Ressourcenreservierung und Task-Plazierung in verteilten Multimedia-Systemen

Der Fakultät Informatik der Universität Stuttgart
zur Erlangung der Würde eines Doktors der
Naturwissenschaften (Dr. rer. nat.) vorgelegte Abhandlung

Vorgelegt von Gabriel Dermler aus Eisenmarkt

Hauptberichter:	Prof. Dr. K. Rothermel
Mitberichter:	Prof. Dr. R. Steinmetz
Tag der mündlichen Prüfung:	21.6.1999

Institut für Parallele und Verteilte Höchstleistungsrechner der Universität Stuttgart
1999

Inhaltsverzeichnis

Abkürzungsverzeichnis

AFS	Anwendungs-Flußspezifikation
A-QoS	Anwendungs-QoS
ATM	Asynchronous Transfer Mode
CINEMA	Configurable Integrated Multimedia Architecture
CPU	Central Processing Unit
EDF	Earliest Deadline First
FPP	Flowgraph Placement Protocol
HS-Modell	Host-Satellite Modell
LK	Lokaler Koordinator
MRU	Maximal Resource Usage
QoS	Quality of Service
RM	Rate Monotonic
R-QoS	Ressourcen-QoS
RSVP	Resource Reservation Protocol
ST-II	Internet Stream Protocol II
NRP	Negotiation and Resource Reservation Protocol
ZK	Zentraler Koordinator

Kurzfassung

Die Integration von Audio und Video in rechnerbasierte Anwendungen führt zu einer Reihe von Anforderungen, die nicht durch Mechanismen gelöst werden können, die für herkömmliche Daten wie Text und Graphik entwickelt wurden. Audio und Video werden als *kontinuierliche Medien* bezeichnet, da sie aus Dateneinheiten bestehen, die periodisch dargestellt werden müssen. Sie können große Datenmengen bedingen, deren Verarbeitung und Präsentation über lange Zeiträume aufrechterhalten werden muß. Eine hierfür vorgesehene Unterstützung sollte sicherstellen, daß sie Endbenutzern unter Einhaltung einer *Dienstgüte* präsentiert werden, die medienspezifische bzw. generische Charakteristika, z. B. die Bildrate einer Videowiedergabe bzw. die Verzögerung zwischen Erzeugung und Wiedergabe, für die Dauer der Präsentation festlegt.

Angesichts beschränkter Systemressourcen werden Mechanismen zur *Ressourcenreservierung* benötigt, um die Darstellung eines kontinuierlichen Mediums mit gleichbleibender Dienstgüte zu ermöglichen. Für eine verteilte Multimedia-Anwendung muß dabei eine Ressourcenreservierung sowohl zur Verarbeitung der Dateneinheiten als auch zu ihrem Transport zwischen verwendeten Rechnern vorgesehen werden.

Um den Entwurf von Multimedia-Anwendungen zu erleichtern, werden Funktionen zur Verarbeitung eines kontinuierlichen Mediums in *Anwendungskomponenten* gekapselt. Eine Multimedia-Anwendung besteht aus mehreren solcher Komponenten, die zusammen eine *Verarbeitungstopologie* bilden. Soll durch eine Anwendung ein kontinuierliches Medium bearbeitet werden, so müssen alle beteiligten Komponenten Ressourcen reservieren. Mechanismen sind erforderlich, die aus einer gewünschten Dienstgüte, die durch die Anwendung erbracht werden soll, Anforderungen für die einzelnen Komponenten ableiten und die Reservierungen der Komponenten so koordinieren, daß für die Anwendung eine möglichst hohe Dienstgüte bereitgestellt werden kann. Solche Mechanismen erfassen medienspezifische Dienstgüteparameter und koordinieren das Verhalten von Anwendungskomponenten. Sie werden daher dem *Ressourcenmanagement auf Anwendungsebene* zugerechnet.

Die Komponenten einer Anwendung werden auf den Rechnern eines verteilten Systems instanziiert. Angesichts potentiell knapper Ressourcen auf einzelnen Rechnern ist es sinnvoll, Komponenten auf diejenigen Rechner zu verteilen, die aufgrund ihrer Ressourcen eine möglichst

hohe Dienstgüte zulassen. Um in einem verteilten System geeignete Rechner zu bestimmen, sind Konzepte und Mechanismen zur *Task-Plazierung* erforderlich, die die Verfügbarkeit von Ressourcen im verteilten System erfassen und eine möglichst „gute“ Verteilung der Komponenten der Verarbeitungstopologie auf die einzelnen Rechner bestimmen können.

Der Entwurf und die Benutzung multimedialer Anwendungen wird durch eine *multimediale Systemplattform* unterstützt. Diese realisiert Dienste zur Konfiguration einer Verarbeitungstopologie aus Komponenten, zur Plazierung der Anwendungskomponenten auf Rechnern, zur Ressourcenreservierung gemäß einer spezifizierten Dienstgüte sowie zur Stromsynchronisation. In dieser Abhandlung werden die Konzepte und Verfahren der Dienste zur Task-Plazierung sowie zur Ressourcenreservierung beschrieben.

Die Entwicklung der Systemdienste beruht auf einer *Dienstgütearchitektur*, die verschiedene Abstraktionsebenen der Dienstgütespezifikation unterscheidet. Auf der Systemebene verwenden die Dienste Mechanismen, welche die Reservierung einzelner Ressourcen auf Endsystemen sowie im Netzwerk ermöglichen. Auf der Anwendungsebene realisieren sie Protokolle, die medienspezische und generische Parameter erfassen und eine Task-Plazierung bzw. Ressourcenreservierung für die Elemente eines Flußgraphen durchführen.

Als Kernelement des Dienstes zur *Ressourcenreservierung* wird ein *Protokoll* vorgestellt, welches die Ressourcenreservierung innerhalb einer Verarbeitungstopologie koordiniert. Es erlaubt eine individuelle Dienstgütevorgabe an jeder Senke der Verarbeitungstopologie und ist in der Lage, die Verfügbarkeit von Rechner- und Netzwerkressourcen sowie funktionale Einschränkungen zu berücksichtigen, die durch das Design der Komponenten definiert sind. Das Protokoll ist für eine breite Klasse von Verarbeitungstopologien anwendbar, wobei die Komponenten der Topologie beliebig auf den Rechnern eines verteilten Systems plaziert sein können.

Zur Realisierung des Dienstes zur Task-Plazierung wird ein Plazierungsmodell beschrieben, das für Multimedia spezifische Anforderungen berücksichtigt. Darauf aufbauend wird ein Plazierungsproblem formuliert, dessen Lösung ein *Plazierungsprotokoll* zum Einsammeln von Plazierungsinformation sowie verschiedene *Algorithmen* zur Berechnung optimierter Plazierungen umfaßt. Auf der Grundlage der vorgestellten Konzepte wird die Beziehung zwischen Task-Plazierung und Ressourcenreservierung dargestellt.

1 Einleitung

1.1 Multimedia-Anwendungen

Die enorme Steigerung der Leistungsfähigkeit der Rechen- und Kommunikationstechnik hat die neue Klasse multimedialer Anwendungen ermöglicht, welche die Darstellung und Manipulation herkömmlicher Daten wie Text und Graphik mit der rechnerbasierten Bereitstellung der neuen Medien Audio und Video verbindet. Das Versprechen dieser Integration ist beträchtlich: Computer und ihre Anwendungen sollen Benutzerschnittstellen verwirklichen, die den audiovisuellen Fähigkeiten von Menschen immer besser gerecht werden und neue Möglichkeiten der Interaktion zwischen Computern und ihren Benutzern eröffnen.

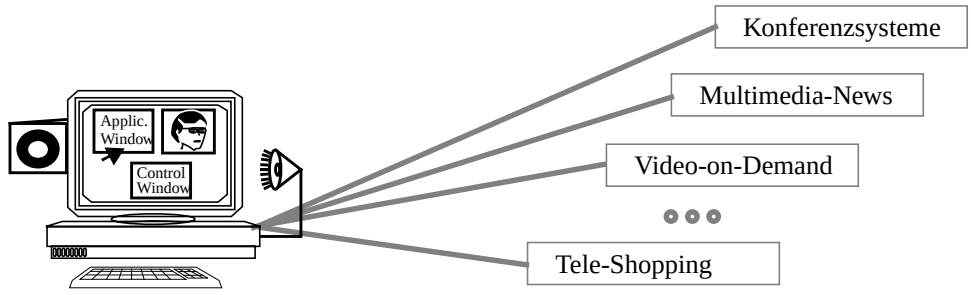


Abb. 1: *Rechner als multimediales Kommunikationsterminal*

Standen zu Beginn lokale Multimedia-Anwendungen (Spiele, Lernprogramme, Informationskioske) im Vordergrund, so gewinnen mit der zunehmenden Verfügbarkeit leistungsfähiger Kommunikationsnetze verteilte Multimedia-Anwendungen an Bedeutung. Konferenz- und CSCW[1]-Systeme zur entfernten Kommunikation zwischen Menschen sowie zur gemeinsamen Manipulation rechnerbasierter Daten ([DeFr93], [DGPR93], [SmPr97]), elektronische Post zum Transfer multimedialer Information ([BoFr93], [AhLi97]) sowie multimediale Abrufdienste wie „Television-on-Demand“, digitale Bibliotheken und „Tele-Shopping“ ([DVV94], [BLMS96]), sind einige wesentliche Schwerpunkte dieser Entwicklung. Virtuelle multimediale Welten reichern zunehmend die Informationsdarstellung und die Benutzerinteraktion in diesen Bereichen

[1] Computer Supported Cooperative Work [Wils91].

an ([SMK96], [HaSm96]) und weisen den Weg zu völlig neuen Anwendungen wie beispielsweise verteilten interaktiven Kinos ([NaTo97]).

1.2 Eigenschaften und Anforderungen kontinuierlicher Medien

Dem potentiellen Nutzen multimedialer Anwendungen steht eine Reihe von Herausforderungen gegenüber, die sich aus den besonderen Eigenschaften von Audio und Video ergeben und die bei ihrer Integration in digitale Rechen- und Kommunikationssysteme berücksichtigt werden müssen.

Ein wichtiges Merkmal von Audio und Video ist ihre *Zeitabhängigkeit*. Anders als herkömmliche Daten wie Text und Graphik, beinhalten sie Information, die sich kontinuierlich verändert, weswegen sie auch als kontinuierliche Medien bezeichnet werden [Herr91]. Diese werden in digitalisierter Form als ein Strom periodischer Mediendateneinheiten (z. B. von Videobildern) dargestellt, in dem die Information nicht nur in den Dateneinheiten gegeben ist, sondern auch durch deren zeitliche Abstände definiert wird. Eine Veränderung dieser Abstände kann die Informationswiedergabe verfälschen, wie dies beispielsweise durch das „ruckweise“ Abspielen eines Films geschehen würde. Anders als für nicht-kontinuierliche Daten, stehen für Audio und Video daher nicht mehr nur eine möglichst schnelle, sondern vor allem auch eine „rechtzeitige“ (d.h. weder zu frühe noch zu späte) Handhabung von Dateneinheiten im Vordergrund, die den periodisch anfallenden Dateneinheiten gerecht wird.

Audio und Video beinhalten Information, die menschlichen Benutzern mit einer bestimmten Qualität - oder *Dienstgüte* [VKBG95] - präsentiert wird. Diese umfaßt zum einen medienspezifische Merkmale wie beispielsweise Größe und Rate der Bilder eines darzustellenden Videos oder die über Abtastrate und Genauigkeit der Abtastwerte steuerbare Qualität eines Audiostroms. Weitere Angaben beschreiben „Ende-zu-Ende“-Eigenschaften des Datentransfers, wie die Verzögerung oder den Datenverlust zwischen Erzeugung und Präsentation von Dateneinheiten eines kontinuierlichen Datenstroms. Während medienspezifische Qualitätsmerkmale im Zusammenhang mit jeder Medienpräsentation zu beachten sind, spielt Verzögerung vor allem in Anwendungsszenarien eine Rolle, in denen Interaktion mit geringer Verzögerung, z. B. zwischen Teilnehmern einer Telekonferenz, stattfinden muß. Ein Multimedia-

System muß für eine Anwendung sicherstellen, daß eine Präsentation mit der erforderlichen Dienstgüte stattfindet.

Kontinuierliche Medien können *große Datenmengen* verursachen mit entsprechend hohen Anforderungen an die Leistungsfähigkeit verwendeter Rechner- und Kommunikationsressourcen. Während Audio in hoher (z. B. CD-) Qualität Datenraten von bis zu 1.5 Mbit/sec bedingt, erfordert hochqualitatives Video Datenraten in der Größenordnung von mehreren 100 Mbit/sec [StNa95]. Mit Hilfe verschiedener Verfahren können Audio und Video komprimiert werden und somit geringere, in der Regel immer noch erhebliche Datenraten verursachen. So erfordert Video, das nach dem MPEG-1-Standard kodiert wird, Datenraten von bis zu 1.8 Mbit/sec [ISO93], während die neuere MPEG-2-Norm Datenraten von bis zu 80 Mbit/sec vorsieht [ISO94].

Ein Video mit einer erhöhten Bildauflösung oder Bildrate stellt in einem weiten Bereich menschlicher Wahrnehmungsfähigkeit ebenso einen Gewinn dar wie eine Audiosequenz, die mit erhöhter Abtastrate und Abtastgenauigkeit generiert wird. Auf der anderen Seite ist eine reduzierte Qualität häufig auch dann akzeptabel, wenn die optimal gewünschte nicht möglich ist. Die *skalierbare Einstellung der Dienstgüte* ermöglicht den Einsatz von Audio und Video häufig auch in Umgebungen, die mit einer begrenzten Rechen- bzw. Kommunikationsfähigkeit ausgestattet sind. Dies ist insbesondere bei dem Einsatz von Video von Bedeutung, dessen Datenrate einerseits enorm sein kann, andererseits in einem sehr weiten Bereich (z. B. von 2 bis 80 Mbit/sec für MPEG-2 Video) eingestellt werden kann.

Aufgrund ihrer Anforderungen an eine rechtzeitige Behandlung von Mediendaten, erfordern Multimedia-Systeme Verfahren, die traditionell in *Echtzeitsystemen* verwendet wurden [BuLi91]. Hierzu gehören insbesondere Scheduling-Verfahren, die rechtzeitige Aufrufe von Verarbeitungs- und Kommunikationsfunktionen ermöglichen sowie Verfahren zur Reservierung von Systemressourcen, um eine erforderliche Dienstgüte gewährleisten zu können. Anders jedoch als traditionelle Echtzeit-Anwendungen, beispielsweise aus dem Bereich der Fabrikautomation, weisen Multimedia-Anwendungen eine gewisse Toleranz gegenüber sporadischen Datenverlusten auf, sofern sie für den Endbenutzer nicht als störend wahrnehmbar sind.

1.3 Bereitstellung von Dienstgüte

1.3.1 Reservierung von Ressourcen

Die Manipulation und Übertragung kontinuierlicher Medien erfordert die Bereitstellung von Ressourcen sowohl auf Endsystemen als auch im Netzwerk. Auf Endsystemen werden zum einen Ressourcen eingesetzt, die auch zur Handhabung nichtkontinuierlicher Daten verwendet werden, darunter Dateisysteme, die CPU, Hauptspeicher und Netzwerkadapter. Diese werden durch medienspezifische Geräte ergänzt, die z. B. zur Generierung bzw. zur Ausgabe kontinuierlicher Medien benötigt werden (Kameraadapter, Audioadapter, etc.). Vom Netzwerk werden Verbindungen zwischen Endsystemen erwartet, die eine erforderliche Kommunikationsbandbreite bereitstellen.

Heutige Multimedia-Systeme weisen potentiell knappe Ressourcen auf [StNa95]. Lediglich für einfache Szenarien kann eine „Überversorgung“ mit Ressourcen angenommen werden, so z. B. zur Ausgabe eines gespeicherten Audiostromes auf einem Endsystem. Viele Multimedia-Szenarien haben jedoch eine weitergehende Komplexität. Zum einen können Anwendungen einen höheren Verarbeitungsaufwand bedingen, wenn sie weitere Funktionen zur Medienverarbeitung wie z. B. zur Kompression oder Formatkonvertierung benutzen. Ferner könnte in einer Anwendung mehr als nur ein kontinuierlicher Strom verwendet werden und ein Endbenutzer könnte auf seinem Rechner mehr als nur eine Anwendung gleichzeitig benutzen wollen. Nicht zuletzt werden Multimedia-Server sowie Netzwerke eine Vielzahl gleichzeitig laufender Anwendungen für verschiedene Endbenutzer unterstützen müssen, wie dies beispielsweise im Fall von „Video-on-Demand“ gegeben ist.

Angesichts potentieller Ressourcenknappheit erfordert die Bereitstellung von Dienstgüte eine geeignete Unterstützung durch das darunterliegende System. Zwei Klassen von Systemen sind in dieser Hinsicht unterscheidbar [CSZ92]. Eine Klasse unterstützt sogenannte adaptive Anwendungen, die gewisse Schwankungen der Dienstgüte in Abhängigkeit von der Verfügbarkeit von Systemressourcen tolerieren. Das in [SCFJ95] vorgestellte adaptive Transportsystem sowie das in [FSVW96] beschriebene „Fast Web Project“ sind Beispiele hierfür. Ein wesentlicher Vorteil solcher Systeme besteht darin, daß sie existierende Netzwerke und Betriebssysteme

umfassen können, die für einzelne Anwendungen keine gleichbleibende Ressourcenverfügbarkeit garantieren.

Um für eine Anwendung eine gleichbleibende Dienstgüte bereitzustellen, müssen Ressourcen reserviert werden, bevor sie zur Bearbeitung kontinuierlicher Medien verwendet werden [CSZ92]. Die Reservierung einer Ressource ordnet einer Anwendung eine bestimmte Kapazität dieser Ressource zu, die nicht durch andere, gleichzeitig laufende Anwendungen benutzt werden kann. Insbesondere ist die Reservierung von CPU-Kapazität Voraussetzung für echtzeitfähige Scheduling-Verfahren, die eine periodische Ausführung von Verarbeitungs- und Kommunikationsfunktionen mit begrenzter Verzögerung ermöglicht [LiLa73]. Ähnlich ist die Reservierung von ausreichend Hauptspeicher bzw. Netzwerkbandbreite Voraussetzung dafür, daß anfallende Mediendateneinheiten in einem Endsystem bzw. im Netzwerk nicht wegen fehlender Ressourcen verworfen werden müssen [BrZi96].

Systeme, die Ressourcenreservierung unterstützen, wurden sowohl auf der Netzwerk- als auch auf der Anwendungsebene realisiert ([ZDES93], [Wolf96]). Im letzteren Fall muß eine Reservierung alle Ressourcen umfassen, die auf Endsystemen und im Netzwerk zur Verarbeitung und Kommunikation benötigt werden und die potentiell mit einer Knappheit ihrer Kapazität konfrontiert sein können. Die im Rahmen dieser Arbeit beschriebene Systemunterstützung für Multimedia-Anwendungen ermöglicht eine gleichbleibende Dienstgüte und integriert die Reservierung aller benötigten Ressourcen auf Endsystemen sowie im Netzwerk.

1.3.2 Ressourcenreservierung auf Anwendungsebene

Rechnerbasierte Multimedia-Anwendungen beinhalten verschiedene Komponenten, die unterschiedliche Funktionen zur Manipulation von Stromdaten realisieren. Komponenten stellen entsprechenden funktionsspezischen Programmcode bereit und machen von benötigten Ressourcen Gebrauch. Beispielsweise wird eine Quellkomponente für gespeichertes Video typischerweise ein Dateisystem benutzen und Code zum periodischen Einlesen von Bilddaten beinhalten, der sowohl CPU- als auch Speicherkapazität benötigt. Weitere Komponenten sind für eine Vielzahl unterschiedlichster Funktionen verwendbar, so z. B. zur Speicherung, Kompression, Präsentation oder zum Transport kontinuierlicher Daten.

