Fakultät für Informatik
Universität Karlsruhe (TH)

von

cand. inform.
Roland Bless

Betreuer:
Prof. Dr. Dr. h.c. Gerhard Krüger
Dipl.-Inform. Claudia Schmidt

Tag der Anmeldung: 1. März 1996
Tag der Abgabe: 26. August 1996

High Performance Networking Group

Ich erkläre hiermit, daß ich die vorliegende Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, den 26. August 1996

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Der zunehmende Einsatz leistungsfähiger, vernetzter Arbeitsplatzrechner hat in Kombination mit der fortschreitenden Entwicklung in der Hochleistungskommunikation die Realisierung vielfältiger neuer verteilter Anwendungen ermöglicht. Während zuvor der zuverlässige Datenaustausch zwischen Rechnern den zentralen Aspekt von Datenkommunikationssystemen darstellte, sind durch die Entwicklung optischer Übertragungstechnologien und multimedialer Peripheriegeräte neue Möglichkeiten zur Übertragung von Medien aller Art geschaffen worden. Die optische Übertragungstechnik ermöglicht hohe Übertragungsraten (mehrere Mbit/s bis Gbit/s) bei gleichzeitig störungsarmer Übermittlung. Erst solche hohen Übertragungsraten machen es möglich, kontinuierliche Medien, wie Bewegtbilder (Video) und Ton (Audio), digitalisiert und in guter Qualität zu übertragen. Heutige Arbeitsplatzrechner sind in der Lage, diese Medien aufzunehmen und wiederzugeben. So wurden verteilte multimediale Anwendungen, wie Video-Konferenzen, Tele-Medizin, Tele-Robotik, computerunterstütztes Gruppenarbeiten, usw. realisierbar.

Kommunikationsdienste zur Übertragung audio-visueller Datenströme sollten daher zusammen mit klassischen Datentransferdiensten von *einem* universellen Telekommunikationssystem angeboten werden (Diensteintegration). Um die Vielfalt der möglichen Dienste zu berücksichtigen, sind erweiterte Dienstprimitive an der Anwendungsschnittstelle und die Unterstützung neuer Dienstqualitäten vorzusehen. Die gleichzeitige Übertragung verschiedener Datenströme an mehrere Teilnehmer sollte ebenfalls möglich sein. Zusätzlich können zwischen einigen Datenströmen Beziehungen bestehen, z.B. zeitlicher Art aufgrund von Synchronisationsanforderungen (beispielsweise Lippensynchronisation). Der von der Anwendung benötigte Kommunikationsdienst zeichnet sich dadurch aus, daß er in Parametern spezifiziert wird, auf denen die Anwendung operiert. Die benötigten Kommunikationsdienste werden jedoch von den beteiligten Systemen noch nicht in ausreichendem Umfang unterstützt, um eine gleichbleibend hohe Qualität zwischen den Kommunikationsteilnehmern zu gewährleisten. Es existieren zwar einzelne Subsysteme, welche die benötigten Grundfunktionen zur Verfügung stellen, jedoch entsteht erst durch das aufeinander abgestimmte Zusammenwirken dieser Subsysteme ein *anwendungsorientierter* Kommunikationsdienst. Solche Subsysteme sind beispielsweise das Kommunikationssubsystem, das Netzwerk und das Betriebssystem.

Die verschiedenen Grundfunktionen der Subsysteme benötigen ein geeignetes *Management*, so daß sie zusammen den gewünschten anwendungsorientierten Kommunikationsdienst realisieren. Ein *integriertes* Management stellt somit den Anwendungen eine Vielfalt an Diensten zur Verfügung, ohne daß diese sich um die Verwaltung der Funktionalität verschiedener Subsysteme kümmern müssen. Das integrierte Management bietet den Anwendungen eine *erweiterte Dienstspezifikation* an, so daß sie ihre Anforderungen anwendungsorientiert formulieren können. Eine weitere Aufgabe des Managements ist die *Dienstabbildung*, welche die Parameter der anwendungsorientierten Spezifikation auf systeminterne Parameter abbildet. Schließlich muß die Funktionalität der einzelnen Subsysteme aufeinander abgestimmt werden, so daß der gewünschte Dienst erbracht wird. Diese *Koordination der Diensterbringung* beinhaltet eine funktionale Konfiguration der Subsysteme und eine *Aushandlung der Dienstqualität* unter den beteiligten Endsystemen. Letzteres ist notwendig, da die Fähigkeiten der einzelnen Kommunikationsteilnehmer stark differieren, einerseits durch unterschiedliche Ausstattung, andererseits durch die momentane Auslastung. Weiterhin muß das Management auch in der Lage sein, Anpassungen während der Kommunikation durchzuführen, da sowohl der Benutzer eine Änderung der Dienstqualität initiieren kann, als auch lastabhängige Schwankungen in der Dienstqualität möglich sind.

1.1 Aufgabenstellung

Die Aufgabe der vorliegenden Diplomarbeit bestand darin, eine Instanz zu entwerfen und prototypisch zu implementieren, die das Management anwendungsorientierter Kommunikationsdienste realisiert. Dieser sogenannte „QoS-Manager" sollte in eine vorgegebene Systemarchitektur (aus [ScZi94]) eingefügt werden und mit den entsprechenden Komponenten dieser Architektur kooperieren. Die Unabhängigkeit von konkreten Implementierungen der Komponenten auf bestimmten Plattformen sollte dabei weitgehend erreicht werden, so daß das integrierte Dienstemanagement auch in heterogenen Umgebungen einsetzbar ist. Wesentliche Teilaufgaben waren die Identifikation der ins Dienstemanagement involvierten Komponenten, die Koordination der Diensterbringung, die Festlegung von Abläufen zur Dienstaushandlung, die Formulierung von Dienstgüteabbildungen und der Entwurf eines Management-Protokolls für die Kommunikation zwischen Managementinstanzen.

Zunächst wurden in einem umfassenden Literaturstudium verschiedene Ansätze zum qualitätsorientierten Management von Kommunikationsdiensten analysiert. Es wurden die Vor- und Nachteile der jeweiligen Ansätze diskutiert. Die Untersuchung ergab, daß die Ansätze oft nur wenige Aspekte eines umfassenden anwendungsorientierten Dienstemanagements berücksichtigen. Die Identifizierung der Aufgaben eines integrierten anwendungsorientierten Dienstemanagements diente dazu, die Aufgaben und Anforderungen an die zu entwerfende Managementinstanz zu formulieren. Die Zerlegung in Teilaufgaben resultierte auch beim Entwurf in eine Zerlegung der Managementinstanz in Teilinstanzen, die jeweils bestimmte Aufgaben übernehmen. Diese Zerlegung in Teilinstanzen findet sich schließlich auch in der Implementierung wieder.

1.2 Gliederung

Kapitel 2 beschreibt zunächst einige Grundlagen, die im weiteren Verlauf der Arbeit verwendet werden. Dabei werden die Anforderungen und Aufgaben eines integrierten Dienstemanagements analysiert und eine Systemarchitektur vorgestellt, die zur Lösung der Aufgabe verwendet werden soll. Im Anschluß daran wird eine Auswahl bestehender Ansätze zum Management qualitätsorientierter Dienste diskutiert. Abschließend werden die Aufgaben der Managementinstanz noch einmal zusammengefaßt.

Kapitel 3 beschreibt den Entwurf der Managementinstanz. Zuerst wird die verwendete Architektur der Instanz beschrieben und anschließend werden die einzelnen Teile der Architektur detailliert erläutert.

In Kapitel 4 wird die Implementierung der in Kapitel 3 entworfenen Instanz beschrieben.

Die Arbeit schließt mit einer kurzen Zusammenfassung und einem Ausblick auf mögliche Erweiterungen in Kapitel 5.

2. Analyse

In diesem Kapitel werden zunächst einige Grundlagen beschrieben, die im weiteren Verlauf der Arbeit verwendet werden. Danach wird eine Auswahl bestehender Ansätze zum Dienstemanagement vorgestellt. Schließlich werden die Aufgaben der zu entwerfenden Managementinstanz formuliert.

2.1 Grundlagen

In diesem Abschnitt werden die grundlegenden Begriffe definiert und erläutert, die im Rahmen dieser Arbeit verwendet werden. Zuerst wird der Begriff des *Kommunikationsdienstes* eingeführt. Anschließend folgt eine Beschreibung der darauf aufbauenden *Diensteintegration*, bei deren Ausführung auch garantierte Dienste, wichtige Dienstgüteparameter, Dienstklassen und die Definition einer „Sitzung“ erläutert werden. Im dann folgenden Abschnitt wird auf die Rolle der *Ressourcen* für die Diensterbringung eingegangen. In Abschnitt 2.1.4 wird der Begriff des *Dienstemanagements* eingeführt und schließlich in Abschnitt 2.1.5 eine Systemarchitektur vorgestellt, die ein solches Dienstemanagement geeignet unterstützen soll.

2.1.1 Kommunikationsdienste

Im traditionellen ISO/OSI-Basisreferenzmodell [JaAg90, LoKK93] ist das Diensteparadigma von zentraler Bedeutung. Das ISO/OSI-Basisreferenzmodell ist modular und schichtenorientiert aufgebaut. Dabei bietet jede Schicht der nächsthöheren ihren *Dienst* als *Dienstgeber* an ihrer *Dienstschnittstelle* an. Dieser Dienst wird von der darübergelegenen Schicht (als sogenannter *Dienstnehmer*) genutzt und funktional erweitert, so daß an ihrer Dienstschnittstelle der erweiterte Kommunikationsdienst der wiederum nächsthöheren Schicht zur Verfügung steht. Somit entsteht eine Dienstehierarchie, die auf einfachen Diensten aufbauend immer komplexere Dienste zur Verfügung stellt. Die konkrete Implementierung der Dienste bleibt dabei im Innern der Schichten verborgen.

Ein Dienst wird vollständig durch eine Menge von *Dienstelementen* und eine damit assoziierte *Dienstqualität* definiert. Diese Dienstelemente stellen spezifische Funktionen des Dienstgebers dar und werden über eine Folge von *Dienstprimitiven* realisiert

(*Dienstprozedur*). Ein Dienstprimitiv ist ein nicht weiter unterteilbarer Bestandteil einer solchen Dienstfunktion. Die vier Arten von Dienstprimitiven sind: Anfrage (*request*), Anzeige (*indication*), Antwort (*response*) und Bestätigung (*confirmation*). *Bestätigte* Dienste umfassen alle vier Arten von Dienstprimitiven, während *unbestätigte* Dienste nur die ersten beiden Arten verwenden (siehe Abbildung 2.1).

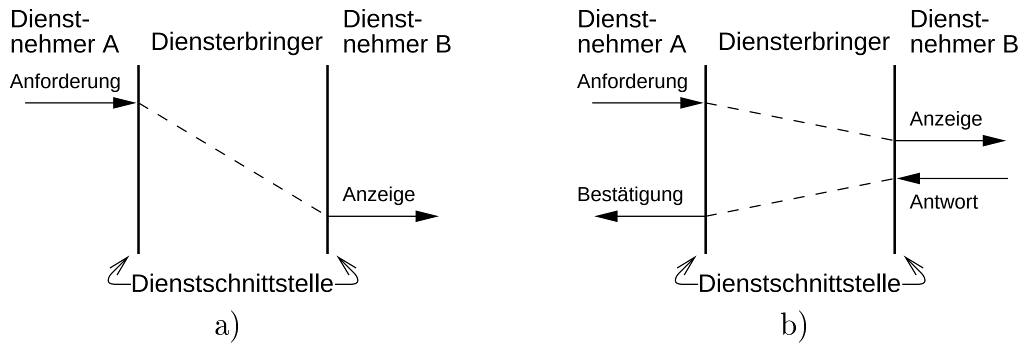

Abbildung 2.1: Unbestätigter Dienst a) und bestätigter Dienst b)

Zur exakten Definition eines Dienstes wird noch die Angabe einer *Dienstqualität* (auch: *Dienstgüte*) benötigt, wodurch die qualitativen Eigenschaften des Dienstes für den Dienstnutzer genauer spezifiziert werden [LoKK93]. Von der ITU-T[1] wird die Dienstqualität (Quality of Service, abgekürzt QoS) definiert als:

> „Collective effect of service performances which determines the degree of satisfaction of a user of a service"

Als Beispiele für traditionelle Dienstgüteparameter seien hier kurz Datenverlust, Fehlerwahrscheinlichkeiten für Verbindungsauf- und -abbau, Durchsatz, Verzögerungszeit, Schutz, Kosten, etc. genannt.

2.1.2 Diensteintegration

Durch die *Diensteintegration* werden dem potentiellen Dienstnutzer qualitativ verschiedenartige Telekommunikationsdienste (beispielsweise Sprach- und Bewegtbildübertragung, sowie Telefax- oder traditionelle Datenübertragungsdienste) angeboten. Ein Beispiel für die Nutzung integrierter Dienste sind multimediale Anwendungen (z.B. Videokonferenzen, Tele-Medizin, Tele-Robotik, Video auf Abruf) oder kooperatives, computerunterstütztes Gruppenarbeiten (Computer Supported Cooperative Work – CSCW), die im allgemeinen verschiedenartige *garantierte Kommunikationsdienste* von Anwendung zu Anwendung („Ende-zu-Ende") benötigen. Dem Nutzer solcher „garantierten" Dienstewird zugesagt, daß die Schwankung in der Qualität der Dienste durch eine von ihm gewählte Schranke begrenzt bleibt.

[1]International Telecommunication Union – Telecommunication Standardization Sector of ITU

So wird beispielsweise für eine Videoübertragung ein gewisser Mindestdurchsatz und eine Beschränkung der Verzögerungsschwankung benötigt, um eine für den Benutzer akzeptable Qualität zu gewährleisten. Die Anforderung eines garantierten Dienstes benötigt eine detaillierte Charakterisierung der gewünschten Qualität und Garantie an der *Dienstschnittstelle*, über welche die Anwendung und das Kommunikationssystem interagieren.

Für diensteintegrierende Kommunikationssysteme ist vor allem die garantierte Einhaltung folgender Dienstgüteparameter von Bedeutung:

- Durchsatz (Throughput)
  Anzahl der übertragenen Nutzdaten pro Zeiteinheit. Für den Dienstnutzer ist ein garantierter Mindestdurchsatz von Interesse.

- Ende-zu-Ende Verzögerung (Delay)
  Zeitspanne zwischen Annahme einer Dateneinheit beim Sender zur Übertragung und der Auslieferung der Dateneinheit beim Empfänger. Besonders für interaktive Anwendungen ist eine Beschränkung durch die Vorgabe einer *maximalen Ende-zu-Ende Verzögerung* wichtig, da Reaktionen ansonsten nicht mehr unmittelbar erscheinen (beispielsweise macht sich eine lange Übertragungsverzögerung beim Telefonieren störend bemerkbar, d.h. der Sprecher ist irritiert, wenn er seine eigene Sprache noch mehr als 100ms nach einer Sprechpause hören kann [Part94]).

- Verzögerungsschwankung (Delay-Jitter)
  Auf einer idealen Übertragungsstrecke erfährt jede Dateneinheit die gleiche Ende-zu-Ende Verzögerung. In der Realität variiert diese Ende-zu-Ende Verzögerung jedoch aufgrund unterschiedlicher Bearbeitungszeiten in den Zwischen- und Endsystemen. Für Medienströme, die einen *gleichmäßigen* Datenfluß erfordern (z.B. Audio und Video), wird es nötig, diese Schwankung zu begrenzen. Das wird durch Vorgabe einer *maximalen Verzögerungsschwankung* erreicht, die die Differenz zwischen der maximalen und minimalen Ende-zu-Ende Verzögerung bezeichnet.

- Zuverlässigkeit (Reliability)
  Die Zuverlässigkeit bezeichnet die Zusammenfassung der beiden Fehlerarten Datenverlust und Datenverfälschung. In heutigen Netzwerken sind die Bitfehlerraten durch den vermehrten Einsatz glasfaserbasierter Medien nur noch sehr gering (bis 10^{-15} [Zitt95]), dafür werden aber durch überlastete Zwischensysteme verursachte Verluste von Dateneinheiten wahrscheinlicher, denn der Pufferplatz in diesen Zwischensystemen ist begrenzt. Daher werden sogenannte „Burst-Fehler“, die einen Verlust mehrerer unmittelbar aufeinanderfolgender Dateneinheiten bezeichnen, öfter auftreten. Für eine Anwendung wird meistens eine verfälschte Dateneinheit nutzlos sein, so daß sie wie eine verlorengegangene Dateneinheit angesehen werden kann. Für zeitkritische Anwendungen

haben zu spät angekommene Dateneinheiten die gleiche Bedeutung wie verlorengegangene [Ferr90]. Diese sollen aber hier nicht mit einbezogen werden. Die Zuverlässigkeit läßt sich detailliert durch ein Tripel (i, m, c) spezifizieren, wobei i ein Intervall von i aufeinanderfolgenden Dateneinheiten bezeichnet, m die maximale Anzahl der fehlerhaften Dateneinheiten innerhalb dieses Intervalls und c die maximale Anzahl der Dateneinheiten, die unmittelbar hintereinander fehlerhaft sein dürfen (es muß gelten $c \leq m \leq i$) [GoPa92]. So bedeutet beispielsweise $(100, 10, 3)$, daß unter 100 unmittelbar aufeinanderfolgenden Dateineinheiten höchstens 10 fehlerhaft sein dürfen und nicht mehr als 3 fehlerhafte direkt aufeinanderfolgen dürfen.

Die ersten drei Parameter stehen in enger Beziehung zueinander, da sie letztlich von der rechtzeitigen Bearbeitung in den End- und Zwischensystemen abhängen. Sie sollen daher unter dem Begriff *leistungsorientierte Dienstgüteparameter* zusammengefaßt werden.

Zu welchem Grad die Garantie eines Parameters erbracht werden soll, läßt sich anhand einer Einteilung in *Dienstklassen* noch genauer spezifizieren [Ferr90, CaCH94, Zitt95]. Es werden drei Klassen unterschieden:

- *deterministisch*
  Parameter dieser Klasse dürfen keine Werte annehmen, die außerhalb der vorgegebenen Grenzen liegen. Der Parameter ist immer, d.h. auch im schlechtesten Fall, zu garantieren. Typischerweise erfolgt die Vorgabe durch ein Intervall mit einer unteren und oberen Schranke. Beispielsweise läßt sich eine Bedingung für die Ende-zu-Ende Verzögerung d so ausdrücken: $20ms \leq d \leq 100ms$. Das Intervall kann auch zu einer Seite offen sein, weil oft nur eine obere oder untere Schranke von Bedeutung ist. So wird häufig eine untere Schranke für den Durchsatz (Mindestdurchsatz), aber eine obere Schranke für die Ende-zu-Ende Verzögerung von Bedeutung sein.

- *statistisch*
  Der Parameter wird mit einer anzugebenden Wahrscheinlichkeit garantiert, d.h. Parameter dieser Klasse dürfen auch Werte außerhalb der vorgegebenen Grenzen annehmen, allerdings nur bis zu einer vorgegebenen Häufigkeit. Es wird daher zu einem Grenzintervall zusätzlich eine *gebrochene Schranke* (a, b) spezifiziert [Ferr90]. Dadurch wird für a aufeinanderfolgende Parameterwerte vorgegeben, wieviele Male b höchstens ein Parameterwert außerhalb der vorgegebenen Grenzen liegen darf (es gilt $b < a$). Zählt eine Funktion $C_a(P)$ für a aufeinanderfolgende Auswertungen der Bedingung P, wie oft P nicht erfüllt ist, so läßt sich für einen Parameter x eine statistische Bedingung wie folgt formulieren: $C_a(x_{\min} \leq x \leq x_{\max}) \leq b$. Der Vorteil der Formulierung der Wahrscheinlichkeit durch eine gebrochene Schranke (a, b) liegt gegenüber einer einzelnen Prozentangabe $p\%$ in der einfacheren Überprüfung und der exakteren Spezifikation, da bei $p\%$ die Länge des betrachteten Intervalls nicht

vorgegeben ist (z.B. $5\% \hat{=} 5/100 = 100/2000$). Ein Parameter der statistischen Klasse bietet aufgrund der gewährten Überschreitung der Grenzen keine so gute Garantie, wie ein Parameter der deterministischen Klasse.

- *bestmöglich*
  Parameter dieser Klasse bieten keinerlei Garantie. Daher ist eine Angabe von Grenzen für Parameterwerte hinfällig, denn dem System wird vollkommen freigestellt, welche Parameterwerte tatsächlich erzielt werden.

Die leistungsorientierten Dienstgüteparameter werden traditionell nur bestmöglich erbracht (z.B. im Internet mit TCP/IP). So wirkt sich die durch andere Nutzer erzeugte Last auch auf die eigene Nutzung einschränkend aus.

Für die weiteren Ausführungen sollen nun die Begriffe *Datenstrom* und *Sitzung* eingeführt werden, die im Rahmen der Arbeit von zentraler Bedeutung sind:

Definition 1 DATENSTROM:
Ein Datenstrom (flow, stream) ist eine unidirektionale, Punkt-zu-Punkt oder Punkt-zu-Mehrpunkt Kommunikationsverbindung von Anwendung zu Anwendung. Diesem Datenstrom ist eine Dienstqualität zugeordnet.

Üblicherweise verlangen komplexere Anwendungen (z.B. multimediale) die gleichzeitige Übertragung mehrerer Datenströme mit unterschiedlicher Dienstqualität, die zusätzlich in einer *zeitlichen* oder *logischen* Beziehung zueinander stehen können [TaFH94]. So stellt die zeitliche Beziehung zwischen einem Videostrom und einem zugehörigen Audiostrom die Notwendigkeit der *Synchronisation* beider Ströme dar. Eine Sitzung beschreibt die Kommunikationsbeziehungen zwischen zwei oder mehreren gleichen Anwendungen, basierend auf Datenströmen:

Definition 2 SITZUNG:
Eine Sitzung umfaßt einen oder mehrere Datenströme, die eine unterschiedliche Dienstqualität besitzen können. Zwischen diesen Datenströmen können zeitliche und logische Beziehungen bestehen. Datenströme können im Verlauf einer Sitzung dynamisch erzeugt, beendet oder in ihrer Dienstqualität verändert werden.

Im folgenden Abschnitt wird erläutert, wie eine Garantie für Dienstqualitätsparameter erreicht werden kann. Dafür wird zunächst der Begriff „Ressource“ eingeführt.

2.1.3 Ressourcen

Die Kommunikationsdienste werden durch die Nutzung von *Ressourcen* (technische Komponenten, Betriebsmittel [Wett93]) erbracht. Es lassen sich *aktive* und *passive* Ressourcen unterscheiden. Aktive Ressourcen sind jegliche Art von eigenständigen Verarbeitungseinheiten (z.B. Zentralprozessor oder eine Netzwerkadapterkarte) und passive Ressourcen sind vornehmlich Speicher oder Bandbreite. In einem End- oder Zwischensystem stehen solche Ressourcen jedoch normalerweise nicht exklusiv nur einer Anwendung zur Verfügung, sondern sie werden unter mehreren geteilt. Somit wird eine Verwaltung dieser Ressourcen durch sogenannte *Ressourcenverwalter* nötig, die den Wettbewerb um diese gemeinsam zu nutzenden Ressourcen regeln und für eine möglichst effiziente Auslastung der Ressourcen sorgen sollen. Die Zuteilung einer aktiven Ressource übernimmt z.B. ein sogenannter *Scheduler*, der die Bearbeitungsreihenfolge der zu bearbeitenden Dateneinheiten festlegt.

Um garantierte Dienste erbringen zu können, muß eine *Reservierung* der Ressourcen bereits vor ihrer ersten Nutzung vorgenommen werden (üblicherweise beim Verbindungsaufbau). Dazu wird das Prinzip der *Zugangskontrolle* (admission control) verwendet: jeder Nutzer der Ressourcen hat vor deren Benutzung eine Anfrage an den Ressourcenverwalter mit dem benötigten Umfang der Ressourcennutzung zu stellen. Beim *Zugangskontrolltest* wird geprüft, ob die Ressourcen noch über ausreichende Kapazität verfügen, um die neue Anforderung zu erfüllen, ohne die bereits bestehenden Dienstgarantien anderer Nutzer zu verletzen. Kann die Nutzung im gewünschten Umfang gewährt werden, so muß die entsprechende Ressourcenmenge gleich im Zuge der Anfrage *vorreserviert* werden, damit nicht zwischen Anfrage und tatsächlicher Reservierung ein anderer Nutzer einen Teil der benötigten Ressourcenmenge reserviert. Kann die Nutzung im benötigten Umfang hingegen nicht gewährt werden, so muß die Forderung abgelehnt und eventuell später wiederholt werden. Auf diese Weise wird sichergestellt, daß andere Nutzer, die bereits reservierte Ressourcen benutzen, von der Last zusätzlicher Benutzer nicht beeinträchtigt werden. Wenn die Ressourcen nicht mehr benötigt werden, erfolgt eine explizite Freigabe der reservierten Ressourcen durch deren Nutzer.

Mit den bereits vorgestellten Dienstklassen erkennt man, daß für die Parameter der deterministischen Klasse Ressourcen *exklusiv* anhand des schlechtesten Falles („worst-case") reserviert werden müssen, da keinerlei Grenzüberschreitungen erlaubt sind. Die Ressourcenauslastung wird allerdings in den meisten Fällen schlecht sein, da typischerweise nicht permanent eine Spitzenlast auftritt und somit im Mittel zu viel Ressourcen reserviert wurden. Für die statistische Klasse können Ressourcen aber anhand von Mittelwerten reserviert werden, da ein kurzzeitiges Überschreiten der Grenzen erlaubt ist. Somit ist eine bessere Ressourcenauslastung als im deterministischen Fall gewährleistet, da weniger Ressourcen reserviert werden und somit die verbleibende Ressourcenkapazität auch von anderen Anwendungen genutzt werden kann. Für die Klasse „bestmöglich" werden keine Ressourcen reserviert. Mit zunehmender Strenge der Garantie wird also die zu erwartende Auslastung der Ressourcen schlechter.

2.1.4 Dienstemanagement

Es besteht auf der einen Seite das Ziel, Anwendungen komplexe Kommunikationsdienste zur Verfügung zu stellen (Diensteintegration, Abschnitt 2.1.2), auf der anderen Seite müssen diese Dienste durch die Nutzung verschiedener Ressourcen erbracht werden. Die Verbindung zwischen beiden Ebenen soll durch ein *Dienstemanagement* geschaffen werden, dessen Aufgaben in Abbildung 2.2 dargestellt sind. Es lassen sich

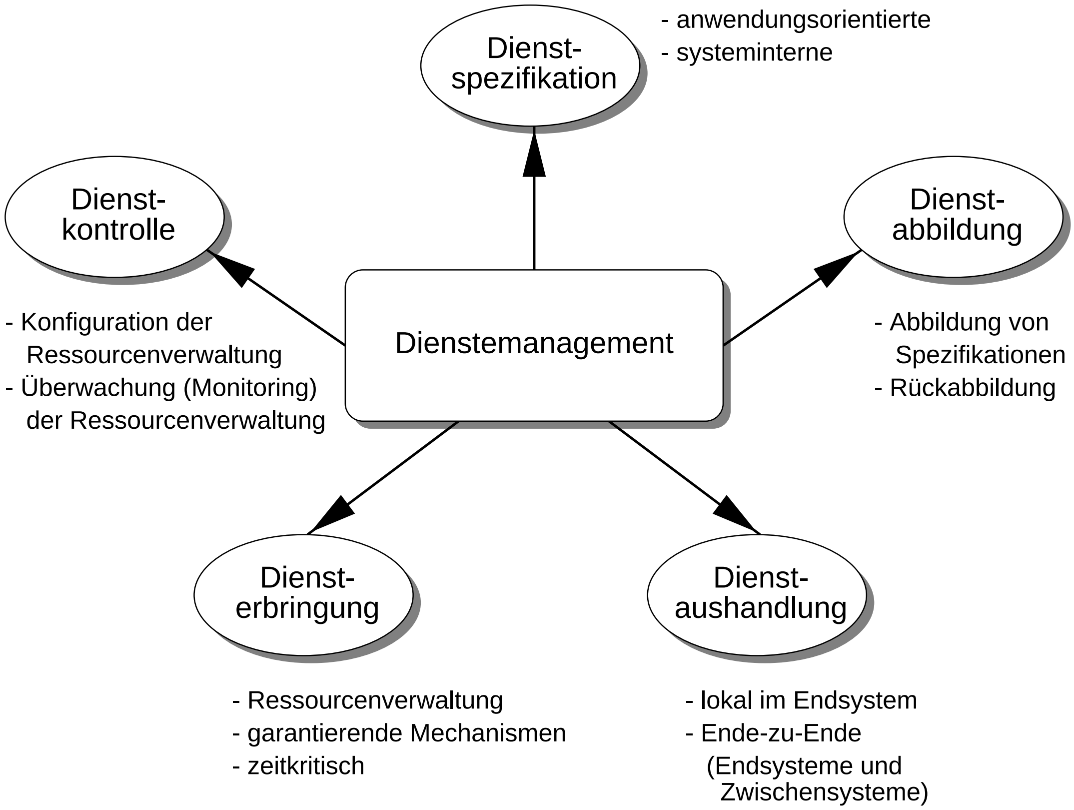

Abbildung 2.2: Aufgaben des Dienstemanagements

fünf Hauptaufgaben identifizieren [Schm96]:

- Dienstspezifikation
  Fortgeschrittene Anwendungen sollten eine anwendungsorientierte Dienstspezifikation benutzen, welche die Bedürfnisse der Anwendung vollständig abdeckt und ohne die Spezifikation interner Parameter (wie z.B. Durchsatz und Jitter) auskommt. Unterhalb der Anwendungsschnittstelle, d.h. *systemintern* werden jedoch erweiterte Dienstspezifikationen für die Systemkomponenten benötigt, um z.B. die erforderlichen Garantien zu formulieren. Eine Dienstspezifikation sollte einen möglichst großen Spielraum für die Aushandlung von Parameterwerten bieten, was z.B. durch die Angabe von Intervallen für akzeptable Parameterwerte realisiert werden kann.

- Dienstabbildung
  Die Dienstabbildung sorgt dafür, daß die Dienstspezifikationen ineinander umgesetzt werden können, denn schließlich müssen die anwendungsorientierten

Parameter auf systeminterne Anforderungen abgebildet werden. Der Abbildung von systeminternen Parametern auf anwendungsorientierte („Rückabbildung“) kommt eine besondere Bedeutung zu, da auch eine Änderung des Status der Systemkomponenten interpretiert werden muß.

- Dienstaushandlung
  Alle an der Kommunikation beteiligten Komponenten müssen sich auf konkrete Parameterwerte aus der Dienstspezifikation einigen. Bei einer „lokalen Aushandlung“ wird zunächst nur lokal im Endsystem die Reservierung der Ressourcen veranlaßt. Aufgrund der Zugangskontrolle können die Ressourcenverwalter bestimmen, welche der Werte aus der Dienstspezifikation akzeptiert werden, deshalb werden hier schon tatsächlich Werte *ausgehandelt*. Scheitert die Reservierung jedoch bereits lokal, kann eine „Ende-zu-Ende“-Aushandlung mit den übrigen involvierten Endsystemen unterbleiben. Die Teilnehmer einer Kommunikationsbeziehung handeln bei deren Aufbau einen sogenannten *Verkehrsvertrag* aus, welcher für alle Teilnehmer dieser Kommunikation bindend ist. Er enthält die genaue Spezifikation des vom Kommunikationssystem zugesicherten und von den Anwendungen vereinbarten Dienstes.

- Diensterbringung
  Wie in Abschnitt 2.1.3 bereits geschildert wurde, müssen geeignete Mechanismen für die Ressourcenverwaltung existieren, um garantierte Dienste zu erreichen. Eine Unterstützung für Dienstgarantien auf Ressourcenebene ist unerläßlich, denn nur in wenigen Fällen ist eine Dienstgarantie ohne entsprechende Garantien der Ressourcenverwaltung möglich. Die Diensterbringung ist als zeitkritisch anzusehen, da sie im Datenpfad, d.h. auf Dateneinheiten operiert.

- Dienstkontrolle
  Es sind verschiedene Mechanismen für die Verwaltung der Ressourcen denkbar (etwa unterschiedliche Scheduling-Algorithmen), um eine gewünschte Funktionalität zu erbringen. Daher soll der Ressourcenverwaltung noch eine Kontrollinstanz vorgeschaltet werden, die jene entsprechend *konfiguriert* und *überwacht*. Die Überwachung (auch als Monitoring bezeichnet) ermöglicht es, auf Schwankungen in der Diensterbringung zu reagieren, was z.B. für Parameter der statistischen Dienstklasse wichtig ist. Es können somit auch Verletzungen des Verkehrsvertrages (z.B. aufgrund aussergewöhnlicher Ereignisse) festgestellt werden. Die Dienstkontrolle ist jedoch weniger zeitkritisch als die Diensterbringung, denn nur die Überwachung sammelt Informationen aus dem Datenpfad.

Im folgenden Abschnitt soll nun eine Systemarchitektur vorgestellt werden, die ein solches Dienstemanagement realisiert [Schm96].

2.1.5 Systemarchitektur

Im Endsystem können mehrere Komponenten identifiziert werden, die verschiedene Ressourcen verwalten und zur Erbringung eines garantierten Dienstes eng kooperieren müssen. Folgende Komponenten sollen unterschieden werden [ScZi94]: *Kommunikationssubsystem*, *Netzwerk*, *Betriebssystem*. Für den Transport der Daten wird ein Basisdienst der Netzwerkkomponente meistens durch die zusätzliche Funktionalität des Protokolls im Kommunikationssubsystem erweitert, so daß beide Komponenten zusammen den geforderten Kommunikationsdienst erbringen können. Eine geeignete Unterstützung im Betriebssystem ist notwendig, um die gewünschte Qualität bis zum Nutzer der Anwendung zu erhalten. Da letztendlich anwendungsorientierte Dienste nur durch die Benutzung realer Ressourcen verwirklicht werden können, müssen diese in einer geeigneten Weise konfiguriert, betrieben und überwacht werden. Jede Komponente besitzt daher eine *Kontrollinstanz*, bestehend aus einem *Kontrollagenten* und einer *Monitorinstanz*. Der Kontrollagent bestimmt die Auswahl und Kontrolle der Mechanismen, welche die Ressourcen verwalten. Diesen Agenten werden noch *Monitorinstanzen* zur Seite gestellt, die eine Überwachung der Mechanismen durchführen und überprüfen, ob die geforderte Leistung erbracht wird [Bles95]. Somit ist es möglich, auf Veränderungen zu reagieren und ein dynamisches Management zu realisieren. Die Kontrollinstanzen übernehmen damit genau die in 2.1.4 beschriebenen Aufgaben der Dienstkontrolle.

Um Anwendungen jedoch integrierte Kommunikationsdienste anbieten zu können, müssen die drei Komponenten Kommunikationssubsystem, Netzwerk und Betriebssystem geeignet zusammenarbeiten. Dieses kann durch ein *integriertes Management* der Komponenten erreicht werden (s. Abbildung 2.3). Der sogenannte *QoS-Manager* stellt die Managementinstanz dar, welche die zusammenfassende *Steuerung und Verwaltung der Komponenten* realisiert. Ein *integriertes* Management ist außerdem vorteilhaft, da so die Wechselwirkungen der Komponenten untereinander besser berücksichtigt werden können. Beispielsweise resultiert die Benutzung einer Vorwärtsfehlerkorrektur im Kommunikationssubsystem in einer erhöhten Durchsatzforderung gegenüber dem Netzwerkdienst, ebenso wird die benötigte Funktionalität im Kommunikationssubsystem durch die Wahl der Funktionalität des Netzwerkdienstes mitbestimmt.

2.1.5.1 Die Netzwerkkomponente

Die Netzwerkkomponente umfaßt typischerweise das physikalische Medium, die Bitübertragungsschicht, und die Netzzugangsschnittstelle. Für das ATM-basierte B-ISDN wäre letztere also eine Schnittstelle zur ATM Adaptionsschicht (ATM Adaption Layer, kurz AAL). Bei traditionellen verbindungslosen lokalen Netzwerken (LANs) wie Ethernet, Token-Ring oder FDDI[2] ist dies für eine globale Kommunikation noch nicht ausreichend. Dort wird zusätzlich die Vermittlungsschicht benötigt,

[2] Fibre Distributed Data Interface

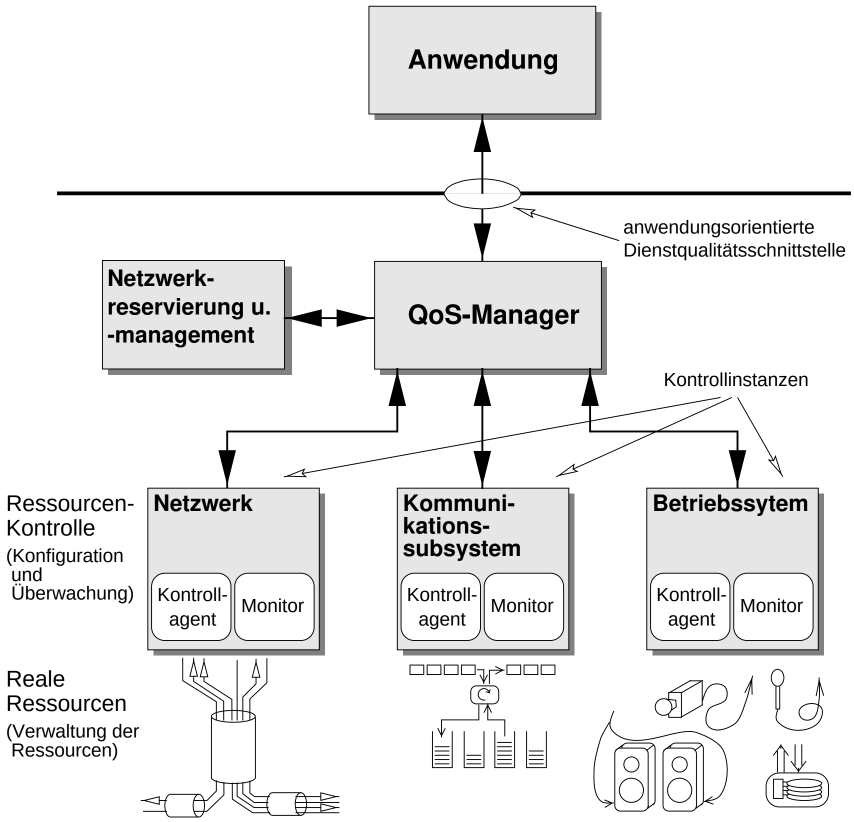

Abbildung 2.3: Umgebung des QoS-Managers

um die Interkonnektivität herzustellen (ein weit verbreitetes Protokoll der Vermittlungsschicht ist beispielsweise IP, das Internet Protocol).