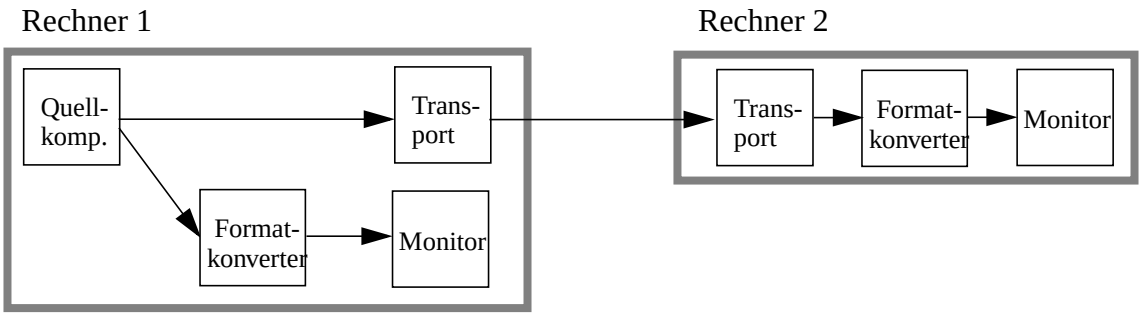


Abb. 2: *Struktur einer Multimedia-Anwendung*

Komponenten zur Bearbeitung kontinuierlicher Medien werden unabhängig voneinander bereitgestellt. Zum einen erlaubt dies eine vereinfachte und spezialisierte Entwicklung einzelner Funktionen, die jeweils eigene Algorithmen (z. B. zur Kompression von Video, zur Ausgabe von Video auf einem Monitor) sowie Schemata zur Benutzung von Ressourcen realisieren können. Zum anderen erhalten Multimedia-Anwendungen die Möglichkeit, benötigte Komponenten auszuwählen und in die angestrebte Datenmanipulation zu integrieren. Nicht zuletzt erzwingt die Verteilung einer Multimedia-Anwendung das Vorhandensein separierter Komponenten auf unterschiedlichen Rechnern.

Als Beispiel ist in Abb. 2 eine Anwendung dargestellt, die einen Videostrom für zwei entfernte Endbenutzer liefert und Komponenten zur Generierung, Konvertierung, Kommunikation sowie zur Darstellung eines Videostromes benutzt. Mit Hilfe der zwei Konverterkomponenten wird für die zwei Endbenutzer die jeweils benötigte Bildgröße bereitgestellt. Eine solche Anwendung kann beispielsweise das in [DGOR94], [DGOP94] beschriebene Szenario realisieren, das das gemeinsame Betrachten multimedialer Information im Rahmen einer Telekonferenz vorsieht, in der Endbenutzer unterschiedlich mächtige Endsysteme benutzen.

Um für eine Anwendung eine bestimmte Dienstgüte zu erzielen, ist es nötig, daß jede der benutzten Komponenten die benötigten Ressourcen im erforderlichen Maß reserviert. Die Konsequenzen dieser Forderung sind vielfältig. So müssen Endbenutzer einer Anwendung die Möglichkeit erhalten, ihre Wünsche bezüglich der zu erzielenden Dienstgüte zu spezifizieren und diese Spezifikationen müssen in Anforderungen an die einzelnen Komponenten umgesetzt wer-

den. Komponenten wiederum müssen in der Lage sein, ihren Ressourcenbedarf (an Speicher, CPU-Verarbeitung, Kommunikationsbandbreite, etc.) entsprechend zu berechnen und zu belegen.

Eine Reihe von Maßnahmen ist erforderlich, um konsistente Reservierungen durch Komponenten sowie eine möglichst hohe Dienstgüte für Endbenutzer zu erzielen. Angesichts potentiell knapper Ressourcen müssen Mechanismen vorhanden sein, die die Zuteilung der Ressourcen an die Komponenten so koordinieren, daß der Anwendung insgesamt eine hohe Dienstgüte ermöglicht wird. Falls eine gewünschte Dienstgüte nicht erreichbar ist, sollten sie in der Lage sein, die Berechnung einer erzielbaren Dienstgüte und entsprechende Ressourcenreservierungen zu veranlassen oder falls nicht anders möglich, die Ressourcenreservierung für die Anwendung abzulehnen. Als Ergebnis muß insgesamt eine konsistente Einstellung aller Komponenten hinsichtlich der zu unterstützenden medienspezifischen Eigenschaften gegeben sein, so daß zwischen Komponenten ausgetauschte Mediendateneinheiten auf beiden Seiten die gleichen erwarteten Merkmale (z. B. Bildgröße) aufweisen.

Die Bereitstellung solcher Mechanismen unterscheidet sich wesentlich von dem Management einzelner Ressourcen. Während die Reservierung einer Ressource sich auf die Bereitstellung entsprechender Kapazität für einen Ressourcenbenutzer bezieht, erfordern die beschriebenen Koordinationsaufgaben Mechanismen, die Einstellungen der Dienstgüte auf der Ebene der Anwendung betrachten und diese sowohl zwischen der Anwendung und ihren Endbenutzern als auch zwischen den einzelnen Komponenten der Anwendung unter Beachtung der Verfügbarkeit benötigter Ressourcen abstimmen. Entsprechend werden diese Mechanismen als Ressourcenmanagement auf Anwendungsebene bezeichnet.

1.3.3 Task-Plazierung

Angesichts potentiell knapper Ressourcen kann es vorkommen, daß bestimmte Rechner nicht die nötigen Ressourcen aufweisen, die zur Ausführung multimedialer Verarbeitungsfunktionen nötig sind. Beispielsweise kann der in Abb. 2 dargestellte Konverter auf Rechner 2 einen hohen Ressourcenbedarf bedingen, so daß in der Konsequenz auf diesem Rechner die Reduzierung der Dienstgüte evtl. bis auf ein Minimum erforderlich ist. Eine Plazierung dieser Komponente auf dem Quellrechner wäre sinnvoll, falls dadurch der überforderte Rechner entlastet und Rechner

1 aufgrund einer hohen Ressourcenverfügbarkeit eine hohe Dienstgüte für beide Endbenutzer ermöglichen würde.

Die Lokationen von Anwendungskomponenten sind häufig nicht von vornherein gegeben. Im obigen Beispiel war die Wahl der Lokation für die Konverterfunktion offen gelassen, weil sie auf beiden Rechnern unterstützbar war. Weitere Szenarien alternativer Plazierungen sind denkbar. Beispielsweise ist eine variable Wahl von Quellrechnern dann gegeben, wenn ein kontinuierliches Medium in abgespeicherter Form auf mehreren Servern bereitliegt, so daß die Auswahl des Servers unter dem Aspekt einer Dienstgüteoptimierung erfolgen kann. Um sowohl Quell- als auch Senkenrechner zu entlasten, können ferner zwischengelagerte Rechner verwendet werden, die ebenfalls Anwendungskomponenten unterstützen können ([Sren96], [SSJH96], [AMZ95]). Die Wahl eines von mehreren möglichen Rechnern kann hier ebenfalls eine Verbesserung der erzielbaren Dienstgüte bewirken.

Um diese potentiellen Verbesserungen der Dienstgüte zu erreichen, sind Verfahren nötig, die die Plazierung von Anwendungskomponenten auf Rechnern eines verteilten Systems - neben der Ressourcenreservierung - als weitere Dimension der Bereitstellung von Dienstgüte miteinbeziehen.

1.4 Multimediale Systemplattformen

Ziel der vorliegenden Abhandlung ist die Beschreibung von Konzepten und Verfahren, die eine Bereitstellung von garantierter und möglichst hoher Dienstgüte für Multimedia-Anwendungen ermöglichen. Solche Verfahren müssen einerseits von Funktionen Gebrauch machen, die von existierenden Betriebs- und Kommunikationssystemen zur Verfügung gestellt werden, einschließlich CPU-Scheduling-Verfahren, Mechanismen zum Zugriff auf Dateisysteme, Ein- und Ausgabegeräte sowie Reservierungsprotokollen, die zum Aufbau von Netzwerkverbindungen bereitgestellt werden.

Auf der anderen Seite sollen die Verfahren den Benutzern von Anwendungen eine möglichst einfache Sicht auf die Bereitstellung von Dienstgüte liefern, die sowohl von Reservierungsinteraktionen mit Ressourcen der Rechen- und Kommunikationsumgebung abstrahiert als auch die nötigen Koordinationsmechanismen zur Ressourcenreservierung und Task-Plazierung ver-

birgt. Die hierzu benötigten anwendungsorientierten Funktionen werden in aller Regel nicht von generischen Betriebssystem- und Kommunikationssystemdiensten angeboten, sondern müssen durch spezielle multimediale Systemplattformen realisiert werden.

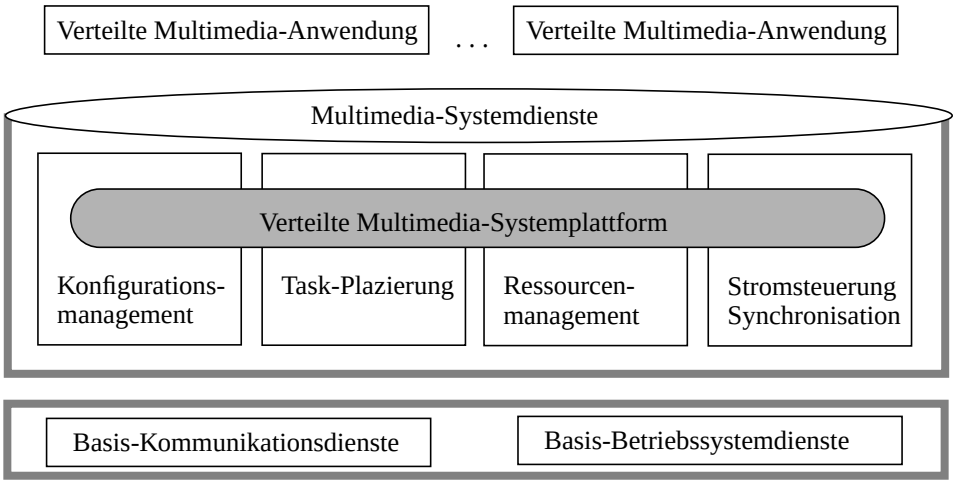


Abb. 3: *Dienste einer multimedialen Systemplattform*

Allgemein bestehen solche Plattformen aus einer Reihe von Diensten, die ihren Klienten verschiedene Funktionen zum Umgang mit Multimedia-Anwendungen anbieten ([IMA93], [RBH95]). Solche Dienste benutzen die Funktionen der Systemumgebung und ergänzen sie um anwendungsorientierte, multimediaspezifische Mechanismen. Zugleich realisieren sie für ihre Klienten Schnittstellen, die über geeignete Abstraktionen die Verwendung der Dienstfunktionalität ermöglichen. In Abb. 3 ist das Dienstkonzept der CINEMA-Plattform[2] ([BDH93], [BHR95], [RBH94]) dargestellt, die in den letzten Jahren an der Universität Stuttgart entwickelt wurde und den Rahmen für diese sowie für weitere Forschungsarbeiten bildet.

1.4.1 Die CINEMA-Systemplattform

CINEMA ([BDHR93], [RBH94], [BHR95]) stellt eine Systemplattform bereit, die sowohl den Entwurf als auch die Benutzung von Multimedia-Anwendungen ermöglicht. Die von CINEMA

[2] Configurable Integrated Multimedia Architecture

realisierte Unterstützung umfaßt dabei Funktionen zur Spezifikation und Instanziierung von Anwendungen auf Rechnern eines verteilten Systems, zur Berechnung günstiger Lokationen für Anwendungskomponenten, zur Reservierung von Ressourcen sowie zur Steuerung und Synchronisation von Datenströmen.

Diese Funktionen werden Klienten von CINEMA über vier multimediale Dienste zugänglich gemacht. Über das *Konfigurationsmanagement* ([Bart98]) erhält ein Klient die Möglichkeit, Verarbeitungstopologien (siehe Abb. 2) zur Manipulation kontinuierlicher Medien zu spezifizieren, wobei die Verarbeitungskomponenten durch Klienten vorgegebenen Rechnern zugeordnet werden müssen. Im Unterschied hierzu bietet der CINEMA-Dienst zur *Task-Plazierung* ([DeIq98]) die Möglichkeit, spezifizierte Verarbeitungstopologien automatisch durch CINEMA innerhalb eines verteilten Systems plazieren zu lassen. Hierbei wird eine Plazierung von Komponenten angestrebt, die aufgrund der zu einem Zeitpunkt gegebenen Ressourcenverfügbarkeit im verteilten System eine möglichst hohe Dienstgüte verspricht. Das *Ressourcenmanagement* ([DFR96], [DeFi96], [DFR97], [RDF97]) ist für die letztliche Aushandlung der Dienstgüte und für die Reservierung von Ressourcen verantwortlich. Auf diese Weise wird sichergestellt, daß die Qualität der Präsentation kontinuierlicher Medien über die Lebensdauer von Anwendungen ohne Schwankungen aufrechterhalten werden kann.

Während die genannten Dienste wesentliche Funktionen bereitstellen, die vor Benutzung einer Anwendung ausgeführt werden, dient der letzte Dienst zur *Stromsteuerung und Stromsynchronisation* ([Helb94], [RoHe95], [RoHe96]) der Kontrolle einer laufenden Anwendung. So bietet der Dienst die Möglichkeit, den Datenfluß von Strömen (durch Kommandos wie Start, Stop, Rücklauf, etc.) zu steuern sowie Synchronisationsbeziehungen zwischen mehreren Strömen ([RoDe92]) zu spezifizieren und zu erzwingen.

1.4.2 Dienste zur Ressourcenreservierung und Task-Plazierung

Im Rahmen dieser Abhandlung werden Konzepte und Verfahren zur Ressourcenreservierung und Task-Plazierung für verteilte Multimedia-Anwendungen in CINEMA beschrieben. Sie umfassen zum einen die Dienstabstraktionen, die Klienten angeboten werden, um verteilte Multimedia-Anwendungen mit garantierter Dienstgüte zu instanziieren. Zum anderen werden die Verfahren ausführlich dargestellt, die zur Durchführung der Ressourcenreservierung und Task-

Plazierung eingesetzt werden müssen. In einem Überblick zusammengefaßt, behandeln und lösen diese vor allem folgende Anforderungen.

Der Begriff von Dienstgüte bei der Reservierung von Ressourcen (z. B. CPU oder Netzwerkkapazität) ist jeweils ressourcenspezifisch definiert. Vor allem ist er auf einer anderen Abstraktionsebene angesiedelt als die Dienstgüte, die Klienten von einer Multimedia-Anwendung erwarten (z. B. Bildgröße, Bildrate) und von Komponenten einer Multimedia-Anwendung bereitgestellt wird. Um diese Begriffe zu separieren und zueinander in Beziehung zu setzen, wird eine Dienstgütearchitektur benötigt, die zugleich die Grundlage für eine Strukturierung des benötigten Ressourcenmanagements bildet.

Um Information über die gewünschte Dienstgüte sowie die von Anwendungskomponenten unterstützbare Dienstgüte zu erfassen, auszutauschen und abzugleichen sind Reservierungsprotokolle erforderlich, die für die Propagierung und den Abgleich der Information in der gesamten Verarbeitungstopologie sorgen. Solche Protokolle müssen mit den Reservierungsmechanismen der einzelnen Ressourcen verzahnt werden, da deren Verfügbarkeit ein entscheidender Faktor bei der Bereitstellung von Dienstgüte ist. Ferner müssen Reduzierungen der Dienstgüte unterstützbar sein, um bei Ressourcenmangel den Benutzern einer Anwendung eine erzielbare Dienstgüte bereitzustellen.

Zur Unterstützung der Task-Plazierung in verteilten Multimedia-Systemen ist sowohl Information über die Verfügbarkeit von Ressourcen im verteilten System als auch über den Ressourcenbedarf für verschiedene Plazierungsalternativen nötig. Je nach vollständiger oder teilweiser Verfügbarkeit solcher Information können Plazierungsverfahren mit unterschiedlicher Mächtigkeit entwickelt werden. Bei gegebener Information sind Plazierungsalgorithmen erforderlich, die eine möglichst optimale Entscheidung im Hinblick auf die erzielbare Dienstgüte liefern. Nicht zuletzt muß das Verhältnis zwischen Task-Plazierung und Ressourcenreservierung spezifiziert werden, da auch im Falle der Task-Plazierung die Instanziierung einer Anwendung mit garantierter Dienstgüte erforderlich ist.

1.5 Einordnung der Arbeit

Die Entwicklung und Bewertung von Protokollen zur Ressourcenreservierung auf der Anwendungsebene sowie von Verfahren zur Task-Plazierung stellen den Hauptbeitrag dieser Arbeit dar. Diese wurden im Rahmen der CINEMA-Systemplattform als Kern der entsprechenden Dienste realisiert und ermöglichen die Plazierung und Instanziierung verteilter Multimedia-Anwendungen, die Verarbeitungstopologien unterschiedlichster Komponenten umfassen können (siehe Abb. 2). Die Plazierungsverfahren zielen auf eine Wahl von Lokationen für Anwendungskomponenten ab, die eine möglichst hohe Dienstgüte zulassen. Die Reservierungsprotokolle sorgen (für plazierte Anwendungen) dafür, daß alle Ressourcen, die zur Verarbeitung und Kommunikation in einer Anwendung erforderlich sind, reserviert werden und somit eine garantierte Dienstgüte bereitgestellt werden kann.

Probleme der Ressourcenreservierung wurden in der Vergangenheit für verschiedene Ressourcen sowohl auf Endsystemen als auch für den Aufbau von Netzwerkverbindungen untersucht [StNa95]). Beispielsweise wurden Scheduling-Verfahren für die CPU ([LiLa73], [Ande93], [MST94]), für multimediale Dateisysteme ([LoSh93], [GVKR95]) und Protokolle zur Ressourcenreservierung auf der Netzwerkebene realisiert ([ZDES93]). Die im Rahmen dieser Arbeit vorgestellten Reservierungsprotokolle der Anwendungsebene setzen die Verfügbarkeit solcher Mechanismen voraus.

Das Ressourcenmanagement auf der Anwendungsebene hat die Reservierung aller Ressourcen zum Ziel, die von einer Anwendung für eine bestimmte Dienstgüte benötigt werden. Frühere Arbeiten haben verschiedene Aspekte dieser Aufgabe behandelt. So wurden unterschiedliche Abstraktionsebenen zur Bereitstellung von Dienstgüte in Multimedia-Systemen beschrieben ([VKBG95]) und in entsprechende Strukturen des Ressourcenmanagements integriert ([StNa95], [Wolf96]). Verfahren wurden entwickelt, die den Ressourcenbedarf (z. B. CPU-Zeit, Speicher) einzelner Anwendungskomponenten ableiten und ihre Gruppierung in Verarbeitungseinheiten der CPU bestimmen können ([Wolf96], [Bart98]). Zur Durchführung der Reservierung für eine gesamte Anwendung wurden ferner verschiedene Ansätze vorgeschlagen ([NaSm95], [Wolf96], [HuSc97], [HWVC97]).

Im Vergleich zu den erarbeiteten Protokollen dieser Abhandlung weisen diese Ansätze eine Reihe von Unterschieden auf. So sehen sie Dienstgüteanforderungen vor, die einheitlich für alle Endbenutzer einer Anwendung spezifiziert und eingestellt werden. Das Konzept der „heterogenen Empfänger“, das eine individuelle Einstellung für jeden Endbenutzer erlaubt und im Kontext von Netzwerkprotokollen vorgeschlagen wurde [ZDES93], wird auf der Anwendungsebene nicht unterstützt.

Des weiteren betrachten die Ansätze keine funktionalen Einschränkungen, die für Komponenten aufgrund ihres Entwurfs vorgegeben sind. Beispielsweise beachten sie die mögliche Beschränkung einer Komponente nicht, lediglich Bildformate bestimmter Größen verarbeiten zu können. Nicht zuletzt weisen sie in der Regel Einschränkungen im Hinblick auf die unterstützten Verarbeitungstopologien auf oder machen einschränkende Annahmen über die Plazierung der Anwendungskomponenten auf den Rechnern eines verteilten Systems.