Zwischensysteme besitzen ebenfalls die Funktionalität einer solchen Netzwerkkomponente (s. Abbildung 2.4). Ressourcen in den Zwischensystemen sind Prozessor

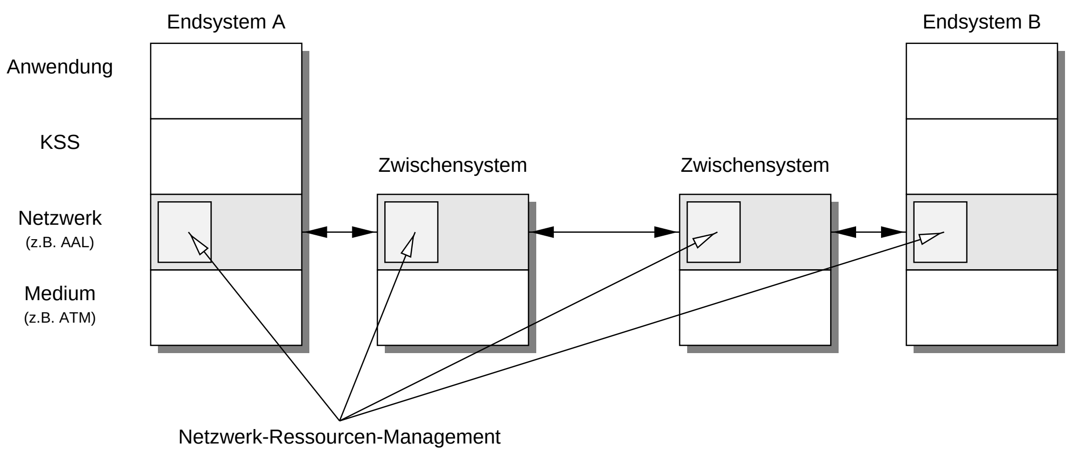

Abbildung 2.4: Netzwerkkomponenten in Zwischensystemen

und Pufferspeicher, die durch einen Ressourcenverwalter zuzuteilen sind. Durch Anwendung der in Abschnitt 2.1.3 eingeführten Ressourcenreservierung und geeigneter Zuteilungsverfahren lassen sich Garantien für Durchsatz, Ende-zu-Ende Verzögerung und Verzögerungsschwankung erreichen. Für das Entgegennehmen, Umsetzen und Weiterleiten solcher Reservierungsanfragen wird eine entsprechende Instanz des Netzwerk-Resourcen-Managements benötigt, die somit in jedem Zwischensystem vorhanden ist.

2.1.5.2 Kommunikationssubsystem

Basierend auf den Netzwerkdiensten erbringt das Kommunikationssubsystem eine erweiterte Funktionalität, die einen vollständigen Kommunikationsdienst abdeckt. Typische Aufgaben sind etwa Verbindungsverwaltung, Ende-zu-Ende Fehlerkontroll- und Fehlerbehebungsmechanismen, Segmentieren und Reassemblieren, Fluß- und Ratenkontrolle. Ein fortgeschrittenes Kommunikationssubsystem läßt sich konfigurieren, so daß nur die wirklich benötigten Funktionen eingeschaltet werden [Brau93]. Reale Ressourcen im Kommunikationssubsystem sind Puffer und Bearbeitungszeit, Verbindungen stellen logische Ressourcen dar. In Abhängigkeit von den Anwendungsanforderungen stellt das Kommunikationssubsystem verschiedene Dienste zur Verfügung, die im traditionellen Schichtenmodell unterschiedlichen Schichten zugeordnet werden. So kann eine Anwendung beispielsweise einen einfachen Transportdienst benötigen oder einen Dienst, der mehrere kontinuierliche untereinander synchronisierte Datenströme verwaltet.

2.1.5.3 Betriebssystem

Einige Datenströme benötigen zusätzliche Unterstützung durch spezielle Geräte im Endsystem. Denkbar sind z.B. Audio-Karten, Video-CODECs (Kodier- und Dekodierkarten, für Datenreduktions- oder Kompressionsverfahren wie beispielsweise MPEG-2), Video-Framegrabber und Disk-Arrays (mehrere zusammengeschaltete Festplatten erzielen höhere Datentransferraten, die z.B. für die Videowiedergabe in hoher Qualität benötigt werden). Somit sollte die Betriebssystemkomponente dafür sorgen, der Anwendung die entsprechende Unterstützung durch solche Geräte zukommen zu lassen. Auch die Anwendung selbst benötigt für die Bearbeitung der Datenströme Zeit und Speicherplatz (Puffer). So sollte für zeitkritische Datenströme z.B. unbedingt ein Speicherbereich gewählt werden, der nicht von der Speicherverwaltung ausgelagert werden kann, da solche Vorgänge auf Hintergrundspeicher wie Festplatten viel zu lange dauern. Eine wichtige Rolle spielt ebenfalls die Unterstützung geeigneter Scheduling-Algorithmen, um Anwendungen ein weitgehend[3] deterministisches Verhalten zu garantieren. Von der konkreten Ausprägung des Betriebssystems mit all seinen Eigenheiten sollte der Anwendungsprogrammierer jedoch nach Möglichkeit verschont bleiben, andernfalls müßte die Anwendung für jedes Betriebssystem neu angepaßt werden.

[3] diese Einschränkung resultiert aus dem Unterbrechungskonzept (Interrupts)

2.2 Bestehende Ansätze zum Dienstemanagement

In diesem Abschnitt wird eine Auswahl bereits bestehender Ansätze für das Management von qualitätsorientierten Kommunikationsdiensten kurz vorgestellt. Es werden drei Ansätze beschrieben: Die OMEGA-Architektur der Universität von Pennsylvania, das MASI-Rahmenwerk der Universität Pierre et Marie Curie in Paris und die QoS-Architektur der Universität Lancaster. Anschließend erfolgt eine Diskussion der drei Ansätze.

2.2.1 OMEGA-Architektur

Die OMEGA-Architektur wurde an der Universität von Pennsylvania von Klara Nahrstedt zur Unterstützung garantierter Dienste von Anwendung zu Anwendung entwickelt [NaSm95, Nahr95]. Das dort zugrundegelegte Kommunikationsmodell eines Endsystems ist in Abbildung 2.5 dargestellt: Es besteht aus den beiden Haupt-

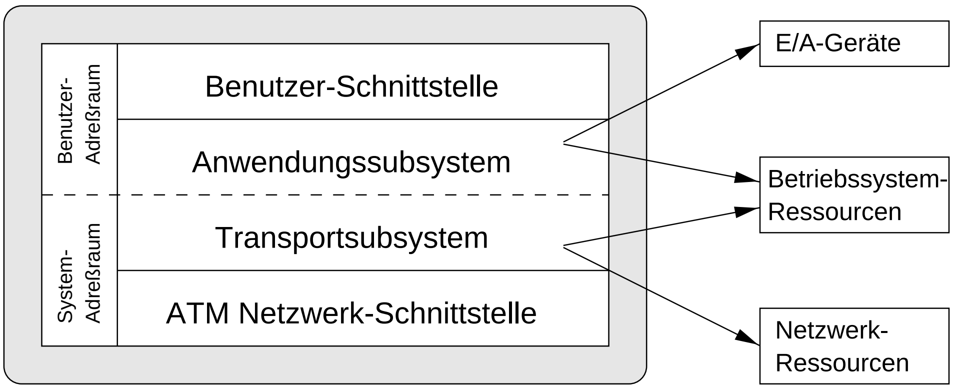

Abbildung 2.5: Modell eines Endsystems

komponenten *Anwendungssubsystem* und *Transportsubsystem*. Das Anwendungssubsystem verwaltet multimediale Verbindungen, Ein-/Ausgabe Geräte und ist für die Synchronisation und Auslieferung der Medienströme verantwortlich. Das Transportsubsystem übernimmt die Verbindungsverwaltung, Fehlerkorrektur, Ratenkontrolle und den Datentransfer von und zum Netzwerkadapter, umfaßt Aufgaben der traditionellen Transport- und Netzwerkschicht und verwendet *Integrated Layer Processing (ILP)* [ClTe90] (die Erläuterung zu ILP folgt in Abschnitt 3.5). Daher existiert keine Grenze zwischen Netzwerk- und Transportschicht, sie sind vielmehr im Transportsubsystem zusammengefaßt. Im Anwendungssubsystem kommt ebenfalls ILP und *Application Level Framing (ALF)* [ClTe90] zum Einsatz.

Es werden drei Arten von Ressourcen unterschieden: Multimediale Geräte, Betriebssystem- und Netzwerkressourcen. Da die Ressourcen des Betriebssystems und des Netzwerks unter mehreren Nutzern geteilt werden, zeigen sie ein dynamisches Verhalten. Das Verhalten der drei Komponenten beeinflußt sich zum großen Teil ge-

genseitig. Daher wird eine Ressourcenverwaltung im Endsystem durch eine einzelne Instanz, den *QoS-Broker* vorgeschlagen. Der QoS-Broker stimmt die benötigten Ressourcen für die Aufgaben des Anwendungssubsystems und des Transportsubsystems miteinander ab und verhandelt mit dem Netzwerk-Ressourcenmanagement und dem QoS-Broker im entfernten Endsystem. Der QoS-Broker kann unterschiedliche Rollen annehmen: Ein *Broker-Buyer* möchte Ressourcen von einem anderen entfernten Endsystem „kaufen". Dort bietet der *Broker-Seller* die zur Verfügung stehenden Ressourcen „zum Verkauf" an. Der Broker-Buyer initiiert dabei den Verhandlungsprozeß. Zusätzlich findet noch eine weitere Einteilung anhand der Richtung des Datenflusses statt, denn es existiert je ein Broker auf Empfangs- und Sendeseite, die entsprechend als *Broker-Receiver* und *Broker-Sender* bezeichnet werden. Beide bestehen jeweils aus einem Broker-Buyer und einem Broker-Seller (s. Abbildung 2.6). Der Broker-Sender ist für Ressourcen von abgehenden Verbindungen

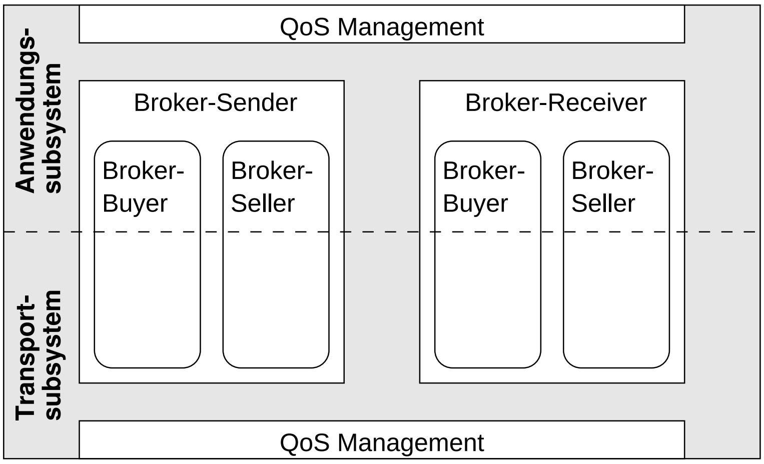

Abbildung 2.6: Architektur des QoS-Brokers

verantwortlich, der Broker-Receiver für ankommende Verbindungen. Die Aushandlung findet dabei immer zwischen einem Broker-Buyer auf der einen Seite und einem Broker-Seller auf der anderen Seite statt. Es werden daher zwei Betriebsarten unterschieden: die *sender-initiierte Verhandlung* (sender-initiated brokerage) und die *empfänger-initiierte Verhandlung* (receiver-initiated brokerage). Im ersten Fall startet der Broker-Buyer beim Sender den Verhandlungsprozeß und erhält beim Empfänger vom Broker-Seller Auskunft über die dort zur Verfügung stehenden Ressourcen. Im zweiten Fall startet der Broker-Buyer beim Empfänger den Verhandlungsprozeß. Broker-Buyer und Broker-Seller bestehen aus je einem anwendungsspezifischen Teil und einem transportspezifischen Teil. Jeder Teil ist für die Aushandlung der Ressourcen in der entsprechenden Schicht verantwortlich. Beide Teile kooperieren mittels einer Schicht-zu-Schicht Kommunikation.

Das Aushandlungsprotokoll (broker protocol) umfaßt ein Broker-Buyer-Protokoll und ein Broker-Seller-Protokoll. Der Broker-Buyer veranlaßt als Initiator der Verhandlung eine Abbildung der anwendungsspezifischen Dienstqualität auf Ressourcenanforderungen für das lokale Betriebssystem. Der Broker-Buyer verhandelt mit dem Betriebssystem durch Zugangskontrolltests auf Anwendungsebene. Beispielsweise wird die Verarbeitungszeit und der Speicherbedarf der Anwendung auf Erfüllbarkeit geprüft. Nachdem die lokalen Ressourcen für den anwendungsorientierten Bedarf reserviert wurden, findet eine Ende-zu-Ende Aushandlung auf Anwendungsebene mit dem Broker-Seller im entfernten Endsystem statt. Die Trennung der Aushandlung zwischen anwendungsorientierter und netzwerkorientierter Dienstgüte erfolgt aus zwei Gründen: erstens ist das Reservieren von Netzwerkressourcen teuer und zweitens werden die Netzwerkressourcen geteilt. Somit sollte eine Reservierung von Netzwerkressourcen so spät wie möglich erfolgen. Die Anforderungen werden (eventuell zusammen mit zusätzlicher anwendungsspezifischer Information) an den Broker-Seller übermittelt, der seinerseits die lokalen Betriebssystem-Ressourcen überprüft und reserviert. Die Antwort an den Broker-Buyer kann eine von diesen dreien sein:

- *accept* – Die gewünschten Ressourcen wurden durch den Broker-Seller reserviert.
- *reject* – Die gewünschten Ressourcen konnten nicht reserviert werden.
- *modify* – Die Anforderungen an die Ressourcen wurden vom Broker-Seller innerhalb der vorgegebenen Grenzen verändert.

Lautet die Antwort *accept* oder *modify* führt der Broker-Buyer im transportsubsystemspezifischen Teil mehrere Schritte zur Aushandlung der Dienstgüte auf Netzwerkebene aus. Der Broker-Buyer im Transportsubsystem führt als ersten Schritt eine Abbildung von anwendungsorientierten auf netzwerkorientierte Dienstqualitätsparameter durch. Die Abbildung wird durch einen QoS-Übersetzer (QoS-Translator) durchgeführt und ist bidirektional. Jedem Medium wird eine Transportverbindung zugeordnet. Danach wird in einem zweiten Schritt ein Zugangskontrolltest für die vom Transportsubsystem benötigten Ressourcen des Betriebssystems durchgeführt. Ist der Test erfolgreich, wird in einem dritten Schritt die netzwerkorientierte Dienstqualität pro Netzwerk-Verbindung ausgehandelt. Dazu ist einen erneute Abbildung der transportorientierten Dienstgüte auf die Dienstgüte des darunterliegenden benutzten Netzwerks (z.B. transportorientierte Dateneinheiten auf ATM-Zellen) und eine Anfrage an das Netzwerk-Ressourcen-Management nötig. Der Broker-Buyer wartet auf eine Antwort des Netzwerk-Ressourcen-Managements und des Broker-Sellers auf der Ebene des Transportsubsystems. Diese Antworten werden zurück in anwendungsorientierte Dienstqualitätsparameter abgebildet. Das Protokoll des Broker-Sellers ist dem des Broker-Buyers ähnlich: Der Broker-Seller wartet auf eine Anfrage des Broker-Buyers, nach deren Empfang die anwendungsorientierten Dienstqualitätsparameter verglichen werden und ein Zugangskontrolltest durchgeführt wird. Das Ergebnis des Tests wird dem Broker-Buyer mitgeteilt. Danach wartet der Broker-Seller auf ein Signal des Netzwerk-Ressourcen-Managements, welches – wie oben beschrieben – vom Broker-Buyer mit der Aushandlung beauftragt wurde. Das Signal initiiert den Zugangskontrolltest im transportsubsystemspezifischen

Teil des Broker-Sellers, dessen Ausgang dem Netzwerk-Ressourcen-Management und dem anwendungsorientierten Teil des Broker-Sellers mitgeteilt wird, damit die zuviel reservierten Ressourcen wieder freigegeben werden können.

Eine prototypische Implementierung des QoS-Brokers wurde für eine Telerobotik-Anwendung eingesetzt. Dafür waren verschiedene Arten von Datenströmen zu übertragen: Video und Audio, sowie sensorische und taktile Daten. Es wurde eine ATM-Verbindung mit 155 MBit/s und als Transportsubsystem das Real-Time Network Protocol (RNTP) der Tenet Gruppe von der Universität Berkeley verwendet.

2.2.2 Das MASI-Rahmenwerk

An der Universität Pierre et Marie Curie in Paris entstand ein Rahmenwerk zum Management von qualitätsorientierten Kommunikationsdiensten, welches mit besonderem Augenmerk auf die Anforderungen multimedialer Anwendungen entwickelt wurde [TaFH94, Tawb93]. Das Rahmenwerk basiert auf einer zentralen anwendungsorientierten Managementkomponente, die den Namen AQOSM (Applications QoS Manager) trägt. In diesem Zusammenhang wurde auch ein Aushandlungsprotokoll HLQNP (High Level QoS Negotiation Protocol) für Dienstqualitäten entworfen. Es werden folgende Aspekte als wichtig angesehen, um eine Verbindung zwischen den Anforderungen der (multimedialen) Anwendungen und den Fähigkeiten der Ressourcen zu schaffen:

- Dienstqualitätsbasierte Formulierung der Anwendungsanforderungen.
- Definition eines Interaktionsmodells für die verschiedenen Ressourcen, die in eine multimediale Sitzung involviert sind.
- Unterstützung für veränderliche Dienstqualitäten während der Ausführung der Anwendung.

Die dienstqualitätsbasierte Formulierung der Anwendungsanforderungen gründet sich auf die Beschreibung einer Menge von Datenströmen (streams), die in verschiedenen Beziehungen zueinander stehen können. Aufgrund ihrer unterschiedlichen Charakteristik werden die Datenströme über separate Verbindungen übertragen. Ein Datenstrom wird durch seinen *Medientyp*, die *Übertragungsart*, *Beziehungen* zu anderen Datenströmen und *dynamische QoS-Änderungen* beschrieben.

Die Angabe des *Medientyps* kann durch optionale Parameter für diesen Medientyp präzisiert werden, andernfalls gelten Standardwerte. Beispielsweise wird ein Medientyp, der ein Video nach dem ITU-Standard H.261 mit einer Bandbreite von 64 kbit/s im Quarter Common Intermediate Format (QCIF) beschreibt, so angegeben: Video.H261[p=1, Frames/s=12, DimensionRatio=3/4, Lines=144, Pixels/line=192]. Die Werte in den eckigen Klammern sind dabei optionale Parameter. Die Abbildung der

Dienstgüte für die verschiedenen Medientypen auf datenstromorientierte Parameter wird durch die Benutzung von Tabellen realisiert.

Die *Übertragungsart* wird durch einen der drei Werte *asynchron* (Asynchronous), *Echtzeit* (RealTime) oder *Echzeit-Interaktiv* (RealTimeInteractive) angegeben. Die asynchrone Übertragungsart stellt keine Zeitanforderungen, die Übertragungsart „Echtzeit" setzt eine Grenze für die maximal erlaubte Ende-zu-Ende Verzögerung, und die Übertragungsart „Echzeit-Interaktiv" stellt als Spezialfall der letzteren Art eine noch härtere Anforderung an die maximal erlaubte Ende-zu-Ende Verzögerung dar. Auch für die Übertragungsart kann ein optionaler Parameter angegeben werden, der den Grad der Anforderung präzisiert. Es kann zwischen *lose* (loose), *mittel* (medium) oder *eng* (tight) gewählt werden. So werden die Angaben RealTime[loose], RealTime[medium] und RealTime[tight] für den Medientyp Video.H261 in 1000ms, 500ms und 250ms als obere Grenze für die Ende-zu-Ende Verzögerung anhand einer Tabelle umgesetzt. Die Angabe RealTimeInteractive ist gleichbedeutend mit einer oberen Grenze von 100ms.

Es werden zwei Arten von *Beziehungen* zwischen Datenströmen unterschieden: *zeitliche* und *logische* Beziehungen. Zeitliche Beziehungen beschreiben die Synchronisation in bezug auf andere Datenströme. Der Grad der Synchronisation kann durch einen optionalen Parameter angegeben werden *(optimal, good, acceptable, not tolerable)*. Logische Beziehungen beschreiben die Abhängigkeiten zwischen Datenströmen, beispielsweise, daß die Existenz eines bestimmten Datenstroms ohne die Existenz eines anderen keinen Sinn macht.

Die *dynamischen QoS-Änderungen* können durch den Benutzer oder Probleme im Kommunikationssystem aufgrund fehlender Ressourcen verursacht werden. Zu solchen Änderungen werden die Verbesserung und die Verschlechterung der Dienstqualität, sowie auch das Hinzufügen von Datenströmen gezählt. Der Benutzer kann beispielsweise durch Angabe einer Reihenfolge von Medientypen vorgeben, wie die Dienstqualität im Falle von Problemen herabgesetzt werden soll.

Das in diesem Rahmenwerk verwendete Management-Modell ist in Abbildung 2.7 dargestellt. Die zentrale Komponente des Modells ist der *AQOSM – Applications QoS Manager*, der in jedem Endsystem existiert. Die Umgebung des AQOSM besteht aus der Anwendung, den lokalen Systemressourcen, die durch mehrere System-Manager verwaltet werden, und einem Zugang zum Kommunikationssubsystem. Die Interaktion mit der Umgebung findet über Anzeigen (Indications) und Benachrichtigungen (Notifications) statt. Der Dialog zwischen verschiedenen AQOSM wird durch das *High Level QoS Negotiation Protocol* (HLQNP) ermöglicht.

Das *Interaktionsmodell* beschreibt, welche Möglichkeiten zur Interaktion zwischen dem AQOSM und den verschiedenen Komponenten bestehen. Der AQOSM kann einen bestimmten Dienst von einer Ressource anfordern, beispielsweise eine bestimmte Menge Speicher von einem Speicherverwalter. Die Verwaltung dieser Ressourcen liegt jedoch nicht beim AQOSM, sondern bei den dafür zuständigen System-Managern. Weiterhin existieren Aufrufe zum Ändern einer Dienstqualität (bezogen

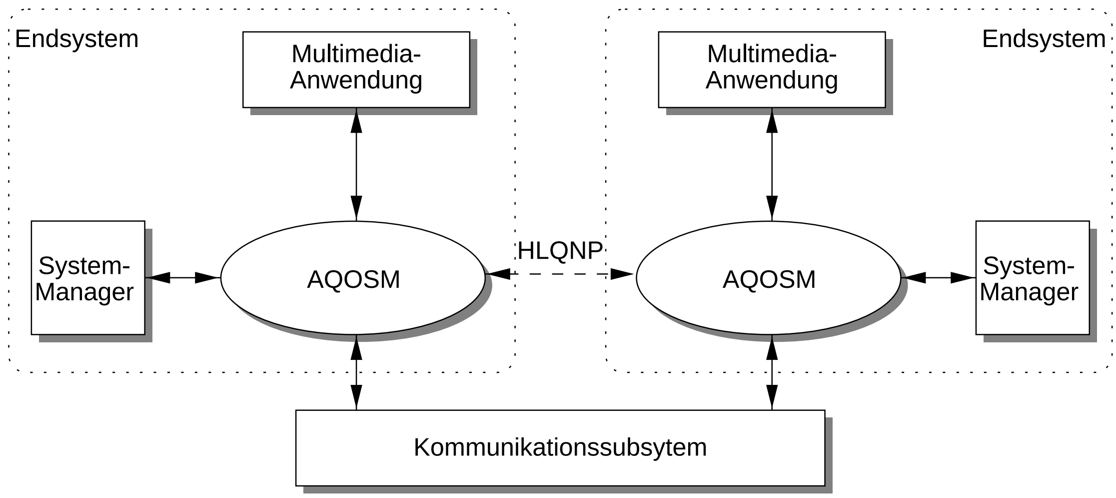

Abbildung 2.7: Das Management-Modell im MASI Rahmenwerk

auf die Ressource), zum Prüfen der Verfügbarkeit von Ressourcen und zum Beenden eines Dienstes, d.h. zur Freigabe der damit verbundenen Ressourcen. Die System-Manager können auf eine Anfrage des AQOSM mit einer Akzeptanz (*accept*), Herabsetzung (*modify*) oder Ablehnung (*refuse*) der geforderten Dienstqualität reagieren. Die beschriebenen Interaktionen dienen zur „lokalen Aushandlung“ der Dienstqualität (local QoS negotiation). Zusätzlich können die Ressourcen unaufgefordert (asynchron) anzeigen, ob die Dienstqualität eines Datenstroms herabgesetzt werden sollte oder der Datenstrom aufgrund von Problemen sogar beendet werden muß. Ebenso kann der AQOSM der Anwendung Probleme oder Änderungen der Dienstqualität anzeigen.

Das zur Kommunikation zwischen AQOSM-Instanzen verwendete HLQNP dient zur initialen Aushandlung und Neuverhandlung der Dienstqualität auf Anwendungsebene. HLQNP verwendet einen zuverlässigen „out-of-band“-Signalisierungskanal und sieht keine Fehlerbehandlung für seine Nachrichten vor. Es unterstützt nur die Aushandlung von einzelnen Simplex-Datenströmen, wobei immer der Sender eines Datenstroms die Aushandlung initiiert. Für mehrere Datenströme müssen separate Aushandlungen durchgeführt werden. Während einer Aushandlung dürfen Dienstqualitäten nur herabgesetzt, also nicht erhöht werden. Es existieren zwei Aushandlungmodi: der *negotiated mode* bezeichnet eine bilaterale Aushandlung, bei welcher der Initiator auf die Antwort des Responders wartet. Der *forced mode* hingegen bezeichnet im eigentlichen Sinne keine Aushandlung mehr, da der Responder die übermittelte Dienstqualität nur akzeptieren oder ablehnen, nicht aber herabsetzen kann. Dies ist nur sinnvoll, wenn es als sicher gelten kann, daß der Empfänger genügend Ressourcen besitzt, um den Datenstrom zu empfangen. Es existieren fünf Nachrichten unter HLQNP: NEGOTIATE, ACCEPT, MODIFY, REJECT und END. Die NEGOTIATE-Nachricht enthält die QoS-Anforderungen eines Datenstroms und wird durch den Sender initiiert. Die ACCEPT-Nachricht ist die Antwort des Empfängers auf eine NEGOTIATE-Nachricht und bezeichnet die Annahme der Dienstgüte ohne Änderungen. Die MODIFY-Nachricht erlaubt einem Empfänger, dem Sender des Datenstroms eine Veränderung der Dienstgüte mitzuteilen. Dies gilt auch während der Phase des Datenaustauschs (Neuverhandlung). Die REJECT-Nachricht zeigt die Ablehnung, oder im Falle von Problemen eine irreguläre Beendigung des Datenstroms an. Die END-Nachricht zeigt den regulären Abbau eines Datenstroms an, bei dem die entsprechenden Ressourcen freigegeben werden.

Eine Reservierung der Netzwerkressourcen findet erst nach der Aushandlung der Dienstqualität über HLQNP statt und wird indirekt bei einer Anforderung der bereits vereinbarten Ressourcen des Kommunikationssubsystems ausgelöst. Dies wird damit begründet, daß die Reservierung von Netzwerkressourcen „teuer" ist und eine unnötige oder unangemessen hohe Reservierung vermieden werden soll. Daher wird zuerst die Dienstqualität zwischen den Anwendungen ausgehandelt, denn beispielsweise kann das Fehlen einer unbedingt notwendigen Hardwarekomponente, wie einer Videodekoderkarte, ein Scheitern der Aushandlung bedeuten. Wären zuvor schon Netzwerkressourcen reserviert worden, stellte sich dies nun als überflüssig heraus.

2.2.3 Die Lancaster QoS-Architektur

Der Unterstützung verteilter multimedialer Anwendungen galt das besondere Interesse beim Entwurf der „Quality of Service Architecture" (QoS-A), die an der Universität Lancaster entwickelt wurde [CCGH93, CaCH94]. Ein weiteres Ziel der QoS-Architektur ist die *systemweite Garantie von Dienstqualitäten*, die Endsysteme, Kommunikationssysteme und Netzwerke umfaßt.

Der Begriff *Flow* wird in [CaCH94] als unidirektionale Punkt-zu-Punkt oder Punkt-zu-Mehrpunkt Verbindung mit einer zugeordneten Dienstqualität definiert. Basierend auf dem Basisreferenzmodell für Breitband-ISDN [ITU I.321] wurde ein Architekturmodell entworfen, das neben der traditionellen horizontalen Schichtenstruktur eine weitere vertikale Einteilung in Ebenen aufweist (s. Abbildung 2.8). Die oberste (horizontale) Schicht ist die Plattform für verteilte Anwendungen, die Dienste zur Multimedia-Kommunikation und Konfiguration der Dienstqualität in einer objektorientierten Umgebung bereitstellt. Die darunterliegende „Orchestration"-Schicht stellt Funktionen zur Synchronisation mehrerer multimedialer Datenströme und zur Jitterkorrektur einzelner Datenströme zur Verfügung. Die Transportschicht umfaßt einen erweiterten multimedialen Transportdienst (Multimedia Enhanced Transport Service – METS), der die Übertragung einzelner multimedialer Datenströme ermöglicht. Die unteren Schichten (Vermittlungsschicht, Sicherungsschicht, Physikalische Schicht) sorgen durch geeignetes Ressourcenmanagement für die notwendige Unterstützung der Dienstqualität im Netzwerk und sind in der Implementierung der QoS-Architektur speziell auf ATM-Netzwerke zugeschnitten. Es existieren drei vertikale Ebenen: die Protokoll-Ebene, die Qualitätsunterstützungs-Ebene und eine Flow-Management-Ebene.

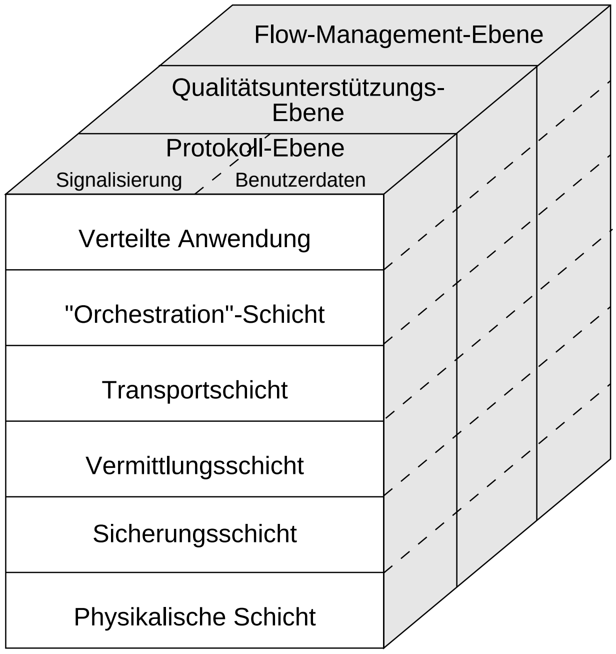

Abbildung 2.8: Die Lancaster QoS-Architektur

- Protokoll-Ebene
  Die Datenübertragung ist die hauptsächliche Aufgabe dieser Ebene. Die Protokoll-Ebene ist wiederum in einen Teil für die Übertragung von Benutzerdaten und einen Teil für die Übertragung von Signalisierungs- und Steuerungsdaten untergliedert. Diese Teilung wird durch die stark unterschiedlichen Anforderungen an die Dienstqualität für Benutzer- und Steuerungsdaten begründet. Zur Übertragung von Steuerungsdaten wird normalerweise ein zuverlässiger voll-duplex Kanal benötigt, der eine geringe Verzögerung aufweist. Die Übertragung multimedialer Benutzerdaten hingegen erfordert oft Dienste, die einen Simplex-Transfer mit hohem Durchsatz und ohne strikte Zuverlässigkeit realisieren.

- Qualitätsunterstützungs-Ebene
  Diese Ebene (QoS maintenance plane) besteht aus schichtenspezifischen QoS-Managern, die für die Überwachung und die Einhaltung der Dienstqualität in den jeweiligen Schichten zuständig sind. Die zugehörigen Protokollinstanzen werden überwacht und die gemessenen Werte mit den Vorgaben aus einem sogenannten *Dienstvertrag* verglichen. Der QoS-Manager versucht bei Abweichungen durch Feinanpassungen der Ressourcen die Erhaltung der Vorgaben zu erreichen (beispielsweise durch Veränderung der Pufferanzahl oder Parametereinstellungen der Scheduling-Mechanismen).

- Flow-Management-Ebene
  In dieser Ebene sind schichtenübergreifende Funktionen für die Etablierung von Flows (einschließlich Zugangskontrolle, Ressourcenreservierung und einem

dienstqualitäts-basierten Routing), Neuverhandlung der Dienstqualität, Abbildung und Adaption der Dienstqualität lokalisiert.

Die Spezifikation der Dienstqualität eines Flows erfolgt über die erweiterte multimediale Transportdienstschnittstelle. Nach einer Aushandlung der Dienstqualität zwischen den Dienstnutzern und dem Diensterbringer ist die zu erbringende Dienstqualität in einem sogenannten *Dienstvertrag* festgelegt. Dieser Dienstvertrag besteht aus folgende Komponenten:

- *flow specification*
  Die Spezifikation eines Flows enthält die Beschreibung des Medientyps, des benötigten Durchsatzes, der Ende-zu-Ende Verzögerung, Verzögerungsschwankung, und tolerierbarer Verlustquote.

- *commitment*
  Diese Angabe legt eine separate *Dienstklasse* für die Parameter Durchsatz, Verlust, Verzögerung und Verzögerungsschwankung fest. Es existieren drei Dienstklassen:

  - *deterministic*
    Die Werte eines Parameters dieser Dienstklasse dürfen von der vereinbarten Vorgabe nicht abweichen.
  - *statistical*
    Für Parameter dieser Dienstklasse kann ein angegebener Prozentsatz an Verletzungen der vereinbarten Dienstqualität toleriert werden. Die vereinbarten Werte stellen daher nur Richtwerte dar, die das System zu erreichen versucht.
  - *best effort*
    Es werden keinerlei Garantien für Parameter dieser Dienstklasse vereinbart.

- *adaptation*
  Hier kann für verschiedene Ereignisse angegeben werden, welche Maßnahmen vom Management ergriffen werden sollen. Für jedes Ereignis können bis zu zwei Maßnahmen angegeben werden. Die Verletzung der vereinbarten Dienstqualität eines bestimmten Parameters (Durchsatz, Verlust, Verzögerung und Verzögerungsschwankung) oder der spontane Verbindungsabbruch sind mögliche Ereignisse. Als Maßnahme kann die Anzeige der Verletzung, die Neuverhandlung (mit Angabe einer neuen Spezifikation) oder der Abbau des Datenstroms angegeben werden.

- *maintenance*
  Über diese Angabe wird dem Management mitgeteilt, ob die Dienstqualität überwacht und dem Dienstnutzer periodisch mitgeteilt werden soll, oder ob versucht werden soll, Feinanpassungen vorzunehmen, die z.B. lokal Anpassungen für das Scheduling oder Änderungen beim Puffern von Daten bewirken.

- *connection*
  Es stehen drei verschiedene Verbindungsaufbau-Varianten zur Auswahl, welche die Ressourcenreservierung beeinflussen: *schnell, im Voraus, ausgehandelt.* Bei der „schnellen" Variante fallen die Phasen der Ressourcenreservierung und des Datentransfers zusammen. Bei der Variante „im Voraus" wird eine Zeit angegeben, wann die entsprechenden Ressourcen benötigt werden, so daß die Reservierung entsprechend vorgemerkt werden kann. Die letzte Variante bezeichnet die vollständige Ende-zu-Ende Aushandlung der Dienstqualität.

- *cost*
  Die Höhe der Kosten, die der Benutzer zu zahlen bereit ist.

Das verwendete multimediale Transportprotokoll stellt einen verbindungsorientierten, unzuverlässigen aber reihenfolgegetreuen Kommunikationsdienst zur Verfügung. Der zugehörige QoS-Manager in der Qualitätsunterstützungs-Ebene ist eng an das Protokoll gekoppelt, um rechtzeitig Anpassungen an Schwankungen in der Leistung des Protokolls vornehmen zu können. Von der Flow-Management-Ebene werden folgende Hauptaufgaben übernommen: Netzwerk-Signalisierung, Ressourcen-Reservierung, Dienstqualitätsanpassung durch Neuverhandlung und Abbildung der Dienstqualitätsspezifikationen zwischen den Schichten.