Im Gegensatz hierzu lassen die nachfolgend vorgestellten Reservierungsprotokolle sowohl die Verwendung heterogener Empfänger als auch die Berücksichtigung funktionaler Einschränkungen zu. Die Protokolle unterstützen komplexe Verarbeitungstopologien, die Multicast- (siehe Abb. 2), Misch- sowie Konferenzszenarien abdecken. Mischszenarien beinhalten die Versorgung einer Senke mit Strömen mehrerer Quellen, während Konferenzszenarien die Darstellung der Ströme mehrerer Quellen an mehreren Senken umfassen können. In allen Fällen sind zwischengelagerte Komponenten zwischen den Quellen und Senken einer Verarbeitungstopologie konfigurierbar. Insbesondere sind die entwickelten Protokolle bei beliebiger Verteilung von Komponenten auf den Rechnern eines verteilten Systems anwendbar.

Verfahren zur Task-Plazierung wurden in der Vergangenheit im Kontext verschiedener Aufgabenstellungen entwickelt [Gosc91]. Insbesondere wurden zur Plazierung paralleler Programme in Multiprozessorumgebungen bzw. verteilten Systemen aus Modulen aufgebaute Programmgraphen betrachtet, die multimedialen Verarbeitungstopologien ähnlich sind ([NoTh93]). Im Kontext multimedialer Anwendungen wurde die Task-Plazierung dagegen bislang nicht untersucht.

Ein wesentliches Ergebnis dieser Arbeit stellt die Beschreibung eines Plazierungsproblems für verteilte Multimedia-Anwendungen dar, das, anders als existierende Aufgabenstellungen aus

dem Bereich paralleler Programme und der Lastbalancierung, für Multimedia spezifische Anforderungen berücksichtigt. Zur Lösung dieses Plazierungsproblems wird eine neuartige Lösung beschrieben, die sowohl ein Protokoll zum Einsammeln von Plazierungsinformation umfaßt als auch verschiedene Algorithmen zur Berechnung günstiger Plazierungen für Anwendungskomponenten beinhaltet und im Vergleich bewertet.

1.6 Gliederung der Abhandlung

Die weiteren Kapitel der Abhandlung sind wie folgt unterteilt. Kapitel 2 beschreibt die Abstraktionen, die zur Instanziierung einer Multimedia-Anwendung mit spezifizierter Dienstgüte angeboten werden. Nach einer Darstellung der Modellierung von Multimedia-Anwendungen und ihrer Dienstgüte-Unterstützung, wird das Konzept der CINEMA-Session eingeführt, die die Einheit zur Anwendungsinstanziierung und Ressourcenreservierung in CINEMA darstellt und (min, max)-Vorgaben von Klienten in Bezug auf die zu erzielende Dienstgüte erlaubt.

Kapitel 3 führt in einem ersten Teil eine Dienstgütearchitektur ein, die die verschiedenen, für Multimedia-Anwendungen relevanten Dienstgüte-Ebenen erfaßt. Darauf aufbauend wird eine Struktur des Ressourcenmanagements entwickelt, die Managementaufgaben in Bezug auf einzelne Ressourcen, Reservierungsprotokolle auf der Anwendungsebene sowie die Kopplung der Mechanismen dieser Ebenen unterscheidet und motiviert. In einem zweiten Teil wird in einem Überblick das Management der CPU und des Hauptspeichers in CINEMA beschrieben und gezeigt, auf welche Weise der entsprechende Ressourcenbedarf vorgegeben bzw. abgeleitet werden kann.

Kapitel 4 beinhaltet eine Zusammenstellung der Anforderungen an die Ressourcenreservierung auf Anwendungsebene, die sich aus dem Session-Modell von CINEMA ergeben. Reservierungsprotokolle auf der Transport- bzw. Netzwerkebene werden im Hinblick auf diese Anforderungen genauso diskutiert wie Ansätze zur Ressourcenreservierung auf der Anwendungsebene, die im Kontext anderer Projekte entwickelt wurden.

Kapitel 5 stellt eine Protokollfamilie zur Ressourcenreservierung vor, die zum Aufbau von CINEMA-Sessions eingesetzt wird. Das Grundprotokoll, das als „Negotiation and Resource Reservation Protocol“ (NRP) bezeichnet wird, ist in der Lage eine CINEMA-Session entspre-

chend (min, max)-Dienstgütevorgaben eines Klienten aufzubauen. Es beruht auf einer verteilten Protokollmaschine und führt die Ressourcenreservierung für baumartige Anwendungstopologien in einer dezentralen Weise aus. Um eine Optimierung der erzielbaren Dienstgüte bei Ressourcenknappheit zu erreichen, werden zwei weitere Protokolle vorgestellt. Das NRP-Min Protokoll baut eine Session gemäß einer minimalen Dienstgütevorgabe auf, während NRP-LK zur Einstellung einer möglichst optimalen Dienstgüte zusätzliche Koordinationseinheiten auf den Endsystemen vorsieht. Die Eigenschaften sowie die Leistungsfähigkeit der Protokolle werden anschließend diskutiert und einander gegenüber gestellt.

In Kapitel 6 werden zunächst grundlegende Aspekte beschrieben, die bei dem Entwurf von Verfahren zur Task-Plazierung für Multimedia-Anwendungen von Bedeutung sind. So werden die Elemente des Plazierungsproblems charakterisiert und es werden allgemeine Anforderungen an die Lösung der Task-Plazierung abgeleitet. Um diese Anforderungen zu erfüllen, wird das Problem der Task-Plazierung im Kontext eines sogenannten „Host-Satellite“-Modells formuliert, das auf eine Vielzahl von Multimedia-Szenarien anwendbar ist. Das Modell liefert Information über den Ressourcenbedarf von Plazierungsalternativen sowie über die Verfügbarkeit von Ressourcen im verteilten System. Anschließend wird die Anwendbarkeit existierender Plazierungsalgorithmen aus dem Bereich paralleler Programme auf dieses Modell diskutiert.

Kapitel 7 stellt das „Flowgraph Placement Protocol“ (FPP) zur Lösung des „Host-Satellite“-Problems vor. Zunächst werden die Mechanismen des Protokolls beschrieben, die zum Einsammeln von Information über den Ressourcenbedarf sämtlicher Plazierungsalternativen sowie der verfügbaren Ressourcen im verteilten System benötigt werden. Anschließend werden zur Berechnung optimierter Plazierungen eine Heuristik sowie ein exaktes Verfahren vorgestellt und im Hinblick auf ihre Leistungsfähigkeit bewertet.

Die wesentlichen Resultate dieser Arbeit werden in Kapitel 8 zusammengefaßt. Abschließend werden Richtungen diskutiert, in denen eine weitere Forschung sinnvoll erscheint.

2 Modellierung verteilter Multimedia-Anwendungen

Um verteilte Multimedia-Anwendungen einfach entwerfen und benutzen zu können, werden Abstraktionen benötigt, welche die Konfigurierung einer Anwendung sowohl hinsichtlich ihrer Struktur als auch der Dienstgüteunterstützung ermöglichen. Die Abstraktionen bilden die Basis für die Schnittstellen der CINEMA-Dienste sowie für die Protokolle und Mechanismen, die innerhalb der Dienste eingesetzt werden. In diesem Kapitel werden zunächst die allgemeinen Konzepte zum Aufbau verteilter Multimedia-Anwendungen beschrieben, die in ähnlicher Form auch in anderen Projekten ([IMA93], [PREM96], [CBRS93], [KHSM95]) verwendet wurden. Anschließend wird ein verfeinertes Modell eingeführt, das für das Ressourcenmanagement von CINEMA grundlegend ist und insbesondere einen Session-Begriff zur Instanziierung einer Anwendung und zur Ressourcenreservierung beinhaltet.

2.1 Konfiguration von Anwendungen

Verteilte Multimedia-Anwendungen dienen zur Erzeugung, Manipulation und Präsentation kontinuierlicher Datenströme, die zwischen verteilten Rechnern ausgetauscht werden. Eine solche Anwendung wird in CINEMA als ein Graph modelliert, der aus Verarbeitungsbausteinen besteht, die über gerichtete Kanten miteinander verbunden sind (siehe Abb. 4). Verarbeitungsbausteine, in CINEMA *Komponenten* genannt, kapseln Verarbeitungsfunktionen für Stromdaten ein. Verschiedene Komponententypen sind unterscheidbar.

Quellkomponenten dienen zur Erzeugung von Datenströmen, so z. B. eine Video-Komponente, die periodisch Bilddaten aus einer Kamera einliest und bereitstellt. Senkenkomponenten dienen der Präsentation (allgemein zum Verbrauch) ankommender Stromdaten. Z. B. kann eine Video-Senke periodisch Bilder auf einem Monitor darstellen. Zwischengelagerte Komponenten werden zur Manipulation von Stromdaten zwischen Quellen und Senken verwendet. Zwei Gruppen werden unterschieden. Filter dienen zur Verarbeitung eines einzigen Datenstroms, wie dies z. B. zur Verkleinerung der Darstellungsgröße eines Videos nötig ist. Mischer werden zur gemeinsamen Verarbeitung mehrerer Ströme benutzt, z. B. zum Mischen mehrerer Audioströme oder zur Berechnung einer Bild-in-Bild-Darstellung, die aus zwei Videoströmen gespeist wird. Zwi-

schengelagerte Komponenten haben sowohl *Eingangs*- als auch *Ausgangs-Ports*, während Quell- bzw. Senkenkomponenten nur Ausgangs- bzw. Eingangs-Ports besitzen.

Komponenten verbrauchen bzw. produzieren Daten durch Zugriffe auf Ports. Die Ports einer Komponente trennen die Verarbeitungsfunktion der Komponente von jeglicher Kommunikationsfuntionalität. Eine Komponente nimmt Daten an Eingangs-Ports entgegen, bearbeitet sie und stellt die bearbeiteten Daten an Ausgangs-Ports zur Verfügung. Die Funktion einer Komponente ist unabhängig davon, wie (lokal oder entfernt) die Daten an bzw. von Ports kommuniziert werden. Genauso ist sie unabhängig davon, ob und welche weiteren Komponenten zur Stromverarbeitung gegeben sind oder nicht. An einem Port werden formatierte Mediendaten (z. B. Videobilder) empfangen bzw. bereitgestellt. Jeder Komponenten-Port ist daher mit einem Stromtyp assoziiert, der das Format der am Port empfangenen bzw. bereitgestellten Daten charakterisiert (siehe nächsten Abschnitt).

Zur Konfiguration einer Multimedia-Anwendung werden Komponenten zu einem Flußgraphen zusammengesetzt, indem *Links* zwischen Eingangs- und Ausgangs-Ports von Komponenten definiert werden. Ein Link zeigt einen gerichteten Stromfluß zwischen zwei Komponenten-Ports an, wobei die Ports denselben Stromtyp haben müssen. Zugleich abstrahiert ein Link von dem zugrundeliegenden Kommunikationsmechanismus, der den Datentransport abwickelt, der sowohl lokal (z. B. über einen gemeinsamen Speicher) als auch entfernt (über ein Transportsystem) stattfinden kann.

Durch die Verwendung von Links können Flußgraphen beliebiger Topologie konfiguriert werden. In Abb. 4 ist ein Beispielflußgraph für ein Videokonferenzszenario dargestellt. Er besteht aus zwei Quellen, die über zwei Datenströme einen Videomischer speisen. Der Mischer stellt einen Strom zur Verfügung, der über einen Multicast-Link an zwei Senken weitergeleitet wird. Vor jeder Senke ist ein Filter eingebaut, der eine Vorverarbeitung des Datenstroms (z. B. Bildverkleinerung) für die jeweilige Senke vornehmen kann.

Ein Flußgraph ist zunächst lediglich eine (gerichtete, azyklische) Graphstruktur, die eine Anwendung aus Komponenten und Links definiert. Je nach betrachteter Aufgabenstellung kann ein Flußgraph als plaziert oder nicht plaziert definiert werden. Das Ressourcenmanagement ohne Task-Plazierung erfordert eine a-priori Allokation, d. h. zu jeder Komponente wird explizit

eine Zuordnung zu einem Rechner vorausgesetzt (siehe Kapitel 5). Verfahren der Task-Plazierung verwenden dagegen Flußgraphen ohne eine (vollständige) Plazierungsvorgabe (siehe Kapitel 6).

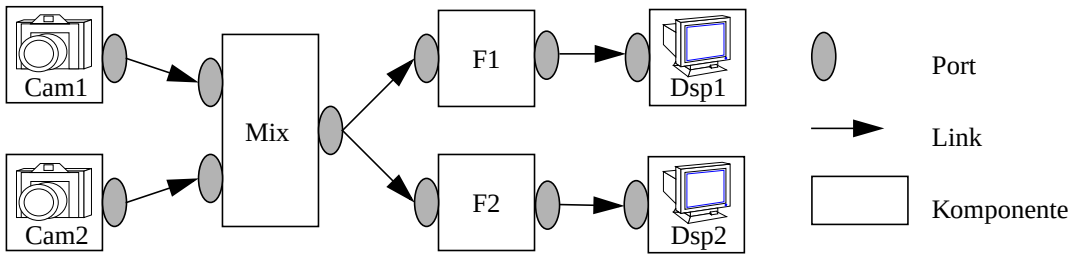


Abb. 4: *Flußgraph für ein Videokonferenzszenario*

Flußgraphen werden durch Klienten konfiguriert. Ein Klient ist der Teil einer Anwendung, der die Systemdienste von CINEMA zum Aufbau und zur Kontrolle von Flußgraphen benutzt. Zur Konfiguration wählt ein Klient aus einer Menge vorhandener Komponenten die gewünschten Komponenten aus und verbindet diese zu einem Flußgraphen. Vorhandene Komponenten können über eine Datenbank zugänglich gemacht werden.

Diese Vorgehensweise realisiert für Klienten ein wichtiges Konfigurationsprinzip: es trennt die Bereitstellung von Komponenten und deren Verknüpfung zu komplexen Flußgraphen („programming in the small“ und „programming in the large“ [DeKr75]). Zum einen lassen sich auf diese Weise Komponenten unabhängig voneinander erstellen. Zum anderen können Klienten fertige Komponenten zu einem beliebigen Zeitpunkt in verschiedenen Flußgraphkonfigurationen (wieder-) verwenden, ohne Details der Programmierung der Komponenten betrachten zu müssen.

2.2 Stromtypen

Ein *Stromtyp* definiert in CINEMA die Formateigenschaften der Mediendateneinheiten, die durch einen Datenstrom zwischen Komponenten-Ports kommuniziert werden. Um verarbeitbar zu sein, muß jeder Strom mit einem eindeutigen Stromtyp assoziiert sein, wie z. B.

„uncompressed_video“, „JPEG_video“ für Videoströme oder „pcm_audio“ für Audioströme. Für einen Komponenten-Port zeigt der assoziierte Stromtyp an, welcher Datenstrom an dem Port ankommen bzw. generiert werden kann.

Jeder Stromtyp spezifiziert sogenannte medienspezifische Parameter, welche die Charakteristik des Stromtyps definieren. Zum Beispiel könnte der Stromtyp „uncompressed_video“ durch Parameter wie Bildgröße, Bildrate und Farbtiefe beschrieben sein, während ein Stromtyp „pcm_audio“ Parameter wie Sample-Abtastrate, Sample-Größe sowie Sample-Paketierung beinhalten kann. Angaben zu diesen Parametern werden von Komponenten benötigt, um sich auf eine korrekte Verarbeitung empfangener Daten einzustellen.

Zwei durch einen Link verbundene Komponenten-Ports müssen denselben Stromtyp unterstützen. Andererseits kann eine Komponente verschiedene Stromtypen an verschiedenen Ports aufweisen. So würde z. B. eine JPEG-Kompressionskomponente am Eingangs-Port „uncompressed_video“ und am Ausgangs-Port z.B. „JPEG_video“ unterstützen, um die Umformatierung durch Kompression anzuzeigen.

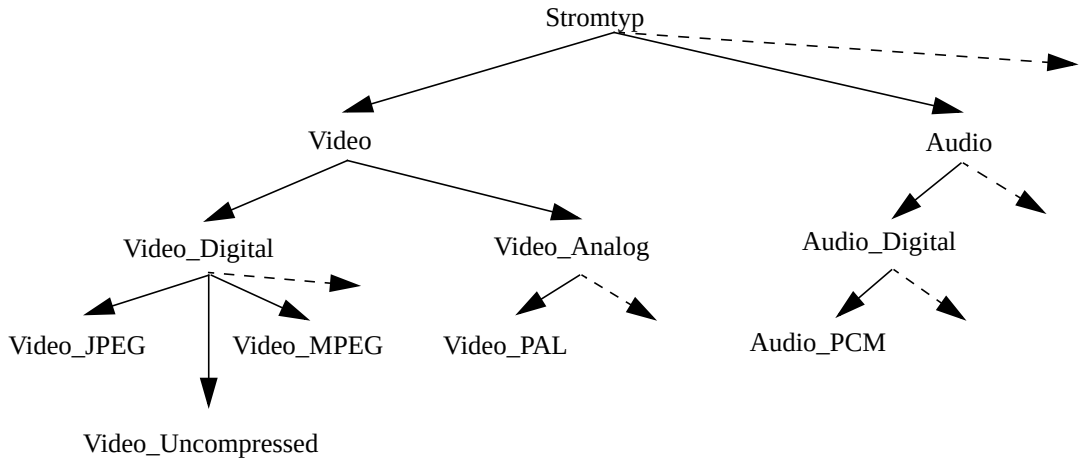


Abb. 5: *Stromtyphierarchie*

Stromtypen werden in CINEMA objektorientiert gemäß einer Stromtyphierarchie definiert (ein ähnlicher Ansatz wird in [IMA93] verfolgt). Ausgehend von gegebenen Stromtypen können durch Ableitung neue Stromtypen definiert werden. Abgeleitete Stromtypen erben alle Parameter (und Methoden) und ergänzen sie durch neue Parameter. Auf diese Weise wird sichergestellt,

daß z. B. alle Stromtypen vom Typ „video“ einen Kernsatz von Parametern (z. B. Bildgröße und Bildrate) unterstützen und daß neue Stromtypen mit weiteren Parametern eingeführt werden können.

Medienspezifische Parameter eines Stromtyps können Werte aus einer gegebenen Wertemenge annehmen. Die Wahl eines festen Wertes für jeden Parameter definiert dabei eine bestimmte Strominstanz. Eine Wertemenge beinhaltet eine Anzahl diskreter Werte, z. B. (8 kHz, 11.025 kHz, 22.05 kHz, 44.1 kHz) für die Abtastrate von „digital_audio“, die die möglichen Stromtypausprägungen beschreibt.[3]

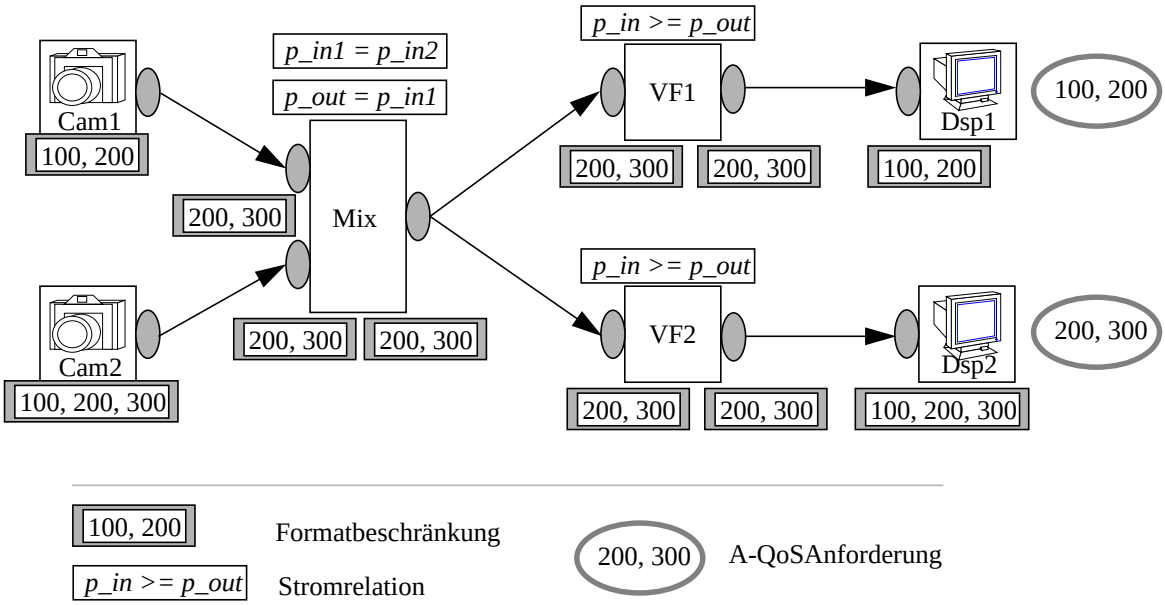


Abb. 6: *Formatbeschränkungen und Stromrelationen in einem Flußgraphen*

Im allgemeinen kann nicht angenommen werden, daß an einem Komponenten-Port für medienspezifische Parameter die jeweils volle Wertemenge unterstützt wird. Das funktionale Design der Komponente wird vielmehr oft Einschränkungen vorgeben. So könnte eine Videoquelle nur zur Unterstützung einer maximalen Bildgröße (z. B. 300x300) entworfen sein. Um funktionale Einschränkungen dieser Art anzuzeigen, kann in CINEMA an jedem Port und zu jedem Para-

[3] Falls die zugrundeliegende Zahlenmenge bekannt ist (z.B. alle Vielfachen von 11.025 kHz), wird die Wertemenge als ein Bereich angegeben, z. B. (11.025 .. 44.1).

meter eine sogenannte *Formatbeschränkung* definiert werden. Eine Formatbeschränkung stellt die Teilmenge der möglichen Werte dar, die durch die Komponente für einen Parameter an einem bestimmten Port funktional unterstützt wird (siehe Abb. 6). Sie wird durch den Komponentenprogrammierer definiert.