Eine ATM-basierte Implementierung der QoS-Architektur verwendet einen erweiterten Chorus-Micro-Kernel zu Unterstützung der Dienstqualitäten im Endsystem. Die Flow-Management-Ebene besteht aus einem Verbindungsmanager und einem Flow-Management-Protokoll (FMP). Letzteres stellt Funktionen zum Management von Netzwerk-Ressourcen in lokalen und Weitverkehrsnetzen zur Verfügung, wobei ein vom Benutzerdatentransfer getrennter Signalisierungskanal verwendet wird. Es werden durch das FMP sowohl die Ressourcen im Endsystem (Prozessorzeit, Speicher) als auch die Ressourcen im Netzwerk reserviert. FMP setzt sich aus einer Teilmenge von RSVP [ZDES93] und einem speziellen ATM-Signalisierungsprotokoll *ATMSig* zusammen.

2.2.4 Diskussion

Der Schwerpunkt der OMEGA-Architektur liegt auf der Aushandlung von Dienstqualitäten. Als positiven, neuen Aspekt kann man die Trennung des Initiierens einer Verbindung (und damit der Aushandlung) von der Senderichtung ansehen. Da zuerst die Dienstqualität für die Anwendungen ausgehandelt wird, kann ein Scheitern der gesamten Aushandlung durch das Fehlen lokaler Ressourcen für das Transportsubsystem erst im zweiten Schritt festgestellt werden. In einem solchen Fall wurde die Aushandlung auf Anwendungsebene umsonst durchgeführt. Die Wichtigkeit einer bidirektionalen QoS-Abbildung wird zwar hervorgehoben, jedoch wird keine Rückabbildung der Dienstgüte beschrieben. Unberücksichtigt bleibt auch die Neuverhandlung, der Dienstabbau oder der Austausch von Monitoring-Information zwischen QoS-Brokern. Die Aushandlung muß für jeden Datenstrom separat durchgeführt werden, die Aushandlung mehrerer Datenströme für eine Sitzung wird daher aufwendig.

Das MASI-Rahmenwerk stellt eine anwendungsorientierte Schnittstelle für multimediale Anwendungen zur Verfügung. Die Abbildung der Dienstgüte erfolgt allerdings anhand von statischen Tabellen, was für viele Fälle zu inflexibel sein dürfte. Eine Rückabbildung der Dienstqualität wird nicht erwähnt. Die Ressourcen im Endsystem werden nicht nach ihrer Verwendung durch die verschiedenen Komponenten klassifiziert und so bleiben eventuelle Abhängigkeiten zwischen diesen unberücksichtigt. Der detaillierte Ablauf der lokalen Aushandlung fehlt. Da HLQNP ein reines Aushandlungsprotokoll ist, sind beispielsweise Nachrichten zum Austausch von Warnungsmeldungen oder Monitoringinformation zwischen zwei AQOSM-Instanzen nicht vorgesehen.

Die Lancaster QoS-Architektur ist durch ihre vielen Schichten und Ebenen sehr komplex. Die Aushandlung der Dienstqualität geschieht nur auf Netzwerkebene im Zuge einer Ressourcenreservierung. Eine detaillierte Beschreibung des Ablaufs einer Aushandlung wird nicht gegeben. Auf Rückabbildungen der Dienstgüte wird nicht eingegangen, obwohl sie wesentlicher Bestandteil der Dienstgüte-Aushandlung sind. Eine Aushandlung der Dienstqualität auf Anwendungsebene erfolgt nur indirekt. Obwohl die Unterstützung der Dienste durch das Betriebssystem einen zentralen Aspekt bei der Implementierung darstellt, findet sich diese Komponente nicht im Architektur-Modell wieder.

2.3 Aufgaben der Managementinstanz

In diesem Abschnitt werden die Aufgaben der zu entwerfenden *Managementinstanz* (QoS-Manager, siehe Abschnitt 2.1.5, Seite 13) erläutert.

Das diensteintegrierende Kommunikationssystem muß neben einer qualitätsbasierten, anwendungsorientierten Dienstschnittstelle auch Mechanismen zur Aushandlung, Unterhaltung, Änderung (d.h. Neuverhandlung) und Anzeige der Dienstqualität bereitstellen. Zur Diensterbringung ist die Funktionalität der in 2.1.5 erwähnten Endsystemkomponenten *Betriebssystem*, *Kommunikationssubsystem* und *Netzwerk* entsprechend den Anwendungsanforderungen auszuwählen und zu konfigurieren. Konkrete Parameterwerte müssen mit diesen Komponenten vereinbart werden (lokale Aushandlung der Dienstqualität). Zusätzlich sind die Fähigkeiten der Kommunikationspartner und der Zwischensysteme in die Betrachtungen mit einzubeziehen (entfernte bzw. Ende-zu-Ende Aushandlung der Dienstqualität). Der QoS-Manager übernimmt diese Aufgaben und entlastet so die Anwendung bzw. die Anwendungsprogrammierer, die ansonsten die Aushandlung der Dienstqualität selbst durchzuführen hätten.

Es lassen sich also drei Hauptaufgaben der Managementinstanz identifizieren:

- Dienstaushandlung
  Aushandlung der Dienstqualität im Endsystem und zwischen den Teilnehmern.
- Koordination der Diensterbringung
  Initiale und Re-Konfiguration der lokalen Endsystemkomponenten, sowie mittelbar auch die der Partner und des zugrundeliegenden Netzwerks. Koordination der Kooperation der Systemkomponenten.
- Dienstabbildung
  Abbildung der Dienstspezifikationen zwischen Anwendungen und Systemkomponenten.

Doch auch während des Datenaustausches zwischen den Anwendungen muß die Managementinstanz auf Schwankungen in der Dienstqualität der verschiedenen Datenströme reagieren und unter Umständen eine *Neuverhandlung* einleiten. Es können aber ebenfalls Datenströme vom Benutzer hinzugefügt oder entfernt werden, oder im Falle einer Gruppenkommunikation auch Sitzungsteilnehmer. Somit müssen auch dynamische Aspekte durch die Managementinstanz berücksichtigt werden.

3. Entwurf

In diesem Kapitel wird der Entwurf der Managementinstanz beschreiben. Zuerst werden einige Ziele formuliert, die beim Entwurf berücksichtigt werden sollten. Anschließend wird die Architektur der Instanz entwickelt und deren Teile ausführlich in den darauffolgenden Kapiteln beschrieben.

3.1 Entwurfsziele

Beim Entwurf der Managementinstanz sollten folgende Aspekte berücksichtigt werden:

- Flexiblität
  Unabhängigkeit von spezielle Ausprägungen der Komponenten, d.h. Definitionen von Schnittstellen, die vom konkreten Betriebssystem, Kommunikationssubsystem oder Netzwerk abstrahieren. Ebenso wird die Unabhängigkeit von der Rolle des Kommunikationspartners (Sender oder Empfänger von Datenströmen) gefordert.

- Schnelligkeit
  Die Aushandlungsdauer sollte möglichst kurz sein, die Aushandlung muß zielstrebig und schnell zum Abschluß kommen. Dazu sollte insbesondere eine Minimierung des Datenaustausches zwischen den Endsystemen angestrebt werden und *während* einer Aushandlungsphase nur eine Herabsetzung der Dienstqualitäten möglich sein.

- Robustheit
  Es sollte unter Ausnutzung aller Freiräume versucht werden, eine Verbindung aufzubauen. Es sollte also besser eine Sitzung mit niedrigerer Qualität aufgebaut werden, als keine. Die Anwendungen sollten nach einer erfolgreichen Aushandlung die gleiche Sicht der vereinbarten Dienstqualitätsparameter haben. Häufige Schwankungen durch Neuverhandlungen aufgrund kurzzeitig verbesserter Dienstqualitäten sollten vermieden werden.

3.2 Architektur des QoS-Managers

Um die in 2.3 dargestellten Aufgaben zu erfüllen, erscheint eine Gliederung des QoS-Managers in folgende logische Instanzen zweckmäßig (s. Abbildung 3.1), die jeweils bestimmte Teilaufgaben übernehmen:

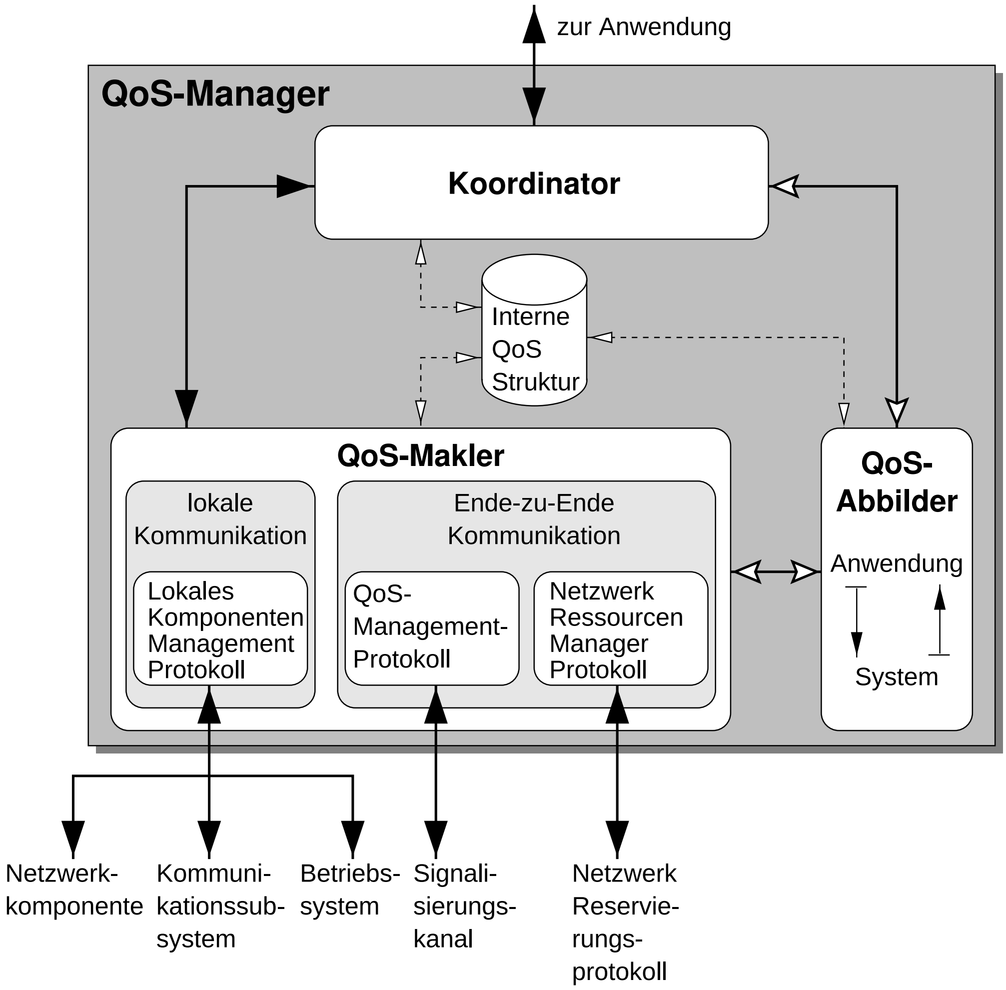

Abbildung 3.1: Architektur des QoS-Managers

- Koordinator
  Der Koordinator steuert den Ablauf im QoS-Manager. Er wickelt die Kommunikation mit der Anwendung ab und sorgt für die Koordination der Diensterbringung.

- QoS-Makler
  Der QoS-Makler verhandelt mit den lokalen und externen Komponenten über die Dienstqualitäten nach festgelegten Protokollen[1].

[1]ein Protokoll ist ein Satz von Regeln, nach denen die Kommunikation zwischen zwei oder mehr Beteiligten abgewickelt werden muß [LoKK93].

- QoS-Abbilder
  Der QoS-Abbilder ist für eine Abbildung der Dienstgüteparameter auf verschiedenen Ebenen verantwortlich, das heißt Dienstgüteparameter der Anwendung werden auf Dienstgüteparameter des Kommunikationssubsystems und des Netzwerks abgebildet und umgekehrt. Er ist keine aktive Instanz, sondern stellt eine Sammlung von Abbildungsfunktionen dar, die vom Koordinator oder QoS-Makler aufgerufen werden.

Diese Instanzen benutzen außerdem eine zentrale Datenstruktur, die Interne QoS-Struktur (IQS), in der Dienstqualitäts- und Managementparameter gehalten werden. In den folgenden Abschnitten werden die Aufgaben und Mechanismen dieser Instanzen im einzelnen dargestellt.

3.3 Koordinator

Der Koordinator nimmt Befehle der Anwendung entgegen, stößt interne Abläufe an und meldet Ergebnisse zurück. Er kann sich aber auch spontan (asynchron) an die Anwendung richten, um diese bei Bedarf über Änderungen oder besondere Vorkommnisse zu unterrichten. Die Anwendung muß daher in der Lage sein, ihren momentanen Ablauf zu verlassen und solche Meldungen sofort entgegenzunehmen. Da die Anwendung auf Sitzungsobjekten operiert, die mehrere Datenströme zusammenfassen, hat der Koordinator die Aufgabe Dienstprimitive, die als Parameter Teilmengen von Datenstromspezifikationen enthalten, entsprechend auf einzelne Datenströme anzuwenden. Der Koordinator kommuniziert also mit zwei Instanzen (s. Abbildung 3.1): mit der Anwendung über die Dienstschnittstelle und mit dem QoS-Makler über interne Signale und Nachrichtenaustausch.

3.3.1 Funktionale Konfiguration

Der Koordinator führt auch die *funktionale Konfiguration* der Komponenten durch. Vor einer *quantitativen Konfiguration* mit konkreten Dienstgüteparameterwerten muß zuerst die Funktionalität der Komponenten festgelegt werden, was durch die Abhängigkeiten zwischen Kommunikationssubsystem und Netzwerk motiviert wird. Das Kommunikationssubsystem erweitert die Funktionalität des Netzwerkdienstes, kann für viele Dienstqualitätsparameter aber keine Garantie ohne einen bereits garantierenden Netzwerkdienst erreichen (s. Tabelle 3.1). Beispielsweise kann das Kommunikationssubsystem eine durch das Netzwerk verursachte Verspätung einer Dateneinheit nicht mehr ausgleichen. Außer den garantierenden Mechanismen existieren sogenannte *assistierende* Mechanismen[Schm96], die eine Diensterbringung nur unterstützen, nicht aber garantieren können. Beispiel hierfür ist die ratenbasierte Flußkontrolle, die den Durchsatz assistierend unterstützt.

Komponente	leistungs-orientierte Parameter	Zuver-lässigkeit	Kommunika-tionsform
Betriebs-system	Scheduling Verwaltung der E/A-Geräte		
Kommunika-tionssub-system		Fehler-kontrolle	Gruppen-verwaltung
Netzwerk	Pufferverwaltung Scheduling	Pufferverwaltung Fehlerkontrolle	Adressierung

Tabelle 3.1: Garantierende Mechanismen

Dem Kommunikationssubsystem muß bekannt sein, auf welche Funktionalität des Netzwerks aufgebaut werden kann. Daher ist die grundlegende Funktionalität beider Komponenten festzulegen, *bevor* die Anfrage mit konkreten Werten der Dienstqualitätsparameter erfolgt (quantitative Konfiguration), denn erst wenn die grundlegenden Mechanismen ausgewählt sind, kann die tatsächliche Beeinflussung der Werte errechnet werden. So wird das Einschalten einer Vorwärtsfehlerkontrolle die Anforderung im Durchsatz stark erhöhen, was nun bei der Reservierung der Bandbreite im Netzwerk zu berücksichtigen ist. Eine funktionale Konfiguration kann auch im Zuge einer Neuverhandlung durchgeführt werden (*Rekonfiguration*) oder nach Scheitern einer *quantitativen Aushandlung* in einem erneuten Versuch.

Da anhand der Anforderungen zuerst die benötigte Netzwerkfunktionalität festgelegt werden muß, sollte eine Auswahl durch den QoS-Manager erfolgen. Die konkreten Mechanismen (z.B. Scheduling-Verfahren) werden später vom Kontrollagenten der Kontrollinstanz festgelegt. Bei der qualitativen Anfrage wird die gewünschte funktionale Konfiguration als Parameter mitgegeben. Es ist festzulegen, nach welchen Regeln das passende Netzwerk ausgesucht wird. Dazu wird folgende Tabelle 3.2 zugrundegelegt: Da die leistungsorientierten Parameter (siehe Abschnitt 2.1.2) auf jeden Fall eine Garantie im Netzwerk benötigen, sollen hier nur die Parameter aufgeführt werden, für die eine Aufgabentrennung zwischen Kommunikationssubsystem und Netzwerk möglich ist. Das sind die *Zuverlässigkeit* und die *Kommunikationsform*. Für die Zuverlässigkeit steht in der ersten Spalte die gewünschte Qualität jeweils anhand ihrer Dienstklasse beschrieben. In der zweiten Spalte sind mögliche Kombinationen der Unterstützung im Kommunikationssubsystem und im Netzwerk beschrieben. Dazu ist in der dritten Spalte unter „Auswirkung“ vermerkt, welche Parameter durch diese Kombination beeinflußt werden („leist.or.“ steht für leistungsorientierte Parameter). In den letzten beiden Spalten sind die zu erwartenden Kosten und die erwartete Last im Endsystem tendenziell aufgeführt. Bei der Initialisierung des QoS-Managers wird festgestellt, welche Kombinationen grundsätzlich vom lokalen Endsystem unterstützt werden.

geforderte Qualität		Unterstützung Netz	KSS	Auswirkung	Kosten	Last
Zuverlässigkeit	bestmöglich	bestmöglich	–	–	gering	–
	statistisch	bestmöglich	statistisch	leist.or.	gering	mittel
		statistisch	bestmöglich	–	mittel	–
		statistisch	statistisch	leist.or.	mittel	mittel
	deterministisch	bestmöglich	determin.	leist.or.	gering	hoch
		statistisch	determin.	leist.or.	mittel	hoch
		determin.	–	–	hoch	–
		determin.	determin.	leist.or.	hoch	hoch
Kommunikationsform	Multicast	Unicast	Multicast		mittel	hoch
		Multicast	Gruppenverw.		niedrig	mittel

Tabelle 3.2: Konfigurationsmöglichkeiten

Die Auswahl eines passenden Netzwerkdienstes erfolgt dann anhand einer weiteren Tabelle im sogenannten *Configuration Profile*. Dort wird die Beschreibung der verfügbaren Netzwerkdienste abgelegt. In der ersten Spalte befindet sich eine ein-

Netz	Netto-Durchsatz (Mbit/s)	MTU-Größe (bytes)	Bitfehler Rate	Auslastung (%)	Multicast Support	Garantie (b,s,d) R	D	J	Z	Kosten
A	134.588	65535	10^{-12}	30	ja	b	b	b	b	...
B	9.699	1480	10^{-6}	2	ja	b	b	b	b	...
C	...	...	...	...	...	.	.	.	.	...

Tabelle 3.3: Netzwerk-Eigenschaften

deutige Kennzeichnung des beschriebenen Netzwerkdienstes. In der zweiten Spalte ist der verfügbare Durchsatz für Nutzdaten angegeben, in der dritten Spalte die maximale Größe einer Nutzdateneinheit. Die Bitfehlerrate dient als ungefährer Anhaltspunkt für bitfehlersensitive Anwendungen, da das Übertragungsmedium auf dem Weg zum entfernten Endsystem vermutlich mehrfach wechselt. Die Angabe bezieht sich daher auf das direkt zugängliche lokale Netzwerk. Die Auslastung wird in regelmäßigen Abständen gemessen und aktualisiert. Ein von der Funktionalität passendes aber bereits ausgelastetes Netzwerk ist von keinem Nutzen und ein alternativer Netzwerkdienst sollte ausgewählt werden. Die Unterstützung für die Gruppenkommunikation wird in der Spalte „Multicast“ beschrieben, in der Spalte „Garantie“ findet sich die Unterstützung für die Rate (R), Verzögerung (D), Verzögerungsschwankung (J) und Zuverlässigkeit (Z) wieder (b=best effort, s=statistisch, d=deterministisch). In der letzten Spalte findet sich ein Hinweis auf die zu erwartenden Kosten. In der Tabelle 3.3 sind beispielhaft die Werte für ein 155Mbit/s ATM-basierten AAL-5 Dienst (A) und IP über ein 10MBit/s Ethernet (B) angegeben. Die Auswahl erfolgt zunächst anhand der benötigten Garantie, danach anhand des benötigten und des zur Verfügung stehenden Durchsatzes.

Die Entscheidungen des Dienstemanagements wirken sich im wesentlichen auf drei Größen aus: die *Qualität*, die *Kosten* und die *Last im Endsystem* (s. Abbildung 3.2). Dabei beschreibt die Qualität, die vom Benutzer empfundene Qualität und

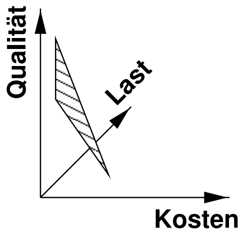

Abbildung 3.2: Vom Management beeinflußte Parameter

entspricht damit der Dienstqualität. Die Kosten entstehen durch den kostenpflichtigen Aufbau und die Nutzung der Verbindungen im Netzwerk. Ohne Kostenstruktur gäbe es für die Nutzer des Netzwerkdienstes keinen Grund, eine andere Qualität als die beste zu wählen. Die Last im Endsystem beschreibt dabei die Auslastung der Prozessoren und steigt dabei z.B. mit der Nutzung komplexer Funktionen im Kommunikationssubsystem. Im allgemeinen ist eine hohe Qualität bei niedrigen Kosten und geringer Last anzustreben.

3.3.2 Quantitative Konfiguration

Die quantitative Konfiguration besteht in der Anfrage an die Kontrollagenten mit konkreten Parameterwerten, die oft als ein gewünschter Wert innerhalb von Grenzen spezifiziert sind. Es gibt drei mögliche Antworten auf eine solche Anfrage:

- *accepted*
  Die vorgegebenen gewünschten Parameterwerte werden so akzeptiert. Sollten noch mehr Ressourcen frei sein, wird dies nicht angezeigt. Die entsprechenden Ressourcen werden vorreserviert.

- *modified*
  Die vorgegebenen Parameterwerte können innerhalb der vorgegebenen Grenzen, jedoch nicht mit dem gewünschten Wert erfüllt werden. Die tatsächlich erfüllbaren Parameterwerte werden dann zurückgegeben und die Ressourcen entsprechend vorreserviert.

- *rejected*
  Die vorgegebenen Parameterwerte können auch nicht innerhalb der angegebenen Grenzen erfüllt werden.

Zu einem späteren Zeitpunkt können die vorreservierten Ressourcen teilweise freigegeben werden („Relaxing"), falls sich im Verlauf der Aushandlung mit den anderen Komponenten abzeichnet, daß geringere Anforderungen genügen.

3.3.3 Interaktion mit der Anwendung

Die Kommunikation des QoS-Managers mit der Anwendung wird durch den Koordinator realisiert. Sitzungen (siehe Definition 2) umfassen einen oder mehrere Datenströme (siehe Definition 1), die in Beziehungen zueinander stehen können. Die Parameter der folgenden Dienstprimitive bestehen daher zum großen Teil aus ein oder mehreren anwendungsorientierten Datenstromspezifikationen, welche jedoch erst im Detail in Abschnitt 3.6 vorgestellt werden.

Die Gruppenkommunikation stellt zusätzliche Anforderungen an den QoS-Manager: es müssen grundsätzlich unterschiedliche Entscheidungen bzw. Antworten der Kommunikationspartner berücksichtigt werden (z.B. unterschiedliche Parameterwerte), die oft einen *Konsens* erfordern (meist Festlegung auf eine Variante), der dann den Kommunikationspartnern wieder mitgeteilt werden muß. Daher wird eine weitere Dienstprimitivart „commit" (Com) eingeführt, die den Beantwortern mitteilt, auf welche Variante sich nun festgelegt wurde. Die Konsensfindung wird beim jeweiligen Initiator durchgeführt.

Die folgenden Aktionen beziehen sich immer auf ein Sitzungsobjekt und tragen daher explizit ein Präfix, welches das Sitzungsobjekt angibt (z.B. Session.OpenReq(...)).

- Sitzungsaufbau
  Der Initiator einer Sitzung veranlaßt deren Aufbau durch das Dienstprimitiv OpenReq. Sämtliche potentielle Teilnehmer der Sitzung erhalten daraufhin die Anzeige OpenInd des Sitzungsaufbauwunsches. Der Wunsch kann von der Anwendung angenommen werden (OpenRsp+), wobei die Änderung der Charakteristik der Datenströme im vom Initiator vorgegebenen Rahmen möglich ist. Im OpenRsp+ werden sowohl akzeptierte als auch abgelehnte Datenströme aufgeführt, wobei letztere als nicht notwendig spezifiziert waren, denn andernfalls kann der Aufbauwunsch als ganzes nicht erfüllt werden. Die Ablehnung des Sitzungsaufbaus erfolgt durch OpenRsp- unter Angabe von Gründen (etwa unpassende oder nicht unterstützte Parameter, oder die Ablehnung notwendiger Datenströme). Der QoS-Manager des Initiators sammelt die Antworten und bildet aus den unterschiedlichen Antworten einen Konsens, dessen Ergebnis den Teilnehmern durch ein OpenCom+ (für OpenCommit) mitgeteilt wird. Potentielle Teilnehmer, die durch die Konsensbildung von der Sitzung ausgeschlossen wurden, erhalten hingegen ein OpenCom-. Der Initiator selbst bekommt die ausgehandelte Spezifikation im Erfolgsfalle durch ein OpenCnf+ angezeigt, bei Scheitern die Ursache durch ein OpenCnf-.

- Sitzungsverwaltung
  Nach Aufbau einer Sitzung können ihre Eigenschaften verändert werden. Es bestehen folgende Möglichkeiten:

  - Hinzufügen von Teilnehmern
    Neue Teilnehmer können durch das Primitiv ModifyJoinReq ihren Teilnahmewunsch äußern. Analog zum Sitzungsaufbau findet eine vollständige Aushandlung statt. Die Existenz eines Verzeichnisses für momentan aktive Sitzungen anderer Endsysteme wird vorausgesetzt. Das Verzeichnis dient zum Herausfinden der entsprechenden Gruppenadresse.
  - Entfernen von Teilnehmern
    Ein Teilnehmer kann jederzeit auf eigenen Wunsch aus der Sitzung ausscheiden. Dies wird durch das Dienstprimitiv ModifyRmEPReq (für RemoveEndPoint) veranlaßt und kann von anderen Teilnehmern nicht abgelehnt werden. Die erfolgreiche Ausführung wird dem Initiator des ModifyRmEPReq durch ein ModifyRmEPCnf angezeigt (es handelt sich hierbei um eine implizite Bestätigung).
  - Hinzufügen von Datenströmen
    Jeder Teilnehmer kann das Hinzufügen von Datenströmen durch ModifyAddReq veranlassen. Es findet eine vollständige Aushandlung für diese Datenströme statt, d.h. Teilnehmer können den Aufbau neuer Datenströme ablehnen.
  - Entfernen von Datenströmen
    Der Abbau eines Datenstroms (ModifyDelReq) kann nur durch den Sender desselben veranlaßt werden. Die Empfänger der Datenströme werden nur über den Abbau informiert, können diesen aber nicht verhindern. Auch dieser Dienst ist nur implizit bestätigt.
  - Modifikation von Datenströmen
    Die Parameterwerte eines Datenstroms können durch ModifyChgReq jederzeit neu ausgehandelt werden. Es findet eine vollständige Aushandlung wie bei der erstmaligen Etablierung eines Datenstroms statt.

  Meldungen können jederzeit vom Koordinator an die Anwendung durch ein NotifyInd weitergereicht werden. Es kann sich hierbei um periodische Statusmeldungen, Warnungen oder Alarme aufgrund von Qualitätsverletzungen handeln. Das Meldungsverhalten wird mittels Managementparametern für jeden Datenstrom einzeln festgelegt.

- Sitzungsabbau
  Nur der Initiator einer Sitzung kann diese auch durch ein CloseReq beenden. Sämtliche Teilnehmer bekommen den Sitzungsabbau durch ein CloseInd angezeigt und können diesen nicht ablehnen. Der Initiator wird nach erfolgtem Sitzungsabbau durch ein CloseCnf darüber informiert (implizite Bestätigung).

Der Initiator kann auch einen abrupten Sitzungsabbau, d.h. einen Sitzungsabbruch veranlassen (AbortReq). In diesem Fall werden alle anderen Teilnehmer der Sitzung benachrichtigt und brechen ebenfalls die Sitzung ab. Da es sich nicht um einen geordneten Sitzungsabbau handelt, ist es möglich, daß noch in der Übertragung befindliche Daten verlorengehen.

Der Schnittstellenautomat, der angibt, welche Dienstprimitive wann, wo und in welcher Reihenfolge an der Anwendungs-Schnittstelle erlaubt sind, ist in [Henr96] dargestellt.

3.3.4 Interner Ablauf

Da der QoS-Manager für alle Anwendungen im System zur Verfügung steht, muß er mehrere Sitzungen gleichzeitig verwalten können. Allerdings soll die Einschränkung gelten, daß eine Anwendung immer nur eine Sitzung unterhält, um das zusätzliche Multiplexen auf Anwendungsebene zu vermeiden, was eine zusätzliche Komplexität beeinhaltet, aber wohl selten von großem Nutzen sein wird (ein Nutzer der Anwendung wird meist nur an einer Sitzung gleichzeitig teilnehmen wollen). Somit muß für jede Sitzung ein *Kontext* im QoS-Manager vorhanden sein, der den jeweiligen Zustand der Sitzung festhält (s. Abbildung 3.3). Der Koordinator nimmt über die Dienstschnittstelle Dienstprimitive entgegen und wendet sie auf die jeweiligen Sitzungen und Datenströme an. Damit sich gleichzeitig mehrere Aktionen auf verschiedenen Datenströmen (Hinzufügen, Verändern, Löschen) während einer Sitzung in Ausführung befinden können, muß auch hier pro Datenstrom ein Kontext existieren, der den Zustand desselben beeinhaltet. Eine zweite Aktion während der Ausführung einer ersten auf demselben Datenstrom ist allerdings unzulässig und wird mit einer Fehlermeldung abgelehnt, es sei denn, es handelt sich um das Schließen der Sitzung, Löschen von Datenströmen oder Entfernen von Teilnehmern während einer Neuverhandlung (s. Abbildung 3.4). Die Aktionen dürfen per Definitionem nicht abgelehnt werden und haben deshalb Vorrang vor einer Neuverhandlung.

Der endliche Automat zur Beschreibung der möglichen Zustände einer Sitzung ist in Abbildung 3.3 zu sehen. Die Nachrichten StartNego, NegoWish, NegoSuccess und NegoFail zum und vom QoS-Makler sind typisiert, d.h. sie enthalten zusätzlich eine Angabe darüber, welchem Dienstprimitiv sie zugeordnet sind (Open, ModifyAdd, ModifyChange, ModifyJoin). Die zugehörigen Aktionen werden auf den einzelnen Datenströmen durchgeführt und verursachen für sie einen entsprechenden Zustandswechsel (s. Abbildung 3.4).

Es folgt eine Beschreibung der Sitzungszustände und der möglichen Übergänge.

- IDLE
  Die Sitzung befindet sich zunächst in diesem Ruhezustand. Auf einen lokalen Sitzungsaufbauwunsch (OpenReq) hin, wird zunächst die funktionale

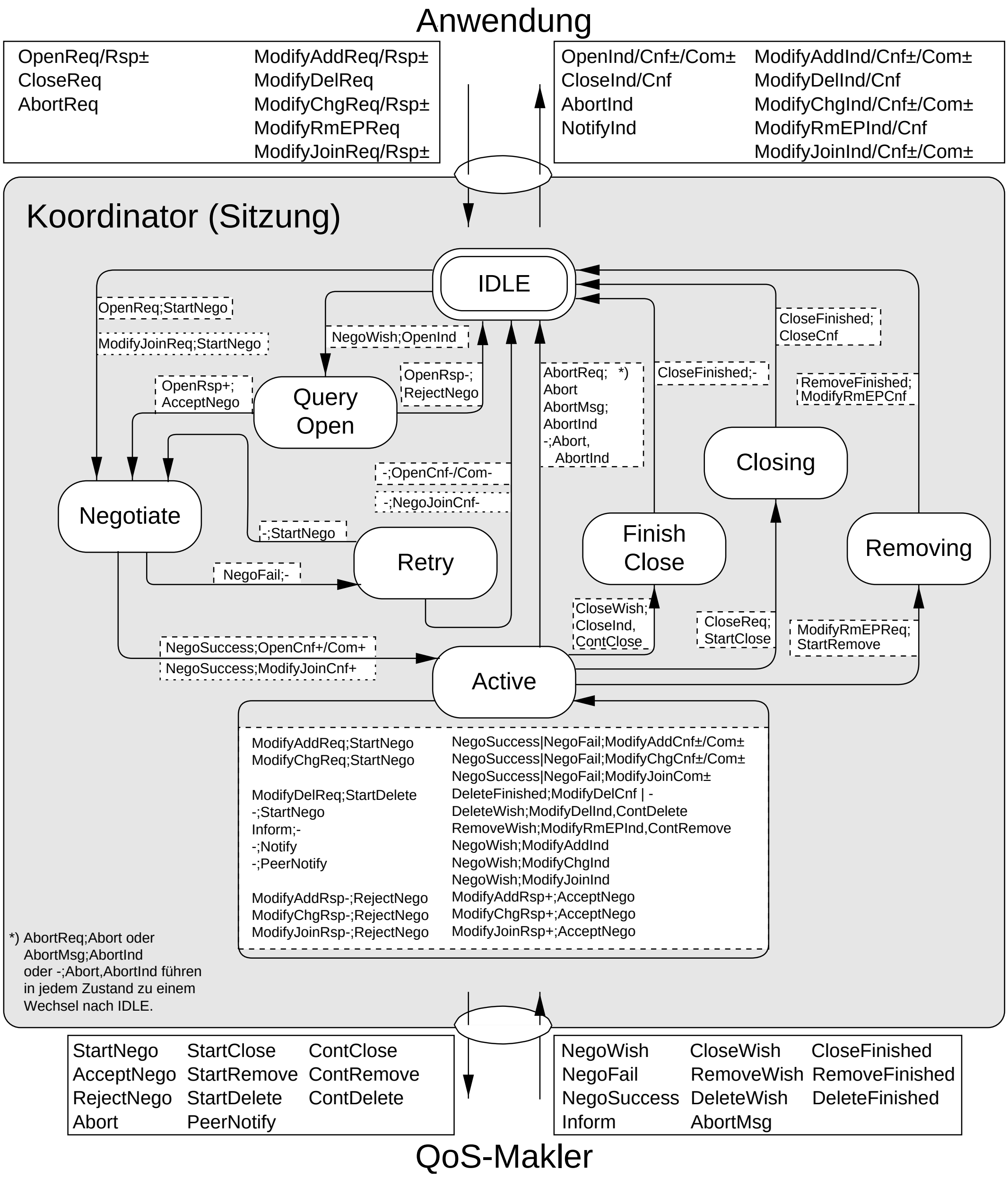

Abbildung 3.3: Die Zustände einer Sitzung

Konfiguration durchgeführt. Danach wechselt sie in den Zustand Negotiate und der QoS-Makler wird mit der Aushandlung der Dienstqualität beauftragt (StartNego). Ein Sitzungsaufbauwunsch von einem anderen Endsystem wird durch den QoS-Makler mittels einer NegoWish-Nachricht angezeigt, welche die Benachrichtigung der Anwendung (OpenInd) mit anwendungsorientierten Dienstgüteparameterwerten zur Folge hat. Nach anschließendem Übergang in den Zustand Query wird auf die Antwort der Anwendung gewartet. Die Teilnahme an einer bereits bestehenden Sitzung wird durch das Dienstprimitiv ModifyJoinReq ausgelöst, welches analog zu einem OpenReq behandelt wird.

- Negotiate
  In diesem Zustand findet die Aushandlung der Dienstqualität statt (sowohl lokal als auch Ende-zu-Ende). Ist die Aushandlung erfolgreich (QoS-Makler signalisiert NegoSuccess), werden die endgültig ausgehandelten anwendungsorientierten Parameter der Anwendung durch ein OpenCnf+ Primitiv mitgeteilt (Übergang in Zustand Active). Scheitert die Aushandlung (NegoFail), so kann zunächst eine erneute Aushandlung mit einer alternativen funktionalen Konfiguration versucht werden (Zustand Retry).

- Query Open
  In diesem Zustand wartet der Koordinator auf eine Antwort der Anwendung, ob sie gewillt ist, den Sitzungsaufbauwunsch anzunehmen. Im Falle einer positiven Antwort (OpenRsp+), die eventuell veränderte anwendungsorientierte Dienstgüteparameter enthält, wird die weitere Aushandlungsphase mit einer AcceptNego-Nachricht an den QoS-Makler eingeleitet, d.h. die Sitzung wechselt in den Zustand Negotiate. Lehnt die Anwendung jedoch den Sitzungsaufbauwunsch ab (OpenRsp– z.B. aufgrund von Sicherheitsaspekten, fehlender Unterstützung von Anwendungsdiensten oder unverträglichen Versionen der Anwendungsprogramme), kehrt die Sitzung in den Zustand IDLE zurück. Die Ablehnung wird unter Angabe von Gründen dem QoS-Makler durch eine RejectNego-Nachricht mitgeteilt, damit auch der Initiator darüber unterrichtet wird.

- Retry
  In diesem Zustand wird entschieden, ob ein erneuter Versuch mit alternativen funktionalen Konfigurationen durchgeführt wird (Übergang mit StartNego nach Negotiate). Scheitert die Aushandlung aufgrund fehlender Alternativen oder Überschreiten einer gewissen Anzahl an Versuchen jedoch endgültig, wird der Anwendung der Grund im OpenCnf– mitgeteilt.

- Active
  In diesem Zustand ist eine Sitzung aufgebaut. Während der Sitzung kann eine Veränderung der Sitzung stattfinden: Es können Datenströme hinzugefügt (ModifyAddReq), verändert (ModifyChgReq) oder gelöscht werden (ModifyDelReq). Die Zustände der betroffenen Datenströme ändern sich entsprechend (s. Abbildung 3.4) und der QoS-Makler wird mit der Ausführung der dazu nötigen Aktionen beauftragt (StartNego bzw. StartDelete); der Zustand der Sitzung ändert sich jedoch nicht. Reaktionen an der Dienstschnittstelle zeigen sich erst nach einer entsprechenden Reaktion des QoS-Maklers (NegoSuccess oder NegoFail bzw. DeleteFinished). Für die ersten beiden Fälle (ModifyAddReq, ModifyChgReq) muß eine vollständige erneute Aushandlung der betroffenen Datenströme durchgeführt werden, weshalb die zugehörigen Dienste explizit bestätigt sind. Das Löschen eines Datenstroms kann nur der Sender des selbigen veranlassen und darf von den Empfängern nicht abgelehnt werden. Der zugehörige Dienst ist allerdings implizit bestätigt, d.h. die Ausführung des ModifyDelReq wird durch ein ModifyDelCnf bestätigt.

Der Koordinator kann jederzeit Benachrichtigungen von den lokalen Komponenten oder den anderen Kommunikationspartnern erhalten (Inform). Dabei kann es sich um Zustandsmeldungen (Monitoringinformation), Warnungen oder Alarme aufgrund von Qualitätsverletzungen handeln. Das Weitergeben dieser Meldungen an die Anwendung (Notify) wird nach einer vorher festgelegten aber wählbaren (und jederzeit änderbaren) Strategie pro Strom durch den Koordinator veranlaßt. Der Koordinator kann auch Meldungen an die Kommunikationspartner verschicken (PeerNotify).

Hat der Koordinator von der Anwendung die Erlaubnis, eine Neuverhandlung für Datenströme selbständig zu initiieren und durchzuführen, so kann eine StartNego-Nachricht an den QoS-Makler auch ohne Stimulus an der Dienstschnittstelle erfolgen. Der Koordinator wird in einem solchen Fall auch manchmal eine funktionale Rekonfiguration vornehmen. Wird eine Neuverhandlung durch einen Partner gewünscht, so zeigt der QoS-Makler dies durch eine (typisierte) NegoWish Nachricht an. Der Koordinator benachrichtigt daraufhin die Anwendung mit einem ModifyAddInd bzw. einem ModifyChgInd oder einem ModifyJoinInd. Die Antworten der Anwendung für die bestätigten Dienste (ModifyAddRsp±, ModifyChgRsp±, ModifyJoinRsp±) werden an den QoS-Makler weitergereicht (AcceptNego, RejectNego). Der Wunsch anderer Teilnehmer, Datenströme abzubauen (DeleteWish) oder aus der Sitzung auszuscheiden (RemoveWish) wird der Anwendung angezeigt und kann von ihr nicht abgelehnt werden. So erhält der QoS-Makler gleich nach dem Absetzen der Anzeige (ModifyDelInd bzw. ModifyRemoveEPInd) den Auftrag, die entsprechenden Aktionen fortzuführen (ContDelete, bzw. ContRemove). Die Sitzung bleibt jedoch weiterhin bestehen, auch wenn soeben der letzte Datenstrom abgebaut oder der vorletzte Teilnehmer entfernt wurde.

Erhält der Koordinator einen Befehl zum Sitzungsabbau (CloseReq) und ist er selbst der Initiator der Sitzung, so veranlaßt er den Sitzungsabbau durch den QoS-Makler (StartClose) und wartet danach im Zustand Closing auf dessen Ausführung. Empfängt hingegen ein Teilnehmer den Sitzungsabbauwunsch des Initiators (CloseWish), so wird dies der Anwendung durch ein CloseInd Primitiv angezeigt (das Senden von Daten ist dann nicht mehr erlaubt) und der lokale Abbau der Sitzung veranlaßt (ContClose), auf dessen Beendigung im Zustand Finish Close gewartet wird. Soll die Teilnahme an der Sitzung beendet werden, wird dies durch ein ModifyRmEPReq dem Koordinator mitgeteilt, der daraufhin den QoS-Makler durch ein StartRemove anweist, die entsprechende Prozedur durchzuführen. Auf deren Fertigmeldung wird daher im Zustand Removing gewartet.

- Closing
  In diesem Zustand der Sitzung wartet der Koordinator auf die Fertigmeldung der Prozedur zum Schließen der Sitzung vom QoS-Makler (CloseFinished), welche dann mit einer Anzeige an die Anwendung (CloseInd) in den Zustand IDLE zurückkehrt.

- Finish Close
  Es wird auf die Beendigung der lokalen Sitzungsabbauprozedur gewartet. Der QoS-Makler zeigt dies durch ein CloseFinished an, wonach die Sitzung wieder in den Zustand IDLE zurückkehrt.

- Removing
  Nachdem der QoS-Makler die Durchführung der Prozedur zum Entfernen des Endsystems aus der Teilnehmermenge gemeldet hat (RemoveFinished), wird die Anwendung ebenfalls darüber informiert (ModifyRmEPCnf). Die Sitzung kehrt dann in den Zustand IDLE zurück.

In jedem Zustand der Sitzung kann ein Sitzungsabbruch durch den Initiator erfolgen (AbortReq), der dem QoS-Makler mitgeteilt wird (Abort). Die Sitzung kehrt unmittelbar in den Zustand IDLE zurück, der QoS-Makler benachrichtigt noch sämtliche Sitzungsteilnehmer. Der Koordinator bei einem entfernten Sitzungsteilnehmer erhält daraufhin ein AbortMsg und zeigt der Anwendung den Sitzungsabbruch durch ein AbortInd an. Die Sitzung befindet sich danach wieder im Zustand IDLE. Ein schwerwiegender interner Fehler kann ebenfalls zum lokalen Abbruch führen (-;Abort,AbortInd). Die übrigen Teilnehmer erfahren durch das AbortMsg vom Ausscheiden des Teilnehmers und lassen die Sitzung bestehen, wenn die AbortMsg nicht vom Initiator stammt. Sollte durch einen Ausfall der Netzwerk-Verbindung keine Kommunikation mehr möglich sein (es kann folglich kein AbortMsg verschickt werden), wird bei den übrigen Teilnehmern der Ausfall des Teilnehmers vom Kommunikationssubsystem oder Netzwerk erkannt und das Entfernen des Teilnehmers eingeleitet.

Nun sollen die Zustände eines Datenstroms erläutert werden: In Abbildung 3.4 sind nur die Stimuli aufgeführt, die den entsprechenden Zustandsübergang auslösen können. Dies können Anweisungen der Anwendung oder Reaktionen des QoS-Maklers sein. Mögliche Alternativen sind durch einen senkrechten Strich (|) voneinander getrennt. Wie bereits bei der Beschreibung der Interaktion mit der Anwendung erwähnt wurde (s. Abschnitt 3.3.3), besitzen die Dienstprimitive als Parameter eine Liste von Datenstromspezifikationen. Im folgenden wird unterstellt, daß der betrachtete Datenstrom als Parameter der dort aufgeführten Dienstprimitive vorhanden ist.

- IDLE
  Ein noch nicht aktiver Datenstrom befindet sich zunächst in diesem Grundzustand. Ist der betroffene Datenstrom in einem der Dienstprimitive OpenReq, ModifyJoinReq oder ModifyAddReq als Parameter aufgeführt, so wechselt sein Zustand in Negotiate, d.h. er befindet sich in der initialen Aushandlungsphase. Die drei Dienstprimitive können auch durch ein entferntes Endsystem ausgelöst worden sein, was durch eine entsprechende NegoWish-Nachricht des QoS-Maklers angezeigt wird. In diesem Fall wechselt der Datenstrom in den Zustand Query Open.

- Query Open
  Nach dem Erhalt eines Aushandlungswunsches (NegoWish) entscheidet die

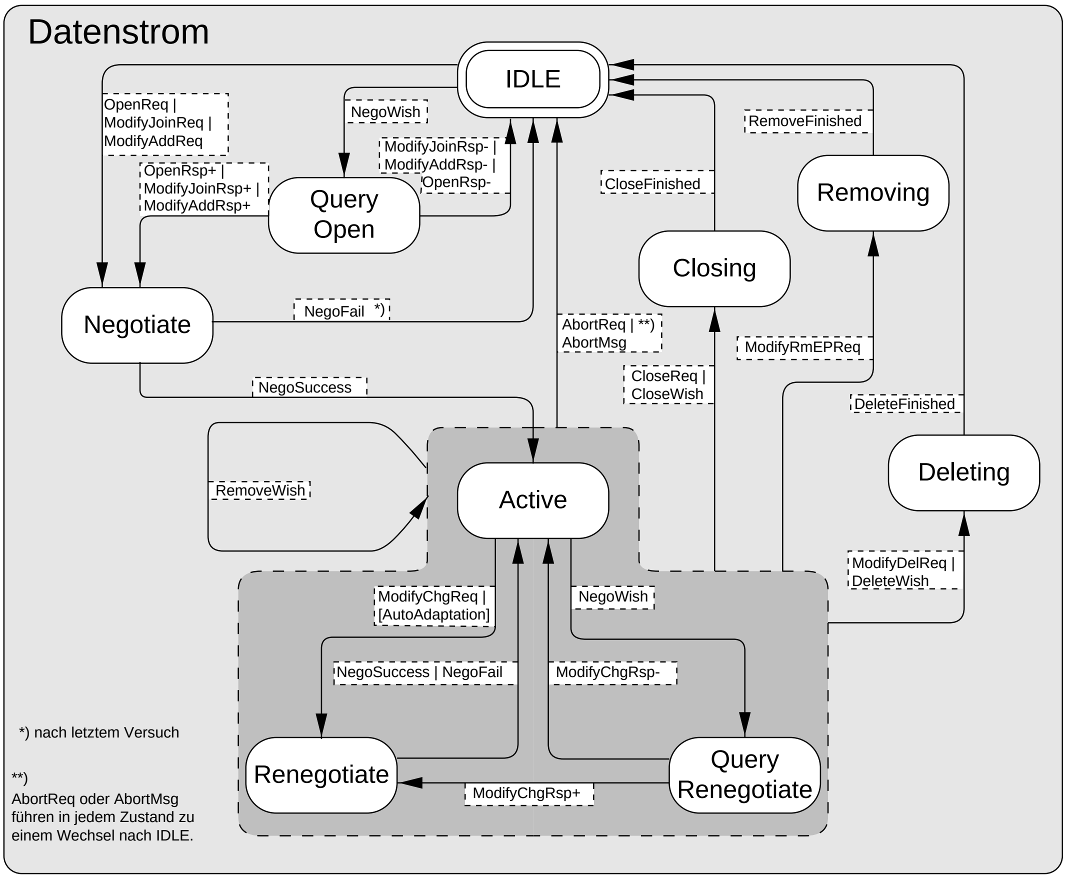

Abbildung 3.4: Zustände eines Datenstroms

Antwort der Anwendung über den weiteren Zustand des Datenstroms. Wird der Datenstrom akzeptiert, wechselt sein Zustand nach Negotiate. Wird der Datenstrom abgelehnt, kehrt er in den Zustand IDLE zurück. Da die Dienstprimitive OpenRsp, ModifyAddRsp, ModifyJoinRsp über ihr positives oder negatives Attribut anzeigen, ob die *Sitzung* als solche akzeptiert oder abgelehnt wird, kann ein einzelner Datenstrom trotz einer positiven Antwort ...Rsp+ abgelehnt werden, sofern er als nicht notwendig gekennzeichnet war. Diese Möglichkeit wurde in der Abbildung 3.4 zur Vereinfachung weggelassen.

- Negotiate
  In diesem Zustand befindet sich der Datenstrom in der initalen Aushandlungsphase. Nach einer für den Datenstrom erfolgreich abgeschlossenen Aushandlung NegoSuccess geht er in den Zustand Active über. Scheitert die Aushandlung für diesen Datenstrom, bleibt er im Zustand Negotiate, solange der Koordinator einen weiteren Aushandlungsversuch unternimmt (s. Abbildung 3.3 Zustand Retry). Andernfalls kehrt er in den Zustand IDLE zurück. Auch hier gilt, daß die Aushandlung für die Sitzung insgesamt durchaus erfolgreich sein kann (NegoSuccess), der betrachtete Datenstrom jedoch ausscheidet.

- Active
  Im Zustand Active ist der Übergang in die beiden Zustände der Neuverhandlung möglich: ein ModifyChgReq bzw. eine vom Koordinator selbständig initiierte Änderung (in der Abbildung angedeutet durch [AutoAdaptation]) führt in den Zustand Renegotiate. Ein NegoWish, welches den betrachteten Datenstrom ändern soll, führt in den Zustand Query Renegotiate. Aus allen drei Zuständen sind Übergänge aufgrund Löschen des Datenstroms (ModifyDelReq oder DeleteWish), Ausscheiden des Teilnehmers (ModifyRmEPReq) oder Beendigung der Sitzung (CloseReq bzw. CloseWish) möglich. Ist eine Neuverhandlung gerade in Bearbeitung, werden diese Befehle bevorzugt behandelt, da sie nicht abgelehnt werden dürfen. Geht der Wunsch eines Teilnehmers ein, aus der Sitzung auszuscheiden (RemoveWish), so kann das Entfernen aus der Empfängerliste ebenfalls während einer Neuverhandlung geschehen.

- Renegotiate
  In diesem Zustand wird analog zum Zustand Negotiate auf das Ergebnis der Neuverhandlung gewartet (NegoSuccess oder NegoFail). Es wird allerdings danach unabhängig vom Ergebnis in den Zustand Active zurückgekehrt.

- Query Renegotiate
  Nach Eingang eines Neuverhandlungswunsches NegoWish wird in diesem Zustand auf die Antwort der Anwendung gewartet. Bei einer Ablehnung (ModifyChgRsp–) kehrt der Datenstrom in den Zustand Active zurück, bei Akzeptanz (ModifyChgRsp+) wechselt der Datenstrom in den Zustand Renegotiate.

- Deleting
  In diesem Zustand wird auf die erfolgreiche Ausführung des Entfernens dieses Datenstroms gewartet. Erst nach Erhalt der Meldung DeleteFinished vom QoS-Makler erfolgt die Rückkehr in den Zustand IDLE.

- Closing
  Nach Erhalt der CloseFinished-Nachricht vom QoS-Makler geht der Datenstrom in den Zustand IDLE über und ist somit geschlossen.

- Removing
  Sind alle Aktionen zum Entfernen des Sitzungsteilnehmers beendet (RemoveFinished), wird auch dieser Datenstrom endgültig geschlossen und kehrt somit in den Zustand IDLE zurück.

3.4 QoS-Makler

Der QoS-Makler übernimmt die *Kommunikationsaufgaben* innerhalb des QoS-Managers. Ziel der Kommunikation ist es, Informationen im gesamten System zu verteilen. Wie in Abbildung 3.1 zu sehen ist, kommuniziert der QoS-Makler mit den drei Kontrollinstanzen des Netzwerks, des Kommunikationssubsystems und des Betriebssystems. Zusätzlich führt er die Ende-zu-Ende Kommunikation mit den QoS-Managern der Kommunikationspartner über einen gesicherten Signalisierungskanal durch und kennt die Schnittstelle sowie den Ablauf des Netzwerkreservierungsprotokolls. Die Kommunikation mit einer lokalen Instanz, d.h. die Benutzung ihrer Schnittstelle, läuft nach gewissen Regeln ab, die zusammengefaßt als Protokoll bezeichnet werden. Die Reihenfolge der Kommunikation mit den Instanzen ist jedoch ebenfalls festzulegen, denn im Zuge der Aushandlung von Dienstqualitäten muß abwechselnd mit lokalen Komponenten, entfernten QoS-Managern und dem Koordinator (bzw. der Anwendung) kommuniziert werden. Diese Aufgabe übernimmt die *Ablaufsteuerung* (s. Abbildung 3.5). Da mehrere Abläufe gleichzeitig in Bearbeitung sein können (etwa Entfernen eines Teilnehmers und Neuverhandeln von bestimmten Datenströmen), ist es auch ihre Aufgabe, die Ereignisse an den Schnittstellen den richtigen Abläufen zuzuordnen. Es wird daher für jeden aktivierten Ablauf ein Kontext angelegt, in dem die aktuelle Phase festgehalten wird. Der QoS-Makler übernimmt die folgenden Funktionen:

- Ende-zu-Ende Dienstaushandlung
  Bei der Etablierung oder Änderungen der Eigenschaften von Datenströmen muß jedesmal eine vollständige Aushandlung erfolgen. Diese beinhaltet auch eine lokale Aushandlung mit den Kontrollinstanzen in den Endsystemen.

- Ende-zu-Ende Dienstabbau
  Werden Datenströme beendet, so müssen alle Kommunikationspartner benachrichtigt, lokale und im Netzwerk reservierte Ressourcen freigegeben werden. Diese Funktionen werden ebenfalls bei Beenden der Sitzung oder Ausscheiden eines Teilnehmers benötigt.

- Austausch von Managementinformationen
  Statusmeldungen der lokalen Komponenten werden an den Koordinator weitergereicht, systemübergreifende Statusmeldungen werden an die Partner verteilt.

Diese Funktionen werden nachfolgend im einzelnen erläutert.

3.4.1 Ende-zu-Ende Aushandlung

Eine Aushandlung wird immer dann nötig, wenn die Werte der Dienstgüteparameter eines Datenstroms festgelegt werden sollen. Dies ist bei folgenden Dienstprimitiven der Fall: Session.OpenReq(), Session.ModifyJoinReq(), Session.ModifyChgReq(), Session.ModifyAddReq(). Die Aushandlung der Dienstqualität gliedert sich grob in vier Schritte: zuerst erfolgt beim Initiator eine Aushandlung mit den *lokalen* Komponenten. Im zweiten Schritt erfolgt eine *Ende-zu-Ende* Aushandlung mit den entfernten

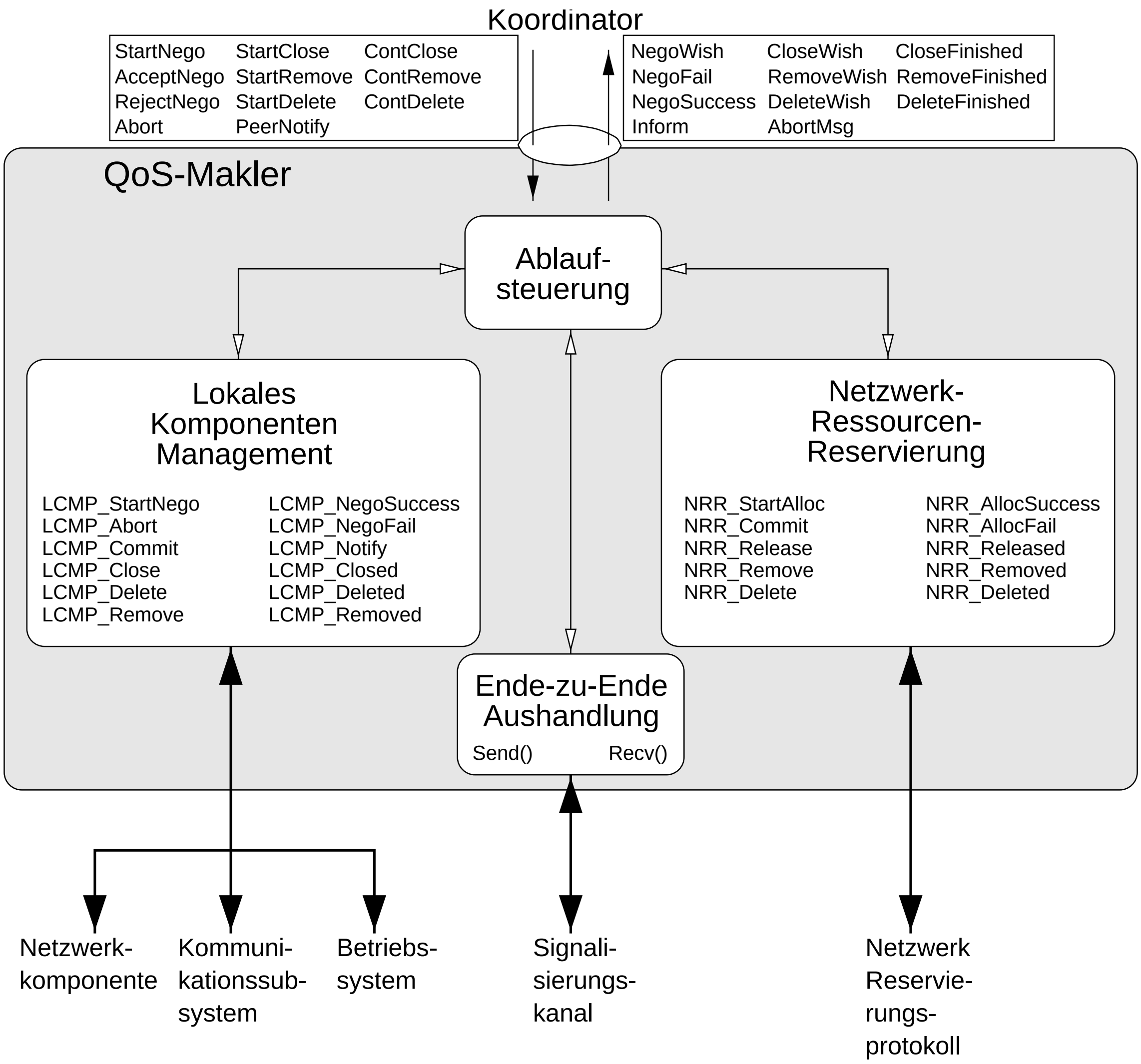

Abbildung 3.5: QoS-Makler mit Kommunikationsschnittstellen

Endsystemen, die dort ihrerseits eine lokale Aushandlung durchführen und die Ergebnisse schließlich dem Initiator zurückschicken. Die Ergebnisse der Empfänger werden beim Initiator zu einem Konsens zusammengeführt und den entfernten Endsystemen mitgeteilt. Erst im dritten Schritt erfolgt eine systemübergreifende *Reservierung der Netzwerkressourcen* durch ein Netzwerkreservierungsprotokoll, da die Reservierung von Netzwerkressourcen Kosten verursacht. Würde eine Reservierung z.B. vor Aushandlung der anwendungsspezifischen Dienstgüteparameter erfolgen, so wären eventuell zuviel Ressourcen im Netzwerk reserviert und bezahlt worden, weil die gerufene Anwendung möglicherweise die gewünschten Dienstqualitätsparameter gar nicht unterstützen kann. Daher sollten erst die kostenpflichtigen Netzwerkressourcen reserviert werden, wenn die Endsysteme sich auf eine gemeinsame Dienstqualität einigen konnten. Abschließend werden im vierten Schritt noch einmal die anwendungsorientierten Dienstgüteparameter an die tatsächlich reservierten Ressourcen im Netzwerk angepaßt, lokal zuviel reservierte Ressourcen in den Komponenten freigegeben und die Anwendung schließlich über die vereinbarte Dienstqualität informiert.

Die Rolle des Initiators einer Aushandlung ist unabhängig davon, ob er Empfänger oder Sender von Datenströmen ist. Üblicherweise werden in einem Endsystem verschiedene Datenströme gleichzeitig empfangen und gesendet. Eine „Video-auf-Abruf"-Anwendung wird typischerweise vom Empfänger des Videostroms (Client) initiiert (Initiator). Der Video-Server (Beantworter) wird daraufhin einen Videostrom und einen Audiostrom aufbauen und zum Client senden. Zusätzlich wird ein zuverlässiger schmalbandiger Rückkanal für die Steuerung durch den Benutzer vom Client zum Video-Server benötigt.

Die Nachrichten vom und zum QoS-Makler sind typisiert (anhand des Dienstprimitivs) und enthalten als Parameter die Beschreibung ein oder mehrerer Datenströme. Erhält der QoS-Makler einen Befehl vom Koordinator, die Aushandlung zu starten (StartNego), so wird folgender Ablauf angestoßen (s. Abbildung 3.6, links): In der ersten Phase (Initiated Local Negotiation) wird die lokale Aushandlung initiiert (LCMP_StartNego, wird anschließend in Abschnitt 3.4.1.1 detailliert erklärt). Scheitert die lokale Aushandlung (LCMP_NegoFail), meldet der QoS-Makler ein NegoFail an den Koordinator und die Aushandlung ist beendet. Andernfalls (LCMP_NegoSuccess) informiert er die übrigen Teilnehmer über den Aushandlungswunsch und die bisher ausgehandelten anwendungsorientierten Parameter mit der anwendungsorientierten Dienstspezifikation und der entsprechenden funktionalen Konfiguration. Dazu verschickt er eine entsprechende Protokolldateneinheit (PDU – Protocol Data Unit) des QoS-Management-Protokolls über einen gesicherten Kanal (Send(NREQ)). Für jede Art von Dienstprimitiven, welche die Aushandlung ausgelöst haben, existiert ein entsprechender PDU-Typ. Das *Paketformat* ist aber für solche PDU-Typen gleich und wird hier mit NREQ (für Negotiation Request) abgekürzt (siehe Abschnitt 4.4.2, Seite 86). Die zweite Phase (Initiated Remote Negotiation) wird fortgesetzt, sobald die Antworten aller Teilnehmer eingetroffen sind. Eine Antwort fällt negativ aus (Recv(NRSPN) – Negotiation Response Negative), wenn die Anforderungen eines unbedingt benötigten Datenstroms oder sämtlicher optionaler Datenströme nicht erfüllt werden konnten. Haben sämtliche Partner abgelehnt, ist die Aushandlung zu Ende und der Koordinator wird über den negativen Ausgang informiert (NegoFail), nachdem die bereits reservierten lokalen Ressourcen wieder freigegeben wurden (LCMP_Abort). Eine positive Anwort (Recv(NRSPP) – Negotiation Response Positive) enthält die beim Partner ausgehandelten Parameter für die akzeptierten Datenströme und auch die Kennzeichnung für abgelehnte optionale Datenströme. Nun können die bei den Partnern ausgehandelten Parameter stark differieren, weshalb der Initiator nun in der dritten Phase einen Konsens finden muß (Find Consensus). Dieser kann mit Hilfe einer zuvor festgelegten Gruppensemantik entschieden werden. Soll zum Beispiel die Gruppe homogen sein, so muß der Initiator einen bestimmten Parametersatz für jeden Datenstrom festlegen, der nachher für jeden Teilnehmer gelten soll. Teilnehmer, welche die geforderten, unbedingt notwendigen Parameter nicht unterstützen können, erhalten dann eine Nachricht, daß für sie die Aushandlung gescheitert ist (Send(NCOMN) – Negotiation Com-

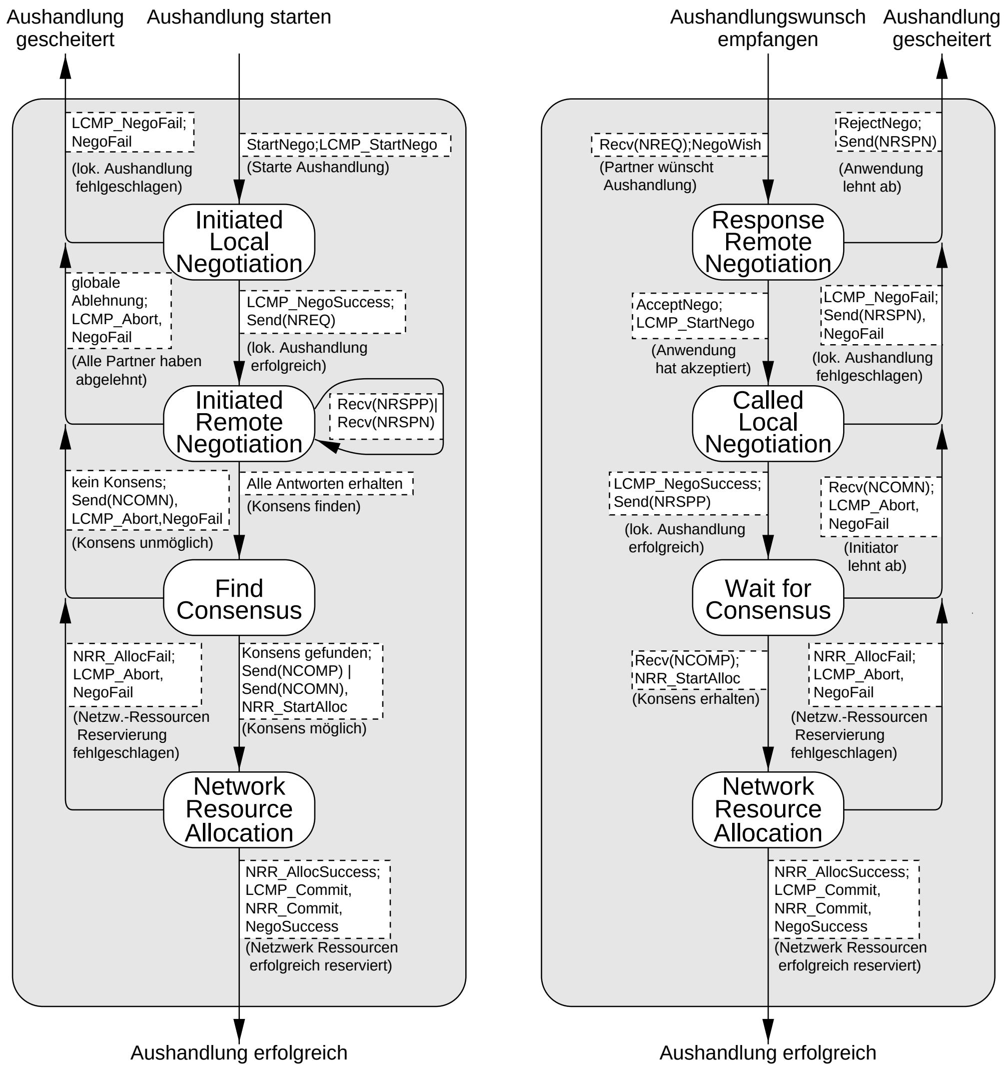

Abbildung 3.6: Ablauf der Aushandlung im QoS-Makler

mit Negative). Alle Teilnehmer, die jedoch die Parameter für den Konsens erfüllen können, erhalten eine Bestätigung vom Initiator mit den von ihm festgelegten Werten (Send(NCOMP) – Negotiation Commit Positive). Der Initiator und die übrigen Teilnehmer leiten dann die vierte Phase der Aushandlung ein, indem sie die Reservierung der Netzwerkressourcen veranlassen (NRR_StartAlloc). Kann in der dritten Phase aber kein Konsens gefunden werden, werden alle Teilnehmer, die eine positive Anwort geschickt hatten, darüber informiert (Send(NCOMN)) und der Initiator zeigt nach Freigabe der bereits reservierten Ressourcen (LCMP_Abort) dem Koordinator das Scheitern der Aushandlung an (NegoFail). In der vierten Phase der Aushandlung (Network Resource Allocation) reserviert der Initiator mittels der Instanz für die Netzwerkreservierung nicht nur Ressourcen für bestimmte Datenströme (z.B. für alle abgehenden), sondern wartet auch auf die Ergebnisse der Ressourcenreservierung im Netzwerk für die übrigen Datenströme (z.B. für alle eingehenden). Scheitert die Reservierung der Netzwerkressourcen (NRR_AllocFail), werden die lokal vorreservierten Ressourcen wieder freigegeben (LCMP_Abort) und der Koordinator informiert (NegoFail). Ist die Reservierung der Netzwerkressourcen erfolgreich, müssen die von der Netzwerkreservierung zugesagten netzwerkorientierten Parameterwerte wieder auf anwendungsorientierte Parameterwerte umgesetzt werden (Rückabbildung). Da einige Werte der anwendungsorientierten Parameter nur diskret verändert werden können, kann es sein, daß nun die entsprechenden Werte der Netzwerkparameter abermals verändert werden. Damit nicht zuviel Ressourcen im Netzwerk und Endsystem reserviert bleiben, erfolgt eine entsprechende lokale Bestätigung mit den nun endgültig vereinbarten Werten (LCMP_Commit, NRR_Commit). Der Koordinator wird über den Erfolg der Aushandlung und die endgültig ausgehandelte Dienstqualität informiert (NegoSuccess). Es wird vorausgesetzt, daß jeder Teilnehmer über die Netzwerkreservierung erfährt, welche Parameterwerte auf Netzwerkebene ausgehandelt wurden. Somit ist kein erneuter Austausch zwischen den QoS-Managern nötig. Die in jedem Endsystem identische Rückabbildung sorgt dann für die gleichen Ergebnisse der Anwendungsparameterwerte bei allen Teilnehmern.

Erhält der QoS-Makler den Aushandlungswunsch eines entfernten Sitzungsteilnehmers (Recv(NREQ)), so folgt der Ablauf aus Bild 3.6 rechts: Der QoS-Makler informiert den Koordinator über den Aushandlungswunsch (NegoWish) und wartet auf dessen Antwort (Response Remote Negotiation). Der Koordinator teilt dann dem QoS-Makler mit, ob der Aushandlungswunsch von der Anwendung akzeptiert (AcceptNego) oder abgelehnt (RejectNego) wurde. Im zweiten Fall wird der Initiator des Aushandlungswunsches über die Ablehnung mit Angabe von Gründen informiert (Send(NRSPN)) und die Aushandlung ist beendet. Der Koordinator erwartet in diesem Fall auch keine Antwort des QoS-Maklers mehr. Im ersten Fall wird mit den Parameterwerten aus der AcceptNego-Nachricht eine lokale Aushandlung gestartet (LCMP_StartNego). Scheitert die lokale Aushandlung (LCMP_NegoFail), wird dies dem entfernten Initiator (Send(NRSPN)) der Aushandlung und dem Koordinator (NegoFail) mitgeteilt. Ist die lokale Aushandlung erfolgreich (LCMP_NegoSuccess), wird das Ergebnis in anwendungsorientierte Parameter zurück abgebildet und dem Initiator mitgeteilt (Send(NRSPP)). Der QoS-Makler erwartet sodann die Nachricht des Initiators über einen Konsens (Wait for Consensus). Kommt kein Konsens zustande oder ist das System nicht im Konsens inbegriffen, erhält der QoS-Makler eine Ablehnung vom Initiator (Recv(NCOMN)) und muß dann die bereits lokal vorreservierten Ressourcen freigeben (LCMP_Abort), sowie den Koordinator über das Scheitern der Aushandlung in Kenntnis setzen (NegoFail). Erhält der QoS-Makler eine bestätigende Anwort des Initiators (Recv(NCOMP)), leitet er die letzte Phase der Aushandlung ein und startet die Reservierung der Netzwerkressourcen (Network Resource Allocation). Schlägt die Reservierung der Netzwerkressourcen fehl, werden die bereits lokal vorreservierten Ressourcen freigeben (LCMP_Abort) und der Koordinator benachrichtigt (NegoFail). Andernfalls werden zuviel reservierte Ressourcen freigegeben (LCMP_Commit, NRR_Commit) und der Koordinator über die endgültig ausgehandelte Dienstqualität informiert (NegoSuccess).

Nun soll noch der Ablauf erklärt werden, der durch LCMP_StartNego ausgelöst wird.

3.4.1.1 Lokale Aushandlung

In jedem Endsystem müssen die drei lokalen Komponenten Betriebssystem, Kommunikationssubsystem und Netzwerk in die Aushandlung integriert werden: ohne eine entsprechende Unterstützung durch die Ressourcen der Komponenten sind keine Dienstgarantien erzielbar. Die Komponenten müssen daher einzeln nach ihrer möglichen Unterstützung befragt werden. Die Reihenfolge, in der die Komponenten befragt werden, sollte jedoch nicht willkürlich gewählt werden, da z.B. das Kommunikationssubsystem auf Diensten des Netzwerks aufsetzt. Die lokale Aushandlung ist durch das *Lokale Komponenten Management Protokoll* (LCMP – Local Component Management Protocol) festgelegt. Der Ablauf der Aushandlung ist in Abbildung 3.7 zu sehen: Die Aushandlung wird auch hier für mehre-

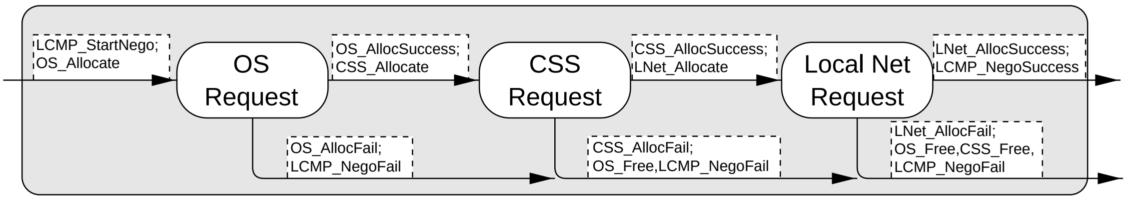

Abbildung 3.7: Ablauf der lokalen Aushandlung

re Datenströme simultan durchgeführt. Zuerst wird mit den anwendungsorientierten Dienstqualitätsparametern bei der Betriebssystemkomponente nach einer möglichen Unterstützung gefragt (OS_Allocate). Diese Unterstützung kann durch spezielle Zusatzgeräte wie Kodier- und Dekodierkarten erfolgen. Diese Zusatzgeräte operieren auf anwendungsorientierten Daten und benötigen daher auch Angaben zur gewünschten Dienstqualität auf dieser Ebene. Kann die gewünschte Güte aufgrund von eingeschränkten Fähigkeiten der Geräte oder bereits bestehender Belastung durch nebenläufige Nutzung nicht erreicht werden, so werden die anwendungsorientierten Dienstqualitätsparameter entsprechend herabgesetzt. Beispiel hierfür sei eine MPEG-Dekoderkarte, die nur 18 statt der gewünschten 24 Bilder pro Sekunde in einer bestimmten Auflösung leisten kann. Die eventuell veränderten Parameter werden an den QoS-Makler zurückgegeben (OS_AllocSuccess). Für die Anwendung muß auch genügend Rechenzeit und Arbeitsspeicher reserviert werden, um die auszutauschenden Daten bereitzustellen oder zu verarbeiten. Es besteht zum einen die Möglichkeit, daß die Anwendung ihre benötigte Rechenzeit selbst angibt, zum anderen kann diese Zeit aus einer Tabelle entnommen werden, die gemessene oder durchschnittliche Werte für diese Art von Anwendung enthält. Zur Zeit wird die letztere Möglichkeit angewendet, um die Anwendung zu entlasten. Kann das Betriebssystem die notwendige Unterstützung nicht gewähren, erhält der QoS-Makler ein OS_AllocFail zurück und die lokale Aushandlung gilt als gescheitert (LCMP_NegoFail). War die Reservierung im Betriebssystem erfolgreich (OS_AllocSuccess), folgt die nächste Phase (CSS Request) zur Reservierung von Ressourcen im Kommunikationssubsystem (CSS – Communication Subsystem). Dafür müssen zuerst die (eventuell veränderten) anwendungsorientierten Dienstqualitätsparameter durch den QoS-Abbilder auf kommunikationsorientierte Dienstqualitätsparameter abgebildet werden, dann erfolgt die Anfrage an das Kommunikationssubsystem mit der vorher durch die funktionale Konfiguration ausgewählten Funktionalität und den konkreten Parameterwerten (CSS_Alloc). Kann das Kommunikationssubsystem die notwendigen Parameterwerte der unentbehrlichen Datenströme nicht zusichern (CSS_AllocFail), scheitert die lokale Aushandlung und die zuvor reservierten Ressourcen des Betriebssystems werden freigegeben (OS_Free). Kann das Kommunikationssubsystem die nötigen Ressourcen reservieren (CSS_AllocSuccess), werden die von ihm zugesagten Werte zurückgegeben. Es werden aber außerdem noch die eventuell erhöhten Anforderungen des Kommunikationssubsystems an die Dienstqualitätsparameter gegenüber dem Netzwerk mitgeteilt. Dies ist notwendig, um der Bearbeitung im Kommunikationssubsystem Rechnung zu tragen, z.B. der erhöhten Durchsatzanforderung durch Benutzung eines Vorwärtsfehlerkorrekturmechanismus. In der letzten Phase wird nun die lokale Netzwerkkomponente mit den neuen Werten befragt (LNet_Allocate). Kann die Netzwerkkomponente die gewünschten Dienste erbringen, erfolgt die Rückgabe der zugesicherten Werte (LNet_AllocSuccess) und die lokale Aushandlung ist erfolgreich abgeschlossen (LCMP_NegoSuccess). Andernfalls (LNet_AllocFail) werden die schon reservierten Ressourcen des Kommunikationssubsystems und des Betriebssystems wieder freigegeben (CSS_Free, OS_Free) und die lokale Aushandlung gilt als gescheitert (LCMP_NegoFail). Ein späteres LCMP_Commit nach der Reservierung der Netzwerkressourcen (s. Abbildung 3.6) sorgt bei allen drei Komponenten für eine Anpassung der Ressourcen (OS_ChgAlloc, CSS_ChgAlloc, LNet_ChgAlloc), wodurch die zuviel reservierten Ressourcen freigegeben werden.