2.3 Stromrelationen

Die Wahl von Parametern an Komponenten-Ports wird ferner durch sogenannte *Stromrelationen* eingeschränkt. Solche Relationen koppeln Parameter an unterschiedlichen Ports einer gegebenen Komponente. So kann z. B. ein Audiomischer die gleiche Sample-Größe an allen Eingang-Ports verlangen, um die Ströme mischen zu können. Genauso kann ein Bild-in-Bild Mischer erfordern, daß die Bildgröße des Vordergrundvideos in einem bestimmten Verhältnis zur Bildgröße des Hintergrundvideos steht.

Stromrelationen werden auch zur Kopplung von Parametern zwischen Ausgangs- und Eingangs-Ports benötigt. So wird in der Regel ein Videomischer ein Video bereitstellen, dessen Bildgröße der des Empfangsvideos für den Mischhintergrund entspricht. Ähnliche Stromrelationen sind für Filter definiert. Viele Filter lassen Parameterwerte zwischen Ein- und Ausgangs -Ports unverändert, weil sie lediglich den Medieninhalt beeinflussen, wie dies z. B. zur Glättung von Bildern nötig ist. Andere Filter werden dagegen Eingangsparameterwerte um einen bestimmten Faktor verändern, z. B. um eine Bildgröße zu halbieren. Wie Formatbeschränkungen, sind Stromrelationen Teil des funktionalen Designs von Komponenten und werden vom Komponentenprogrammierer bereitgestellt.

In CINEMA sind verschiedene Stromrelationen formal definiert:

(1) param: @inport_i = $f_{i,j}$ * @inport_j

(2) param: @outport = F * @inport

Für Mischerkomponenten, sind Mischrelationen (1) angebbar zwischen Eingangs-Ports mit der Bedeutung, daß der Wert des Parameters param am Port inport_i $f_{i,j}$ mal so groß sein muß wie der Wert von param an inport_j. Um die Relation zwischen einem Ausgangs- und einem Eingangs-Port (eines Mischers oder Filters) zu beschreiben, wird eine ähnliche Filterrelation (2)

mit Hilfe eines Filterfaktors F definiert. Diese zwei Relationen genügen, um die Abhängigkeiten in den oben gegebenen Beispielen zu modellieren.

Stromrelationen, die für eine Komponente definiert werden, beeinflussen zum einen die medienspezifischen Parameter an den Ports der Komponente selbst. Zum anderen haben sie Auswirkungen auf die Parameter aller Komponenten im Flußgraphen. Betrachtet man den Flußgraphen in Abb. 4 unter der Annahme, daß an den Eingangs-Ports des Mischers derselbe Parameterwert, z. B. Bildgröße, erwartet wird und daß die Filter F1 und F2 die Bildgröße nicht verändern, so ist offensichtlich, daß, um allen Stromrelationen und dem Multicast-Link zwischen dem Mischer und F1 bzw. F2 Rechnung zu tragen, dieselbe Bildgröße an allen Komponenten-Ports eingestellt werden muß.

Dies gilt insbesondere auch für die Eingangs-Ports der Senkekomponenten Dsp1 und Dsp2. Haben diese Ports nicht überlappende Formatbeschränkungen (d. h. keinen gemeinsamen Wert), so ist keine konsistente Parametereinstellung im Flußgraphen möglich. Selbst wenn eine Überlappung gegeben ist, erfordert der Multicast-Link, daß derselbe Wert für beide Senken - Ports einzustellen ist, obwohl evtl. beide Senken höhere, aber unterschiedliche Parameterwerte an ihren Ports hätten unterstützen können.

2.4 Variable Filter

Um solche Situationen zu vermeiden, werden zwei Klassen von Filtern eingeführt: *fixe* und *variable Filter.* Fixe Filter entsprechen Filtern mit einer festen Filterrelation, wie sie oben beschrieben wurde. Variable Filter werden benutzt, um die Abhängigkeiten, die durch Multicast-Links und Stromrelationen bedingt werden, zu begrenzen. Ein variabler Filter ist über seine Fähigkeit definiert, einen Parameterwert an seinem Eingangs-Port auf einen beliebigen niedrigeren oder gleichen Wert am Ausgangs-Port konvertieren zu können. Die entsprechende Stromrelation hat die Form:

(3) param: @inport >= @outport

Der wichtigste Unterschied zu einem fixen Filter besteht darin, daß der Filterfaktor nicht apriori festgelegt ist. Wie später verdeutlicht wird, werden die Filterfaktoren variabler Filter im Rahmen der Dienstgüteaushandlung durch das Reservierungsprotokoll ermittelt. Dabei werden

Formatbeschränkungen, Stromrelationen sowie Ressourcenverfügbarkeit in einer integrierten Weise betrachtet, so daß keine dieser Randbedingungen verletzt wird.

Variable Filter ermöglichen insbesondere sogenannte „heterogene Empfänger“, d. h. Komponenten, deren Parameter unabhängig voneinander eingestellt werden können. Wird in Abb. 6 vorausgesetzt, daß VF1 und VF2 variable Filter sind, so kann die Bildgröße für Dsp1 und Dsp2 unterschiedlich eingestellt werden. Jede Senke kann dabei die maximale Bildgröße erhalten, die sie unterstützt, sofern diese vom Mischer bereitgestellt werden kann.[4] Die nötige Konvertierung auf den gewünschten Wert wird individuell durch VF1 bzw. VF2 vorgenommen. Die zuvor beschriebenen Abhängigkeiten durch Multicast-Links und fixen Stromrelationen werden somit aufgehoben.

2.5 Abstraktion einer Session

Eine konfigurierte Multimedia-Anwendung muß instanziiert werden, bevor sie durch einen Klienten benutzt werden kann. Die Instanziierung beinhaltet mehrere Schritte. Zum einen muß für Komponenten und Links auf den zugeordneten Rechnern der ablauffähige Code bereitgestellt werden, der die gewünschte Stromverarbeitungsfunktion erbringt. Zum anderen müssen für Komponenten und Links benötigte Ressourcen (z. B. CPU-Zeit, Speicher oder Kommunikationsbandbreite) reserviert werden, um eine gewünschte Dienstgüte zu erbringen. Ressourcenreservierung wird benötigt, um eine einmal eingestellte Dienstgüte durch eine exklusive Nutzung von Ressourcenkapazität garantieren zu können.

Zur Instanziierung eines Flußgraphen stellt CINEMA die Abstraktion einer *Session* zur Verfügung. Allgemein kann ein Teil eines Flußgraphen für den Session-Aufbau spezifiziert werden [Bart98]. Zur Vereinfachung wird im folgenden immer ein vollständiger Flußgraph als Teil einer Session vorausgesetzt.

Sessions sind mit Dienstgüteparametern assoziiert, aus denen zum einen die vom Klienten gewünschte Dienstgüte hervorgeht, zum anderen der Bedarf an Ressourcen abgeleitet werden

[4] Reduktionen von Parameterwerten wegen Ressourcenknappheit werden später betrachtet. Reduktionen für Dsp1 und Dsp2 können auch in diesem Fall unabhängig voneinander vorgenommen werden.

kann. Im Rahmen dieser Arbeit werden die Dienstgüte- und Ressourcenmangementaspekte einer Session behandelt. Eine Session wird daher als die Einheit der Dienstgütespezifikation und Ressourcenallokation für einen gegebenen Flußgraphen aufgefaßt. Für eine Beschreibung des Aspekts der Code-Instanziierung wird auf [Bart98] verwiesen.

Der Aufbau einer Session wird von einem Klienten durch die Spezifikation gewünschter Dienstgüte an den Ports von Senkekomponenten gestartet. Da Senken für die wahrnehmbare Präsentation von Datenströmen verantwortlich sind, legt eine sich darauf beziehende Spezifikation die für den Endbenutzer sichtbare Dienstqualität fest. Da verschiedene Senken unabhängigen (heterogenen) Endbenutzern dienen können, kann für jede Senke individuell eine gewünschte Dienstgüte (*Application Quality-of-Service*, A-QoS) spezifiziert werden. Wie im nächsten Kapitel ausgeführt wird, beinhaltet eine A-QoS Vorgaben sowohl für medienspezifische Parameter (z. B. Bildgröße, Bildrate) als auch für sogenannte generische Parameter (z. B. Verzögerung). Solche Vorgaben können in Form von (min, max)-Wertebereichen angegeben werden.

Der Aufbau einer Session wird durch CINEMA mit Hilfe eines Reservierungsprotokolls der Anwendungsebene durchgeführt (siehe Kapitel 5). Bei erfolgreichem Aufbau werden als Ergebnis die ursprünglichen A-QoS abgeändert, indem für jeden Parameter der eingestellte Wert eingetragen wird. Der Erfolg oder Mißerfolg eines Session-Aufbaus wird dem Klienten mitgeteilt.

3 Dienstgütearchitektur und Ressourcenmanagement

Der Aufbau einer Session erfordert die Betrachtung von Dienstgüte auf verschiedenen Abstraktionsebenen. Eine Dienstgütearchitektur wird benötigt, um diese Ebenen zu identifizieren und zueinander in Beziehung zu setzen. Das Ressourcenmanagement realisiert die verschiedenen Dienstgüteabstraktionen und stellt die Mechanismen bereit, die den Aufbau einer Session durchführen. Es orientiert sich in seiner Struktur an den Ebenen der Dienstgütearchitektur und muß Funktionen zur Dienstgütespezifikation und zur Ressourcenreservierung bereitstellen.

In diesem Kapitel werden die Grundlagen des CINEMA-Ressourcenmanagements beschrieben. Sie liefern die Basis für die folgenden Kapitel, in denen die Aspekte der Ressourcenreservierung auf Anwendungsebene sowie der Task-Plazierung untersucht werden. Im Abschnitt 3.1 wird zunächst die Dienstgütearchitektur von CINEMA vorgestellt. Die darauf aufbauende Struktur des CINEMA-Ressourcenmanagements wird im Abschnitt 3.2 erläutert. Das Management einiger wichtiger Systemressourcen (CPU und Hauptspeicher) wird im Abschnitt 3.3 beschrieben, bevor im Abschnitt 3.4 die Zusammenfassung von Komponenten und Links zu Verarbeitungs-Threads dargestellt wird.

3.1 Dienstgütearchitektur

Die Bereitstellung von Dienstgüte für verteilte Multimedia-Anwendungen bedingt Dienstanforderungen auf verschiedenen Ebenen ([VKBG95], [StNa95]). CINEMA unterscheidet drei Ebenen ([RDF97], [BDF95]).

Auf der *Systemebene* sind die Ressourcen von Endsystemen, z. B. CPU und Speicher, sowie von Kommunikationssystemen, z. B. Transportbandbreite, sichtbar. Diese Ressourcen werden zur Stromverarbeitung in Komponenten sowie zur Kommunikation benötigt. Dienstanforderungen auf dieser Ebene beinhalten ressourcenorientierte Dienstgüteangaben (R-QoS), die spezifisch für jeden betrachteten Ressourcentyp sind. Z. B. kann von einem Transportsystem die Bereitstellung einer Verbindung mit garantierter (Transport-) Dienstgüte angefordert werden, indem eine anvisierte Verkehrscharakteristik (Paketgröße, durchschnittliche Paketrate, etc.) sowie eine maximal zulässige Verzögerung spezifiziert werden (siehe Tab. 1). Eine Anforderung von CPU-

Kapazität erfordert dagegen in der Regel die Angabe einer periodisch wiederkehrenden Verarbeitungsdauer sowie der damit verbundenen Verarbeitungsperiode (siehe 3.3).

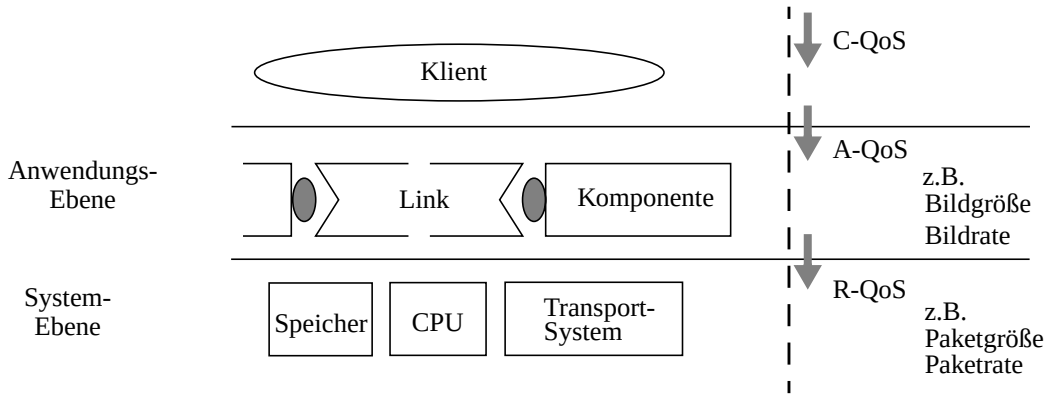


Abb. 7: *Dienstgütearchitektur von CINEMA*

Auf der *Anwendungsebene* sind dagegen die Objekte der Anwendung (d. h. Komponenten und Links) sichtbar, sowie ihre Fähigkeit, Medienströme zu verarbeiten und zu kommunizieren. Medienspezifische Parameter, wie sie im letzten Kapitel eingeführt wurden, sind nötig um ihre Funktion festzulegen. Einschränkungen funktionaler Art, durch Formatbeschränkungen und Stromrelationen definiert, sind auf dieser Ebene angesiedelt. Diese Aspekte sind durch (medienunspezifische) Angaben der Systemebene wie Bandbreite oder Verarbeitungsdauer nicht beschreibbar. Insbesondere sind Komponenten erst anhand medienspezifischer Parameterwerte (an ihren Ports) in der Lage, ihren Bedarf an Ressourcen eindeutig abzuleiten.

Dienstgüteanforderungen eines Klienten an einen Flußgraphen (A-QoS) werden, wie früher erwähnt, an den Senken-Ports eines Flußgraphen spezifiziert. Eine A-QoS beinhaltet Angaben zu *medienspezifischen* und *generischen Parametern*. Der medienspezifische Teil beschreibt den erwarteten Medienstrom an dem betrachteten Senken-Port und umfaßt Parameter des entsprechenden Stromtyps. Wird z. B. eine A-QoS für eine Videosenke spezifiziert (Abb. 6), so könnten darin Anforderungen an Bildgröße, Bildrate oder Farbtiefe angegeben werden, sofern diese Teil des Stromtyps am Senken-Port sind.

Der generische Teil einer A-QoS erfaßt Parameter, die „Ende-zu-Ende“-Eigenschaften eines Flußgraphen beschreiben und nicht medienspezifisch sind. Verzögerung und Jitter sind Bei-

spiele hierfür, wobei im Rahmen dieser Arbeit lediglich die Behandlung von Verzögerung untersucht wird. Die Spezifikation einer maximal erlaubten Verzögerung erfolgt in CINEMA individuell an jedem Senken-Port und legt die „Ende-zu-Ende“-Verzögerung zwischen einer beliebigen Quelle und der betrachteten Senke fest.

CINEMA definiert eine sogenannte *Klienten-Ebene* oberhalb der Anwendungsebene. Diese erlaubt es Klienten, den (menschlichen) Endbenutzern angepaßte Dienstgüteabstraktionen wie z. B. „hohe“, „mittlere“ oder „niedrige“ Videoqualität anzubieten anstelle der technisch orientierten Parameter der Anwendungsebene. Solche Abstraktionen können beliebig gewählt werden, z. B. in Abhängigkeit von der gegebenen Anwendung oder der zu unterstützenden Endbenutzer. Die Abbildung auf die A-QoS Spezifikationen der Anwendungsebene ist im Klienten gekapselt und hat keine Auswirkungen auf das CINEMA Ressourcenmanagement, das lediglich die Anwendungs- und Systemebenen einbezieht.

Folgende Tabelle verdeutlicht als Beispiel die unterschiedlichen Abstraktionen der Dienstgütespezifikation in CINEMA:

Klientenebene	Präsentationsqualität	
	Qualität: hoch, mittel, niedrig	
Anwendungsebene	Stromdeskriptor	Generische Dienstgüteanforderungen
	Bildgröße Bildrate Farbtiefe	Verzögerung Jitter
Transportebene (aus [Ferr94])	Verkehrsdeskriptor	Generische Dienstgüteanforderungen
	Max Msg Size Min Intermsg Time Average Intermsg Time Averaging Interval	Verzögerung Jitter

Tabelle 1: Dienstgüte-Spezifikationen

3.2 Managementstruktur zur Ressourcenreservierung

Entsprechend der vorgestellten Architektur umfaßt das Ressourcenmanagement Funktionen auf der Anwendungs- und der Systemebene. Auf der *Systemebene* werden einzelne Ressourcen verwaltet und Komponenten und Links zugeteilt. Zu jeder einzelnen Ressource gehört ein entsprechender Manager. Beispielsweise verwaltet ein CPU-Manager die Kapazität einer CPU, während ein Speichermanager den vorhandenen Hauptspeicher eines Rechners verwaltet. Transportsysteme stellen Kommunikationsbandbreite für Transportverbindungen bereit und machen ihrerseits von Ressourcen der Endsysteme (z. B. CPU) und des Netzwerks Gebrauch.

Ressourcenmanager werden durch Komponenten oder Links zur Ressourcenreservierung aufgerufen. Die erwünschte Reservierung wird in Form von ressourcenspezifischen Parametern angegeben. Transportsysteme werden zum Verbindungsaufbau durch Links aufgerufen, wobei die gewünschte Qualität in Form von Parametern des verwendeten Transportsystems ausgedrückt wird.