Die Anfrage an die *lokale* Netzwerkkomponente wird folgendermaßen motiviert: Im allgemeinen werden mehrere Verbindungen zu verschiedenen Endsystemen „gleichzeitig" über einen Netzzugang durch die Netzwerkadapterkarte unterhalten (bei verbindungslosen Diensten sind es Datagramme unterschiedlicher Herkunft oder Bestimmung). Somit wird die verfügbare Bandbreite unter mehreren Nutzern aufgeteilt. Eine Anfrage nach zur Verfügung stehender Bandbreite zwecks Reservierung sollte daher sinnvollerweise in eine *lokale* und eine *entfernte* Anfrage aufgeteilt werden. Für eine lokale Anfrage wird nur die momentane Auslastung der *lokalen* Netzwerkkomponente im Endsystem berücksichtigt. Ist diese schon ausgelastet, erübrigt sich eine entfernte Anfrage an Zwischensysteme und die Partner-Endsysteme. Für die Erbringung garantierter Dienste ist eine Unterstützung allen involvierten Zwischensystemen und dem entfernten Endsystem notwendig. Dort müssen ebenfalls Ressourcen (Puffer und Bearbeitungszeit) reserviert werden, um Garantien einhalten zu können. Die Anforderungen müssen daher an alle Zwischen- und Endsysteme verteilt werden (entfernte Anfrage), damit die lokalen Ressourcenverwalter dort entsprechende Zugangskontrolltests und Reservierungen durchführen können. Für diesen Zweck wurden Reservierungsprotokolle wie ST-2+ [DeBe95] und RSVP [ZDES93] entwickelt. Üblicherweise erfolgt beim Verbindungsaufbau auf dem Weg vom Sender zu den Empfängern die Reservierung der Ressourcen, eine Bestätigung mit der endgültigen Dienstqualität wird zurückgeschickt. Auf dem Rückweg werden eventuell zuviel reservierte Ressourcen freigegeben. Die Reservierung für das initiierende System darf natürlich nur einmal bei der lokalen Anfrage vorgenommen werden, d.h. das Netzwerkreservierungsprotokoll muß die bereits bestehende Reservierung der lokalen Netzwerkressourcen berücksichtigen.

3.4.1.2 Beispielszenario einer Ende-zu-Ende Aushandlung

Die Ende-zu-Ende Aushandlung zwischen zwei Kommunikationspartnern ist in Abbildung 3.8 dargestellt:

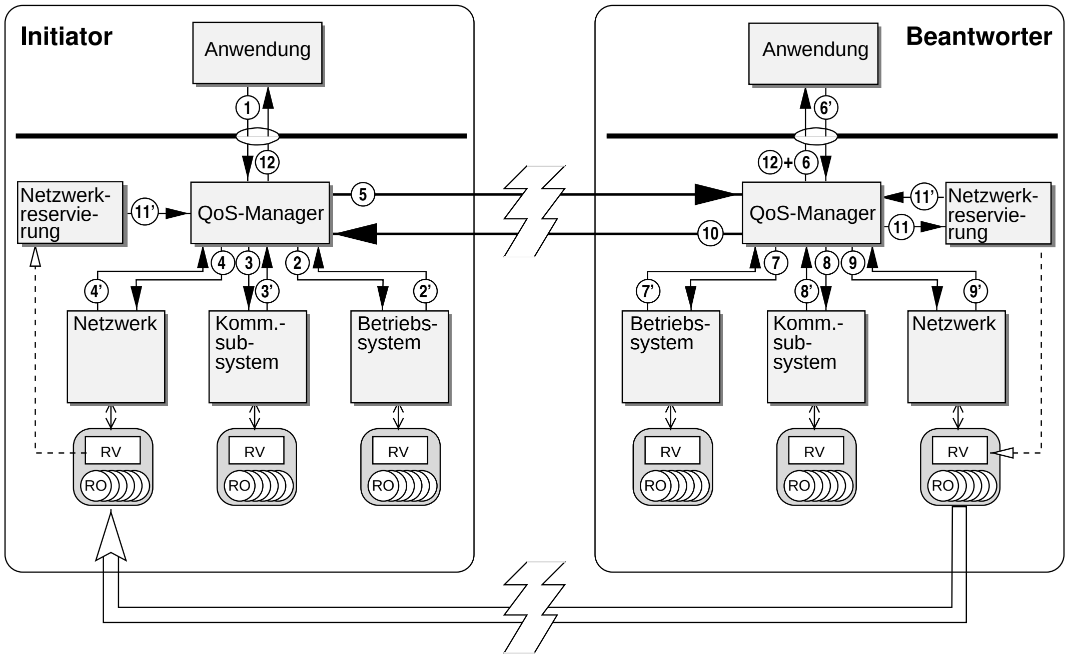

Abbildung 3.8: Aushandlungsszenario

1. Die initiierende Anwendung spezifiziert ihre Anforderungen für eine Sitzung, die aus mehreren Datenströmen unterschiedlicher Spezifikation bestehen kann. Der QoS-Manager wählt aufgrund der Anforderungen und anhand einer Tabelle der zur Verfügung stehenden Netzwerkdienste nach festgelegten Regeln einen Netzwerkdienst und einen Dienst des Kommunikationssubsystems aus (*funktionale Konfiguration*).

2. Der QoS-Manager fragt mit den anwendungsspezifischen Dienstgüteparametern nach einer Unterstützung im Betriebssystem. Für jeden Datenstrom veranlaßt der Kontrollagent eine Reservierung der benötigten Ressourcen. Dazu zählen auch spezielle multimediale Geräte, die in irgendeiner Form belegt werden müssen. Das Betriebssystem gibt daraufhin eine Antwort (2'): *Akzeptanz* oder *Ablehnung*. Der QoS-Manager muß gegebenenfalls veränderte Werte der anwendungsspezifischen Dienstgüteparameter berücksichtigen, die nun wieder

auf die systemorientierten Parameter für das Kommunikationssubsystem abgebildet werden müssen.

3. Der QoS-Manager gibt den Typ des Netzwerkdienstes und Dienstqualitätsanforderungen an das Kommunikationssubsystem weiter. Dieses reserviert ebenfalls vor und gibt eventuell veränderte Größen zurück (3'). Zusätzlich wird die Anforderung an den Netzwerkdienst formuliert und mitgeteilt.

4. Der QoS-Manager fordert bei der lokalen Netzwerkkomponente einen Dienst mit der bisher benötigten Funktionalität und Qualität an. Die tatsächlich erreichbare Dienstqualität wird von der Netzwerkkomponente zurückgegeben (4') und Ressourcen entsprechend vorreserviert. Geänderte Werte müssen nun mittels einer Rückabbildung auf anwendungsorientierte Parameter berücksichtigt werden.

5. Nachdem alle lokalen Ressourcen reserviert werden konnten, startet der initiierende QoS-Manager die Ende-zu-Ende Aushandlung und kontaktiert den Partner QoS-Manager. Übermittelt werden die anwendungsorientierten Dienstqualitätsparameter, sowie Konfigurationshinweisen für das Kommunikationssubsystem und den Netzwerkdienst.

6. Der entfernte QoS-Manager kontaktiert die adressierte Anwendung und übergibt Anwendungs-QoS. Die Anwendung nimmt den Verbindungswunsch an oder lehnt ihn ab. Bei Ablehnung erfolgt eine negative Rückantwort an den initiierenden QoS-Manager (10.).

7. Es wird nach einer lokalen Unterstützung durch das Betriebssystem gefragt. Die Antwort (7') spiegelt sich in veränderten Werten der anwendungsorientierten Dienstqualitätsparameter wieder.

8. Mit der vom Initiator festgelegten Konfiguration des Kommunikationssubsystems wird nun bei diesem mit den aktuellen systemorientierten Parametern angefragt (eine funktionale Konfiguration wie beim Initiator findet nicht mehr statt). Auch hier können veränderte Werte zurückgeliefert werden (8').

9. Schließlich wird bei der lokalen Netzwerkkomponente mit den entsprechenden Parametern angefragt, die erfüllbaren Werte zurückgegeben (9'). Eine Rückabbildung auf Anwendungs-QoS muß nun für den nächsten Schritt durchgeführt werden.

10. Der antwortende QoS-Manager informiert nun den initiierenden QoS-Manager über die dort ausgehandelte Dienstqualität.

11. Der QoS-Manager, dessen Endsystem eine oder mehrere unidirektionale Datenstrom-Verbindungen aufbaut, kontaktiert die Netzwerkreservierungsinstanz und bekommt reservierte Größen zurück (11'). Nun muß eine Rückabbildung

auf Anwendungs-QoS und interne QoS-Parameter erfolgen, damit lokal Ressourcen freigegeben werden können, die zuvor überreserviert wurden. Der Partner QoS-Manager erhält ebenfalls Informationen über reservierte Verbindungen durch die Netzwerkreservierungsinstanz (11'). Auch hier werden Rückabbildungen vorgenommen, zuviel belegte Ressourcen freigegeben.

12. Die Anwendungen werden über endgültig ausgehandelte QoS informiert und können ab dann Daten senden.

Um die Darstellung einfach zu halten, wurde ein Szenario gewählt, welches die Aushandlung lediglich zwischen zwei Teilnehmern beschreibt. Dadurch wird hier keine Konsensfindung und das anschließende „Commit“ dargestellt, welche allerdings für eine Gruppenkommunikation mit mehr als zwei Teilnehmern notwendig ist.

3.4.2 Ende-zu-Ende Dienstabbau

Es gibt vier mögliche Ursachen für einen Dienstabbau: Der Initiator der Sitzung schließt die Sitzung (Session.CloseReq()), ein Teilnehmer möchte aus einer bestehenden Sitzung ausscheiden (Session.RmEPReq()), ein Datenstrom soll geschlossen werden (Session.ModifyDelReq()) oder einen abrupten Sitzungsabbau (Session.AbortReq()). Grunsätzlich verläuft der Dienstabbau in zwei Phasen: Der den Abbau initiierende QoS-Makler informiert die übrigen Teilnehmer über den durchzuführenden Abbau, veranlaßt die lokale Beendigung des Dienstes und wartet danach auf die Bestätigungen der übrigen Teilnehmer. Da die entfernten Teilnehmer die Aufforderung zur Beendigung des Dienstes nicht ablehnen dürfen, führen sie ebenfalls einen lokalen Abbau durch, informieren anschließend den initiierenden QoS-Makler über den vollzogenen lokalen Dienstabbau und beginnen die mit dem Dienst assoziierten Netzwerk-Ressourcen freizugeben. Nachdem der initiierende QoS-Makler alle Antworten erhalten hat, beginnt auch er die systemübergreifenden reservierten Netzwerk-Ressourcen freizugeben.

Der Ablauf im QoS-Makler für den Abbau einer Sitzung ist in Abbildung 3.9 dargestellt: Der Koordinator weist den QoS-Makler an, die Sitzung zu schließen (StartClose). Der QoS-Makler verschickt daraufhin eine Nachricht an alle Teilnehmer, daß die Sitzung beendet wird (Send(CLREQ)) und veranlaßt den lokalen Abbau der Sitzung (LCMP_Close). Ist die Sitzung lokal abgebaut (LCMP_Closed), wartet der Initiator auf die impliziten Bestätigungen aller Teilnehmer (Recv(CLRSP)). Erst danach wird die Netzwerkressourcenreservierung angewiesen, die Ressourcen im Netzwerk wieder freizugeben (NRR_Release) und damit die Netzwerkverbindung zu beenden. Nachdem dies abgeschlossen ist (NRR_Released), meldet der QoS-Makler dem Koordinator den Vollzug des Sitzungabbaus (CloseFinished). In Abbildung 3.9 ist unten dargestellt, wie ein eingehender Wunsch des Initiators zum Sitzungsabbau (Recv(CLREQ)) beantwortet wird: Zunächst wird dem Koordinator der Wunsch angezeigt (CloseWish), der die Anweisung zur Fortsetzung des Sitzungsabbaus (Cont-

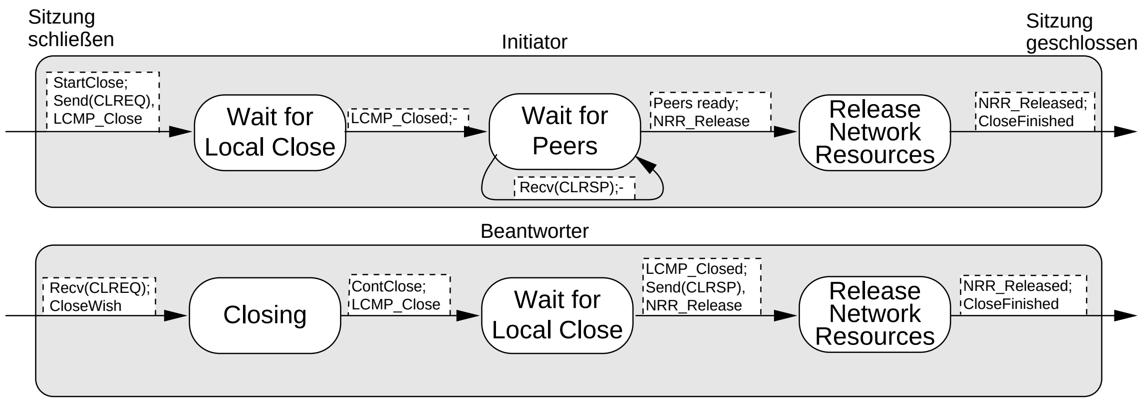

Abbildung 3.9: Ablauf bei Schließen der Sitzung

Close) gibt, sobald er die Anwendung darüber informiert hat. Anschließend erfolgt der zum Initiator analoge Ablauf: zunächst wird die Sitzung lokal abgebaut (LCMP_Close), danach wird die Bestätigung, daß nun alles lokal abgebaut wurde, an den Initiator zurückgeschickt (Recv(CLRSP)). Schließlich werden die im Netzwerk reservierten Ressourcen freigegeben (NRR_Release).

Der Ablauf eines LCMP_Close ist in Abbildung 3.10 zu sehen: Zuerst wird der Ab-

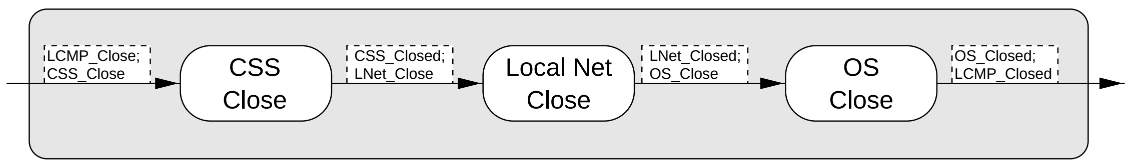

Abbildung 3.10: Lokales Schließen der Sitzung

bau aller Datenströme durch das Kommunikationssubsystem ausgelöst (CSS_Close). Hierbei handelt es sich um einen geordneten Abbau, bei dem dafür gesorgt wird, daß noch alle in der Übertragung befindlichen Dateneinheiten zugestellt werden. Zuerst werden alle Datenströme beendet, die von diesen Endsystem aufgebaut wurden (z.B. alle abgehenden), anschließend wird darauf gewartet, daß die übrigen Datenströme (z.B. alle ankommenden) durch die anderen Teilnehmer vom Kommunikationssubsystem beendet wurden. Erst dann werden im Kommunikationssubsystem die zugehörigen Ressourcen freigegeben und angezeigt, daß alle Datenströme beendet wurden (CSS_Closed). Anschließend erfolgt die entsprechende Prozedur in der Netzwerkkomponente (LNet_Close) bei der nur die zugehörigen Ressourcen freigegeben werden. Als letzte Aktion beim lokalen Schließen der Sitzung werden die Ressourcen im Betriebssystem freigegeben (OS_Close).

Möchte ein Teilnehmer aus einer bestehenden Sitzung ausscheiden, so erhält sein QoS-Makler die Nachricht StartRemove. Dieser informiert alle übrigen Teilnehmer (Send(RMEPREQ)) und beginnt wie oben die Sitzung lokal abzubauen, anschließend wartet er auf die Bestätigung der entfernten Teilnehmer, daß sie ihrerseits die Entfernung des Teilnehmers lokal durchgeführt haben. Die übrigen Teilnehmer bauen nämlich nicht die Sitzung ab, sondern entfernen den Teilnehmer und die damit verbundenen Datenströme und Ressourcen lediglich aus ihrem Sitzungskontext. Ein eingehender Wunsch eines Teilnehmers aus der Sitzung auszuscheiden (Recv(RMEPREQ)) wird daher zuerst dem Koordinator mitgeteilt (RemoveWish), damit dieser die Anwendung darüber informieren kann. Danach weist der Koordinator den QoS-Makler an, mit dem Entfernen des Teilnehmers fortzufahren (ContRemove). Der QoS-Makler informiert seinerseits die lokalen Komponenten über die Änderung (LCMP_Remove), die daraufhin die notwendigen Operationen ausführen, um den Teilnehmer zu entfernen. Anschließend wird dem Initiator mitgeteilt, daß er lokal aus der Sitzung entfernt wurde (Send(RMEPRSP)) und die Netzwerkreservierung angewiesen, die mit dem ausgeschiedenen Teilnehmer verbundenen Ressourcen wieder freizugeben (NRR_Remove). Der Ablauf für den Initiator des Session.ModifyRmEP-Req() und die Beanworter ist in Abbildung 3.11 dargestellt.

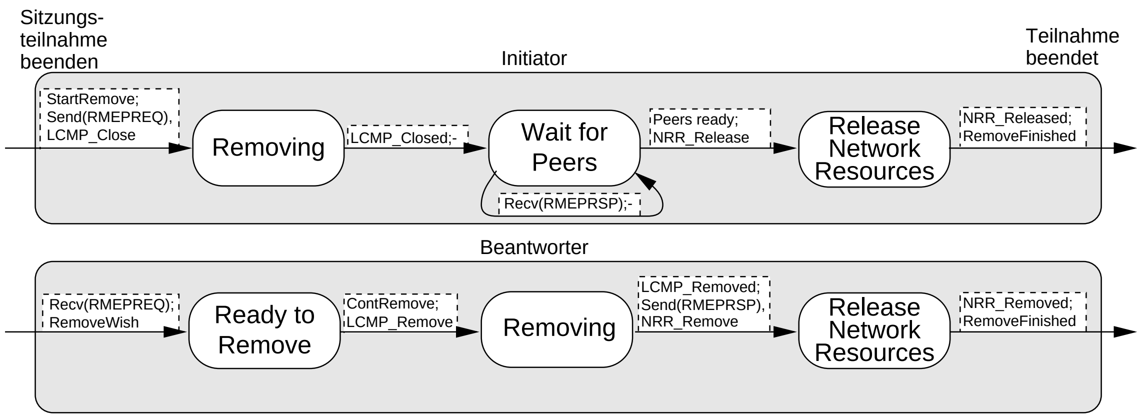

Abbildung 3.11: Entfernen eines Sitzungsteilnehmers

Sollen ein oder mehrere Datenströme geschlossen werden, so folgt der in Abbildung 3.12 dargestellte Ablauf. Der Koordinator beim Sender der Datenströme weist den QoS-Makler an, diese zu schließen (StartDelete). Der QoS-Makler informiert die übrigen Teilnehmer darüber (Send(DELREQ)) und beginnt die Datenströme lokal abzubauen (LCMP_Delete – analog zu einem LCMP_Close, nur daß lediglich die Ressourcen der abzubauenden Datenströme freigegeben werden). Die übrigen Teilnehmer bestätigen dem Initiator die Durchführung (Send(DELRSP)). Anschließend werden beim Initiator die mit den Datenströmen verbundenen Netzwerkressourcen wieder freigegeben (NRR_Release).

Ein abrupter Sitzungsabbau wird dem QoS-Makler durch eine Abort-Nachricht vom Koordinator oder durch den Empfang einer Abort-Nachricht (Recv(ABORT)) vom Initiator mitgeteilt. Der QoS-Makler veranlaßt daraufhin die sofortige Freigabe aller lokalen Ressourcen (LCMP_Abort – analog zu einem LCMP_Close, nur daß die Ressourcen sofort freigegeben werden, auch wenn ihre Nutzung noch nicht abgeschlossen ist) und der im Netzwerk reservierten Ressourcen (NRR_Release).

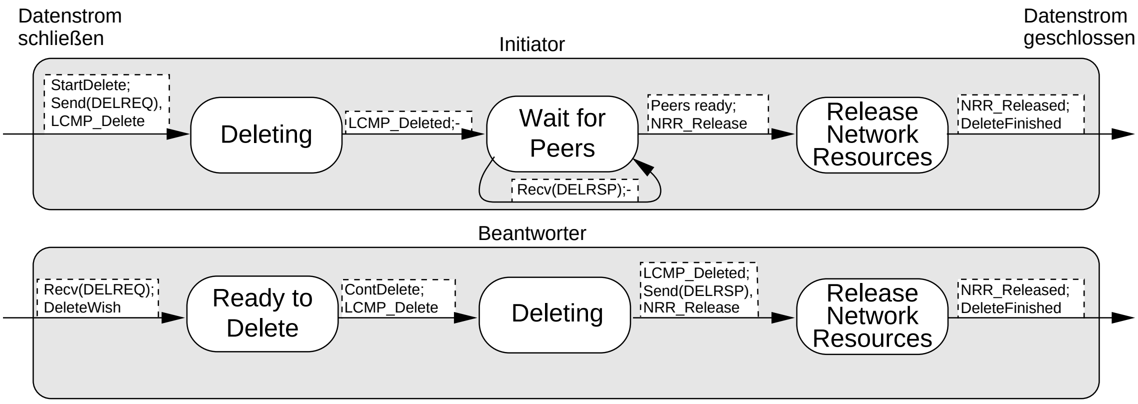

Abbildung 3.12: Löschen von Datenströmen

3.4.3 Austausch von Managementinformation

Der Austausch von Monitoringinformation oder Meldungen über Verletzungen der vereinbarten Dienstqualität werden ebenfalls über den QoS-Makler abgewickelt. Dabei kann es sich um lokale Meldungen oder entfernte Meldungen handeln. Lokale Meldungen von den drei Komponenten (LCMP_Notify) werden durch den QoS-Makler an den Koordinator weitergereicht (Inform). Entfernte Meldungen (Recv(INFO)) werden ebenfalls durch Inform an den Koordinator weitergereicht. Der Koordinator kann seinerseits den QoS-Makler beauftragen (PeerNotify), Meldungen an andere Teilnehmer zu übermitteln (Send(INFO)).

3.5 QoS-Abbilder

Da die Komponenten unterschiedliche Dienstspezifikationen verwenden, müssen die Dienstgüteparameter entsprechend aufeinander abgebildet werden. Beispielsweise ist für einen Videostrom auf Anwendungsebene die Bildgröße, Farbtiefe und Bildrate in einen entsprechenden Durchsatz umzurechnen. Das Kommunikationssubsystem fügt aber weitere Daten zur Protokollverarbeitung hinzu, so daß sich der Durchsatzbedarf gegenüber dem Netzwerk erhöht. Besonders starken Einfluß auf solche Abbildungen haben Segmentier- und Reassemblierfunktionen, durch die größere Dateneinheiten in kleinere zerlegt (und später wieder zusammengesetzt) werden. Es hat sich gezeigt [ClTe90], daß es vorteilhaft für die weitere Protokollverarbeitung ist, das Segmentieren und Reassemblieren (Segmentation and Reassembly – SAR) so früh wie möglich durchzuführen, damit nachfolgende Operationen auf Dateneinheiten gleicher Größe durchgeführt werden können (Integrated Layer Processing – ILP). Beim Prinzip des „Application Level Framing“ (ALF) produziert die Anwendung direkt Datenpakete der „passenden“ Größe (sogenannte Application Data Units – ADUs). Dabei wird eine ADU so definiert, daß sie die Einheit bildet, auf der sämtliche Manipulationen ausgeführt werden (eine ADU tritt somit an die Stelle eines „klassischen“ Datenpakets). Als grundlegende Charakteristik einer ADU wird in [ClTe90] definiert, daß jede ADU ohne Bezug zu den anderen ADUs verarbeitet werden kann. Somit sind ADUs diejenigen Einheiten, auf denen z.B. auch Ende-zu-Ende Fehlererkennung und Fehlerbehebung, sowie das Verschieben in den Anwendungsadreßraum basiert.

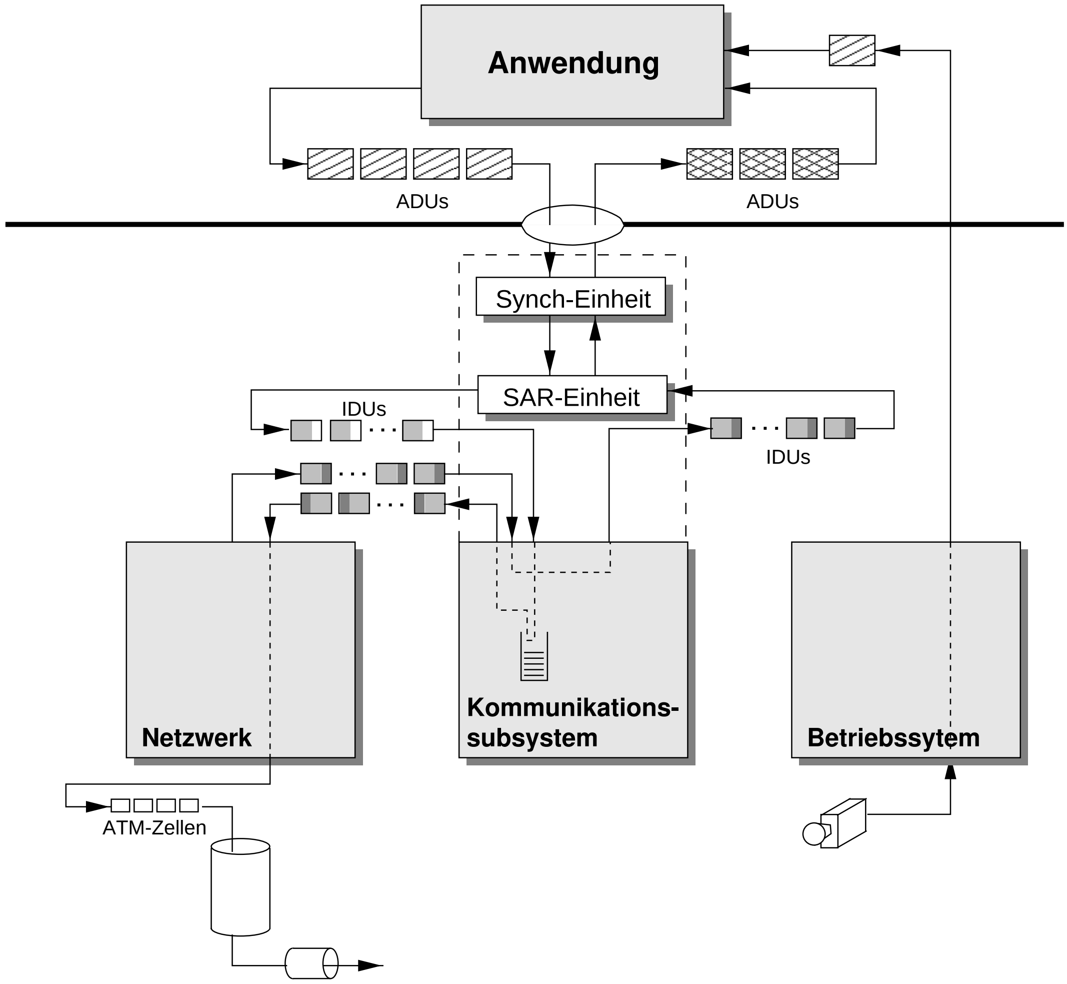

Abbildung 3.13: Datenfluß mit SAR-Einheit

Die Anwendungsdateneinheiten können sehr groß sein (z.B. ein einzelnes Videobild in hoher Auflösung) und müssen daher für die Übertragung in kleinere Segmente zerlegt werden, die der Netzwerkdienst noch verarbeiten kann. Um die Dienstgüteabbildungen einfach zu halten und eine gemeinsame Bezugsgröße für weitere Komponenten zu haben, sollte schon direkt vor oder nach der anwendungsorientierten Dienstschnittstelle zum Kommunikationssystem segmentiert werden (s. Abbildung 3.13). Somit können alle nachfolgenden Komponenten auf Dateneinheiten gleicher Größe operieren, welche die Bezeichnung IDU (Internal Data Unit) tragen. Die Größe der IDUs wird aufgrund der maximalen Größe einer Nutzdateneinheit des benutzten Netzwerks (Maximum Transfer Unit size – MTU size) und der hinzuzufügenden Protokollinformation (Protokoll Control Information – PCI) des Kommunikationssubsystems bestimmt. Es wird angenommen, daß die Größe des Protokollkopfes – unabhängig von der tatsächlich genutzten Protokollfunktionalität – konstant bleibt, da dies auch eine einfachere Verarbeitung bei hohen Geschwindigkeiten erlaubt. So wird z.B. ein Feld für die Prüfsumme immer vorhanden sein, unabhängig davon, ob der Mechanismus zur Fehlererkennung ein- oder ausgeschaltet ist.

Der *QoS-Abbilder* ist dafür zuständig, die anwendungsorientierten auf die systemorientierten Parameter abzubilden (Hinabbildung) und umgekehrt (Rückabbildung). Die Dienstabbildung von anwendungsorientierten Parametern auf systemorientierte verläuft in zwei Schritten (s. Abbildung 3.14):

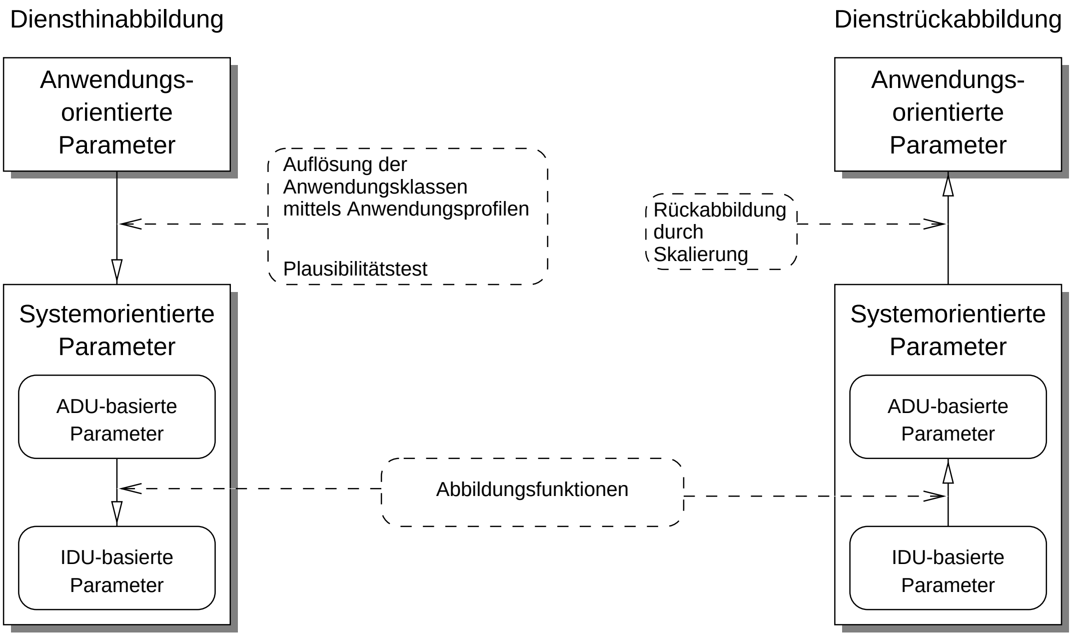

Abbildung 3.14: Die Abbildung der Dienstgüteparameter

1. Zuerst werden nicht spezifizierte Parameter durch Standardbelegungen ersetzt und anhand von sogenannten *Anwendungsprofilen* in konkrete Werte der Struktur aus Tabelle 3.6 umgesetzt (Auflösen der Standardwerte durch eine Datenbasis). Anschließend werden diese Werte mittels einer zum Medientyp gehörenden Abbildung (*medientyp-inhärente Abbildung*) auf die ADU-basierten Parameter abgebildet und einer *Plausibilitätsprüfung* unterzogen. Die *medientyp-inhärente Abbildung* sorgt beispielsweise dafür, daß die ADU-Größe berechnet wird, Werte für den Delay-Jitter eingesetzt und die Dienstklasse der einzelnen Parameter bestimmt werden. Die Plausibilitätsprüfung überprüft die Spezifikation der Datenströme auf Konflikte, denn manche Kompressionsverfahren implizieren bestimmte Parameterbedingungen [Henr96]. Die Synchronisationsanforderungen durch die Interstrombeziehungen müssen ebenfalls auf Erfüllbarkeit überprüft werden.

2. Die ADU-basierten Parameter werden auf IDU-basierte Parameter abgebildet.

3.5.1 Medientyp-inhärente Abbildung

Es soll nun ein kurzer Überblick über die medientyp-inhärenten Abbildungsvorschriften gegeben werden. Die folgenden Angaben beziehen sich auf unkomprimierte Datenströme. Die ADU-basierten Parameter für komprimierte Datenströme hängen stark vom verwendeten Kompressionsverfahren ab und sollen hier nicht näher betrachtet werden. Die Ermittlung der am besten passenden Werte ist für einige Parameter nur empirisch möglich. Diese Werte werden dann in sogenannten *Anwendungsprofilen* festgehalten.

- ADU-Größe
  Die Größe einer ADU ist für die unkomprimierten Datenströme nicht schwierig zu bestimmen. Für Video und Graphik berechnet sich die ADU-Größe zu Höhe×Breite×Farbtiefe, für Audio-Daten ist es die Sample-Größe, für allgemeine Daten die angegebene Größe. Diese Größe bleibt bei unkomprimierten Datenströmen konstant. Für komprimierte Datenströme wird hingegen die Größe einer ADU deutlich variieren, so daß eine Beschreibung durch den Mittelwert und die beiden Extrema erfolgt.

- ADU-Rate
  Für Video ist die ADU-Rate mit der Bildrate gleichzusetzen, da eine ADU genau einem Videobild entspricht. Die zugehörige Dienstklasse ist deterministisch. Für Audiodaten ergibt sich als ADU-Rate die Sample-Rate, ebenfalls mit einer deterministischen Verpflichtung. Für Graphik und Daten ist im allgemeinen die Dienstklasse bestmöglich, weshalb eine Vorgabe der Rate wenig Sinn macht. Ist eine *Zwischenankunftszeit* spezifiziert, so kann ihr Kehrwert als Anhaltspunkt für eine Rate eingesetzt werden. Weiterhin ist es möglich, daß für diese Medientypen eine spezifizierte maximale Ende-zu-Ende Verzögerung garantiert werden soll. Dann entspricht die Rate dem Kehrwert dieser Ende-zu-Ende Verzögerung (nähere Erläuterung dazu in Abschnitt 3.5.2.1).

- ADU-Verzögerung
  Die Verzögerung wird aus den bereits an der anwendungsorientierten Schnittstelle spezifizierten Werten (Zeitbeziehung und Verzögerung) bestimmt. Dabei entspricht der Zeitbeziehung *interaktiv* eine statistische, der Zeitbeziehung *Realzeit* eine deterministische Dienstklasse und der Zeitbeziehung *asynchron* eine Dienstklasse vom Typ „bestmöglich“.

- ADU-Jitter
  Die maximal zulässige Verzögerungsschwankung ergibt sich aus dem Medientyp. Die Begrenzung des Delay-Jitters spielt nur eine Rolle für kontinuierlichen Medientypen, wie Video oder Audio. Es handelt sich dabei um vorwiegend empirisch ermittelte Werte der deterministischen Dienstklasse. Für die diskontinuierlichen Medientypen genügt eine „bestmögliche“ Diensterbringung.

- ADU-Zuverlässigkeit
  Die benötigte Zuverlässigkeit variiert stark mit den Medientypen und der darin enthaltenen Redundanz. Für unkomprimierte Datenströme gilt: deterministisch für Graphik und Daten mit totaler Zuverlässigkeit (1,0,0) und statistisch für Video- und Audiodaten.

Die medientyp-inhärente Abbildung ist ausführlicher in [Henr96] behandelt. Die Festlegung von Schwellwerten erfolgt durch den QoS-Manager erst, nachdem die Parameter ausgehandelt wurden. Dabei werden die Schwellwerte empirisch ermittelt. Beispielsweise kann ein oberer Schwellwert um p Prozent (etwa mit $p \in [5, 10]$) unterhalb der oberen Grenze liegen:

$$\text{upper\_threshold(x)} = x_{\max} - \frac{p}{100} \cdot (x_{max} - x_{min})$$

3.5.2 Systeminterne Abbildung

In den folgenden Abschnitten wird erläutert, wie die einzelnen Parameter für Anwendungsdateneinheiten (ADUs) auf Parameter für interne Dateneinheiten (IDUs) abgebildet werden. Die Schwellwerte für IDU-basierte Parameter werden analog zu denen der ADU-basierten Parameter behandelt, aber nicht abgebildet.

Variable	Bedeutung	gegeben	berechenbar
s_A	Größe der ADU in bytes	×	
r_A	ADU-basierte Rate in ADUs/s	×	
d_A	ADU-basierte Ende-zu-Ende Verzögerung	×	
thr_A	ADU-basierte Durchsatz in bytes/s		×
s_h	Größe des Protokollkopfes in bytes	×	
s_I	Größe der IDU in bytes	×	
r_I	IDU-basierte Rate in IDUs/s		×
d_I	IDU-basierte Ende-zu-Ende Verzögerung		×
n_I	Anzahl der IDUs, die einer ADU entsprechen		×
thr_I	IDU-basierte Durchsatz in bytes/s		×
IT_A	Zwischenankunftszeit der ADUs		×
IT_I	Zwischenankunftszeit der IDUs		×
T_V	Verarbeitungszeit im KSS (SAR)	×	
pkthr_N	maximale Durchsatzanforderung an Netzwerk		×
pkthr_A	maximale Durchsatzanforderung, ADU-basiert		×

Tabelle 3.4: Bedeutung der benutzten Variablen

Die Größen in der Tabelle 3.4 werden – soweit nicht anders angegeben – als konstant betrachtet. Variable Größen können durch die drei Parameter Minimum (min), Durchschnittswert (avg) und Maximum (max) charakterisiert werden. So kann eine veränderliche Größe der Anwendungsdateneinheit (ADU) durch das Tripel ($s_A^{\min}$, s_A^{avg}, $s_A^{\max}$) genauer beschrieben werden. Es gilt für die Zwischenankunftszeit $IT_A := \frac{1}{r_A}$ und den Durchsatz $\text{thr}_A := r_A \cdot s_A$, bzw. $IT_A^{\min} = \frac{1}{r_A^{\max}}$ und $IT_A^{\text{avg}} = \frac{1}{r_A^{\text{avg}}}$, sowie $thr_A^{\max} := r_A^{\max} \cdot s_A^{\max}$ usw. Unter der Voraussetzung, daß die Größe s_I einer IDU und die Größe s_h des Protokollkopfes konstant sind, errechnet sich Anzahl n_I der Segmente bei konstanter Größe s_A zu $n_I = \lceil \frac{s_A}{s_I - s_h} \rceil$. Die Verarbeitungszeit T_V im Kommunikationssubsystem ist von n_I abhängig und kann bei Sender und Empfänger aufgrund unterschiedlicher Systemleistung und -belastung durchaus verschieden sein (Sender: T_V^a und Empfänger: T_V^b). Zur Vereinfachung soll die Verarbeitungszeit als konstant angenommen werden. Zur weiteren Vereinfachung kann in den folgenden Gleichungen der Term $T_V^a + T_V^b$ durch $2T_V$ abgeschätzt werden, da die Verarbeitungszeit des Empfängers beim Sender nur selten bekannt sein wird. Die Verarbeitungzeit verteilt sich bei einer Fließbandverarbeitung im Kommunikationssubsystem nahezu gleichmäßig auf die einzelnen Segmente (IDUs). Für die folgenden Berechnungen ist jedoch nur die gesamte Verarbeitungszeit T_V von Interesse.

3.5.2.1 Durchsatz

Zunächst gilt für die Rate:

$$r_I = \lceil \frac{s_A}{s_I - s_h} \rceil \cdot r_A = n_I \cdot r_A \tag{3.1}$$

woraus sich unmittelbar der Durchsatz thr_I ergibt:

$$\text{thr}_I = s_I r_I = s_I n_I r_A \tag{3.2}$$

Die Verarbeitungszeit T_V im Kommunikationssubsystem kann hierbei unberücksichtigt bleiben, wenn man davon ausgeht, daß der Netzwerkadapter die Zentraleinheit zum Senden und Empfangen von Dateneinheiten nicht benötigt, weil dieser in der Regel über einen eigenen Prozessor und Zugriff auf einen Puffer für Dateneinheiten verfügt. Somit können Netzwerkadapter und Zentraleinheit weitgehend unabhängig voneinander operieren. Daher kann der Netzwerkadapter noch mit dem Versenden der letzten Segmente einer ADU beschäftigt sein, während das Kommunikationssubsystem schon eine neue ADU entgegennimmt und verarbeitet. Somit bleibt für das Versenden der n_I IDUs tatsächlich die gesamte Zeit $IT_A = 1/r_A$ (siehe Abbildung 3.15).

Der Vollständigkeit halber sei noch folgender Fall betrachtet: benötigt ein einfacherer Netzwerkadapter die Zentraleinheit für das Versenden und Empfangen von Dateneinheiten, muß die Verarbeitungszeit T_V berücksichtigt werden. Es stünde somit weniger Zeit als $1/r_A = IT_A$ zur Verfügung, um die n_I Segmente zu verschicken oder zu empfangen. Daher ist also von der Zeitspanne $1/r_A$ noch die Verarbeitungszeit T_V abzuziehen. Um r_I für diesen Fall zu erhalten, ist (analog zu Gleichung 3.1) n_I wieder mit einer Rate zu multiplizieren. Dazu wird statt r_A nun der Kehrwert von $\frac{1}{r_A} - T_V$ als Faktor eingesetzt, bzw. man dividiert n_I durch $\frac{1}{r_A} - T_V$ und erhält $r_I = n_I \cdot \frac{1}{\frac{1}{r_A} - T_V} = n_I \cdot \frac{r_A}{1 - r_A T_V}$.

Die Berechnung der Rate r_I nach Gleichung 3.1 erfolgte ohne die Berücksichtigung einer Anforderung an die maximale Ende-zu-Ende Verzögerung $d_A^{\max}$. Diese kann nämlich unter Umständen eine andere Berechnung für r_I erfordern. Bei genauerer Betrachtung müssen daher zwei Fälle unterschieden werden (siehe dazu auch Abbildung 3.15):

Fall 1: $d_A^{\max} < 1/r_A$

Die Senderate ist „niedrig“, die maximal erlaubte Ende-zu-Ende Verzögerung ist jedoch noch geringer als die Zwischenankunftszeit der ADUs. Damit muß eine ADU mit einer höheren Geschwindigkeit übertragen werden, als die Rate r_A allein vorgibt. Der Verkehr zeigt daher eine „burstartige“ Natur, d.h. eine kurze Zeitspanne in der Daten übertragen werden und darauffolgend eine längere Zeitspanne ohne Datenübertragung. Es folgt:

$$\mathrm{pkthr}_A \geq s_A^{\max} \cdot \frac{1}{d_A^{\max} - (T_V^a + T_V^b)} \tag{3.3}$$

und daraus

$$\mathrm{pkthr}_N \geq s_I \cdot n_I^{\max} \frac{1}{d_A^{\max} - (T_V^a + T_V^b)} \tag{3.4}$$

pkthr_A (für „peak throughput“) bezeichnet dabei die ADU-basierte Durchsatzanforderung, die erfüllt werden muß, um die vorgegebene Verzögerung von $d_A^{\max}$ überhaupt einhalten zu können. Man beachte, daß sich die *Durchsatzanforderung* pkthr_A *aus der Verzögerungsanforderung* berechnet und daher nicht zwangsläufig mit $\mathrm{thr}_A^{\max} = 1/r_A^{\max}$ identisch ist, welches die maximale *Durchsatzanforderung der Anwendung* bezeichnet. pkthr_N bezeichnet dementsprechend die durch die Verzögerung implizierte IDU-basierte Durchsatzanforderung gegenüber dem Netzwerk.

Fall 2: $d_A^{\max} \geq 1/r_A$

Die Senderate ist hoch und die maximal erlaubte Ende-zu-Ende Verzögerung ist größer als die Zwischenankunftszeit der ADUs. Die Verzögerung stellt damit keine höhere Anforderung an den Durchsatz und somit gilt Gleichung 3.1.

3.5.2.2 Ende-zu-Ende Verzögerung

Wenn $n_I \equiv 1$, also kein Segmentieren notwendig ist, ergibt sich die einfache Abbildungsvorschrift $d_I := d_A - (T_V^a + T_V^b)$. Muß aber segmentiert werden, reicht diese einfache Abbildung nicht mehr aus. Mit Hilfe von Abbildung 3.15 läßt sich die gewünschte Größe d_I ermitteln. Eine Anwendung A möchte mit konstanter Rate r_A gleichgroße Dateneinheiten (ADUs) senden. Dazu reicht sie diese im Zeitabstand IT_A

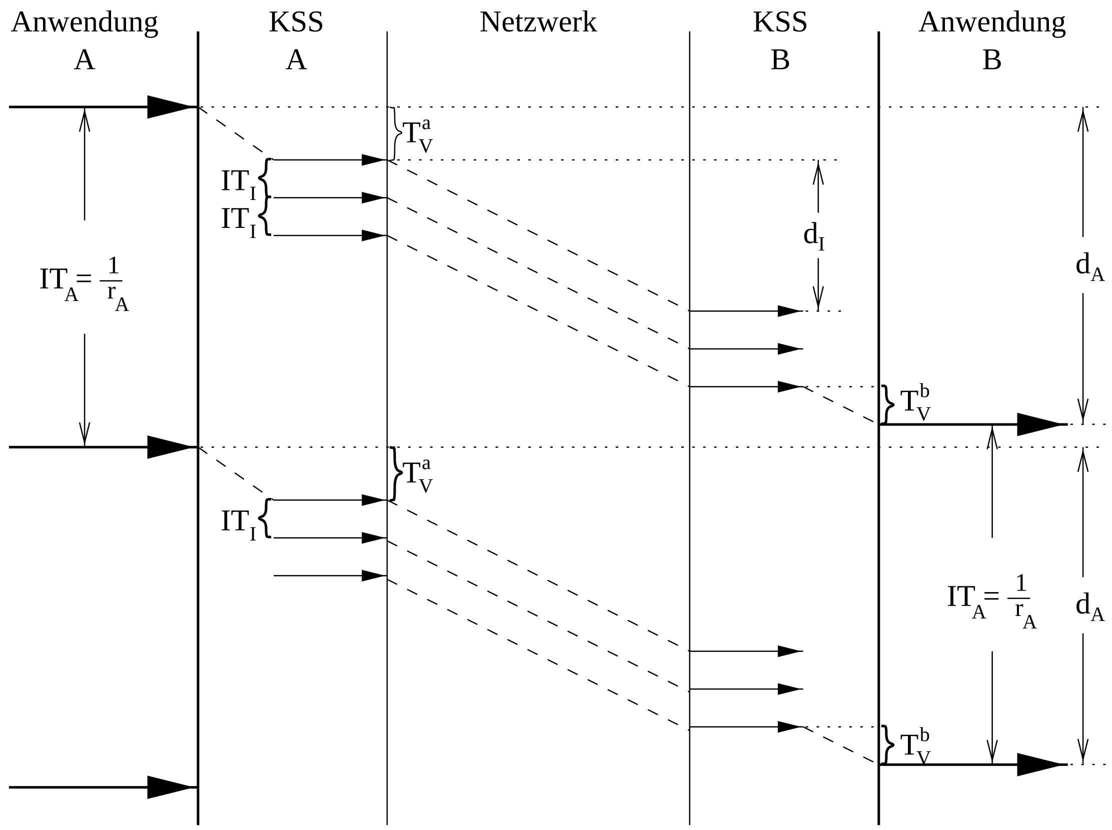

Abbildung 3.15: Zur Abbildung der Ende-zu-Ende Verzögerung

an das Kommunikationssubsystem (KSS) über die entsprechende Dienstschnittstelle weiter. Das Kommunikationssubsystem muß nun die eine ADU in mehrere IDUs zerlegen. Es wird angenommen, daß das Segmentieren und die übrige Protokollverarbeitungszeit zusammen die Zeit T_V^a beanspruchen (bei einer Fließbandverarbeitung ist diese Zeit auf die Segmente verteilt). Nach Zerlegung und Füllen der Protokollinformationsfelder im Kopf der IDU, werden die n_I IDUs mit maximaler Rate an das Netzwerk übergeben (als „Burst“ mit dem Durchsatz pkthr$_N$, IT_I soll hier nicht von r_A wie in Gleichung 3.1 abgeleitet werden). Die erste IDU erreicht das Kommunikationssubsystem B nach der Übertragungsverzögerung d_I. Erfahren die restlichen $(n_I - 1)$ IDUs die gleiche Verzögerung, so benötigen sie die zusätzliche Zeit $(n_I - 1) \cdot IT_I$ und die Zeit T_V^b zum Reassemblieren sowie für die übrige Protokollfunktionalität. Damit ergibt sich Gleichung 3.5:

$$d_A = d_I + (n_I - 1) \cdot IT_I + (T_V^a + T_V^b) \tag{3.5}$$

woraus sich direkt die Gleichung für d_I ableiten läßt:

$$d_I = d_A - (n_I - 1) \cdot IT_I - (T_V^a + T_V^b) \tag{3.6}$$

$$d_I = d_A - (n_I - 1) \cdot \frac{s_I}{\text{pkthr}_N} - (T_V^a + T_V^b) \tag{3.7}$$

Problematisch wird es im Fall $d_A^{\max} < 1/r_A$ aus Abschnitt 3.5.2.1, da dann zwei unbekannte in Gleichung 3.7 zu bestimmen sind: d_I und pkthr_N. Ein höherer anzufordernder Spitzendurchsatz wird dem Netzwerk eine größere Verzögerung d_I zugestehen. Es läßt sich allerdings eine untere Schranke für pkthr_N angeben:

$$\mathrm{pkthr}_N \geq s_I \cdot n_I \frac{1}{d_A - (T_V^a + T_V^b)} \tag{3.8}$$

Nutzt man die zur Verfügung stehende Zeitspanne $d_A - (T_V^a + T_V^b)$ für ein gleichmäßiges Versenden der IDUs voll aus, so erhält man diese untere Schranke. Dieser Durchsatz wird jedoch für die meisten Fälle noch zu gering sein, denn die einzuhaltende Verzögerung d_I liegt dann im Bereich der Zwischenankunftszeit IT_I. Dies läßt sich auch so zeigen: die untere Grenze der Bedingung 3.8 ($\mathrm{pkthr}_N = s_I \cdot n_I \frac{1}{d_A - (T_V^a + T_V^b)}$) in Gleichung 3.7 eingesetzt, ergibt $d_I = \frac{d_A - (T_V^a + T_V^b)}{n_I}$. Mit $IT_I = s_I / \mathrm{pkthr}_N$ und $\mathrm{pkthr}_N = s_I \cdot n_I \frac{1}{d_A - (T_V^a + T_V^b)}$ ergibt sich $IT_I = \frac{d_A - (T_V^a + T_V^b)}{n_I}$ und damit $d_I = IT_I$). Die untere Grenze der Bedingung 3.8 stellt daher eine zu hohe Anforderung an d_I, weshalb hier nur eine Heuristik weiterhilft. Eine initiale Anfrage beim Netzwerk kann mit dem Wert $d_I = k \cdot d_A$ erfolgen, wobei $0 < k < 1$, also z.B. $k = 0.8$, sowie einem entsprechenden Spitzendurchsatz pkthr_N. Können diese Werte nicht erfüllt werden, so kann anhand der vom Netzwerk zurückgelieferten möglichen Werte und Kosten eine Nachverhandlung die passenden Werte finden.

3.5.2.3 Verzögerungsschwankung

Eine Unterstützung für die Begrenzung der Verzögerungsschwankung im Netzwerk ist nicht zwingend notwendig, um eine maximale Verzögerungsschwankung für die ADUs zu garantieren, da „zu früh“ ankommende ADUs vom Kommunikationssubsystem in Puffern zwischengespeichert werden können und eine maximale Ende-zu-Ende Verzögerung wie eben beschrieben vom Netzwerk garantiert werden kann. Nachteilig ist jedoch der eventuell hohe Pufferbedarf im Endsystem. Daher gibt es Ansätze, um bereits die Verzögerungsschwankung im Netzwerk zu begrenzen [ZaFe93, VeZF91]. Soll diese Netzwerkunterstützung in Anspruch genommen werden, ist daher zu einem $d_I^{\max}$ noch ein $d_I^{\min}$ zu bestimmen. Dafür ist Gleichung 3.6 mit den entsprechenden Größen zu verwenden:

$$d_I^{\min} = d_A^{\min} - (n_I - 1) \cdot IT_I - (T_V^a + T_V^b) \tag{3.9}$$

3.5.2.4 Zuverlässigkeit

Die Zuverlässigkeitsspezifikation (i, m, c) aus 2.1.2 bezieht sich auf Anwendungsdateneinheiten, abweichend von [GoPa92], wo sich die Angabe auf Dateneinheiten des Transportsystems bezieht und somit die Anwendung Kenntnis von der Segmentierung erhalten muß, was hier jedoch vermieden werden soll. Die Abbildung soll daher die Umsetzung der Zuverlässigkeitsspezifikation von Anwendungsdateneinheiten auf interne Dateneinheiten leisten. Dies läßt sich allerdings nicht ohne Probleme bewerkstelligen, wenn man die Semantik der Zuverlässigkeitsspezifikation für IDUs übernimmt. Daher wird eine *neue* IDU-basierte Zuverlässigkeitsspezifikation eingeführt.

Zunächst sollen die Probleme aufgezeigt werden, die sich ergeben, wenn man versucht, die auf ADUs bezogene Zuverlässigkeitssemantik auf IDUs zu übertragen. Falls keine Segmentierung der ADUs notwendig ist, ließe sich auch unter Beibehaltung der Zuverlässigkeitssemantik abbilden, denn dann entspricht eine IDU genau einer ADU. Hier zeigt sich noch kein Problem. Im Falle einer Segmentierung der ADUs ist zu berücksichtigen, daß das Kommunikationssubsystem und das unterliegende Netzwerk auf IDUs operieren. Die „Gleichbehandlung" der IDUs im Netzwerk läßt keine Zuordnung mehr zu den ADUs erkennen. Die konkrete Verteilung der fehlerhaften IDUs bestimmt aber die Anzahl der fehlerhaften ADUs. So können zwei fehlerhafte IDUs, die direkt aufeinanderfolgen, den Verlust einer oder zweier ADUs (letztes Segment einer ADU und direkt darauffolgendes Segment fehlerhaft) bedeuten. Daher muß die IDU-basierte Zuverlässigkeitsspezifikation stärker auf die Verteilung der fehlerhaften IDUs eingehen. Das betrachtete Intervall von i aufeinanderfolgenden ADUs kann zuerst problemlos auf ein Intervall von $i \cdot n_I$ IDUs abgebildet werden. Die Werte m und c des Tripels (i, m, c) kann man dagegen nicht so einfach mit n_I multiplizieren, da der Verlust oder die Verfälschung eines Fragments (d.h. einer IDU) ohne Fehlerbehebungsmechanismen den Verlust einer ganzen ADU nach sich zieht. Es soll stattdessen eine Abschätzung der eventuell betroffen ADUs vorgenommen werden: Da maximal c ADUs hintereinander verloren gehen dürfen, kann der Verlust von mehr als $(c-1) \cdot n_I + 1$ *aufeinanderfolgender IDUs* nicht mehr toleriert werden, weil ansonsten im ungünstigsten Fall schon mehr als c ADUs betroffen sind. Nun bleibt noch m abzubilden, die maximale Anzahl der fehlerhaften oder verlorenen ADUs unter i aufeinanderfolgenden. Da der Verlust einer IDU bereits den Verlust einer ganzen ADU zur Folge haben kann, würde die restriktivste Abbildung auch insgesamt höchstens m Verluste von IDUs zulassen, denn diese liegen ungünstigstenfalls stark verteilt innerhalb der $i \cdot n_I$ IDUs, d.h. alle mit einem Abstand von mindestens n_I zueinander, wodurch jeweils eine andere ADU betroffen ist. Eine praktikablere Lösung besteht darin, höchstens $\lfloor \frac{m}{c} \rfloor$ solcher „Fehlerbursts" von $(c-1) \cdot n_I + 1$ verlorenen Dateneinheiten zuzulassen. Damit ergibt sich folgende Abbildung:

$$(i, m, c)_A \longmapsto (n_I \cdot i, \lfloor \frac{m}{c} \rfloor, (c-1) \cdot n_I + 1)_I =: (I, MB, EBL)_I \tag{3.10}$$

Es ist insbesondere die unterschiedliche *Semantik* beider Spezifikationen zu beachten: das erste Tripel $(i, m, c)_A$ bedeutet, daß unter i unmittelbar aufeinanderfolgenden Dateneinheiten höchstens m fehlerhaft sein dürfen und maximal c fehlerhafte unmittelbar hintereinander folgen dürfen. Das zweite Tripel $(I, MB, EBL)_I$ bedeutet hingegen: Unter I unmittelbar aufeinanderfolgenden Dateneinheiten dürfen höchstens EBL hintereinander fehlerhaft sein (maximale Länge eines „Fehlerbursts") und es dürfen höchstens MB solcher „Bursts" unter den I Dateneinheiten vorkommen.

Eine einzelne fehlerhafte Dateneinheit wird als Beginn eines Bursts und damit zu MB gezählt. Weitere Einzelfehler innerhalb der EBL Dateneinheiten ab dem ersten Einzelfehler können dabei ebenfalls toleriert werden, aber Einzelfehler, die weiter als $EBL - 1$ Dateneinheiten auseinanderliegen, werden dann separat zu MB gezählt. Damit geht die Definition für die Zuverlässigkeit der IDUs gut auf das für die Hochleistungskommunikation typische „burst-artige" Fehlerverhalten (siehe [Zitt95]) ein.

3.5.3 Rückabbildung

Die Rückabbildung wird nötig, da während der Aushandlung – und auch später während des Betriebs – Komponenten die angeforderten systemorientierten Parameterwerte herabsetzen können. Die Rückabbildung von IDU-basierten auf ADU-basierte Parameter ist mit den in Abschnitt 3.5.2 vorgestellten Methoden möglich.

Mehr Schwierigkeiten bereitet die Rückabbildung von ADU-basierten auf anwendungsorientierte Parameter. Diese Rückabbildung ist für solche Fälle nicht trivial, bei denen z.B. mehrere Parameter auf Anwendungsebene in einen Parameter auf Systemebene abgebildet werden. Beispiel hierfür ist der in Abschnitt 3.5.1 angeführte unkomprimierte Videostrom, bei dem die Bildgröße und die Farbtiefe zusammen die ADU-Größe ausmachen und gemeinsam mit der ADU-Rate direkten Einfluß auf den benötigten Durchsatz haben. Ändert sich nun der Durchsatz eines Datenstroms, muß der QoS-Manager entscheiden, welche anwendungsorientierten Parameter angepaßt werden. Dies geschieht in einem iterativen Prozeß, bei dem in jedem Schritt ein anwendungsorientierter Parameter verändert (je nach Parameter kontinuierlich oder diskret in festgelegten Schritten, sogenannten *Degradierungsstufen*) und anschließend eine Diensthinabbildung ausgeführt wird, die den zugehörigen Wert des betrachteten Systemparameters berechnet. Diese „Skalierung" der anwendungsorientierten Parameter wird solange fortgesetzt, bis der angestrebte Wert des Systemparameters erreicht oder „unterschritten" wird (es wird sich nicht immer genau der angestrebte Wert erreichen lassen). Die Auswahl des zu skalierenden anwendungsorientierten Parameters geschieht über *Gewichtsfunktionen*, die das Ziel verfolgen, den angestrebten Durchsatzgewinn durch eine möglichst gleichmäßige Qualitätsherabsetzung aller betroffenen anwendungsorientierten Parameter zu erreichen. Die Gewichtsfunktionen sind ausführlich in [Henr96] diskutiert, weshalb sie im Rahmen dieser Arbeit nur kurz vorgestellt werden.

Um eine „gleichmäßige" Skalierung der beteiligten Parameter zu erreichen, muß ein relatives Maß für die Qualität des Parameters existieren. Dieses Maß wird durch das *Qualitätsgewicht* ausgedrückt. Es setzt den momentanen Parameterwert x zum minimalen ($x_{\min}$) und maximalen Parameterwert ($x_{\max}$) in Relation:

$$q_w(x) = \frac{x_{\max} - x_{\min}}{x - x_{\min}}$$

Das zweite Gewicht dient zur Beurteilung des „Durchsatzgewinns" bei Herabsetzung des Parameters um eine Degradierungsstufe. Es setzt den momentan erreichbaren

Durchsatzgewinn $\text{thr}_g(x)$ (throughput gain) in Relation zum gewünschten Durchsatzgewinn (thr_t) (throughput target):

$$\text{thr}_w(x) = \frac{\text{thr}_A - \text{thr}_t}{\text{thr}_g(x)}$$

thr_A bezeichnet den Durchsatz während der Skalierung.

3.6 Die Interne QoS-Struktur

Die interne QoS-Struktur (IQS) enthält alle für das Management relevanten Parameter einer Sitzung. Dazu zählen insbesondere die Dienstspezifikationen der verschiedenen Datenströme. Es existiert eine separate interne QoS-Struktur für jede Sitzung und pro Sitzung eine Teilstruktur je Datenstrom (s. Abbildung 3.16). In

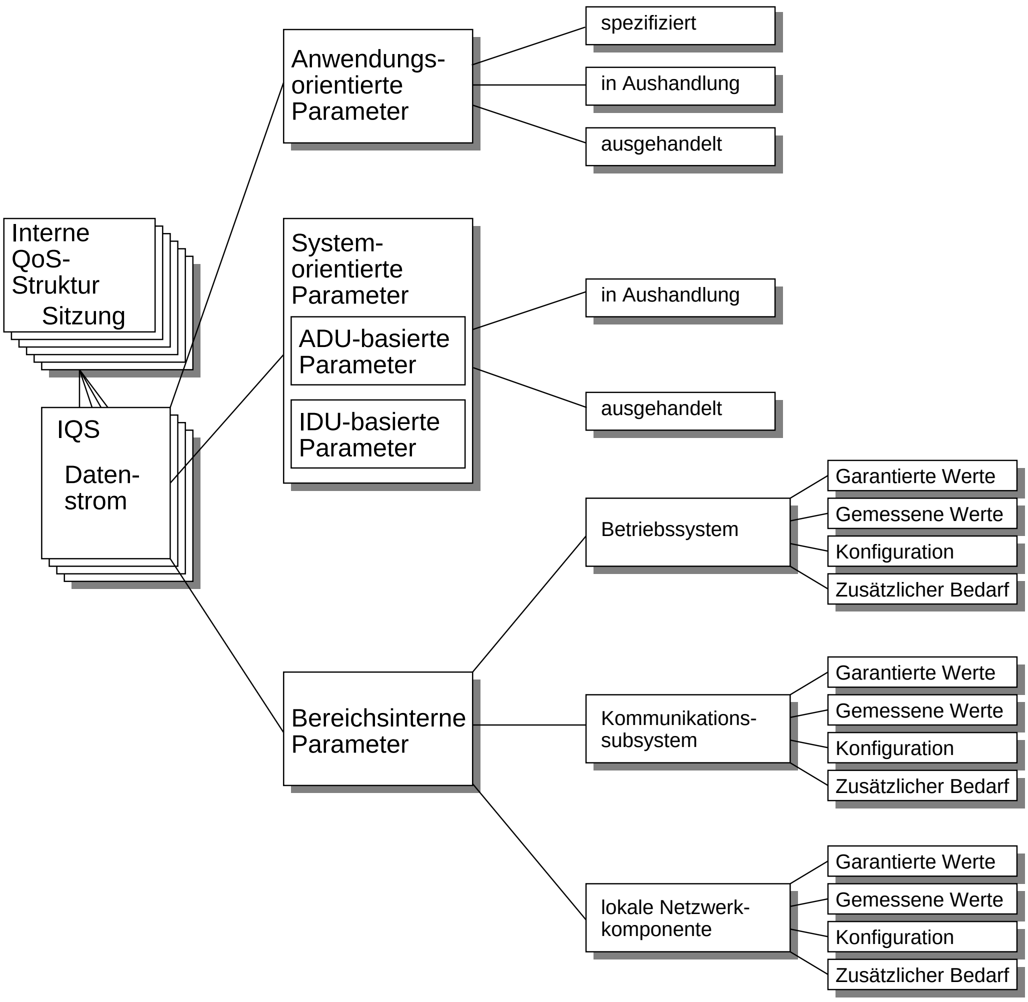

Abbildung 3.16: Grundsätzlicher Aufbau der internen QoS-Struktur

den folgenden Abschnitten werden die einzelnen Bestandteile der IQS vorgestellt und erläutert. Zuerst wird die interne *Struktur einer Sitzung* beschrieben. Im darauffolgenden Abschnitt wird die interne *Struktur eines Datenstroms* erläutert, die sich in die Teilstrukturen für *anwendungsorientierte Parameter*, *systemorientierte Parameter*, und *bereichsinterne Parameter* untergliedert.

3.6.1 Sitzungsstruktur

Jede Spezifikation einer Sitzung besteht aus einer Liste von mehreren Datenstromspezifikationen und einer Beschreibung der Interstrombeziehungen. Die Datenstrom-

Sitzung	
Datenströme	Datenstrom A
	$\vdots$
	Datenstrom N
Strombeziehungen	Strombeziehung 1 = Datenstrompaar, Beziehung
	$\vdots$
	Strombeziehung m = Datenstrompaar, Beziehung

Tabelle 3.5: Struktur einer Sitzung

struktur wird in Abschnitt 3.6.2 erläutert. Die *Strombeziehungen* werden durch eine Liste von paarweisen Strombeziehungen angegeben, d.h. es wird jeweils die Relation zweier Datenströme zueinander beschrieben. Ein Paar (X, Y) gibt dabei an, daß die angegebene *Beziehung* für die Datenströme X und Y gilt. Man beachte dabei, daß die Spezifikation des Datenstrompaares (X, Y) eine unsymmetrische Relation darstellt (also $(X, Y) \neq (Y, X)$!). Eine Beziehung kann *zeitlicher* oder *logischer* Art sein. Für jedes Datenstrompaar können mehrere Beziehungen definiert sein, das heißt, es können eine zeitliche und mehrere logische Beziehungen gleichzeitig definiert sein. Eine Beziehung wird durch ihre *Art* und ein *Attribut* spezifiziert. Die Art ist dabei entweder *logisch* oder *zeitlich*. Für logische Beziehungen existieren folgende Ausprägungen, die durch das zugehörige *Attribut* angegeben werden:

- Lebenszeitrelation
  Die Lebenszeit des Datenstromes Y hängt von der Lebenszeit des Datenstromes X ab. Das heißt, wenn Datenstrom X beendet wird, so wird auch Y beendet (deshalb auch $(X, Y) \neq (Y, X)$). Die Lebenszeitrelation berücksichtigt die Situationen, in denen die Existenz eines Datenstroms ohne den anderen sinnlos ist. Als Beispiel sei ein Videostrom angeführt, der ohne den zugehörigen Audiostrom wenig Sinn macht (der umgekehrte Fall – Audio ohne Video – ist dagegen in vielen Fällen akzeptabel). Denkbar ist auch, daß ein Datenstrom durch einen anderen gesteuert wird. Fällt der steuernde Datenstrom aus, ist der gesteuerte Datenstrom in vielen Fällen alleine nicht mehr sinnvoll.

- Identische Teilnehmerstruktur
  Datenstrom Y erhält die gleiche Kommunikationsbeziehung (also Quelle, Senken und Gruppensemantik) wie Datenstrom X. Dies ist zur Vereinfachung der Spezifikation vorgesehen.

- Identische Dienstqualität
  Datenstrom Y soll die gleiche Spezifikation der Dienstqualität erhalten, wie Datenstrom X. Gleichartige Datenströme mit unterschiedlicher Teilnehmerstruktur sind somit einfach anzugeben.

Eine Beziehung zeitlicher Art beschreibt die Synchronisation zweier Datenströme zueinander. Daher enthält das *Attribut* der zeitlichen Beziehung die Angabe über den erlaubten maximalen *zeitlichen Versatz* beider Datenströme zueinander (das sogenannte *Skew-Intervall*). Ein typisches Beispiel für die Notwendigkeit der zeitlichen Synchronisation zweier Datenströme zueinander ist die *Lippensynchronisation*: Der Ton muß rechtzeitig und damit passend zur Lippenbewegung einer sprechenden Person wiedergegeben werden. Das Skew-Intervall $[\alpha, \beta]$ spezifiziert dabei die Grenzen für den erlaubten Versatz, das heißt die Differenz $D_X - D_Y$ der Ende-zu-Ende Verzögerung beider Datenströme muß innerhalb von $[\alpha, \beta]$ liegen. Bei der Lippensynchronisation sind das z.B. $[-80\text{ms}, +80\text{ms}]$, bei einer getrennten Übertragung zweier Stereokanäle lediglich $[-0.011\text{ms}, +0.011\text{ms}]$ (siehe [BlSt96]). Das Skew-Intervall steht mit der Ende-zu-Ende Verzögerung und der Verzögerungsschwankung der zu synchronisierenden Dateneinheiten in einer engen Beziehung. Das Skew-Intervall stellt Bedingungen für die erlaubten Werte der Ende-zu-Ende Verzögerung der beiden Datenströme [Henr96].

3.6.2 Datenstromstruktur

Die Struktur ist in drei Bereiche unterteilt: In die *anwendungsorientierten* Parameter, in die *systemorientierten* Parameter und in die *bereichsinternen* Parameter.

Die Struktur für die *anwendungsorientierten* Parameter besteht wiederum aus drei Teilen. In dem ersten Teil werden die durch die Anwendung *spezifizierten* Werte gespeichert. Im zweiten Teil werden Parameterwerte während der *Aushandlung* abgelegt und im dritten Teil, die zuletzt *ausgehandelten* Werte festgehalten. Die durch die Anwendung spezifizierten Werte werden nicht verändert und dienen z.B. bei einer Rekonfiguration, als Ausgangsbasis. Während einer Aushandlung werden die Werte ebenfalls festgehalten und können durch die verschiedenen Komponenten, die an der Aushandlung beteiligt sind, verändert werden. Kommt die Aushandlung zu einem Abschluß, werden die ausgehandelten Parameter im dritten Teil gesondert festgeschrieben und bleiben dort solange unverändert, bis das Ergebnis einer erneuten erfolgreichen Aushandlung dort übernommen wird.

Die systemorientierten Parameter beschreiben den eigentlichen Datenstrom auf unterschiedlichen Ebenen: auf Anwendungsebene bezogen auf Anwendungsdateneinheiten (ADUs – Application Data Units), auf Netzwerkebene bezogen auf systeminterne Dateneinheiten (IDUs – Internal Data Units), die durch Segmentierung der Anwendungsdateneinheiten entstehen. Die Dienstgüteabbildung aus 3.5 sorgt für eine Umsetzung der Werte zwischen diesen beiden Parametersätzen. Auch hier existieren Strukturen für beide Parametersätze, wovon eine die Werte während der Aushandlung und die andere ausgehandelte Werte enthält.

Schließlich spiegeln die bereichsinternen Parameter die Konfiguration der drei Bereiche Betriebssystem, Kommunikationssubsystem und lokale Netzkomponente wider. Pro Bereich werden die von ihm zugesagten, d.h. garantierten Werte, sowie die gemessenen Werte gespeichert. Die Meßwerte liefert der Monitor der Kontrollinstanz entsprechend der gewählten Meldungsstrategie. Die Konfiguration der Komponente wird ebenfalls in einer Teilstruktur festgehalten. Schließlich werden die von dem Bereich zusätzlich benötigten Werte gespeichert, beispielsweise die von einer Komponente benötigte Verarbeitungszeit oder zusätzlichen Durchsatz.

3.6.2.1 Anwendungsorientierte Parameter

Die anwendungsorientierten Parameter sind in Tabelle 3.6 angegeben. Der Benutzer hat allerdings die Möglichkeit auf Standardbelegungen zurückzugreifen und unter vordefinierten Werten auszuwählen (*selektive* Parameter), so daß sich im allgemeinen die Spezifikation auf wenige Angaben beschränkt (eine ausführliche Beschreibung ist in [Henr96] zu finden). Beispielsweise kann die Angabe der Audioqualität durch eine Auswahl unter den vordefinierten Standardbelegungen Telefon, Sprache, CD-mono oder CD-stereo erfolgen. Die vordefinierten Standardbelegungen werden dann über *Anwendungsprofile* auf die im folgenden beschriebenen Werte abgebildet [Henr96].