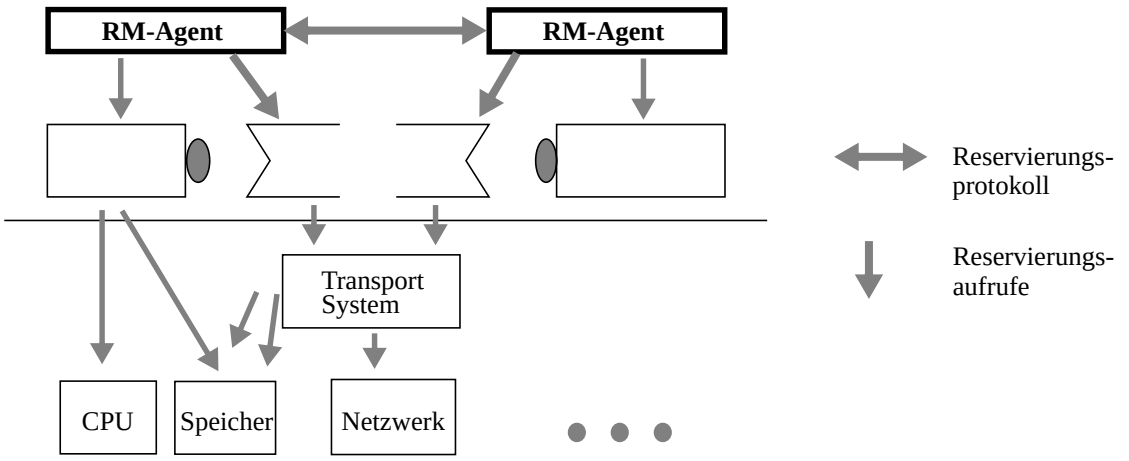


Abb. 8: *Struktur des Ressourcenmanagements*

Das Ressourcenmanagement auf der *Anwendungsebene* koordiniert die Parameterauswahl und die Ressourcenreservierung für Komponenten und Links mit den A-QoS Spezifikationen des Klienten. Eine solche Koordination wird zum einen wegen der Abhängigkeiten medienspezifischer Parameter in einem Flußgraphen benötigt, wie sie durch Stromrelationen und Links

bedingt werden (siehe Kapitel 2). Zum anderen kontrolliert sie die Festlegung generischer Parameter, indem sie „Ende-zu-Ende“-Berechnungen durchführt, die alle Komponenten und Links zwischen Quellen und Senken von Strömen berücksichtigen.

Das Ressourcenmanagement auf der Anwendungsebene ist in verteilten Multimedia-Systemen in sogenannte *RM-Agenten* strukturiert. Aufgaben und Funktionsweise von RM-Agenten unterscheiden sich je nach der erbrachten Unterstützung und dem verfolgten Ressourcenmanagementansatz. Verschiedene bestehende Ansätze werden im nächsten Kapitel untersucht. Generell gilt, daß für jeden Rechner, auf dem Komponenten gegeben sind, (zumindest) ein RM-Agent existiert. RM-Agenten realisieren die Dienstschnittstelle des Ressourcenmanagements (zum Klienten hin) und tauschen untereinander medienspezifische und generische Parameterinformation aus, um eine flußgraphenweite Ressourcenreservierung sicherzustellen. Hierzu wird auf der Anwendungsebene ein *Reservierungsprotokoll* benötigt, das den Austausch o. g. Information geeignet abwickelt.

Interaktionen zur Ressourcenreservierung finden sowohl auf der Anwendungs- als auch auf der Ressourcenebene statt. Auf der Anwendungsebene werden in CINEMA Komponenten und Links durch RM-Agenten zur Ressourcenreservierung entsprechend vorgegebener medienspezifischer Parameter aufgerufen. Ihrerseits rufen Komponenten und Links auf der Ressourcenebene Manager von Systemressourcen sowie Transportsysteme zur Reservierung auf. Um den Bedarf an diversen Ressourcen aus medienspezifischen Angaben abzuleiten sind *Abbildungsvorschriften* zwischen medienspezifischen und ressourcenbezogenen Anforderungen erforderlich. Beispielsweise muß für einen gegebenen Videostrom bekannt sein, wie Videobilder in Transportpaketen zu übertragen sind und welche Paketgröße sowie Paketrate damit verbunden ist [NaSm95]. Solche Vorschriften des „QoS-Mapping“ wurden in der Literatur für diverse Medien und Ressourcen untersucht [WWV94], [Garg95], [KiNa97].

In CINEMA wird für jede Komponente und jeden Link eine Ressourcenbedarfstabelle (RB-Tabelle) gefordert, die für jeden möglichen medienspezifischen Parameterwert (bzw. alle Kombinationen möglicher Werte unterschiedlicher Parameter) an den Ports die Art und das Ausmaß der benötigten Systemressourcen beschreibt. Diese Tabellen sind Teil der Komponenten- bzw. Link-Beschreibung und werden zur Programmierzeit erstellt. Dabei wird der Pufferbedarf direkt aus der Funktionsweise einer Komponente abgeleitet, während die Verarbeitungsdauer

pro Aufruf durch Messungen ermittelt wird [Lind95], [WWV94]. In Abb. 9 ist der (worst-case) Ressourcenbedarf für einen (fixen) Videofilter mit einem Eingangs- und einem Ausgangs-Port dargestellt, aufgrund dessen Ressourcenreservierungen vorgenommen werden.[5]

Bildrate	Bildgröße	CPU	Pufferbedarf	. . .
. . .				
5 Bld./s	640x480	Rate= 5 1/s Dauer= 10 ms	Verarbeitungspuffer = 0 Eingangsspuffer = 900 kByte Ausgangsspuffer = 900 kByte	
. . .				

Abb. 9: *Ressourcenbedarfstabelle für einen Videofilter*

Ein Ressourcenmanagement für verteilte Multimedia-Systeme muß zur Durchführung von Ressourcenreservierung alle erwähnten Teile beinhalten: RM-Agenten und ein Reservierungsprotokoll auf der Anwendungsebene, Abbildungsmechanismen zwischen der Anwendungs- und der Ressourcenebene sowie Ressourcenmanager bzw. Transportsysteme auf der Ressourcenebene. Der Entwurf von Reservierungsmechanismen der Anwendungsebene bildet den Schwerpunkt dieser Arbeit und wird in späteren Kapiteln behandelt. Die Verwendung einiger, wichtiger Systemressourcen wird im folgenden in einer Übersicht beschrieben.

3.3 Reservierung von Systemressourcen

Multimedia-Anwendungen belegen auf jedem beteiligten Rechner Systemressourcen. Da flexible Multimedia-Systeme sowohl software- als auch hardwaregestützte Verarbeitung benötigen, werden vielfältige Ressourcen benutzt. Softwareverarbeitung erfordert insbesondere CPU- und Hauptspeicherressourcen, während hardwaregestützte Verarbeitung sich in der Regel auf DSP-CPUs auf Zusatzboards stützt. Die Speicherung von Multimediadaten erfordert entsprechende Speichergeräte und Dateisysteme, während zur Datenkommunikation Netzwerkadapter eingesetzt werden.

[5] Die Unterscheidung der Pufferarten wird in Abschnitt 3.3.4 erläutert.

Ressourcen werden durch unterschiedliche Teile einer Anwendung oder auch durch mehrere Anwendungen beansprucht. Um konkurrierende Anforderungen gegeneinander abzuschirmen, wird Ressourcenreservierung benötigt. Erst diese garantiert, daß eine Ressource für mehrere Benutzer zur Verfügung steht und daß dabei eine bestimmte Dienstgüte eingehalten wird. Im folgenden wird ein allgemeines Modell zur Reservierung einer Ressource eingeführt. Anschließend wird das Management der CPU und des Hauptspeichers im Kontext von CINEMA beschrieben.[6]

3.3.1 Allgemeines Modell

Jede Ressource wird durch einen Manager verwaltet. Dieser realisiert die Schnittstelle zur Belegung einer Ressource und verwaltet deren Kapazität. Nach [StNa95] kann die Verwaltung allgemein in folgende Phasen unterteilt werden:

1. Einplanbarkeitstest: Der Manager überprüft anhand der von der Anwendung geforderten Parameterwerte (die die geforderte Kapazität angeben), ob die gegenwärtig freie Kapazität für die geforderte Belegung ausreicht.

2. Dienstgüteberechnung: Nach erfolgtem Einplanungstest berechnet der Manager die bestmögliche Dienstgüte (z. B. Verzögerung), die die Ressource garantieren kann.

3. Ressourcenreservierung: Der Ressourcenmanager reserviert die geforderte Kapazität entsprechend der berechneten Dienstgüte.

4. Ressourcen-Zuteilung: Der Manager teilt zur Laufzeit die Ressource gemäß der berechneten Dienstgüte zu. Hierzu wird ein geeigneter Scheduling-Algorithmus benötigt, dessen Beschaffenheit auch die Schritte 1 bis 3 beeinflußt.

Die Reservierung einer Ressource findet in den ersten drei Phasen statt, während die vierte Phase zur Verwaltung einer Ressource zur Laufzeit dient. Die Interaktion mit einem Ressourcenmanager entspricht in der einfachsten Form einem „Request-Reply“-Schema. Der Benutzer fordert eine Reservierung unter Angabe der benötigten (Kapazitäts-) Werte an und der Manager antwortet mit einer Erfolgsanzeige sowie der garantierten Dienstgüte.

---

[6] Eine Beschreibung des Managements weiterer Systemressourcen ist z. B. in [StNa95] gegeben.

3.3.2 Management der CPU

Kontinuierliche Datenströme bestehen aus periodisch zu verarbeitenden Dateneinheiten. Eine Verarbeitung erfordert entsprechend eine periodische Zuteilung der CPU. Da Multimedia-Anwendungen nur begrenzte Verzögerungen zwischen Erzeugung und Präsentation von Dateneinheiten tolerieren, muß ein Verarbeitungsschritt auf der CPU (wie auf anderen Ressourcen auch) eine garantierte und zugleich niedrige Verzögerung erfahren. Um diese Anforderungen zu erfüllen, wird zur CPU-Verwaltung Echtzeit-Scheduling eingesetzt.

Für die Verarbeitung periodischer Datenströme sind vor allem zwei Scheduling-Verfahren bekannt: das „Rate-Monotonic“ (RM, [LiLa73], [LSD89]) und das „Earliest Deadline First“ (EDF, [LiLa73]) Verfahren. Weitere, weniger häufig eingesetzte Verfahren, die häufig aus RM oder EDF abgeleitet sind, wurden in der Literatur untersucht ([StNa95]). Das RM-Verfahren bildet die Grundlage des CPU-Schedulings in CINEMA. RM teilt die CPU den Threads (Ausführungseinheiten) der Komponenten und Links in Abhängigkeit von deren Aufrufrate zu. Threads, die eine hohe Aufrufrate haben, erhalten eine entsprechend hohe Priorität. Sind zwei Threads zur Ausführung bereit, so wird zuerst dem Thread mit der höheren Priorität die CPU zugeteilt.[7]

Die Reservierung von CPU-Kapazität für das RM-Verfahren erfordert folgende Schritte. Die Reservierungsanforderung wird von der Anwendung in Form einer Kapazitätsbeschreibung bestehend aus (Aufrufperiode P_n, Verarbeitungsdauer pro Aufruf E_n) geliefert. Der Einplanbarkeitstest für RM ist erfolgreich, falls folgende Relation durch die neue sowie frühere Anforderungen erfüllt ist:

$$\frac{E_1}{P_1} + \ldots + \frac{E_n}{P_n} \leq n\left(2^{\frac{1}{n}} - 1\right)$$

Die Einplanung gemäß RM garantiert eine Begrenzung der Verzögerung (die Zeit bis zum garantierten Abschluß der Verarbeitung) auf eine Periode. Die neue Reservierung wird durch die Erhöhung der belegten CPU-Kapazität um den angeforderten CPU-Anteil berücksichtigt, der als das Verhältnis der Verarbeitungsdauer zur Verarbeitungsperiode berechnet wird. Bei

---

[7] Das RM-Verfahren erfordert, daß die Threads in periodischen Abständen zur Verarbeitung bereit sind [LiLa73]. Der hierfür nötige Laufzeitmechanismus ist in [Fied98] beschrieben.

erfolgreicher Reservierung wird der Anwendung eine entsprechende Anzeige samt garantierter Verarbeitungsverzögerung geliefert.

Der Vorteil von RM liegt in der einfachen CPU-Zuteilung, die einzig über die statischen Prioritäten der Threads erfolgt. Die Prioritätenreihenfolge ändert sich nur bei Hinzunahme eines neuen Threads aufgrund einer Reservierungsanforderung, nicht jedoch bei jedem Datenempfang, wie dies z. B. bei EDF der Fall ist. Der Nachteil von RM liegt darin, daß die CPU-Auslastung unter ca. 69% liegen muß, damit die Verarbeitungsgarantien eingehalten werden können. Dieser Nachteil tritt jedoch auf Systemen in den Hintergrund, die sowohl Echtzeit- als auch zeitunabhängige Verarbeitung unterstützen. Für letztere kann die durch RM nicht verwendbare CPU-Kapazität vorgesehen werden.

RM garantiert eine maximale Verzögerung von einer Periode pro Verarbeitungsaufruf. Für Threads mit niedrigen Aufrufraten kann dies eine erhebliche Verzögerung bedeuten. In CINEMA wurden deswegen auch erweiterte CPU-Scheduling-Verfahren mit mehreren Dienstklassen betrachtet ([Bart94], [Bart98], [Fied98]). In [Fied98] wird z. B. eine Dienstklasse vorgesehen, die das 0.3-fache der Periode als Verzögerung garantiert. Als Preis dafür kann in den anderen Dienstklassen die einfache Periode nicht generell als maximale Verzögerung eingehalten werden. Für einen weiten Bereich der Aufrufraten kann jedoch in allen Dienstklassen eine Garantie gegeben werden, die die Periode nicht beträchtlich überschreitet.

3.3.3 Management des Hauptspeichers

Das Speichermanagement für Multimedia-Anwendungen muß effiziente Datenzugriffs- und Datentransfermechanismen anbieten, um den zeitlichen Anforderungen der Anwendungen zu genügen. Die üblicherweise verwendeten Betriebssystemmechanismen reichen hierzu nicht aus bzw. können nicht unverändert benutzt werden.

Ein effizienter Datenzugriff erfordert insbesondere das Bereithalten der Daten im realen Hauptspeicher. Das in heutigen Systemen gebräuchliche Konzept des virtuellen Speichers erlaubt zwar eine wesentlich erweiterte Adressierung im virtuellen Speicherraum, bedingt aber Kopieraktionen zwischen dem Haupt- und dem Sekundärspeicher. Diese sind in heutigen Systemen zu langsam, um Echtzeitanforderungen von Multimedia zu genügen. Für Multimedia-Systeme ist

daher das Auslagern der Daten auf Sekundärspeicher zu vermeiden, was durch „Pinning“ benötigter Speicherbereiche (Seiten) im Hauptspeicher erreicht werden kann.

Multimedia-Anwendungen bedingen auf jedem beteiligten Rechner einen periodischen Datentransfer zwischen Komponenten und Links. Da die involvierten Datenmengen häufig sehr groß sind, muß ein effizienter Datentransfermechanismus insbesondere Kopieroperationen vermeiden. In CINEMA, wie in anderen Multimedia-Systemen ([Wolf96]), wird ein Poolkonzept zur Verwaltung von Hauptspeicher verwendet. Im Pool sind unbelegte und belegte Puffer registriert, wobei ein Puffer einen Speicherbereich variabler Größe darstellt, der durch die Komponenten und Links einer Anwendung reserviert und freigegeben wird. Puffer erlauben in CINEMA Lese- und Schreibzugriffe der Komponenten und Links und können zwischen diesen per Referenzübergabe, d. h. ohne Kopieren des Pufferinhalts, ausgetauscht werden.

Da ein Puffer durch verschiedene Anwendungsteile verwendet werden kann, realisiert CINEMA eine Zugriffsverwaltung, um Zugriffe zu synchronisieren. Zu jedem Puffer werden hierzu Lese- und Schreibrechte erzeugt und an Komponenten und Links gereicht. Zu jedem Zeitpunkt stellt die Zugriffsverwaltung sicher, daß es zu einem Puffer entweder (a) beliebig viele Leserechte oder (b) genau ein Schreibrecht gibt. So kann nur eine Komponente auf einmal die Daten des Puffers verändern.

Leserechte werden durch alle Komponenten benötigt, um Zugriff auf die Daten zu haben. Da Komponenten Daten manipulieren, brauchen diese in der Regel auch Schreibzugriffe. Eine Ausnahme bilden Senken, die Daten evtl. nur einlesen und präsentieren. Komponenten, die Daten verändern, müssen in CINEMA einen Puffer für die manipulierten Daten reservieren, d.h. der empfangene Puffer für Eingangsdaten wird nicht beschrieben.

Abb. 10 zeigt exemplarisch die Verwendung von Puffern und Zugriffsrechten. Die Quelle hat Puffer 1 reserviert und dafür ein Lese-/Schreibrecht erhalten. Nach Ablegen der Daten im Puffer, gibt die Quelle dieses Recht auf. Komponente 2 erhält ein Leserecht auf den Puffer, sobald das Schreibrecht der Quelle gelöscht ist. Da sie die Daten manipuliert, reserviert sie Puffer 2 samt Lese-/Schreibbrecht. Nach Ablegen der manipulierten Daten im Puffer 2, werden das Leserecht für Puffer 1 und das Lese-/Schreibrecht für Puffer 2 gelöscht. Komponenten 3 und 4

können hierauf Leserechte erhalten und die Datenpräsentation durchführen. Danach werden die Leserechte gelöscht.

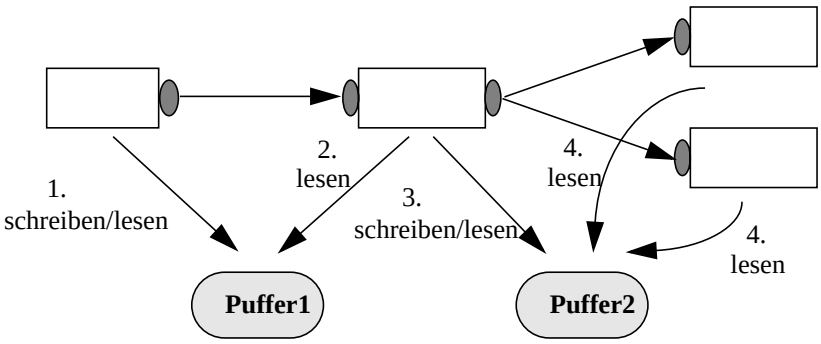


Abb. 10: *Pufferweitergabe in CINEMA*

Dieser Ablauf bedingt folgende Eigenschaften für die Reservierung von Puffern. Puffer werden immer von Komponenten reserviert, die Daten manipulieren. Insbesondere müssen lokale Links keine Puffer (oder andere Ressourcen) reservieren, da sie Pufferdaten nicht kopieren.[8] Dies gilt auch für einen lokalen Multicast-Link. Dieser muß Puffer nicht replizieren, da diese durch nachfolgende Komponenten nie überschrieben werden.

Puffer werden freigegeben, sobald alle einen Puffer empfangenden Komponenten ihre Leserechte abgegeben haben. Die Pufferverwaltung muß hierzu die Zahl solcher Komponenten im voraus kennen (in Abb. 10 zwei für Puffer 2). Diese ist durch die Zahl der Empfänger an jedem lokalen Link gegeben. Da diese Angaben topologiespezifisch sind, müssen die entsprechenden Werte während der Flußgraphenkonfiguration für jeden lokalen Link festgestellt werden und der (aus Sicht des Links) sendenden Komponente mitgeteilt werden. Wann immer eine solche Komponente zur Laufzeit einen Puffer für (an Ausgangs-Ports) bereitzustellende Ausgangsdaten anfordert, teilt sie die Zahl der Empfängerkomponenten der Pufferverwaltung mit, so daß diese die Freigabe der Leserechte durch alle Komponenten erkennen kann.

[8] Dies gilt natürlich nicht für Links zur entfernten Kommunikation. Diese können vielmehr als Komponenten betrachtet werden, die Ressourcen reservieren (siehe 5.4 zur Analogie von Links und Komponenten).

Der Bedarf an Puffer ist nicht direkt topologieabhängig. Komponenten können prinzipiell davon ausgehen, daß empfangene Puffer durch vorgelagerte Komponenten (bzw. entfernte Links) reserviert werden. Außerdem ist der Bedarf an Puffern für interne Verarbeitungsfunktionen (Verarbeitungspuffer) sowie zur Bereitstellung der Daten an Ausgangs-Ports (Ausgangspuffer) zur Zeit der Erstellung der Komponente bekannt und kann somit (in der Ressourcenbedarfstabelle der Komponente) erfaßt werden. Auf der anderen Seite beeinflußt die (topologieorientierte) Zusammenfassung von Komponenten zu Verarbeitungs-Threads den Pufferbedarf. Wie im folgenden Abschnitt dargestellt wird, ist dieser Einfluß in CINEMA zur Zeit der Konfiguration eines Flußgraphen erkennbar, so daß er von Komponenten während des Session-Aufbaus berücksichtigt werden kann.