Zunächst ist die Angabe eines *Medientyps* notwendig. Zur Auswahl stehen vier Klassen: Audio, Video, Graphik und Text bzw. allgemeine Daten. Für jede Klasse sind die in Tabelle 3.6 angegebenen *medienspezifischen* Parameter anzugeben (auch indirekt über die oben beschriebenen Standardbelegungen). Für alle Klassen existiert eine Auswahl an unterschiedlichen Kompressions- und Datenreduktionsverfahren. Die spezifischen Parameter für jedes Verfahren lassen sich ebenso angeben, wie *Intrastromprioritäten* für hierarchisch codierte Ströme (beispielsweise JPEG[2] oder MPEG[3]). Diese Intrastromprioritäten bezeichnen dann die Wichtigkeit der Daten der unterschiedlichen Hierarchiestufen, so daß z.B. im Falle eines Engpasses Daten bestimmter Hierarchiestufen entfallen bzw. gefiltert werden können. Audio und Video sind *kontinuierliche Medien* während hingegen Graphik und allgemeine Daten eine diskontinuierliche Natur aufweisen. Sollen jedoch auch mehrere Einheiten verschickt werden, so kann zusätzlich eine *Zwischenankunftszeit* dieser Einheiten angegeben werden, die zumindest eine Abschätzung des zu erwartenden Verkehrs erlaubt.

[2]JPEG – Datenkompressions- und -reduktionsverfahren für Bilddaten der Joint Photographic Expert Group

[3]MPEG – Verfahren der Motion Picture Expert Group für Bewegtbilder

Parameter	Zusammensetzung		
Medium	Audio	Audioqualität:	(Sample-Größe, Sample-Rate)
		Kanaltrennung:	mono | stereo
		Kompression:	(Verfahren, Intrastromprioritäten)
	Video	Bildrate:	(Minimum, Gewünscht)
		Bildgröße:	(Höhe, Breite, Seitenverhältnis)
		Farbtiefe:	(Minimum, Gewünscht)
		Kompression:	(Verfahren, Intrastromprioritäten)
	Graphik	Bildgröße:	(Höhe, Breite, Seitenverhältnis)
		Farbtiefe:	(Minimum, Gewünscht)
		Kompression:	Verfahren
		Zwischenankunftszeit:	(Minimum, Maximum)
	Text, allgemeine Daten	Größe:	(Minimum, Maximum)
		Kompression:	Verfahren
		Zwischenankunftszeit:	(Minimum, Maximum)
Zeitbeziehung	interaktiv | Realzeit | asynchron		
Verzögerung	(Minimum, Maximum)		
Zuverlässigkeit	(Intervallänge, Maximum, Maximaler Burst), Auslieferung		
Sicherheit	keine | Verschlüsselung | Authentifizierung | Zeitstempel		
Kommunikationsbeziehungen	(Sender, Empfängerliste, Gruppensemantik)		
Priorität	(Ordnungszahl, Relevanz)		
Meldungsverhalten	keine | periodische | Warnungen | Verletzungen		
Reaktionsverhalten	(Ereignis, Reaktion),. . . ,(Ereignis, Reaktion) Ereignis=Warnung | Verletzung | Fehler Reaktion=keine | Benachrichtigung | Rückfragen | Adaption | Abbau		
Kosten	Maximale Kosten		

Tabelle 3.6: Anwendungsorientierte Parameter

Für jeden Datenstrom ist eine *Zeitbeziehung* zu spezifizieren, welche die Ende-zu-Ende Verzögerung genauer beschreibt. Mit ihrer Hilfe kann aus den Anwendungsprofilen ein Wert und die Dienstklasse für die Ende-zu-Ende Verzögerung bestimmt werden. So stellen *interaktive* Anwendung eine starke Anforderung an die maximale Ende-zu-Ende Verzögerung (ca. 100ms [Part94, Tawb93]), allerdings reicht eine *statistische* Dienstklasse für diesen Parameter, da gelegentliche Überschreitungen toleriert werden können. Soll der Datenstrom jedoch *Realzeit*-Bedingungen genügen (z.B. ein Meß- oder Steuerdatenstrom für Aufgaben in der Regelungstechnik), ist die unter *Verzögerung* spezifizierte Schranke unbedingt einzuhalten, d.h. nur die *deterministische* Dienstklasse bietet eine ausreichende Garantie. Eine *asynchrone* Zeitbeziehung erlaubt eine beliebige Ende-zu-Ende Verzögerung und wird damit auf die Dienstklasse *bestmöglich* abgebildet. Die Angabe eines konkreten Intervalls für die Ende-zu-Ende Verzögerung ist prinzipiell nur für die Zeitbeziehung *Realzeit* notwendig.

Die Spezifikation der Werte für die *Zuverlässigkeit* (Definition in Abschnitt 2.1.2), leitet sich normalerweise aus dem konkreten Medientyp ab. Die Sensibilität gegenüber fehlerhaften oder fehlenden Anwendungsdateneinheiten hängt vom Medientyp und der in den Daten enthaltenen Redundanz ab. Datenkompressions- oder Datenreduktionsverfahren reduzieren die natürliche Redundanz der Daten und stellen damit stärkere Anforderungen an die Zuverlässigkeit als unkomprimierte Daten. Die Zuverlässigkeit wird – der Definition aus 2.1.2 folgend – durch ein Tripel spezifiziert: Die *Intervallänge* gibt die Anzahl der aufeinanderfolgenden Dateneinheiten an, auf die sich die *maximale Anzahl* der erlaubten Fehler/Verluste und die *maximale Anzahl der unmittelbar aufeinanderfolgenden* Fehler/Verluste bezieht. Zusätzlich kann angegeben werden, ob die Anwendung eine *Auslieferung fehlerhafter Dateneinheiten* wünscht (z.B. weil sie in der Lage ist, die Fehler selbst zu korrigieren oder die Fehler aufgrund der vorhandenen Redundanz akzeptiert werden können, wie beispielsweise etwa einige fehlerhafte Pixel bei unkomprimiertem Video).

Zum Schutz vor unautorisierten Zugriffen und „Mithören“ kann durch den Parameter *Sicherheit* (security) ein Sicherheitsmechanismus spezifiziert werden. Denkbar ist die Verwendung gängiger Verschlüsselungsverfahren wie RSA[4] und DES[5] oder die Authentifizierung durch Passwörter oder die Mitgabe von Zeitstempeln.

Die *Kommunikationsbeziehungen* beschreiben die Teilnehmerstruktur für diesen Datenstrom. Ein Datenstrom hat genau einen Sender und einen oder mehrere Empfänger (Multicast, 1:n-Kommunikationsbeziehung). Da der Initiator einer Aushandlung auch den Aufbau eines Datenstroms zwischen anderen Teilnehmern anfordern kann (er braucht also weder Sender noch Empfänger des Datenstroms zu sein), ist die Angabe der *Senderadresse* notwendig. Da der Datenstrom an mehrere Empfänger übermittelt werden kann, ist die Angabe einer Liste der *Empfängeradressen* vorgesehen. Die *Gruppensemantik* enthält zusätzliche Attribute über die Multicast-Gruppe. Beispielsweise kann damit angegeben werden, ob die Gruppe *homogen* (alle Teilnehmer erhalten die gleiche Dienstqualität) oder *heterogen* (die Teilnehmer können durchaus unterschiedliche Dienstqualitäten erhalten) sein soll. Ebenso kann eine Zuverlässigkeitssemantik vorgegeben werden, welche beschreibt, wie die Zuverlässigkeit in Bezug auf einzelne Teilnehmer zu interpretieren ist (z.B. Abbau des Datenstroms bei Ausscheiden eines Teilnehmers oder Unterschreiten einer gewissen Teilnehmeranzahl).

Die *Priorität* eines Datenstroms gibt an, wie der Datenstrom bei Problemen im Verhältnis zu anderen Datenströmen zu behandeln ist. Die Reihenfolge der Datenströme auf denen eine Aktion (beispielsweise die Herabsetzung der Dienstqualität) ausgeführt wird, ist durch eine *Ordnung* vorgegeben. Die *Ordnungszahl* eines Datenstroms bestimmt seinen Platz in der Ordnung. Zwei Datenströme (auch aus unterschiedlichen Sitzungen) dürfen die gleiche Ordnungszahl besitzen (die Ordnung ist

[4]asymmetrisches Verschlüsselungsverfahren mit öffentlichem Schlüssel
[5]Data Encryption Standard – symmetrisches Verschlüsselungsverfahren

also nicht notwendigerweise streng monoton). Außerdem erhält jeder Datenstrom einen Vermerk über seine Wichtigkeit für die Sitzung (*Relevanz*). Die Relevanz hat zwei mögliche Ausprägungen: *entbehrlich* oder *unentbehrlich*. Entbehrliche Datenströme können bei Engpässen wegfallen, ohne daß eine Aufgabe der Sitzung notwendig wäre. Anders wird hingegen bei unentbehrlichen Datenströmen verfahren: können sie nicht etabliert oder aufrechterhalten werden, so scheitert die gesamte Sitzung.

Durch die Angabe des *Meldungsverhaltens* wird für jeden Datenstrom festgelegt, welche Ereignisse an die Anwendung gemeldet werden sollen. Die Anwendung kann *keine* Meldungen wünschen, *periodische Meldungen* (Monitoring-Information) verlangen oder möchte nur im Falle von drohenden Verletzungen der Dienstqualität (*Warnungen*) oder bei Eintreten von *Verletzungen* informiert werden. Es können auch mehrere dieser Optionen gleichzeitig aktiviert sein, sofern ihre Kombination sinnvoll ist.

Das *Reaktionsverhalten* bestimmt, wie der QoS-Manager auf welches *Ereignis* reagieren soll. Daher wird das Reaktionsverhalten als Liste von Tupeln angegeben, die einem Ereignis jeweils eine Reaktion zuordnen. Mögliche Ereignisse sind: das Auftreten einer Warnung, die Meldung einer Verletzung der vereinbarten Dienstqualität oder das Auftreten eines Fehlers. Jedem Ereignis kann eine von fünf möglichen Reaktionen zugeordnet werden: das Eintreten des Ereignisses kann ignoriert werden (Einstellung: *keine* Reaktion) oder aber der Anwendung gemeldet werden (Einstellung: *Benachrichtigung*). Bei der Alternative *Rückfragen* wird die Anwendung über die Qualitätsverletzung informiert und auf eine Reaktion ihrerseits gewartet. Die Möglichkeit der *Adaption* beinhaltet eine automatische Neuverhandlung der Dienstqualität durch das QoS-Management. Erst wenn eine neue Dienstqualität für diesen Datenstrom ausgehandelt wurde, wird die Anwendung informiert. Die Option *Abbau* ist die restriktivste Maßnahme und leitet den Abbau des Datenstroms bei Eintreten des Ereignisses ein.

Unter *Kosten* kann eine obere Schranke für die Kosten vorgegeben werden. Zur Zeit besteht diese Schranke aus einer positiven Ganzzahl einer nicht näher festgelegten Einheit. Da die Kostenstruktur für diensteintegrierende Netzwerke momentan noch völlig unbekannt ist, kann der Angabe daher noch keine richtige Semantik zugeordnet werden. Allerdings würde ohne eine Kostenstruktur kein Benutzer die Veranlassung haben, eine andere Dienstqualität als die beste zu wählen.

3.6.2.2 Systemorientierte Parameter

Die systemorientierten Parameter beschreiben die Dienstspezifikation auf zwei verschiedenen Ebenen: einmal auf anwendungsorientierte Dateneinheiten (ADUs) und einmal auf systeminterne Dateneinheiten (IDUs) bezogen.

Die ADU-basierten Parameter werden durch medienspezifische Abbildungen gewonnen (siehe Abschnitt 3.5). Die anwendungsorientierten Dateneinheiten bestehen im

ADU-basierte Parameter	Zusammensetzung
ADU-Größe	(Minimum, Durchschnitt, Maximum)
ADU-Rate	(Minimum, Durchschnitt, Maximum), Schwellwerte, Dienstklasse
ADU-Verzögerung	(Minimum, Durchschnitt, Maximum), Schwellwerte, Dienstklasse
ADU-Jitter	Maximum, Schwellwert, Dienstklasse
ADU-Zuverlässigkeit	(Intervallänge, Maximum, Maximaler Burst), Schwellwerte, Dienstklasse
IDU-basierte Parameter	**Zusammensetzung**
IDU-Größe	Größe
IDU-Rate	(Minimum, Durchschnitt, Maximum), Schwellwerte, Dienstklasse
IDU-Verzögerung	(Minimum, Durchschnitt, Maximum), Schwellwerte, Dienstklasse
IDU-Jitter	Maximum, Schwellwert, Dienstklasse
IDU-Zuverlässigkeit	(Intervallänge, Maximale Anzahl Bursts, Maximaler Burst), Schwellwerte, Dienstklasse

Tabelle 3.7: Systemorientierte Parameter

Normalfall aus den kleinsten semantischen Einheiten, welche die Anwendung verarbeitet. Beispielsweise ein einzelnes Bild eines Films oder eine quantisierte Amplitude („Sample“) eines Audiokanals. Für die *ADU-Größe* kann ein minimaler, ein erwarteter Durchschnittswert und ein maximaler Wert vorgegeben werden. Damit können für das System auch in der Größe variierende Anwendungsdateneinheiten beschrieben werden, die z.B. durch die Anwendung von Kompressionsverfahren entstehen. Da die *Rate* in der anwendungsorientierten Dienstspezifikation als ein Bereich angegeben wird, der von der minimalen akzeptablen und der gewünschten Rate begrenzt ist, existieren für die ADU-basierten Parameter auch Werte für eine minimale und eine maximale Rate. Die Angabe eines Durchschnittswertes ist zwar nicht notwendig und kann aber der Ressourcenreservierung als Hinweis für die im Mittel bereitzustellenden Ressourcen dienen. Die zusätzliche Spezifikation von bis zu zwei *Schwellwerten* ermöglicht in Verbindung mit einem eingeschalteten Monitoring das Aktivieren eines Vorwarn-Mechanismus, der bei Überschreiten des Schwellwertes dem QoS-Management bzw. der Anwendung eine Warnungsmeldung zukommen läßt. Dadurch ist das Management oder die Anwendung frühzeitig in der Lage eine Anpassung (z.B. Neuverhandlung) durchzuführen [Bles95]. Die Spezifikation der Schwellwerte erfolgt durch das Management und erst nachdem konkrete Werte ausgehandelt wurden. Zu jedem Parameter wird noch eine *Dienstklasse* angegeben (deterministisch, statistisch, bestmöglich). Für die statistische Dienstklasse werden zusätzlich die beiden Parameter der gebrochenen Schranke (siehe Definition der statistischen Dienstklasse, Abschnitt 2.1.2, Seite 8) angegeben. Welche Dienstklasse für einen Parameter festgelegt wird, ist in der medientyp-inhärenten Abbildung (also den Anwendungsprofilen) enthalten. Für die *Ende-zu-Ende Verzögerung* der ADUs ist die gleiche Art der Spezifikation wie bei der Rate vorgesehen. Bei der *Verzögerungsschwan-*

kung hingegen (Jitter), macht nur die Vorgabe eines Maximalwertes Sinn, die untere Schranke ist per Definitionem 0 (Null). Für die Zuverlässigkeit wird das Tripel aus der anwendungsorientierten Dienstspezifikation (siehe Tabelle 3.6) übernommen. Die Dienstklasse wird wieder aus dem Medientyp abgeleitet. Zusätzlich können auch hier Schwellwerte angegeben werden.

Die IDU-basierten Parameter sind in Analogie zu den ADU-basierten Parametern definiert. Unterschiede existieren bei der IDU-Größe, die für ein ausgewähltes Netzwerk fest vorgegeben ist, und bei der Spezifikation der Zuverlässigkeit. Letztere wird im Abschnitt 3.5.2.4 detailliert beschrieben.

3.6.2.3 Bereichsinterne Parameter

Für jeden Bereich (Betriebssystem, Kommunikationssubsystem, Netzwerk) werden die von ihm *garantierten* und die tatsächlich *gemessenen* Parameterwerte abgelegt. Die Strukturen entsprechen denen aus dem anwendungsorientierten und dem systeminternen Teil. So wird das Betriebssystem auf anwendungsorientierten Parametern operieren, das Kommunikationssubsystem auf systeminternen und die Netzwerkkomponente nur auf IDU-basierten.

Die *Konfiguration* enthält Einstellungen für die Kontrollinstanz, z.B. ob das Monitoring der Komponente ein- oder ausgeschaltet ist und wie das Meldungsverhalten definiert ist. Das Monitoring wird aktiviert, wenn der QoS-Manager die Dienstqualität selbständig anpassen darf (siehe Reaktionsverhalten in Tabelle 3.6) oder die Anwendung Meldungen vom QoS-Manager erhalten möchte (siehe Meldungsverhalten in Tabelle 3.6). Das Monitoring wird ebenfalls für Parameter aus der statistischen oder deterministischen Dienstklasse aktiviert, um die Einhaltung der spezifizierten Dienstgüte zu überwachen und eventuell Gegenmaßnahmen im Falle einer Warnung oder Verletzung einzuleiten. Das Meldungsverhalten der Bereiche entspricht dem geforderten Meldungsverhalten der Anwendung. Ist die Adaption durch den QoS-Manager erlaubt, benötigt er alle Arten von Meldungen der drei Bereiche.

Der Bedarf an Ressourcen der jeweiligen Komponente wird unter der Rubrik „zusätzlicher Bedarf“ abgelegt. Dort sind beispielsweise Verarbeitungszeiten der Komponenten oder zusätzlicher Durchsatzbedarf vermerkt.

4. Implementierung

In diesem Kapitel werden die implementierungstechnischen Aspekte der Arbeit erläutert. Zunächst werden einige Mechanismen des Betriebssystems vorgestellt, die zur Implementierung verwendet werden sollen. Obwohl die Erläuterung anhand eines konkreten Betriebssystems erfolgt, existieren verwandte Mechanismen auch in anderen Betriebssystemen, so daß die Implementierung nicht auf diese konkrete Plattform beschränkt ist. Im weiteren wird die zur Implementierung verwendete Architektur des QoS-Managers dargestellt. Anschließend wird der Aspekt der objektorientierten Programmierung anhand der Sitzungs- und Datenstromobjekte erläutert. Schließlich werden noch die Protokolldateneinheiten des QoS-Management-Protokolls vorgestellt.

4.1 Betriebssystemaspekte

Die konkrete Implementierung erfolgte unter *Digital-Unix* 4.0 auf Rechnern der Serie DEC Alpha. Digital-Unix ging aus dem Betriebssystem OSF/1 [OSF93] hervor und basiert auf dem *Mach*-Kernel der Carnegie-Mellon Universität. Das Betriebssystem stellt viele der benötigten Konzepte wie *Interprozeßkommunikation* und *Leichtgewichtsprozesse* (englisch: *Threads*) zur Verfügung. Ähnliche Konzepte sind in vielen modernen Betriebssystemarchitekturen enthalten und daher übertragbar. Die in der Implementierung verwendeten Objekte wie Threads, Ports und Messages werden nun kurz erläutert (die Abbildung 4.1 zeigt Tasks, Threads, Ports und Messages).

- Task und Threads
  Unter Mach (und damit auch Digital-Unix) wird eine Prozeßumgebung als *Task* bezeichnet. Die Umgebung beinhaltet die vom Prozeß benötigten Systemressourcen, die den benötigten (virtuellen) Speicheradreßraum einschließen. Innerhalb eines Tasks existieren ein oder mehrere Threads als ausführbare Einheiten, die sich die Systemressourcen des Prozesses teilen können. Ein klassischer Unix-Prozeß wird durch eine Task realisiert, die nur einen Thread besitzt. Mehrere Threads einer Task werden nebenläufig ausgeführt (d.h. auf einer Mehrprozessormaschine tatsächlich parallel, [Wett93]) und das Umschalten zwischen diesen Threads verursacht keinen Adreßraumwechsel (abgesehen

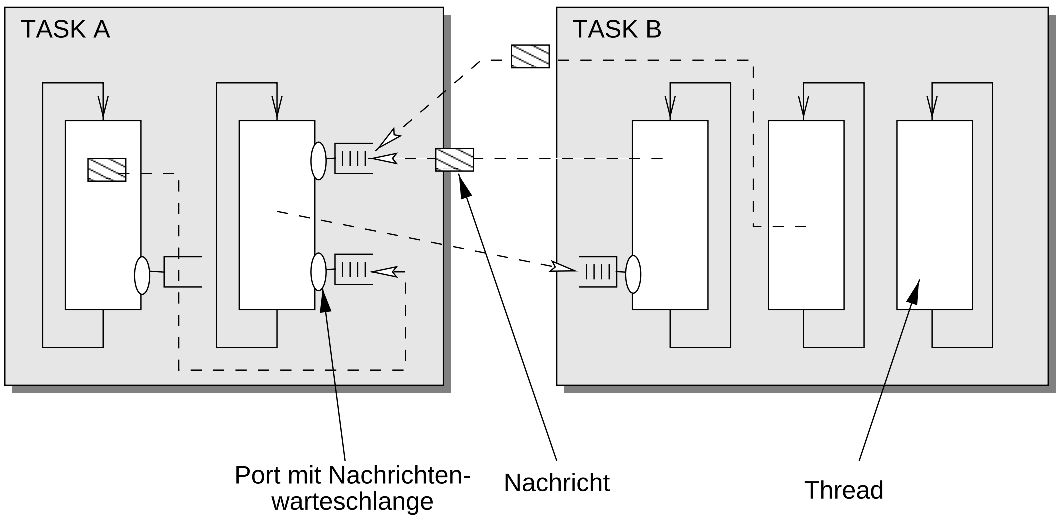

Abbildung 4.1: Tasks, Threads, Ports und Messages

vom Wechsel in den Kernadreßraum), da sie innerhalb eines gemeinsamen Prozeßkontextes ablaufen. Deshalb muß beim Wechsel zwischen zwei Threads innerhalb einer Task auch nicht der Prozeßkontext „gerettet“ werden, sondern lediglich der Kontext des Threads, der wesentlich weniger Information bereithalten muß als ein Prozeßkontext. Daher werden Threads auch gelegentlich als „Leichtgewichtsprozesse“ bezeichnet.

- Ports
  Als Interprozeßkommunikation stellt Mach *Messages* (Nachrichten) und *Ports* („Tore“, siehe [Wett93]) zur Verfügung. Ports sind unidirektionale n:1-Kommunikationskanäle, die zu einer Task gehören und Nachrichten aufnehmen können. Ein Port ist genau einem empfangenden Thread zugeordnet und kann Nachrichten von mehreren Threads aufnehmen. Für die Ports können Schreib- und Leserechte vergeben werden. Das Leserecht kann innerhalb einer Task von Thread zu Thread wechseln. Das Schreibrecht wird von dem Thread vergeben, der momentan das Leserecht besitzt. Schreibrechte können von der übergeordneten Task vererbt werden, sie können mittels Nachrichten weitergegeben werden oder aufgrund des Systemaufrufs `port_allocate()` vom Kernel erhalten werden. Der leseberechtigte Thread ist auch schreibberechtigt, so daß er Nachrichten an sich selbst schicken kann. Bei Erzeugung eines Threads wird diesem vom Kernel automatisch ein Port mit den nötigen Rechten zugeordnet. Der Thread wird vom Kernel durch diesen Kernel-Port identifiziert, er dient sozusagen als Systemname. Im Kernel wird ein Verzeichnis der vergebenen Ports samt ihren Rechten gehalten. Ein „Name-Server“ übernimmt die Umsetzung eines symbolischen Namens auf Portnummern [OSF91], wobei jeder Prozeß das Recht hat, bei diesem Name-Server anzufragen und einen Reply-Port für die Anwort des Name-Servers anzugeben.

- Messages
  Nachrichten mit beliebigem Inhalt (die Menge der Daten ist nur durch die Größe des Adreßraums begrenzt) können von einem Thread an einen anderen geschickt werden, auch zwischen Threads zweier Tasks, d.h. über Adreßraumgrenzen hinweg. Der Nachrichtentransport über Adreßraumgrenzen hinweg, geschieht dabei sehr effizient, da Mach einen sogenannten *copy-on-write* Mechanismus verwendet. Das bedeutet, daß eine Nachricht erst dann physikalisch in den Adreßraum der empfangenden Task kopiert wird, wenn einer der beiden Threads eine Schreibzugriff auf diese Daten durchführen möchte. Das Prinzip, eine Operation aufzuschieben, bis sie notwendigerweise durchgeführt werden muß, wird auch als *„lazy evaluation"* bezeichnet. Das Aufschieben der Operation geschieht in der Hoffnung, daß die Operation nicht ausgeführt werden muß. Nachrichten können in allen Kopplungsarten [Wett93], d.h. *synchron* und *asynchron* gesendet und empfangen werden. Nachrichten werden durch `msg_send()` verschickt und mit `msg_receive()` empfangen. Die abgeschickten Nachrichten werden in einer Nachrichten-Warteschlange am Emfangsport abgelegt und verbleiben dort bis zu ihrer Abholung durch den empfangenden Thread. Ist die Warteschlange leer, wenn der Thread ein `msg_receive()` aufruft, wird er blockiert, bis eine Nachricht eintrifft. Die Nachrichten werden in Empfangsreihenfolge ausgeliefert. Es besteht auch die Möglichkeit, mehrere Ports zu einer Menge zusammenzufassen, so daß sie zusammen nur noch eine einzelne Warteschlange besitzen. Die Nachrichten selbst sind strukturiert, d.h. sie bestehen aus einem Kopf (header) und einem Rumpf (body). Im Kopf der Nachricht sind unter anderem die Adresse des Empfängerports, die Größe der Nachricht und Angaben zur Art der Nachricht enthalten. Auch der Rumpf der Nachricht ist strukturiert. Er kann mehrere Datenfelder enthalten, die durch vorangestellte Beschreibungsfelder (Deskriptoren) beschrieben werden. Ein Deskriptor beschreibt den Typ und die Größe des folgenden Datenfeldes. Durch die Angabe der Größe des Datenfeldes, kann auf die Position des nächsten Deskriptors geschlossen werden.

4.2 Implementierungsarchitektur

Der QoS-Manager soll als *zentrale Instanz* genau einmal in jedem Endsystem vorhanden sein. Dies ermöglicht erst die Koordination und Integration der drei Bereiche Betriebssystem, Kommunikationssubsystem und Netzwerk. Damit existiert der QoS-Manager als *Server-Instanz* in einem eigenen Adreßraum. Die Kommunikation mit der Anwendung, den Kontrollinstanzen der Bereiche muß daher über Adreßraumgrenzen hinweg möglich sein. Dazu wird die Interprozeßkommunikation benutzt. Innerhalb des QoS-Managers werden der Koordinator und der QoS-Makler als Threads realisiert, die sich ebenfalls über Ports Nachrichten schicken (siehe Abbildung 4.2). Dies entspricht der in Abschnitt 3.2 dargestellten Architektur (s. Abbildung 3.1, Seite 30). Der Koordinator wartet auf Nachrichten der Anwendungen und auf Nach-

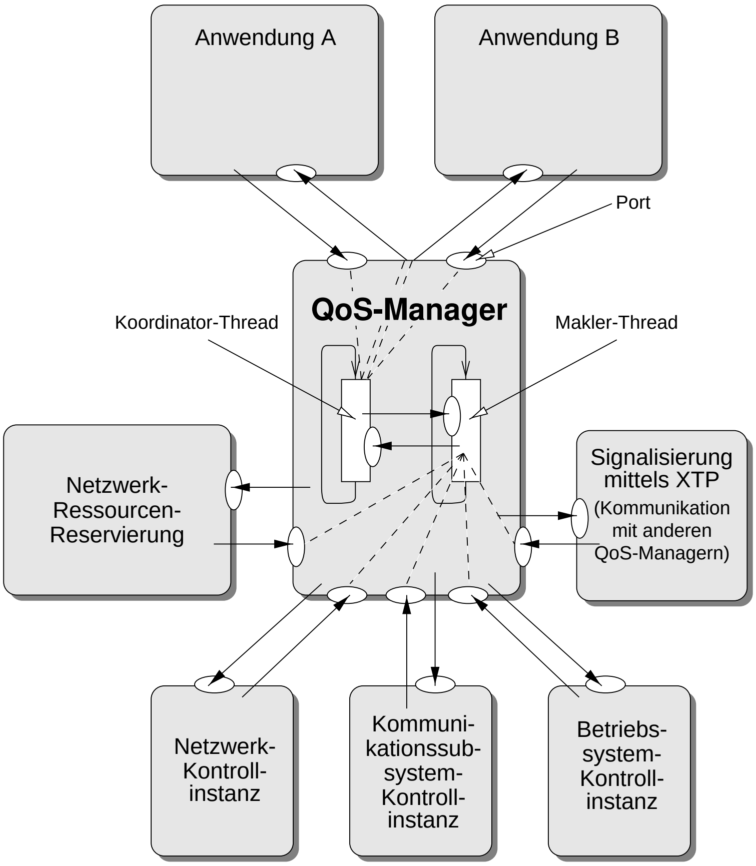

Abbildung 4.2: Implementierung des QoS-Managers durch Threads

richten des QoS-Maklers. Trifft eine solche Nachricht ein, ermittelt der Koordinator zunächst den Absender und sucht den zugehörigen Kontext. Danach erfolgt eine Prüfung, ob die gewünschte Operation auch in dem momentanen Zustand erlaubt ist und durchgeführt werden kann. Dann wird die Operation ausgeführt. Dazu wird in der Regel der QoS-Makler beauftragt, falls der Auftraggeber die Anwendung war. Kam die Nachricht vom QoS-Makler, wird der Anwendung das Ergebnis der Operation mitgeteilt. Der Ablauf ist in Abbildung 4.3 schematisch dargestellt. Der Koordinator-Thread besteht aus einer Schleife (eine sogenannte Ereignisschleife), in der er auf Ereignisse (Nachrichten) wartet und bei Eintreten eines Ereignisses entsprechende Aktionen durchführt. Der Koordinator ist jedoch für einen Auftrag einer anderen (oder der gleichen) Instanz frei, sobald er den vorigen Auftrag abgearbeitet hat, d.h. er arbeitet die Aufträge der unterschiedlichen Instanzen im Wechsel ab. Dies wird durch das Speichern der Zustände in den entsprechenden Kontexten ermöglicht, so daß mehrere Sitzungen in unterschiedlichen Zuständen verwaltet werden können. Ebenso können sich die Datenströme einer Sitzung in unterschiedlichen Zuständen befinden. Daher wird nach Erhalt eines Auftrags nicht nur der entsprechende Sitzungskontext gesucht, sondern auch für jeden Datenstrom der zugehörige Datenstromkontext.

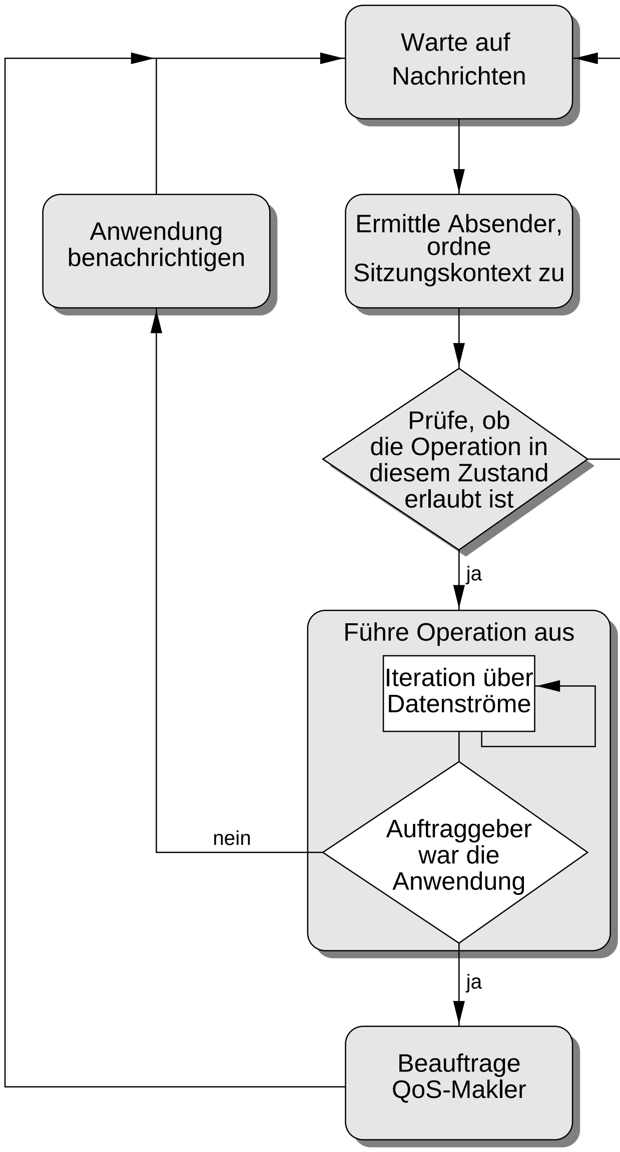

Abbildung 4.3: Ablauf im Koordinator

Innerhalb des QoS-Maklers wird auf die Nachrichten des Koordinators, der drei Kontrollinstanzen, der Instanz für die Ressourcenreservierung im Netzwerk und des Signalisierungprotokolls gewartet. Die Nachrichten enthalten dabei immer einen Identifikator für die Sitzung, so daß der QoS-Makler den entsprechenden Kontext zuordnen kann. In diesem Kontext werden die Phasen der momentan aktivierten Abläufe festgehalten, so daß der zur Nachricht gehörende Ablauf fortgesetzt werden kann. Der QoS-Makler setzt den Ablauf fort, in dem er die entsprechenden Instanzen kontaktiert und die neue Phase im Kontext vermerkt. Anschließend wartet er auf die nächste Nachricht einer Instanz.

4.3 Objektorientierung

Die Implementierung soll weitgehend *objektorientiert* erfolgen. Der Vorteil einer objektorientierten Programmierung liegt vor allem in einer leichteren Wartbarkeit und Erweiterbarkeit des Programmcodes gegenüber herkömmlichen Programmierparadigmen [Stro91]. Ein wesentliches Konzept der objektorientierten Programmierung ist der *Vererbungsmechanismus*, durch den Gemeinsamkeiten der Objekte ausgedrückt werden können, bzw. der erlaubt Spezialisierungen durch Ergänzungen vorzunehmen, ohne die bestehende Programmstruktur zu verändern. Ein *Objekt* ist eine Instanziierung eines bestimmten „Datentyps". Dieser „Datentyp" wird in der objektorientierten Programmierung als *Klasse* bezeichnet. Klassen beinhalten im allgemeinen Datenstrukturen und Funktionen (sogenannte *Methoden*), die auf diesen Datenstrukturen operieren. Normalerweise sind die Datenstrukturen eines Objekts nicht von außerhalb des Objekts direkt zugänglich, sondern können nur indirekt über die erlaubten Methoden des Objekts beeinflußt werden. Die *Methoden* führen also Operationen auf einem Objekt aus, die konkrete Implementierung des Objekts ist von „aussen" nicht sichtbar (Konzept des abstrakten Datentyps). Eine neue Klasse kann von einer bestehenden Klasse *abgeleitet* werden, d.h. sie erbt sämtliche Eigenschaften ihrer *Basisklasse*. Die neue Klasse fügt diesen bestehenden Eigenschaften weitere hinzu.

Als objektorientierte Programmiersprache wurde *C++* gewählt [Stro91], da es die benötigten objektorientierten Konzepte effizient unterstützt (z.B. erfolgen alle Typüberprüfungen bereits zum Zeitpunkt der Übersetzung und nicht während der Ausführung eines Programmcodes) und als Obermenge der Programmiersprache *C* zusätzlich eine „systemnahe" Programmierung ermöglicht. Im folgenden Abschnitt werden einige für die Implementierung entworfenen Objekte kurz vorgestellt.

4.3.1 Objekte

Aus der Perspektive der Anwendung operiert sie auf einem *Sitzungsobjekt* und mehreren *Datenstromobjekten*. Dabei stellt das Sitzungsobjekt genau die Methoden zur Verfügung, die in Abschnitt 3.3.3 als diejenigen Dienstprimitive angegeben wurden, die von der Anwendung aufgerufen werden (also nur Anfragen und Antworten). Weil Anzeigen und Bestätigungen von der Anwendung entgegen genommen werden müssen, können sie nicht von ihr als Methode aufgerufen werden. Es besteht aber die Möglichkeit Prozeduren der Anwendung anzugeben, die bei einer Anzeige oder Bestätigung aufgerufen werden sollen (sogenannte „Callbacks", dieses Prinzip wird z.B. unter Motif für X-Windows angewandt). Als Alternative kann die Anwendung aber auch auf Nachrichten warten, die im Rahmen der Interprozeßkommunikation verwendet werden, und die Anzeige oder Bestätigung enthalten. In der Implementierung wird diese letzte Möglichkeit verwendet. So existiert auch in der Anwendung eine Ereignisschleife, die Methoden auf dem Sitzungobjekt aufrufen kann, Reaktionen auf Benutzereingaben, Anzeigen und Bestätigungen vorsieht.