Der Ablauf zur Reservierung von Puffern ist daher ähnlich wie für andere Ressourcen. Ein Flußgraphenelement fordert einen benötigten Puffer an. Der Pufferverwalter überprüft, ob ein solcher Puffer vorhanden ist und, falls ja, wird ein entsprechender Speicherbereich reserviert. Dem Benutzer wird als Ergebnis ein Zeiger auf den Puffer zurückgeliefert. Eine Dienstgüte (wie Verzögerung bei der CPU) ist bei der Pufferreservierung nicht gegeben. Intern erfordert die Pufferverwaltung einen evtl. komplexen Mechanismus, um Speicherfragmentierung und somit eine schlechte Speicherausnutzung zu vermeiden. Die Implementierung sowohl der Puffer- als auch der Zugriffsverwaltung in CINEMA ist in [Mode94] beschrieben.

3.3.4 Thread-Bildung und Ressourcenreservierung

Komponenten und Links sind zunächst logische Objekte, die Code zur Ausführung von Verarbeitungs- und Kommunikationsfunktionen bereitstellen. Dieser Code wird zur Laufzeit in Scheduling-Einheiten der CPU, sogenannten Threads ausgeführt, wobei sich prinzipiell mehrere Komponenten in einem Thread zusammenfassen lassen. Wird ein Thread durch den CPU-Scheduler aktiviert, so werden die darin enthaltenen Komponenten nacheinander ausgeführt. In [Bart98] ist hierzu ein Laufzeitmechanismus beschrieben, der die Übergabe des Kontrollflußes zwischen den Komponenten in einem Thread bewerkstelligt.

Die einfachste Art Komponenten und Links auf Threads abzubilden besteht darin, für jedes solche Objekt ein eigenes Thread vorzusehen. In [Bart96] wird anhand des RM-Scheduling-Verfahrens dargelegt, daß dies zu einer Zunahme des Gesamtverzögerung aufgrund des Thread-

Schedulings führt. Hieraus wird gefolgert, daß insbesondere jede lineare Komponentenkette in einem Thread zusammengefaßt werden soll, um die damit verbundene Verzögerung auf eine Scheduling-Periode zu reduzieren (anstatt n Perioden im Falle von n unabhängigen Threads). Eine weitergehende Zusammenfassung wird nicht angestrebt, um Nebenläufigkeit in unabhängigen Verarbeitungsketten und dynamische Änderungen eines Flußgraphen zur Laufzeit[9] zu ermöglichen.

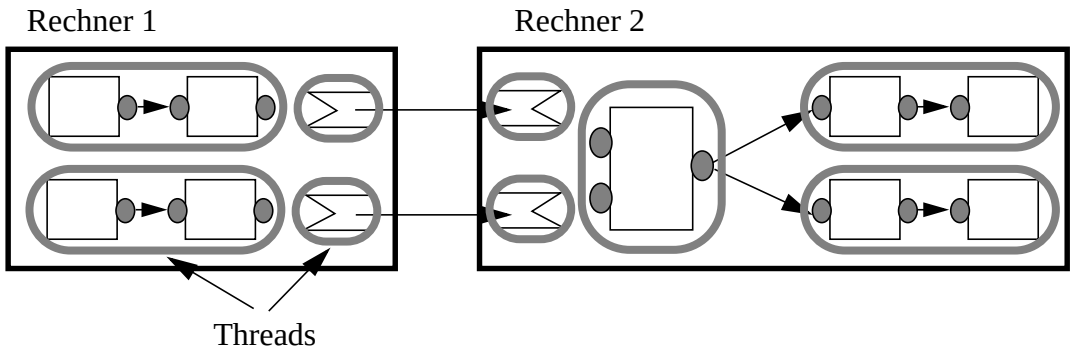


Abb. 11: *Thread-Bildung in CINEMA*

Eine Zusammenfassung einer Komponentenkette in einem Thread ist möglich, falls alle Komponenten mit derselben Aufrufrate aktiviert werden. Dies ist für fixe Filter sichergestellt, deren Aufrufrate von der Rate ankommender Mediendateneinheiten bestimmt wird und die diese Rate zwischen ihrem Eingangs- und Ausgangs-Port nicht verändern (d. h. die Filter generieren für jede erhaltene Medieneinheit eine Medieneinheit an dem Ausgangs-Port). Ein fixer Filter, der mehrere Dateneinheiten (z. B. Audio-Pakete) benötigt, um eine neue Einheit zu generieren, benötigt eine Aufrufrate, die dieser Generierung entspricht. Ein solcher Filter wird daher einem (von davorliegenden Komponenten) unabhängigen Thread zugeordnet, in dem aber (linear) nachfolgende Komponenten mit unveränderter Aufrufrate enthalten sein können.

Bei variablen Filtern werden zwei Arten unterschieden. Eine Gruppe umfaßt Filter, die den Inhalt medienspezifischer Dateneinheiten bearbeitet und hierbei Konversionen medienspezifi-

[9] Dynamische Änderungen umfassen die Hinzunahme bzw. Entfernung von Komponentenketten, die mit neuen Quellen und Senken verbunden sind.

scher Parameter, z. B. eine Bildverkleinerung, bewirkt. Die Rate generierter Dateneinheiten bleibt gegenüber der Empfangsrate unverändert. Eine zweite Gruppe umfaßt variable Filter, die die Empfangsrate auf eine niedrigere Ausgangsrate reduzieren können. Die Filter werden gemäß der reduzierten Rate aufgerufen, daher werden sie, analog zu oben, unabhängigen Threads zugeordnet.

Wie in 5.4 gezeigt wird, werden entfernte Links als senderseitige und empfängerseitige Link-Objekte strukturiert. Jedes dieser Objekte ist einem eigenen Thread zugeordnet, da es zur Übertragung von Medieneinheiten eigene Sende- bzw. Empfangsraten für Transportpakete verwendet und somit eine entsprechende Aufrufrate benötigt.

Obige Regeln zur Zuordnung von Komponenten und Link-Objekten zu Threads sind statischer Art, d. h. sie hängen lediglich vom Typ der betrachteten Komponenten sowie der Links (lokal oder entfernt) ab. Die Zuordnung kann daher sofort nach der Konfiguration des Flußgraphen stattfinden. Diese Aufgabe nimmt in CINEMA das Konfigurationsmanagement wahr, indem es jeder Komponente und jedem Link-Objekt einen Thread-Bezeichner zuweist. Komponenten, die dem selben Thread zugeordnet werden, erhalten den gleichen Bezeichner.

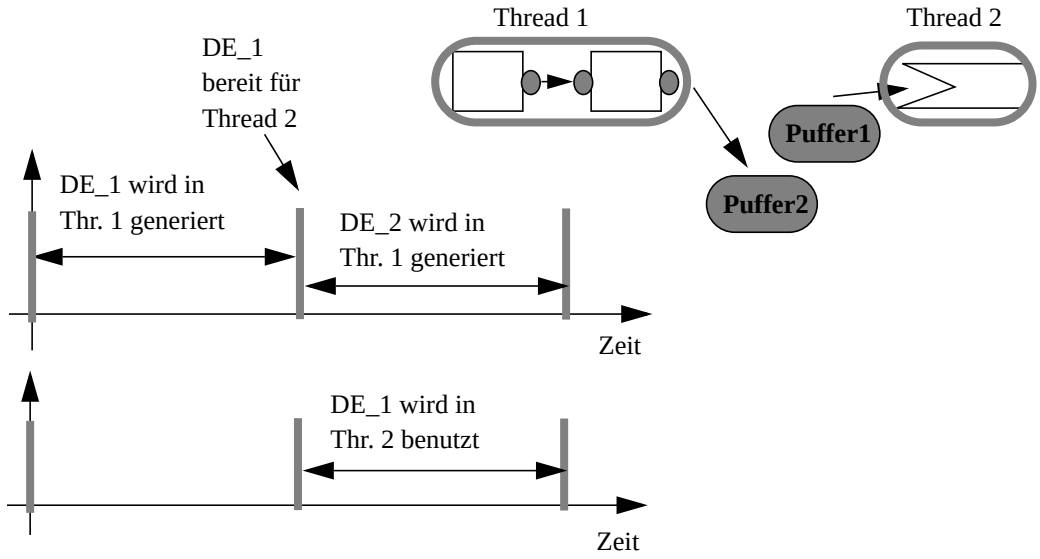


Abb. 12: *Speicherbedarf bei Datenübergabe zwischen Threads*

Wenn Komponenten und Links Ressourcen bei den Ressourcenmanagern anfordern, wird der Thread-Bezeichner übergeben. Jeder Ressourcenmanager ist in der Lage, die Anforderungen, die zum selben Thread gehören, zu aggregieren. Eine Aggregation ist in Bezug auf die CPU unerläßlich, da diese das Scheduling (und somit die Ressourcenvergabe) nur Thread-orientiert durchführen kann. Die Verarbeitungsdauer für ein Thread wird aus der Summe der Verarbeitungsdauern der „Thread-Komponenten“ bestimmt, während die Aufrufrate des Threads der (einheitlichen) Aufrufrate der Komponenten entspricht. Der Pufferbedarf wird dagegen aufgrund der maximalen Pufferanforderung (für jede Pufferart) berechnet, die durch eine der Komponenten bedingt wird. Diese Betrachtung ist möglich, da zu jedem Zeitpunkt nur eine Komponente Puffer benötigt.

Der Pufferbedarf für eine Komponente enthält zunächst durch den in 3.3.3 beschriebenen Pufferbedarf für eine für den Ausgangs-Port generierte Medieneinheit. Verarbeitet eine Komponente n empfangene Dateneinheiten auf einmal, so muß sie bei der Reservierung einen entsprechend höheren Bedarf (von n-1 Puffern für den Eingang und einen Puffer für den Ausgang) anmelden.

Für den Austausch von Dateneinheiten über Thread-Grenzen hinweg entsteht zusätzlicher Pufferbedarf, da die beteiligten Komponenten in unabhängigen Threads eingeplant werden. Dies wird anhand des in Abb. 12 dargestellten Falls deutlich, in dem eine von Thread 2 (über den Eingangs-Port der ersten Komponente) empfangene Dateneinheit während der Bearbeitung für die Dauer einer Periode bereitgehalten werden muß, eine Zeit, in der die nächste Dateneinheit vom vorangegangen Thread bereits erstellt wird.

Daher müssen gleichzeitig zwei von Thread 1 angelegte Ausgangspuffer existieren. Um diesen Effekt bei der Pufferbelegung zu erfassen, wird bei der Konfiguration der Threads der ersten Komponente von Thread 2 mitgeteilt, daß sie ihren Pufferbedarf entsprechend zweier Verarbeitungsaufrufe zu berechnen hat. Dies verdeutlicht einen weiteren Vorteil der Zusammenfassung von Komponenten in Threads, nämlich den eines reduzierten Pufferbedarfs.

3.4 Zusammenfassung

In diesem Kapitel wurden die wesentlichen Designentscheidungen beschrieben, die der Struktur des Ressourcenmanagements in CINEMA zugrundeliegen. Als Basis dient eine dreischichtige Dienstgütearchitektur, die Dienstgüteanforderungen auf verschiedenen Abstraktionsebenen definiert. Die Struktur des Ressourcenmanagements orientiert sich an der Dienstgütearchitektur und realisiert Mechanismen sowohl auf der Anwendungs- als auch auf der Systemebene. Auf der Systemebene sind insbesondere die Verwaltung und Bereitstellung der Kapazität von Endsystem- und Netzwerkressourcen zu realisieren, während auf der Anwendungsebene die Koordination der Ressourcenreservierung zwischen den Anwendungsteilen bereitzustellen ist.

Das Management der CPU in CINEMA stützt sich auf das bekannte „Rate-Monotonic“-Verfahren, während für die Vergabe von Hauptspeicher ein Pufferpool realisiert wurde, in dem Pufferbereiche belegt werden. Um lokale Puffertransfers effizient zu gestalten, werden Puffer lokal per Referenz übergeben. Das Konzept der Pufferverwendung erfordert, daß Komponenten für manipulierte Daten neue Puffer anfordern. Zugleich stellt es sicher, daß lokale Links nie Daten kopieren. Zur Kontrolle der Pufferfreigabe wird topologieabhängige Information benötigt, die von dem Konfigurationsmanagement bereitgestellt wird.

Die Abbildungen zwischen medienspezifischen Beschreibungen der Anwendungsebene und den Ressourcenanforderungen der Systemebene werden durch Ressourcenbedarfstabellen ermöglicht, die zur Zeit der Komponentenerstellung durch Messung festgestellt werden. Der Ressourcenbedarf wird ferner durch den Pufferbedarf beeinflußt, der aufgrund der Thread-Festlegung verursacht wird.

In CINEMA werden lineare Komponentenstrukturen in gemeinsamen Threads zusammengefaßt, sofern ihre Aufrufrate mit der Rate ankommender Mediendateneinheiten übereinstimmt. Filter, die dies nicht erfüllen, werden unabhängigen Threads zugeordnet. Die Thread-Festlegung wird durch das Konfigurationsmanagement vor Instanziierung eines Flußgraphen festgelegt. Dies ermöglicht insbesondere auch die Anzeige eines erhöhten Pufferbedarfs für Komponenten, die Daten über Thread-Grenzen hinweg empfangen.

Die Aggregation des Ressourcenbedarfs für Komponenten eines Threads wird in den Ressourcenmanagern der Systemebene durchgeführt, so daß Komponenten und Links keine Kenntnis

über die Art der Thread-Bildung besitzen müssen. Diese Eigenschaft ermöglicht in den folgenden Kapiteln eine Koordination der Ressourcenreservierung auf Anwendungsebene, die unabhängig von Aspekten der Thread-Bildung durchgeführt werden kann.

4 Ressourcenreservierung auf Anwendungsebene

In diesem Kapitel werden die Anforderungen und Voraussetzungen betrachtet, die für die Entwicklung von Reservierungsprotokollen ohne Task-Allokation wichtig sind. Solche Protokolle setzen eine a-priori Zuordnung von Komponenten zu Rechnern voraus, sie müssen also keine Mechanismen zur Rechnerauswahl bereitstellen. Ihr Design hängt von einer Reihe von Faktoren ab, deren unterschiedliche Festlegung zu unterschiedlichen Protokollen führen kann. In diesem Kapitel werden zunächst die Anforderungen dargestellt, die für CINEMA-Sessions gegeben sind. Anschließend werden bestehende Protokollansätze aus zwei Bereichen betrachtet: Reservierungsprotokolle für Transportsysteme bzw. Netzwerke (Abschnitt 4.2) sowie bestehende Reservierungsansätze auf der Anwendungsebene (Abschnitt 4.3). Ihre Eignung wird im Hinblick auf die formulierten Anforderungen diskutiert.

4.1 Ziele und Anforderungen

Ein Reservierungsprotokoll der Anwendungsebene ist ein wesentlicher Bestandteil des Session-Aufbaus. Es ist für Koordinationsaufgaben verantwortlich, die einen gesamten Flußgraphen betreffen. Es hat dafür zu sorgen, daß die von einem Klienten vorgegebenen A-QoS-Anforderungen für medienspezifische und generische Parameter eingehalten werden. Entsprechend muß es eine Koordination in Bezug auf beide Parameterarten realisieren.

Die *Koordination für medienspezifische Parameter* hat eine konsistente Wahl von Parameterwerten an allen Komponenten-Ports zum Ziel. Die Wahl von Parameterwerten wird durch A-QoS Anforderungen, durch funktionale Vorgaben von Komponenten und durch eine evtl. beschränkte Verfügbarkeit von Ressourcen limitiert. Funktionale Vorgaben sind durch Formatbeschränkungen und Stromrelationen definiert. Im Abschnitt 2.2 wurde gezeigt, daß bereits ihre Berücksichtigung flußgraphenweite Zusammenhänge bewirkt, die durch das Reservierungsprotokoll einzuhalten sind.

Eine verteilte Multimedia-Anwendung erbringt sowohl Kommunikations- als auch Verarbeitungsfunktionen. Sie benötigt daher eine *globale Reservierung* aller verwendeten Ressourcen auf Rechnern (Endsystemen) und in Netzwerken. Die Reservierung solcher Ressourcen kann

nicht isoliert betrachtet werden. Ermöglichen zum Beispiel Quelle und Senke eines Stromes eine hohe Qualität, so muß auch die dazwischenliegende Kommunikationsstrecke eine hohe Bandbreite ermöglichen. Umgekehrt gilt, daß wenn keine ausreichende Kommunikationsbandbreite vorhanden ist, auch die Verarbeitung nur eine verringerte Qualität liefern kann. Es ist erforderlich, daß ein Protokoll die Ressourcenreservierungen auf unterschiedlichen Rechnern (hier der Quelle und der Senke) sowohl miteinander als auch mit Reservierungen im Netzwerk koordiniert.

In vielen Fällen muß ein Rechner mehrere Komponenten eines Flußgraphen aufnehmen können. In Konferenzsituationen sind typischerweise auf jedem Rechner eines Endbenutzers eine Quelle und eine Senke plaziert. Häufig sind Filter oder Mischer zusätzlich gegeben. Die Vergabe von Ressourcen an solche Komponenten sollte so erfolgen, daß sie eine möglichst hohe Dienstgüte erlaubt. Es ist z. B. nicht sinnvoll, einer Senke sehr viel an Ressourcen zuzuteilen, wenn die Quelle anschließend keine oder nur noch wenig an Ressourcen erhalten kann. Ein Reservierungsprotokoll muß daher neben der Reservierungskoordination zwischen verteilten Rechnern auch eine rechnerbezogene Koordination durchführen. Letztere wird im folgenden als *lokale Koordination* bezeichnet.

Ein Reservierungsprotokoll muß sicherstellen, daß die vom Klienten an den Senken spezifizierten A-QoS eingehalten werden. Da ein A-QoS für medienspezifische Parameter *Wertebereiche* enthalten kann, hat das Protokoll die Freiheit, einen möglichen Wert für jeden Parameter an jeder Senke auszuwählen. Dies hat zwei Folgen. Zum einen können geeignete Parameterwerte auch dann bestimmt werden, falls die maximal möglichen Werte, z. B. aufgrund von Ressourcenmangel, nicht unterstützt werden können. Dies nimmt dem Klienten die Last ab, wiederholt den Aufbau einer Session mit schrittweise reduzierten A-QoS anzustoßen. Zum anderen soll das Protokoll im Rahmen der Koordination der Parameterwertewahl möglichst hohe A-QoS ermöglichen. Das Protokoll soll insbesondere gewährleisten, daß Parameter-Reduktionen für einzelne, *heterogene Senken* (siehe 2.2) auch isoliert vorgenommen werden können.

Die *Koordination für generische Parameter* hat die Ende-zu-Ende-Berechnung der Verzögerung zum Ziel, die an Senken eines Flußgraphen wahrgenommen wird. Da Verzögerung durch Verarbeitungskomponenten und Kommunikation erzeugt wird, muß ein Reservierungsprotokoll alle solchen Anteile geeignet zwischen Quellen und Senken aufaddieren. Dabei muß überprüft

werden, ob für jede Senke der vorgegebene maximale Wert aus der A-QoS eingehalten werden kann. Ähnlich wie für medienspezifische Parameter kann für jede Senke eine eigene Obergrenze für die Verzögerung spezifiziert werden, wobei das Protokoll die Freiheit hat, einen beliebigen Wert unterhalb dieser Obergrenze zu bestimmen.