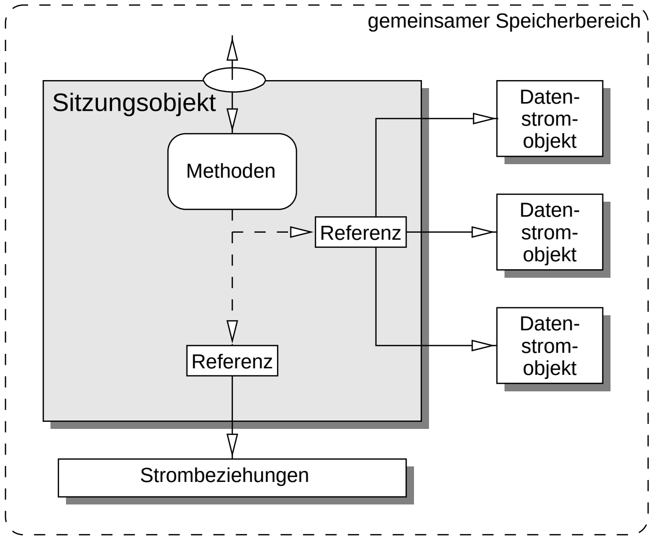

Abbildung 4.4: Das Sitzungsobjekt als Kommunikationsschnittstelle

Jedes Sitzungsobjekt (s. Abbildung 4.4) beeinhaltet eine Liste von *Datenstromobjekten* und *Datenstrombeziehungsobjekten*. Die Datenstromobjekte beeinhalten die Spezifikation für die Datenströme der Sitzung, die Beziehungsobjekte definieren die Beziehungen zwischen diesen Datenströmen wie in Abschnitt 3.6.1 beschrieben. Das Sitzungsobjekt dient nun als Kommunikationsschnittstelle zwischen Anwendung und QoS-Manager. Daher befindet es sich in einem geteilten Speicherbereich („shared memory"), auf den beide Prozesse Zugriff haben. Die Anwendung spezifiert zum Beispiel andere Parameter für einen Datenstrom, in dem sie die Parameter des zugehörigen Datenstromobjekts ändert und anschließend ein SessionObj.ModifyChgReq(&DataStreamObj) ausführt. Diese Methode löst erstmal eine Überprüfung im Sitzungsobjekt aus, ob das Dienstprimitiv in diesem Zustand der Sitzung überhaupt erlaubt ist (siehe dazu den Schnittstellenautomaten in [Henr96]). Ist dies der Fall, wird automatisch eine Nachricht per Interprozeßkommunikation an den QoS-Manager verschickt. Dieser erhält zusätzlich zur Angabe des Dienstprimitivs, eine Referenz auf das Sitzungsobjekt im geteilten Speicherbereich. Der QoS-Manager analysiert daraufhin das Dienstprimitiv und prüft anhand seiner Datenstromkontexte und den dort gespeicherten Zuständen, ob die Aktion durchgeführt werden kann. Wenn dies der Fall ist, werden die Parameter des Datenstromobjekts in der internen QoS-Struktur (IQS) vermerkt. Die IQS besteht selbst aus Objekten der Klassen für Datenströme. So können die Klassen, die bereits für die Anwendungsschnittstelle definiert wurden, wiederverwendet werden. Zusätzlich existieren noch weitere Objekte in der IQS, welche die systeminternen Parameter aufnehmen. Werden neue Parameter für den Datenstrom ausgehandelt, schreibt der QoS-Manager sie zurück in das Datenstromobjekt und führt auf dem Sitzungsobjekt ein SessionObj.ModifyChgCnf(&DataStreamObj) aus. Dadurch wird eine Nachricht an die Anwendung geschickt, die daraufhin die neu ausgehandelten Werte dem Datenstromobjekt entnimmt. Das Datenstromobjekt bleibt für einen weiteren Zugriff des Absenders solange gesperrt, bis der Empfänger es ausgelesen hat (es darf immer nur eine Aktion auf demselben Datenstromobjekt ausstehen). Auch das Senden von Daten wird über das Sitzungsobjekt veranlaßt. Die Anwendung gibt dazu nur eine Referenz auf den Datenstrom und die zu sendenden Daten an: SessionObj.SendData(&DataStreamObj, &DataBuffer). Diese Methode sorgt dann dafür, daß das Kommunikationssubsystem eine Nachricht erhält, die entsprechenden Daten zu übertragen.

Es existieren je nach Medientyp, unterschiedliche Klassen für die Datenstromobjekte. Dabei werden die Datenströme für Video, Audio und Graphik und sonstige Daten von einer Basisklasse Datenstrom abgeleitet, d.h. sie stellen eine Spezialisierung der Basisklasse Datenstrom dar und verfügen daher über zusätzliche Methoden. Da eine Sitzung meist aus heterogenen Datenstromobjekten besteht, wird dies

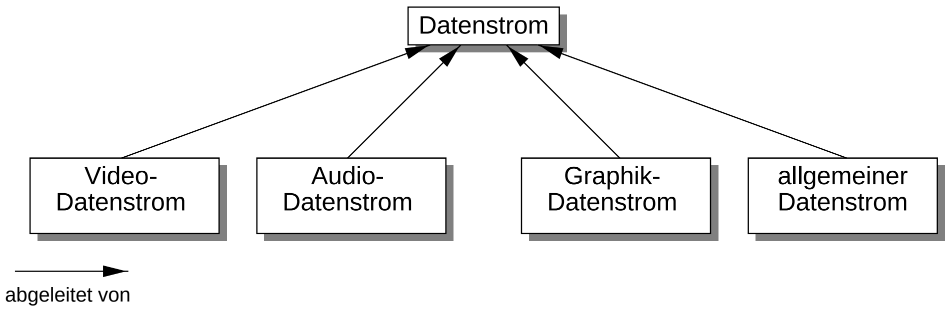

Abbildung 4.5: Die Vererbungshierarchie für Datenstromobjekte

durch eine Liste von Referenzen auf die Datenstromobjekte der Basisklasse realisiert. Die abgeleiteten Datenstromobjekte können dabei eine besondere Behandlung durch die sogenannten „virtuellen Funktionen“ der Basisklasse erfahren [Stro91]. Letztere ermöglichen eine *implizite* Typunterscheidung, ohne daß dazu ein Typfeld zur Laufzeit des Programms überprüft werden muß (*explizite* Typunterscheidung).

4.4 Das QoS-Management Protokoll

In diesem Abschnitt werden die konkreten Protokolldateneinheiten (PDUs) und Formate des QoS-Management Protokolls (QMP) vorgestellt. Zur Vereinfachung wird angenommen, daß die Dateneinheiten des Protokolls über einen gesicherten Signalisierungs-Kanal übertragen werden. Momentan wird dieser gesicherte Signalisierungs-Kanal durch Benutzung von XTP[1] realisiert. Ansonsten müßte man zusätzlich die Funktionalität eines Transportprotokolls zur gesicherten Datenübertragung in QMP integrieren, um es auch über unzuverlässige Kanäle betreiben zu können. Als Byte-Reihenfolge für Ganzzahlen wird die übliche Netzwerk-Reihenfolge definiert, d.h. das höchstwertige Byte wird zuerst übertragen („Big Endian“) definiert. Im folgenden Abschnitt werden die Protokolldateneinheiten vorgestellt.

[1] Xpress Transfer Protocol

4.4.1 Die Protokolldateneinheiten

Das Grundformat der QMP-PDUs ist in Abbildung 4.6 zu sehen: Eine QMP-PDU

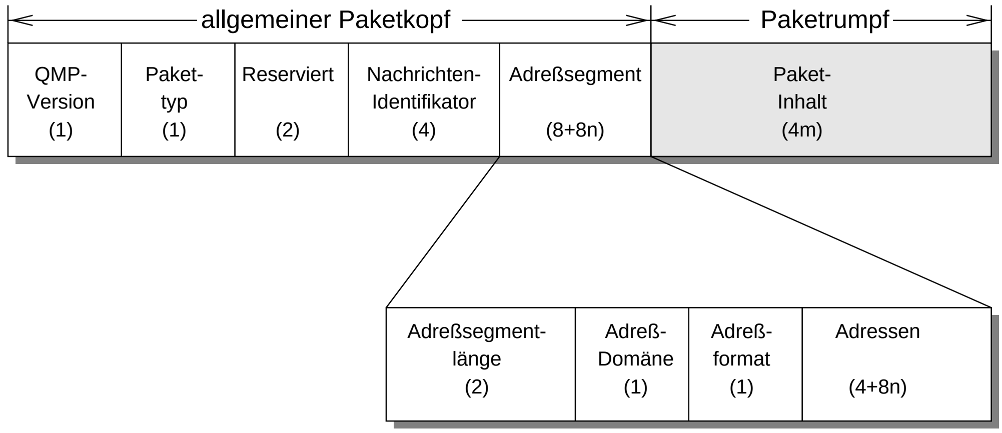

Abbildung 4.6: Allgemeiner Paketkopf mit Paketrumpf

besteht aus einem allgemeinen Paketkopf und einem Paketrumpf. Der Paketkopf ist für alle Pakettypen gleich. Der Inhalt und Aufbau der Nutzlastinformation im Paketrumpf variiert in Abhängigkeit vom Pakettyp. Der allgemeine Paketkopf besteht aus einem Versionsfeld, einer Angabe des Pakettyps, einem noch unbenutzten Feld für Erweiterungen, einem Nachrichtenidentifikator und einem Adreßsegment. Das *Versionsfeld* dient zur Unterscheidung verschiedener Versionen des QoS-Management Protokolls und belegt 1 Byte. Die momentan verwendete Versionsnummer ist 1. Die Angabe des *Pakettyps* dient dazu, die verschiedenen PDU-Typen zu kennzeichnen und anhand dieser auf den Aufbau des Paketinhalts zu schließen. Es können maximal 256 verschiedene Pakettypen unterschieden werden. Der *Nachrichtenidentifikator* (32 bit) dient der Zuordnung von Antworten und Bestätigungen zu einer Anfrage. Jede Anfrage erhält vom QoS-Makler vor dem Absenden einen eindeutigen Nachrichtenidentifikator. Die entfernten QoS-Makler senden ihre Antwort mit dem selben Nachrichtenidentifikator zurück, so daß der initiierende QoS-Makler die eingehenden Antworten der entsprechenden Anfrage zuordnen kann. Das *Adreßsegment* enthält die Adressen des Senders und des Empfängers bzw. der Empfängergruppe. Das Format des Adreßsegments wurde aus der Spezifikation der XTP Version 4.0 [XTPF95] übernommen, weil es eine flexible Adressierung durch Verwendung unterschiedlicher Adreßformate erlaubt. Die Zusammensetzung eines Adreßsegments ist ebenfalls in Abbildung 4.6 dargestellt. Die *Adreßsegmentlänge* gibt die Länge des gesamten Adreßsegments (also einschließlich des anfänglichen 4-Byte Deskriptors) in Bytes an. Die Länge des Adreßsegments ist immer ein Vielfaches von 8 Bytes, daher besteht ein minimales Adreßsegment aus 8 Bytes. Die *Adreßdomäne* beinhaltet die Protokolladreßinformation und dient daher als Adreß-Demultiplexer für die Transportdienste. Dies ist notwendig, weil einige Adreßformate gleichzeitig in mehreren Adreßdomänen verwendet werden: so benutzen z.B. TCP[2], UDP[3] und XTP zur Adressierung Portnummern, wobei durchaus dieselbe Portnummer von allen Protokollen verwendet werden darf, d.h. TCP Port 42 ist verschieden von UDP Port 42. Der Wert der Adreßdomäne ist 6 für TCP, 17 für UDP und 36 für XTP. Das Feld *Adreßformat* enthält eine Nummer, die das Format der folgenden Adressen festlegt. Die Nummern 0–127 werden vom XTP Forum festgelegt, die Nummern 128–255 stehen für lokale Testumgebungen und Implementierungen zur Verfügung. Folgende Adreßformate wurden bisher berücksichtigt:

Wert	Adreßformat
0	Null Adresse
1	Internet Protocol Adresse
2	ISO CNLP Adresse
3	Xerox Network System Adresse
4	IPX Adresse
5	Lokale Adresse
6	Internet Protocol version 6 Adresse

Eine Beschreibung aller Adreßformate findet man in [XTPF95]. Die Null Adresse ist z.B. für dedizierte Punkt-zu-Punkt Verbindungen gedacht. Das Adreßfeld für das Internet Protocol besteht aus 12 Bytes: 4 Bytes Zieladresse, 4 Bytes Quelladresse, 2 Bytes Zielportnummer, 2 Bytes Quellportnummer. Es wird angenommen, daß die Anwendung über eine solche Adresse lokalisiert werden kann, so daß eine Identifikation für die Sitzung nicht benötigt wird, da momentan pro Anwendung nur eine Sitzung zugelassen ist.

Der *Paketrumpf* trägt in Abhängigkeit vom verwendeten Pakettyp unterschiedliche Information. Die Länge des Paketrumpfes ist ein ganzzahlig Vielfaches von 4 Bytes. Auf die Länge des Paketrumpfes kann aus der Gesamtlänge des Pakets und des allgemeinen Paketkopfes geschlossen werden. Es wird angenommen, daß beim Empfang eines Pakets vom Signalisierungskanal die Gesamtlänge mit angezeigt wird. Im folgenden Abschnitt werden die verschiedenen Pakettypen und die zugehörigen Paketformate vorgestellt.

4.4.2 Pakettypen

Die existierenden Pakettypen sind in Tabelle 4.1 aufgeführt. Jedem Pakettyp ist ein bestimmtes Paketformat zugeordnet, welches die Struktur des Paketinhalts festlegt.

Der Aufbau der verschiedenen Paketformate wird in den folgenden Abschnitten erläutert. Grundsätzlich kann der Inhalt eines bestimmten Paketformats verschieden lang sein. Dies wird durch Teile des Paketformats verursacht, die eine variable Länge aufweisen. Dies sind zum Beispiel Teile, die Listen enthalten. Ein variabler Paketteil

[2] Transport Control Protocol
[3] User Datagram Protocol

Pakettyp	Paketformat	Erläuterung
OpenRequest OpenResponsePositive OpenCommitPositive	NEGOTIATE	Sitzungsaufbau
ModifyAddRequest ModifyAddResponsePositive ModifyAddCommitPositive	NEGOTIATE	Hinzufügen von Datenströmen
ModifyChangeRequest ModifyChangeResponsePositive ModifyChangeCommitPositive	NEGOTIATE	Ändern der Eigenschaften von Datenströmen
ModifyJoinRequest ModifyJoinResponsePositive ModifyJoinCommitPositive	NEGOTIATE	Hinzufügen eines Teilnehmers
OpenResponseNegative OpenCommitNegative	REJECT	Sitzungsaufbau (Ablehnung)
ModifyAddResponseNegative ModifyAddCommitNegative	REJECT	Hinzufügen von Datenströmen (Ablehnung)
ModifyChangeResponseNegative ModifyChangeCommitNegative	REJECT	Ändern der Eigenschaften von Datenströmen (Ablehnung)
ModifyJoinResponseNegative ModifyJoinCommitNegative	REJECT	Hinzufügen eines Teilnehmers (Ablehnung)
ModifyDeleteRequest ModifyDeleteResponse	DELETE	Löschen eines Datenstroms
ModifyRemoveEndPointRequest ModifyRemoveEndPointResponse	CLOSE	Entfernen eines Teilnehmers
CloseRequest CloseResponse	CLOSE	Beenden der Sitzung
Information Abort	INFORM INFORM	diverse Meldungen Abbruchsmitteilung

Tabelle 4.1: Pakettypen und Paketformate

enthält zuerst einen 2 Byte großen Zähler, der die Größe des folgenden Teils in der Anzahl an 4 Byte großen Worten angibt.

4.4.2.1 Das NEGOTIATE-Paketformat

Das NEGOTIATE-Paket wird, wie in Tabelle 4.1 zu sehen ist, für alle Pakettypen verwendet, die eine Ende-zu-Ende Aushandlung erfordern. Das NEGOTIATE-Paketformat muß daher die Spezifikation der Sitzung übermitteln. Dazu gehören insbesondere die Datenstromspezifikationen und deren funktionale Konfiguration. Das NEGOTIATE-Paketformat ist in Abbildung 4.7 dargestellt: Entsprechend der Sitzungsdefinition sind im NEGOTIATE-Paket die Strombeziehungen (siehe Abschnitt 3.6.1, Seite 68) und die Datenstromspezifikationen aufgeführt. Die *Größe der Strom-*

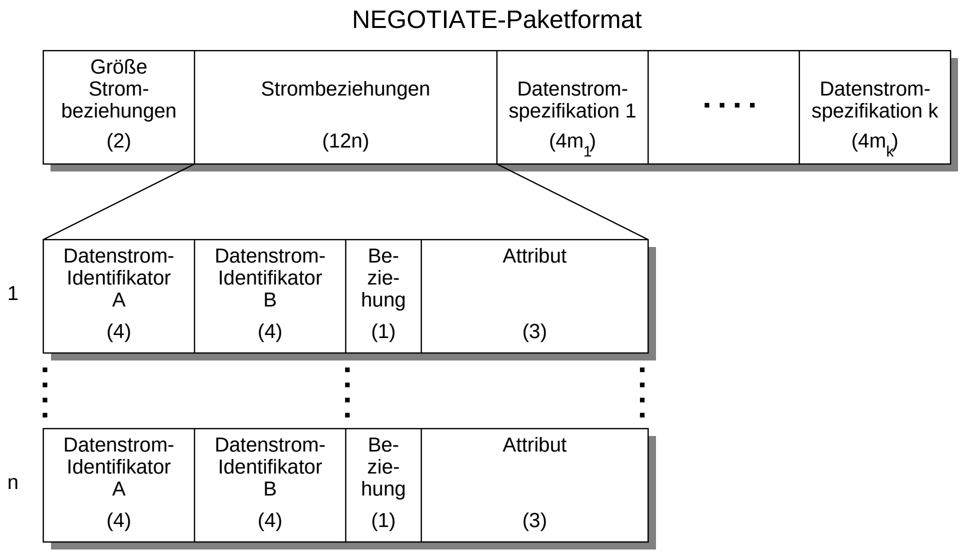

Abbildung 4.7: Das NEGOTIATE-Paketformat

beziehungen beschreibt dabei, aus wievielen 32-bit Worten die folgenden Strombeziehungen insgesamt bestehen. Auch die Größe 0 (Null) ist erlaubt, welche die Abwesenheit von Strombeziehungen anzeigt. Die Beschreibung einer Relation zwischen einem Datenstrompaar besteht aus den beiden Datenstromidentifikatoren, der Beziehung und einem Attribut. Die *Datenstromidentifikatoren* müssen innerhalb einer Sitzung für alle Teilnehmer eindeutig sein. Das wird dadurch erreicht, daß dasjenige System, welches einen neuen Datenstrom initiiert, einen noch freien Datenstromidentifikator per Zufall auswählt. Initiieren zwei Systeme gleichzeitig einen neuen Datenstrom, ist die Wahrscheinlichkeit einer Kollision durch Verwendung der gleichen Datenstromidentifikatoren gering. Die *Beziehung* kann die Werte 0 für eine *logische* und 1 für eine *zeitliche* Beziehung annehmen. Im Fall der logischen Beziehung enthält das folgende Attribut-Feld die Art der logischen Beziehung: 0 für die *Lebenszeitrelation*, 1 für die *identische Teilnehmerstruktur*, 2 für die *identische Dienstqualität*. Im Falle einer zeitlichen Beziehung enthält das Attribut-Feld die Angabe des Skew-Wertes, also des erlaubten zeitlichen Versatzes. Dabei ist der ganzzahlige 24-bit Wert die Anzahl der μs (10^{-6}s), um die die Datenströme gegeneinander versetzt sein dürfen, d.h. es wird immer ein symmetrisches Skew-Intervall angegeben. Der maximal spezifizierbare Skew-Wert beträgt somit ca. $\pm$16s.

Anschließend folgen die Datenstromspezifikationen (s. Abbildung 4.8). Die Datenstromspezifikationen können unterschiedliche Längen aufweisen und beginnen mit dem Datenstromidentifikator. Im darauffolgenden *Status*-Feld wird bei Antworten angegeben, ob der Datenstrom von antwortenden System *akzeptiert* (accepted), *verändert* (modified) oder *abgelehnt* (rejected) wurde. Anschließend folgen drei Teile variabler Länge, denen deshalb ein Größenfeld vorangestellt wird, welches die Länge des Teils in 4 Byte großen Worten angibt. Die drei Teile sind: die *anwendungsorientierte Spezifikation*, die *funktionale Konfiguration* und die *Kommunikationssubsystem Konfiguration*. Die anwendungsorientierte Spezifikation enthält die

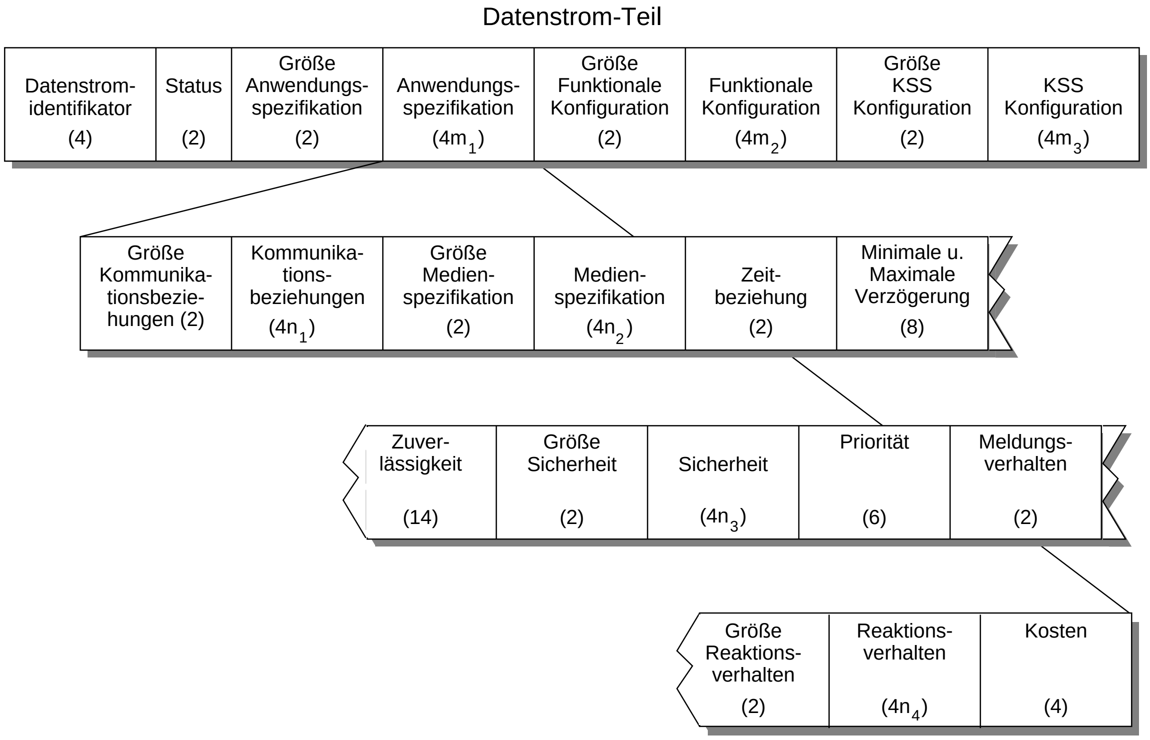

Abbildung 4.8: Die Datenstromspezifikation

anwendungsorientierten Parameter aus Tabelle 3.6. Die funktionale Konfiguration beeinhaltet Managementhinweise, wie das Kommunikationssubsystem und die Netzwerkkomponente zu konfigurieren sind. Der dritte Teil (Kommunikationssubsystem Konfiguration) enthält interne Konfigurationsdaten für das Kommunikationssubsystem und wird von QMP transparent weitergereicht. Solche Konfigurationsdaten beschreiben beispielsweise welche internen Mechanismen zum Einsatz kommen sollen. Es kann dort zum Beispiel auch das FIRST-Paket von XTP übertragen werden, wenn XTP als Kommunikationssubsystem zum Einsatz kommt.

Nun soll noch die anwendungsorientierte Spezifikation näher erläutert werden. Sie ist ebenfalls in Abbildung 4.8 zu sehen. Die *Kommunikationsbeziehungen* sind, ebenso wie die darauffolgende *Medienspezifikation*, Teile mit einer variablen Länge. Dies resultiert aus der variablen Empfängerliste in den Kommunikationsbeziehungen und aus der Angabe eines Kompressionsverfahrens mit zugehörigen Parametern und Intrastromprioritäten bei der Medienspezifikation. Die *Zeitbeziehung* enthält nur einen codierten Wert für interaktiv, Realzeit, und asynchron. Die Angabe der minimalen und maximalen *Verzögerung* erfolgt durch je ein 32-bit Wort, dessen Wert die Anzahl der Mikrosekunden (μs) angibt. Die *Zuverlässigkeit* wird durch das in Abschnitt 2.1.2 definierte Tripel und die Auslieferungsstrategie angegeben, wobei jede Komponente des Tripels als 32-bit Wort abgelegt wird. Die Auslieferungsstrategie (2 Bytes) enthält nur den Hinweis, ob fehlerhafte ADUs ausgeliefert werden sollen. Die Angabe der *Sicherheit* erfolgt durch einen variablen Teil, der zuerst einen codierten Wert für das eingesetzte Sicherheitsverfahren und eventuell zusätzlich benötigte Parameter für das jeweilige Verfahren enthält (siehe Tabelle 3.6). Die *Priorität* enthält die Ordnungsnummer als 32-bit Wert und einen codierten Wert (16 bit) für die Relevanz (entbehrlich, unentbehrlich). Das *Meldungsverhalten* wird durch ein Bitmuster angegeben, da mehrere Meldungsarten gewünscht werden können. Das *Reaktionsverhalten* ist wieder ein Feld variabler Länge, weil es aus einer Liste von Tupeln (Ereignis, Reaktion) besteht. Schließlich werden die maximal tolerierbaren Kosten durch einen 32-bit Wert mitgeteilt.

4.4.2.2 Das **DELETE**-Paketformat

Dieses Paketformat enthält nur eine Liste der Datenstromidentifikatoren, welche die zu löschenden Datenströme spezifizieren (s. Abbildung 4.9 oben). Die Anzahl der Datenstromidentifikatoren läßt sich aus der gesamten Paketlänge ermitteln, da die Länge des Paketkopfes und der Datenstromidentifikatoren nicht variabel ist.

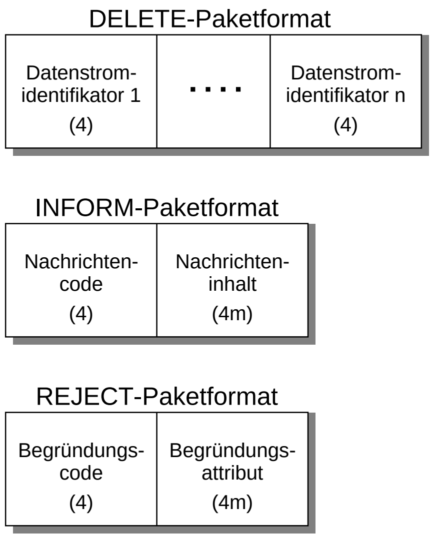

Abbildung 4.9: Die Paketformate DELETE, REJECT, und INFORM

4.4.2.3 Das **INFORM**-Paketformat

Das Paketformat für Nachrichtenpakete setzt sich aus einem Nachrichtencode und in Abhängigkeit davon aus einem variablen Nachrichteninhalt zusammen. Monitoringdaten, Warnungen und Meldungen über die Verletzung der Dienstqualität können beispielsweise durch den Pakettyp Information mit einem entsprechenden Nachrichtencode und Nachrichteninhalt übermittelt werden. Dieses Paketformat wird auch für Pakete vom Typ „Abort“ verwendet. Der Nachrichtencode gibt in diesem Fall einen Hinweis auf die Abbruchursache.

4.4.2.4 Das REJECT-Paketformat

Bei einer Ablehnung wird ein codierter Wert für die Begründung mitgeliefert. In Abhängigkeit von der Begründung kann ein zusätzliches *Begründungsattribut* angegeben werden, welches das Problem näher erläutert, in dem z.B. eine Liste der Identifikatoren von unentbehrlichen Datenströmen mitgegeben wird, die Ursache der Ablehnung waren. Das Format ist in Abbildung 4.9 unten dargestellt. Beispiele für Begründungen sind Tabelle 4.2 zu entnehmen.

Begründungscode	Begründung	Attribut vorhanden
NO_SUCH_APPLIC	Anwendung mit dieser Adresse antwortet nicht	nein
APPLIC_REJECT	Anwendung lehnt ab	ja
CSS_REJECT	Kommunikationssubsystem lehnt ab	ja
LNET_REJECT	lokale Netzwerkkomponte lehnt ab	ja
OS_REJECT	Betriebssystem lehnt ab	ja
COLLISION	Andere Operation wird bereits ausgeführt	ja
⋮	⋮	⋮
UNKNOWN	Grund für Ablehnung unbekannt	nein

Tabelle 4.2: Begründungscodes für ein REJECT

4.4.2.5 Das CLOSE-Paketformat

Momentan ist dieses Paketformat leer, da keine weitere Information zum Schließen einer Sitzung oder Entfernen eines Sitzungsteilnehmers übertragen werden muß.

5. Zusammenfassung und Ausblick

In dieser Diplomarbeit wurde eine Instanz für das integrierte Management qualitätsorientierter Kommunikationsdienste entworfen und implementiert. Die Instanz (der „QoS-Manager") wurde in eine Systemarchitektur integriert, die das Management anwendungsorientierter Kommunikationsdienste ermöglicht. Wesentliche Aufgaben bestanden in der Entwicklung eines universellen Konzepts zur integrierten Aushandlung von Dienstqualitäten und zur Koordination der Diensterbringung sowie in der Formulierung von Dienstgüteabbildungen.

Am Anfang dieser Diplomarbeit stand die Literaturrecherche. Es wurden verschiedene Ansätze zum Management qualitätsorientierter Kommunikationsdienste untersucht und Anregungen für den Entwurf der Managementinstanz gewonnen. Dann wurden die Aufgaben der Managementinstanz formuliert. Eine Gliederung der Aufgaben führte schließlich zur Dekomposition der Managementinstanz in Teilinstanzen, die jeweils bestimmte Teilaufgaben (Koordination der Diensterbringung, Dienstaushandlung, Dienstabbildung) übernehmen. Die Analyse der Abhängigkeiten zwischen den Systemkomponenten lieferte den benötigten internen Ablauf zur Koordination der Diensterbringung. Eine weitere wichtige Teilaufgabe bestand in der Festlegung von Abläufen zur Dienstaushandlung. Es wurde ein Ablauf im lokalen System sowie auch ein Ablauf zwischen den an der Kommunikation beteiligten Endsystemen festgelegt. Es wurden daraus verschiedene Protokolle definiert, unter anderem das QoS-Management-Protokoll (QMP), welches zum Informationsaustausch zwischen QoS-Managern dient. Die Spezifikation dieser Protokolle erfolgte durch eine Beschreibung mittels endlicher Automaten. Die entwickelten Dienstgüteabbildungen, welche zur Aushandlung der Dienstgüte benötigt werden, bilden anwendungsorientierte Parameter auf systeminterne ab (und umgekehrt).

Die Gliederung in Teilinstanzen ließ sich bei der Implementierung direkt auf entsprechende Prozesse abbilden. Auch die Spezifikation der Protokolle durch endliche Automaten konnte direkt verwendet werden. Die Implementierung zeigte die Realisierbarkeit der entwickelten Konzepte zum automatischen und transparenten Management qualitätsorientierter Kommunikationsdienste für Anwendungen.

Die Komponenten der oben erwähnten Systemarchitektur waren zum Zeitpunkt der Implementierung nur teilweise verfügbar. Erste Tests konnten jedoch auch ohne die Existenz einiger Systemkomponenten (wie z.B. Kontrollagenten) durchgeführt werden. Eine zukünftige Erweiterung kann darin bestehen, Fehlerbehebungsmechanismen in das QoS-Management-Protokoll zu integrieren, so daß auch unzuverlässige Datagramm-Dienste zur Signalisierung einsetzbar sind. Weiterhin kann die Konsensfindung so erweitert werden, daß auch Gruppen mit heterogenen Dienstqualitäten verwaltet werden können.

Literaturverzeichnis

[BlSt96] Blakowski, Gerald; Steinmetz, Ralf
„A Media Synchronization Survey: Reference Model, Specification, and Case Studies“
IEEE Journal on selected Areas in Communications, Vol. 14, No. 1, Januar 1996

[Bles95] Bless, Roland
„Entwurf und Implementierung eines Monitors für Dienstqualitäten“
Studienarbeit, Universität Karlsruhe, Institut für Telematik, Fakultät für Informatik, 1995

[Brau93] Braun, Torsten
„Ein paralleles Transportsubsystem für zellenbasierte Hochgeschwindigkeitsnetze“
Dissertation, Universität Karlsruhe, Institut für Telematik, Fakultät für Informatik,
VDI Fortschritt-Berichte, Reihe 10, Nr. 264, VDI-Verlag Düsseldorf 1993

[CaCH94] Campbell, Andrew; Coulson, Geoff; Hutchison, David
„A Quality of Service Architecture“
ACM Computer Communications Review, Volume No. 24, Issue 2, Seite 6-27, April 1994

[CCGH93] Campbell, Andrew; Coulson, Geoff; García, Francisco; Hutchison, David; Leopold, Helmut
„Integrated Quality of Service for Multimedia Communications“
Proceedings IEEE INFOCOM'93, Seite 732-739, San Francisco, USA, April 1993

[ClTe90] Clark, David D.; Tennenhouse, David L.
„Architectural Considerations for a New Generation of Protocols“
Proceedings ACM SIGCOMM'90, Philadelphia, September 1990

[DeBe95] Delgrossi, L.; Berger, L. (Editors)
„Internet Stream Protocol Version2 (ST2) – Protocol Specification Version ST2+“
Request For Comments (RFC) 1819, ST2 Working Group, IETF, August 1995

[Ferr90] Ferrari, Domenico
„Client Requirements for Real-Time Communication Services“
IEEE Communications Magazine, Vol. 28, Nr. 11, November 1990, S. 65-72

[GoPa92] Gong, Fengmin; Parulkar, Gurudatta
„An Application-oriented Error Control Scheme for a High-Speed Transport Protocol“
Technical Report WUCS-92-37, Washington University, USA, 1992

[Henr96] Henrich, Clemens
„Eine qualitätsorientierte Anwendungsschnittstelle für multimediale Kommunikationsdienste“
Diplomarbeit, Universität Karlsruhe, Institut für Telematik, Fakultät für Informatik, 1996

[ITU I.321] International Telecommunication Union
„B-ISDN Protocol Reference Model and its Application“
ITU-T Recommendation I.321, Genf, 1991

[JaAg90] Jain, B.N.; Agrawala, A.K.
„Open Systems Interconnections: its Architecture and Protocols“
Elsevier Science Publishers B.V., 1990

[LoKK93] Lockemann, Peter; Krüger, Gerhard; Krumm, Heiko
„Telekommunikation und Datenhaltung“
Carl Hanser Verlag, 1993

[Nahr95] Nahrstedt, Klara
„An Architecture for End-to-End Quality of Service Provision and its Experimental Validation“
Dissertation an der Universität Pennsylvania, USA, 1995

[NaSm95] Nahrstedt, Klara; Smith, Jonathan M.
„The QoS Broker“
IEEE Multimedia, Spring 1995, S. 53–67

[OSF91] O'Reilly & Associates, Inc.
„Guide to OSF/1 – A Technical Synopsis“
O'Reilly & Associates, 1991

[OSF93] Open Software Foundation
„Design of the OSF/1 Operating System – Release 1.2“
Prentice-Hall, 1993

[Part94] Partridge, Craig
„Gigabit Networking“
Addison-Wesley, 1994

[Schm96] Schmidt, Claudia
„Integrierte Managementarchitektur für qualitätsorientierte Kommunikationsdienste“
Internes Dokument, Universität Karlsruhe, Institut für Telematik, Fakultät für Informatik, 1996

[ScZi94] Schmidt, Claudia; Zitterbart, Martina
„Towards Integrated QoS Management“
Proceedings of the 5th IEEE Computer Society Workshop on Future Trend of Distributed Computing Systems, Cheju Island, Korea, August 1995

[ScZi95] Schmidt, Claudia; Zitterbart, Martina
„Reservierung von Netzwerk-Ressourcen – Ein Überblick über Protokolle und Mechanismen“
PIK – Praxis in der Informationsverarbeitung und Kommunikation, Heft 3/1995, Seite 140-147, Saur-Verlag, München

[Stro91] Stroustrup, Bjarne
„The C++ Programming Language (second edition)“
Addison-Wesley, 1991

[TaFH94] Tawbi, Wassim; Fladenmuller, Anne; Horlait, Eric
„End-System QoS Management of Multimedia Applications“
erschienen in: „Architecture and Protocols for High-Speed Networks“, Herausgeber: Spaniol O., Danthine A., Effelsberg W.
Editions Kluwer Academic Publishers, Seiten 203-213, 1994

[Tawb93] Tawbi, Wassim M.
„La Qualité de Service Dans les Systèmes de Communication Multimédia: Un Cadre d'Etude et Spécification d'un Protocole de Négocation entre Applications“
Dissertation an der Universität Pierre et Marie Curie, Paris, Frankreich, 1993

[VeZF91] Verma, Dinesh; Zhang, Hui; Ferrari, Domenico
„Delay Jitter Control for Real-Time Communication in a Packet Switching Network“
Universität von Kalifornien, Berkeley, erschienen in Proceedings Tricomm 1991

[Wett93] Wettstein, Horst
„Systemarchitektur“
Hanser Verlag, 1993

[XTPF95] XTP Forum
„Xpress Transport Protocol Specification – XTP Revision 4.0“
XTP Forum, Santa Barbara, Kalifornien, USA, 1. März 1995

[ZaFe93] Zhang, Hui; Ferrari, Domenico
„Rate-Controlled Static-Priority Queueing“
Universität von Kalifornien, Berkeley, erschienen in Proceedings IEEE INFOCOM 1993

[ZDES93] Zhang, L.; Deering, S.; Estrin, D.; Shenker, S.; Zappala, D.
„RSVP: A New Resource Reservation Protocol“
IEEE Network, September 1993

[Zimm95] Zimmermann, Peter
„Implementierung des Transportsystems PATROCLOS unter OSF/1“
Diplomarbeit, Universität Karlsruhe, Institut für Telematik, Fakultät für Informatik, 1995

[Zitt95] Zitterbart, Martina
„Hochleistungskommunikation“
Band 1: Technologie und Netze
Oldenbourg Verlag, München/Wien, 1995

Index