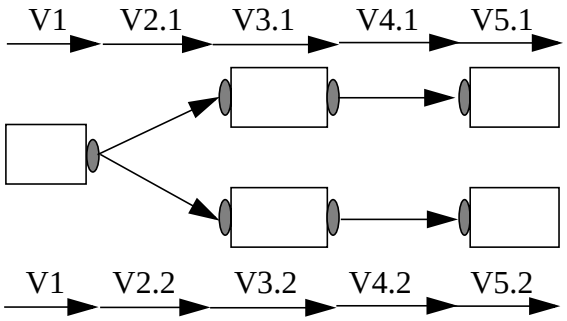
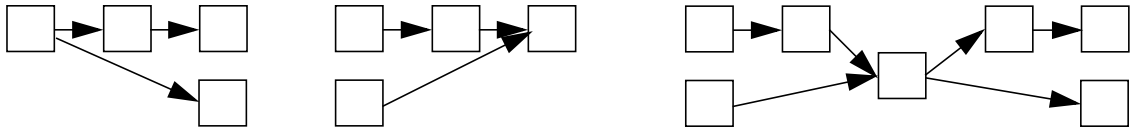


Abb. 13: *Ende-zu-Ende Verzögerung in einem Flußgraphen*

In Abschnitt 3.3 wurde gezeigt, daß die Ermittlung der Verzögerung für Komponenten von den eingestellten medienspezifischen Parameterwerten abhängt. Ein ähnlicher Zusammenhang gilt für Links, die Transportsysteme benutzen. Z. B. spezifiziert das IETF eine Transportdienstschnittstelle, die die Verzögerung explizit von der angeforderten Bandbreite abhängig macht [IETF96]. Die endgültige Berechnung der Verzögerung muß sich daher auf eine endgültige Wahl medienspezifischer Parameterwerte stützen können. Die Koordination für generische und medienspezifische Parameter muß entsprechend aufeinander abgestimmt sein.



(a) Multicast-Topologie (b) Mischer-Topologie (c) Konferenz-Topologie

Abb. 14: *Flußgraphentopologien*

Multimedia-Anwendungen bedingen *Flußgraphen unterschiedlicher Topologie und Verteilung*. Um möglichst allgemein anwendbar zu sein, sollte ein Reservierungsprotokoll alle gängigen Topologien abdecken. Insbesondere soll es Reservierung auch für *zwischengelagerte Kompo-*

nenten (Filter und Mischer) ermöglichen. Gängige Topologien für Multimedia umfassen: Multicast-Topologien, in denen mehrere Senken aus einer Quelle gespeist werden, Mischer-Topologien, in denen eine Senke aus mehreren Quellen gespeist wird sowie Konferenztopologien, in denen mehrere Quellen Mischer speisen, die gemischte Ströme an Senken verteilen (Abb. 2).

Entsprechend der erwähnten Szenarien soll ein Reservierungsprotokoll unabhängig von der Verteilung der Komponenten auf Rechnern sein. Insbesondere sollen Quellen und Senken auf demselben Rechner vorhanden sein können, wie dies für Konferenzsituationen typisch ist. Genauso sollen Filter und Mischer auf Rechnern gegeben sein können, die von Quellen und Senken unabhängig sind. Dies entspricht der Situation, in der Anwendungskomponenten auch auf *zwischengelagerten Rechnern* gegeben sind, die spezielle Verarbeitungsfunktionen bieten. Solche Rechner („High-Level Gateways“) wurden im Kontext verschiedener Multimedia-Anwendungen realisiert [Sren96], [SSJH96], [AMZ95].

4.2 Reservierungsprotokolle auf Transport- und Netzebene

Die ersten Reservierungsprotokolle wurden entworfen, um Transportverbindungen zwischen Endsystemen aufzubauen, die eine gewünschte Dienstgüte garantieren können. Um dies zu erreichen, werden Kommunikationsressourcen auf allen beteiligten Knoten, d.h. auf Endsystemen und Knoten im Netzwerk (Router, Switches) reserviert. Zur Koordination dieser Reservierungen werden entsprechende Reservierungsprotokolle benötigt, die im Kontext von Netzwerktechnologien auch als Signalisierungsprotokolle bezeichnet werden.

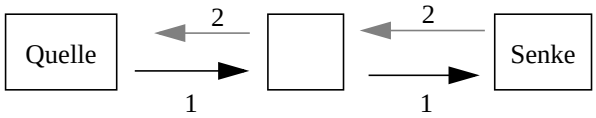
Aus Sicht von Sender und Empfänger von Datenströmen realisiert ein Signalisierungsprotokoll einen Dienst, der die Spezifikation gewünschter Dienstgüte erlaubt und eine entsprechende Verbindung durch das Netzwerk bereitstellt. Die Dienstgüte kann sich dabei sowohl auf Verkehrsparameter als auch auf generische Parameter wie Verzögerung beziehen (siehe 3.1). Verschiedene Dienstklassen wurden definiert, um die Art der Garantie für die erbrachte Dienstgüte zu unterscheiden [BrSt97], [BCS94], [ATM96a], [ATM96b]. Für verteilte Multimedia-Systeme sind die Klassen von Interesse, die Garantien zumindest in Bezug auf Verkehrsparameter liefern. Für Konferenzsituationen mit Echtzeitanforderungen werden zusätzlich Garantien in Bezug auf eine begrenzte Verzögerung benötigt.

Sowohl innerhalb der Internet- als auch der ATM-Gremien (IETF, ITU, ATM-Forum) sind Signalisierungsprotokolle definiert worden, die Dienstgütegarantien erbringen. Im folgenden werden die wichtigsten von ihnen, ST-II, RSVP und die ATM-Signalisierung, eingeführt, wobei der Schwerpunkt auf die von ihnen erbrachten Dienste gelegt wird. Für eine ausführliche Darstellung der internen Mechanismen zur Dienstrealisierung wird auf die entsprechende Literatur verwiesen ([BrZi96]).

4.2.1 Internet Stream Protocol II (ST-II)

ST-II ([Topo90], [DeBe95]) verwendet das sogenannte „Stream Control Message Protocol“ (SCMP) um Verbindungen aufzubauen. SCMP ermöglicht den Aufbau von Unicast- bzw. Multicastverbindungen zwischen einem Sender und einem bzw. mehreren Empfängern. Der Aufbau beider Verbindungsarten wird senderseitig initiiert (Abb. 15).

Die Dienstgüte für eine Verbindung wird vom Sender spezifiziert und umfaßt einen Verkehrsdeskriptor (Paketgröße, Bandbreite) sowie die Angabe von Verzögerung und Jitter. Für jeden Parameter ist ein Ziel- und ein minimal (oder maximal) akzeptabler Wert angebbar. Aufgabe der Signalisierung ist es, eine Verbindung aufzubauen, die Dienstgütewerte innerhalb der Wertebereiche, möglichst nahe bei den Target-Werten aufweist. Dabei erfordert ST-II, daß die Dienstgüte für alle Empfänger gleich ist, heterogene Empfänger werden nicht unterstützt.



1: connect, 2: accept

Abb. 15: *Ablauf des ST-II Verbindungsaufbaus*

Aus Sicht von Sender und Empfänger läuft der Verbindungsaufbau wie folgt ab. In einer ersten Phase spezifiziert der Sender die gewünschte Dienstgüte und übergibt diese an den ST-II-Agenten der Sendermaschine. ST-II Agenten propagieren „FlowSpecs“ durch das Netzwerk zum Empfänger, wobei auf jedem durchlaufenen Netzwerkknoten ein ST-II-Agent lokale Ressourcen reserviert, den nächsten anzusprechenden Knoten auswählt und die Link-Ressourcen zu

diesem Knoten reserviert. Auf der Empfängermaschine wird der Empfänger über die Dienstgüte benachrichtigt, die der Sender gewünscht hat und die vom Netzwerk unterstützt werden kann. Insbesondere werden einem Empfänger die maximal mögliche Verkehrslast und die minimal mögliche Verzögerung angezeigt. Der Empfänger hat die Möglichkeit, die Verbindung anzunehmen (wobei die „FlowSpec“-Parameter verändert werden können) oder abzulehnen. Die veränderte „FlowSpec“ wird in der zweiten Phase durch das Netzwerk zum Sender propagiert, wobei dem Sender die Dienstgüte angezeigt wird, die aufgrund der Reservierungen im Netzwerk und bei dem Empfänger erbracht werden kann. SCMP sorgt in dieser Phase dafür, daß Ressourcenreservierungen gemäß der evtl. angepaßten „FlowSpec“-Parameter freigegeben werden.

Für Multicast-Verbindungen erfordert SCMP zwei Schritte. In einem ersten Schritt (*Connect*) wird die Verbindung zu allen Empfängern wie oben aufgebaut und dem Sender die Dienstgüte signalisiert, die für alle Empfänger in gleicher Weise erbracht werden kann. Um diese endgültige Dienstgüte an die Empfänger mitzuteilen wird ein zweiter Schritt (*Change*) benötigt, bei dem SCMP die Reservierungen entsprechend der endgültigen Parameterwerte für alle Empfänger gleich einstellt.

4.2.2 Resource Reservation Protocol (RSVP)

RSVP wurde dediziert als eigenständiges Signalisierungsprotokoll entwickelt, das unabhängig von der Verwendung eines bestimmten Datentransferprotokolls existiert ([ZDES93]). Es wurde vor allem auch entwickelt, um Ressourcenreservierung in Szenarien zu unterstützen, in denen eine Vielzahl von Empfängern Ströme von einem oder mehreren Sendern empfangen.

Dies wird vor allem dadurch erreicht, daß Empfänger keine Protokollnachrichten an Sender verschicken. Ein Sender stellt über sogenannte Pfadnachrichten an alle potentiellen Empfänger Information zur Verfügung, die u. a. das Format der Datenströme („SenderTemplate“), den bereitgestellten Verkehr („SenderTrafficSpec“) sowie eine sogenannte „Advertisement Specification“ enthält. Letztere umfaßt Angaben, die eine Dienstgütewahl eines Empfängers beeinflussen können, wie z. B. Schätzungen der gegenwärtig verfügbaren Netzwerkbandbreite und Angaben von Größen, die die Verzögerung zwischen Sender und Empfänger beschreiben. Pfad-

nachrichten legen zu jedem Empfänger eine eindeutige Route fest, sie veranlassen aber keine Ressourcenreservierung.

Jeder Empfänger kann aufgrund erhaltener Pfadnachrichten eine Reservierungsnachricht zusammenstellen, die in umgekehrter Richtung zu den Pfadnachrichten durch das Netz propagiert wird. Eine solche Nachricht enthält Angaben zu den gewünschten Datenströmen („FilterSpec“) sowie der gewünschten Dienstgüte („FlowSpec“). Ein Filter kann dabei mehrere Ströme identifizieren, die empfangen werden sollen, wobei über verschiedene Filtertypen verschiedene Benutzungsstile (z. B. „alle Ströme werden auf einmal benötigt“, „immer nur ein Strom wird benötigt“) beschrieben werden können.

Die „FlowSpec“ beschreibt die gewünschte Dienstgüte wie z. B. die erforderliche Bandbreite, Paketgröße sowie die Verzögerung, aufgrund derer entlang des Pfades zu einem Sender hin Ressourcen belegt werden sollen. Hierbei wird der Pfad benutzt, der durch die Pfadnachrichten, die den Empfänger vom Sender erreicht haben, festgelegt wurde, wobei eine Reservierungsnachricht nur bis zum ersten Netzknoten propagiert wird, der bereits mit den angeforderten Strömen versorgt wird. RSVP unterstützt heterogene Empfänger, indem es für jeden Empfänger eine individuelle Spezifikation gewünschter Dienstgüte zuläßt.

4.2.3 ATM-Signalisierung

Die ATM-Signalisierung ([ATM94], [ATM96b]) ermöglicht den Aufbau eines sogenannten „Virtual Circuits“ (VC) zwischen einem Sender und einem Empfänger. Anders als für die bislang erwähnten Protokolle umfaßt ein VC bidirektionale Kommunikationskanäle. Der Aufbau eines VC kann von jeder Seite initiiert werden. Neben Unicast-VCs, werden auch Multicast-VCs unterstützt. Diese sind unidirektional und werden grundsätzlich vom Sender aus initiiert.

Die Behandlung von Dienstgüte für VCs wird für ATM über die Definition von Dienstklassen unterschieden ([ATM96a]). Dienstgütegarantien für die Verkehrslast (Bandbreite) liefern die Klassen „Constant Bit Rate“ (CBR), „Real-Time Variable Bit Rate“ (rt-VBR), „Non-Real-Time Variable Bit Rate“ (nrt-VBR) und - für eine minimale Verkehrslast - „Available Bit Rate“ (ABR). Garantien für die Übertragungsgüte (Verzögerung, Jitter, Verlustrate) liefern CBR und rt-VBR, während nrt-VBR lediglich eine maximale Verlustrate garantiert.

Für Dienstgüteparameter sind keine Wertebereiche angebbar. Vielmehr sind eindeutige Werte erforderlich, für die der Aufbau einer Verbindung versucht wird. Der Initiator einer Verbindung wird lediglich über Erfolg oder Mißerfolg des Verbindungsaufbaus informiert, ohne Anzeige einer reduzierten Dienstgüte, auch wenn diese im Prinzip möglich ist. Dem Initiator bleibt nur übrig, den Verbindungsaufbau erneut mit reduzierter Dienstgüte anzustoßen.

Der Verbindungsaufbau ist ähnlich zu der SCMP-Vorgehensweise und wird durch das Signalisierungsprotokoll Q.2931 in zwei Phasen abgewickelt. In der ersten Phase werden „Set-Up"-Nachrichten vom Sender zum Empfänger durch das Netzwerk propagiert, wobei beteiligte Knoten benötigte Ressourcen reservieren. Der Empfänger hat die Möglichkeit die Verbindung anzunehmen oder abzulehnen. Zur Annahme wird eine „Connect"-Nachricht zum Sender zurückpropagiert. Multicast-Verbindungen werden durch schrittweise Hinzunahme eines neuen Empfängers initiiert, wobei für jede neue Hinzunahme im Kern die Signalisierung für die neue Unicast-Verbindung wiederholt wird.

4.2.4 Bewertung der Ansätze

Die vorgestellten Reservierungsprotokolle sorgen für die Bereitstellung der Ressourcen, die für Verbindungen zwischen Endsystemen erforderlich sind. Sie sind in der Lage, Dienstgütegarantien sowohl in Bezug auf Verkehrsparameter als auch für generische Parameter zu liefern. Verteilte Multimedia-Anwendungen, die eine Dienstgüte auf der Anwendungsebene sicherstellen müssen, erfordern solche Garantien für jede der benötigten Verbindungen. Die Reservierung auf Anwendungsebene muß daher die Reservierungsprotokolle der Transport- bzw. Netzebene miteinbeziehen. Auf der anderen Seite erfordert die Reservierung auf Anwendungsebene Mechanismen, die neben der Kommunikation auch die Verarbeitung im Zusammenhang berücksichtigen.

Da die betrachteten Reservierungsprotokolle für die Vermittlungs- bzw. Transportebene entwikkelt wurden, behandeln sie nicht die im Abschnitt 4.1 beschriebenen Anwendungsaspekte. Sie realisieren keine flußgraphenweite Koordination für medienspezifische Parameter. Reservierungen finden vielmehr aufgrund von Verkehrsparametern statt und die Koordination gilt lediglich Verbindungen zwischen unmittelbar kommunizierenden Endsystemen. Eine Reservierung für Verarbeitungskomponenten mit Hilfe medienspezifischer Parameter ist genauso wenig erfaßt

wie die Koordination dieser Reservierungen zwischen Komponenten auf einem oder mehreren Rechnern. Insbesondere werden funktionale Einschränkungen wie Stromrelationen und Formatbeschränkungen nicht berücksichtigt. Da eine verteilte Anwendung zudem mehrere Transportverbindungen gleichzeitig umfassen kann, muß die Reservierung nicht nur einzeln für jede Verbindung erfolgen, sondern auch eine Koordination zwischen den Reservierungen für die einzelnen Verbindungen beinhalten.

Gleiches gilt für die Koordination generischer Parameter. Um Verzögerung zwischen Quellen und Senken zu berechnen, müssen alle Anteile berücksichtigt werden. Werden zwischen einer Quelle und einer Senke auch zwischengelagerte Rechner benutzt, so müssen neben Verzögerungen aufgrund der Verarbeitung auch die Verzögerungsanteile mehrerer Verbindungen gleichzeitig berücksichtigt werden.

4.3 Reservierungsprotokolle auf Anwendungsebene

4.3.1 Connection Manager

In [HuSc97] wird ein Konzept beschrieben, das den Aufbau von Verbindungen zur Übertragung von Audio- oder Videodaten für Anwendungen ermöglicht, die aus einer Quelle auf einem Quellrechner und einer Senke auf einem Senkenrechner bestehen. Falls nötig, kann auf dem Quellrechner eine sogenannte Skalierkomponente vor der Übertragung eingesetzt werden, um die Verkehrslast zu verringern.

Die Art des zu kommunizierenden Datenstromes wird über einen Medientyp, z. B. „Video“ spezifiziert. Ein Medientyp legt generische Parameterwerte („geringe Verzögerung“, „geringer Jitter“) und die Menge verwendbarer Formate fest. Ein Format beschreibt eindeutig die Charakteristika eines Stroms, z. B. PCM mit 11.025 kHz / 16 bit Abtastung. Sowohl Quelle als auch Senke können mehrere Formate unterstützen. Das beschriebene Verfahren ermöglicht die Bereitstellung des „richtigen“ Formats in Abhängigkeit von der verfügbaren Transportdienstgüte.

Als zentrales Managementmodul dient ein „Connection Manager“ (CM) auf jedem Rechner. Die Anwendung ruft dieses zur Anforderung einer Verbindung mit einer Medientypangabe auf.

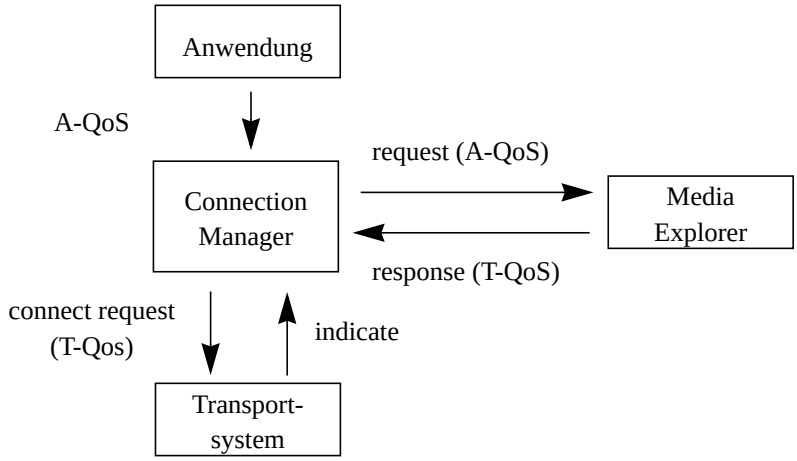


Abb. 16: *Der Connection Manager Ansatz*

Der CM übergibt die Spezifikation an einen sogenannten Media-Explorer, der zunächst das beste verfügbare Format auswählt und die dafür benötigte Transportdienstgüte bestimmt. Sowohl die Formatauswahl als auch die Abbildung auf die Transportdienstgüte sind im voraus festgelegt, wobei die Festlegung im System konfigurierbar ist. Der CM ruft mit der abgeleiteten Dienstgüte ein Transportsystem zum Aufbau einer Transportverbindung auf. Gelingt der Aufbau, so wird das ausgewählte Format der Skalierkomponente mitgeteilt. Ist kein Aufbau möglich, weil Ressourcen nicht ausreichend vorhanden sind, fordert der CM vom Media Explorer das Format mit minimaler Transportdienstgüte an und stößt den Verbindungsaufbau erneut an.

Das vorgeschlagene System realisiert im wesentlichen Module, die für die Formatauswahl, die Abbildung zwischen Parametern der Anwendungs- und Transportebene und der Steuerung des Aufbaus von Transportverbindungen benötigt werden. Auf der anderen Seite unterstützt das System lediglich einfache Erzeuger/Verbraucher-Topologien und schließt die Reservierung von Ressourcen auf den Endsystemen nicht mit ein.

4.3.2 QoS - Broker

Der „QoS-Broker“ ([NaSm95]) ermöglicht die Bestimmung der Dienstgüte für Anwendungen, in denen eine Quelle mit mehreren verteilten Senken verbunden ist oder mehrere Quellen mit einer entfernten Senke kommunizieren. Zwischengelagerte Komponenten werden nicht unterstützt. Zur Dienstgüteaushandlung unterscheidet das Konzept sogenannte „QoS-Buyer“ und

„Seller“. Jedem Endsystem ist eine solche Einheit zugeordnet, die einerseits für das lokale (endsystembezogene) Ressourcenmanagement zuständig ist, andererseits mit einem Netzwerksystem zum Aufbau von Verbindungen interagiert. Der Ablauf zum Aufbau einer Session ist wie folgt.

Der „QoS-Buyer“ (z. B. der Senke einer Anwendung zugeordnet) ermittelt zuerst lokal, für welche medienspezifische Dienstgüte Ressourcen ausreichend vorhanden sind. Anschließend stellt er in Interaktion mit dem Netzwerk fest, welche Transportressourcen (z. B. Bandbreite) auf dem Weg zum „QoS-Seller“ verfügbar sind. Schließlich fragt er den „QoS-Seller“, welche Dienstgüte aufgrund der Ressourcenlage im Netzwerk und auf dem Rechner des „QoS-Buyers“ durch den „QoS-Seller“ unterstützt werden kann. Das Modul des „QoS-Sellers“ ermittelt die unterstützbare Dienstgüte aufgrund der lokalen Ressourcenlage und teilt sie dem „QoS-Buyer“ mit.

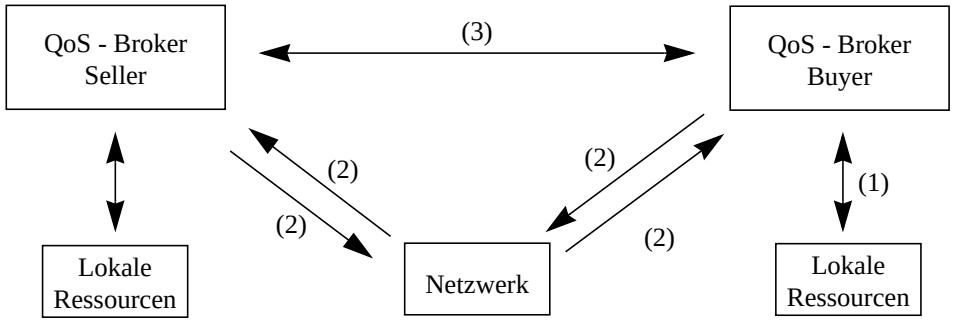


Abb. 17: *Der QoS-Broker Ansatz*

Der Ansatz ermöglicht sowohl die Koordination für medienspezifische als auch generische Parameter (Verzögerung), wobei Wertebereiche für Diensgüteanforderungen unterstützt werden. Es wird die bestmögliche Dienstgüte eingestellt und es werden Ressourcen sowohl auf den Endsystemen und im Netzwerk reserviert. Auf der anderen Seite werden keine heterogenen Empfänger unterstützt und die betrachteten Flußgraphen lassen keine zwischengelagerten Komponenten zu.

4.3.3 Graph Manager

Das in [Wolf96] beschriebene Konzept erlaubt eine Ressourcenreservierung für komplexere Flußgraphen. Es verwendet einen Quellrechner, der über (Unicast- oder Multicast-) Verbindungen mit einem oder mehreren Senkenrechnern verbunden ist. Zwischengelagerte Verarbeitungsrechner sind nicht vorgesehen. Auf jedem Rechner ist ein sogenannter „Stream Handler Graph“[10] (SHG) gegeben, der den (als zusammenhängend vorausgesetzten) Teilgraphen der Anwendung auf diesem Rechner bildet. Der SHG wird auf jedem Rechner durch den lokalen „Graph Manager“ (GM) und das lokale Ressourcenmanagementsystem (RMS) verwaltet. Ziel der Ressourcenreservierung ist es, für eine Anwendung eine möglichst hohe Dienstgüte in Bezug auf medienspezifische Parameter zu garantieren.

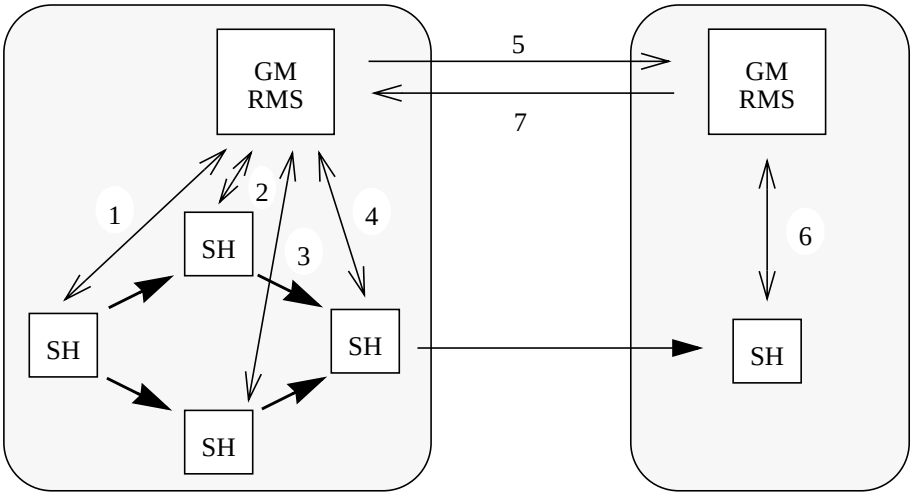


Abb. 18: *Der Graph Manager Ansatz*

Die Anwendung startet die Reservierung durch die Spezifikation der geforderten (medienspezifischen) Parameterwertebereiche für den SGH auf dem Quellrechner. Der GM sorgt dafür, daß alle SH ihren Ressourcenbedarf an das RMS melden, wobei der Ressourcenbedarf entsprechend der höchsten Dienstgüteeinstellung ermittelt wird. Das RMS überprüft, ob die Reservierung für den SHG möglich ist. Falls nicht, reduziert der GM die Dienstgüteanforderung auf die nächstniedrigere Stufe. Das Verfahren wird wiederholt, bis eine Stufe erreicht ist, für die die Reservie-

[10] Eine Komponente in CINEMA entspricht einem Stream Handler.

rung erfolgreich ist. Anschließend wird der Aufbau von Verbindungen zu Senkenrechnern mit ST-II angestoßen, wobei die geforderte Verkehrslast aus der ermittelten Dienstgüte abgeleitet wird. Der Aufbauwunsch wird jedem Senkenrechner durch ST-II angezeigt.

Der GM eines Senkenrechners führt die lokale Reservierung für seinen SHG durch, bevor er eine ST-II-Verbindung akzeptiert. Ist die Reservierung nicht möglich, wird die Verbindung abgelehnt. Der GM auf dem Quellrechner reduziert in diesem Fall für den SHG die Dienstgüte und stößt den Verbindungsaufbau erneut an. Dieser Ablauf wiederholt sich, bis der Senkenrechner für seinen SHG und die vorgegebene Dienstgüte reservieren kann. Im schlimmsten Fall kann für jede Dienstgütestufe ein Verbindungsaufbau versucht werden.

Der Ansatz hat eine Reihe von Einschränkungen. So können keine zwischengelagerten Verarbeitungsrechner verwendet werden. Da das Verfahren keine variablen Filter unterstützt, ist keine entkoppelte Einstellung von Dienstgüte für heterogene Senken möglich. Ferner ist die Art der Dienstgütereduzierung für medienspezifische Parameter unter Umständen sehr aufwendig. Die Signalisierung für jede ST-II Verbindung wird im schlimmsten Fall k mal durchgeführt, falls auf der Anwendungsebene k Dienstgütestufen gegeben sind.

4.3.4 Ripple Scheduling

„Ripple Scheduling“ [HWVC97], [HWK96] ermöglicht Ressourcenreservierung für Flußgraphen mit beliebiger Baumtopologie, wobei Komponenten auf Quell-, Senken- und zwischengelagerten Rechnern plaziert sein können. Komponenten und Links, die auf demselben Rechner plaziert sind, werden als ein einziges Verarbeitungsmodul aufgefaßt. „Ripple Scheduling“ wikkelt die Koordination der Reservierung zwischen den verteilten Modulen ab.

Die Aushandlung wird an einem besonderen Modul, dem Koordinator, initiiert. Die Wahl des Koordinators ist beliebig. Jedem Modul (und damit jedem Rechner) ist ein „Distributed System Resource Manager“ (DSRM) zugeordnet, welcher in Interaktion mit verbundenen DSRM das „Ripple Scheduling“ realisiert. Hierzu tauschen die DSMR Nachrichten aus, die den gewünschten Wertebereich der Dienstgüte enthalten. Der Austausch der Nachrichten wird in drei Phasen abgewickelt.

In der ersten Phase (*Reserve*) werden Nachrichten vom Koordinator zu den Blättern des Flußgraphen propagiert. Jedes Modul reserviert dabei lokale Ressourcen und baut Verbindungen (mit entsprechender Transportdienstgüte) zu benachbarten Modulen auf. Bei jedem Schritt kann der Wertebereich der Dienstgüte eingeschränkt werden, falls Ressourcen nur begrenzt vorhanden sind. In der zweiten Phase (*Acknowledge*) werden die Wertebereiche von den Blättern zum Koordinator zurückpropagiert, wo die Bestimmung der endgültigen Dienstgüte erfolgt. Diese wird in einer dritten Phase den Blättern mitgeteilt (*Commit*), in der Ressourcen entsprechend der evtl. reduzierten Dienstgüte freigegeben werden.

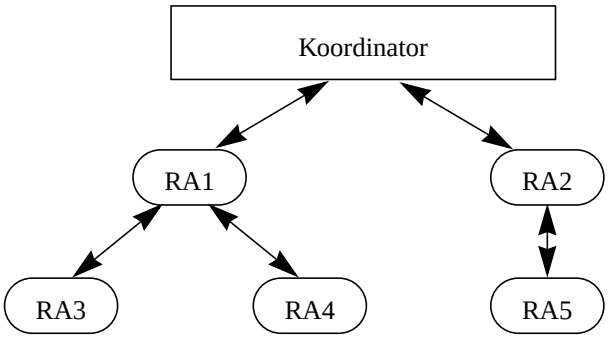


Abb. 19: *Struktur des Ripple Scheduling*

Der Vorteil des Ansatzes liegt in seiner dezentralen Organisation. Es gibt keinen zentralen Koordinator, der den Zustand des Flußgraphen verwalten muß und der deswegen einen entsprechenden Mehraufwand bedingt. Das Verfahren führt eine globale Reservierung durch. Die geforderte Dienstgüte wird zentral am Koordinator für den gesamten Flußgraphen spezifiziert, eine Unterstützung für heterogene Empfänger ist somit nicht vorgesehen. Ferner werden keine Formatbeschränkungen berücksichtigt und die Behandlung generischer Parameter ist nicht beschrieben.

4.3.5 Hierarchical Negotiation

Ein zentralisierter Ansatz zur Ressourcenreservierung wurde in [Hafi95] vorgestellt. Der Aufbau einer Session wird in zwei Schritten durchgeführt. In einem ersten Schritt werden aufgrund einer Dienstspezifikation die benötigten Komponenten bestimmt und geeignete Rechner zur

Plazierung dieser Komponenten ausgewählt. In einem zweiten Schritt wird die Ressourcenreservierung durchgeführt. Zum Session-Aufbau wird eine Dienstgüteanforderung spezifiziert, die medienspezifische Parameter beinhaltet.

Das Dienstgüte-Management ist durch eine Hierarchie von QoS-Managern realisiert. Auf der untersten Stufe sind QoS-Agenten (QA) für das Management einzelner Komponenten und Links zuständig. Mehrere QoS-Agenten können auf einer höheren Stufe durch „QoS-Domain-Manager“ (DM) verwaltet werden. Z. B. ist ein DM für jeden Rechner vorhanden. Auf der höchsten Stufe ist ein zentraler QoS-Manager (ZM) für die Anwendung zuständig. Der Ablauf der Reservierung ist wie folgt.

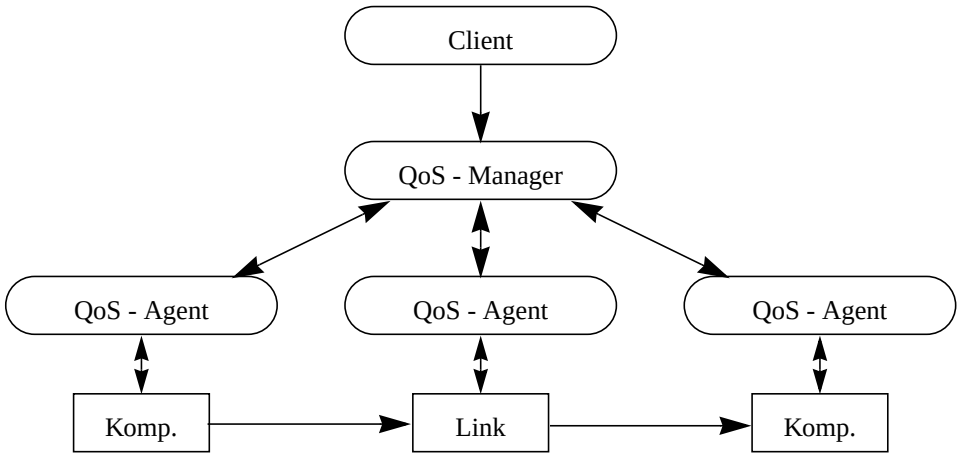


Abb. 20: *Hierarchical Negotiation*

Der ZM leitet aus der Dienstgüteanforderung für die Anwendung eine Anforderung für jede Komponente und Link ab. Über die DM werden diese Anforderungen an die QA der Komponenten und Links geleitet. Diese nehmen die jeweilige Reservierung für die bestmögliche Dienstgüte vor. Dabei können sie miteinander kommunizieren, um eine (rechner-) lokale Koordination der Reservierungen durchzuführen. Die jeweils eingestellte Dienstgüte wird von den QA an den ZM gemeldet, der eine globalen Abgleich der Dienstgüte-Wahl durchführt. Die eingestellte Dienstgüte wird den QA mitgeteilt, die eine Reservierungsrelaxierung durchführen können.

Der Ansatz hat die Beschreibung einer Management-Architektur zum Ziel. Diese erfordert eine zentrale Instanz für die letzte Entscheidung über die einzustellende Dienstgüte. Der Ansatz beschreibt zwar die Kommunikationsstruktur, die zur Aushandlung nötig ist, läßt aber offen, welche Verfahren zur globalen und lokalen Koordination der Reservierung anzuwenden sind. Dementsprechend werden keine Aussagen über die Art der auszuhandelnden Parameter sowie der unterstützbaren Flußgraphentopologien und der verwendeten Komponenten gemacht.

4.3.6 Bewertung der Ansätze

Die vorgestellten Verfahren ermöglichen Ressourcenreservierung auf der Anwendungsebene in unterschiedlichem Ausmaß. Die frühen Ansätze („Connection Manager“, „QoS Broker“) sind vor allem als Ergänzung zur Ressourcenreservierung im Netzwerk entwickelt worden und stellen Ergänzungen zur Verfügung, die den Aufbau von Unicast- oder Multicast-Verbindungen unter Einschluß von Quell- und Senkekomponenten unterstützen. Die neueren Ansätze erlauben komplexere Flußgraphen, die insbesondere zwischengelagerte Komponenten sowie Verarbeitungsrechner enthalten, die zwischen Quell- und Senkenrechnern zwischengeschaltet sind. Folgende Tabelle faßt die wesentlichen Eigenschaften der unterstützten Multimedia-Anwendungen entsprechend der Anforderungen von 4.1 zusammen:

	Topologie	Komponenten-Verteilung	Format-beschränkungen	Strom-relationen
Connection Manager	1 Quelle / 1 Senke	1 Quellrechner 1 Senkenrechner	ja	für Skalier-komponente
QoS Broker	1 Q / n S n Q / 1 S	1 QR / n SR n QR / 1 SR	nicht unterstützt	n. u.
Graph Manager	beliebig	1 QR / n SR	n. u.	n. u.
Ripple Scheduling	Baum-topologie	beliebig	n. u.	n. u.

Tabelle 2 : Eigenschaften unterstützter Flußgraphen

Alle Verfahren koppeln die Reservierung an die Aushandlung der Dienstgüte auf der Anwendungsebene. Für einen Flußgraphen wird eine einzige Dienstgüte spezifiziert und abgeleitet.

Die Spezifikation unterschiedlicher Dienstgüten für heterogene Senken, wie im Kapitel 2 motiviert, ist bei keinem der Ansätze möglich. Alle Verfahren beziehen medienspezifische Parameter in die Aushandlung ein. Außer für den „Ripple-Scheduling“-Ansatz gilt dies auch in Bezug auf die Verzögerung. Dienstgüteanforderungen in Form von Wertebereichen werden durch alle Verfahren unterstützt.

	Heterogene Empfänger	Parameter	Werte-bereiche	Globale Reservierung	Lokale Koordination
Connection Manager	n. u.	ms-Parameter Verzögerung	ja	n. u.	n. u.
QoS-Broker	n. u.	ms-Parameter, Verzögerung	ja	ja	ja
Graph Manager	n. u.	ms-Parameter, Verzögerung	ja	ja	ja
Ripple Scheduling	n. u.	ms-Parameter	ja	ja	ja

Tabelle 3 : Bereitstellung von Dienstgüte

Mit Ausnahme des „Connection-Manager“-Konzepts führen alle Verfahren eine globale Reservierung von Endsystem- und Netzwerkressourcen durch. Eine Entkopplung von Reservierungen für einzelne Flußgraphenteile, wie sie durch variable Filter ermöglicht wird, wird nicht unterstützt. Bei allen Verfahren wird die Verfügbarkeit von Ressourcen berücksichtigt. Formatbeschränkungen sowie Stromrelationen für Komponenten werden dagegen nicht miteinbezogen.

Falls mehrere Komponenten eines Flußgraphen auf demselben Rechner plaziert sind und eine Ressourcenknappheit auf dem Rechner besteht, sollte die Ressourcenvergabe lokal koordiniert werden, um erzielbare Dienstgüten zu optimieren. Obwohl in einfachen Szenarien davon ausgegangen werden kann, daß lokal eine Überversorgung mit Ressourcen gegeben ist, gilt dies im allgemeinen nicht für Multimedia-Systeme, die zur Verarbeitung insbesondere zahlreiche Software-basierte Komponenten benutzen, die dieselben Ressourcen, vor allem CPU und Hauptspeicher, benötigen.

Der „Graph-Manager“-Ansatz garantiert durch den wiederholt ausgeführten Reservierungsversuch für eine Session, daß eine optimale Dienstgüte eingestellt wird. Allerdings bedingt diese Vorgehensweise einen entsprechend hohen Aufwand, insbesondere durch den wiederholten Aufbau von Netzwerkverbindungen. Andere Ansätze („QoS-Broker“, „Ripple Scheduling“) beschränken sich auf eine optimierte Ressourcenreservierung für Verarbeitungsfunktionen, die auf dem selben Rechner zu erbringen sind und vermeiden somit diesen Nachteil.

Jedes erwähnte Verfahren bietet Lösungen unter Betonung unterschiedlicher Aspekte. Diese ergeben sich aus begrenzt definierten Problemszenarien hinsichtlich Topologie und Verteilung der Flußgraphen, funktionalen Einschränkungen und der Art und Weise, wie Dienstgüte spezifiziert und erbracht werden kann. Verfahren, die Reservierungen auf Anwendungsebene unter Einschluß unabhängiger Dienstgüteanforderungen an den Senken, zwischengelagerter Komponenten mit fixen und variablen Stromrelationen sowie zwischengelagerter Verarbeitungsrechner sind in der Vergangenheit nicht untersucht worden. Die Entwicklung solcher Verfahren wird im folgenden Kapitel beschrieben.

5 Das NRP - Protokoll

Um die im vorigen Kapitel diskutierten Anforderungen an Dienstgüteaushandlung und Ressourcenreservierung erfüllen zu können, benötigen verteilte Multimedia-Anwendungen Reservierungsprotokolle, die auf der Anwendungsebene angesiedelt sind. In diesem Kapitel wird das „Negotiation and Resource Reservation Protocol“ (NRP) beschrieben, das für das CINEMA-System entwickelt wurde und entsprechende Mechanismen realisiert. In einer Einleitung im nächsten Abschnitt werden zunächst die wichtigsten Merkmale des Protokolls dargestellt. Im Abschnitt 5.3 wird dann das Protokoll in Bezug auf die Aushandlung medienspezifischer Parameter ausführlich beschrieben. Die Kopplung des Protokolls mit der Reservierung auf Transportebene wird im Abschnitt 5.4 erläutert, während im Abschnitt 5.5 die Berechnung der Verzögerung behandelt wird. In den Abschnitten 5.6 und 5.7 werden zwei Protokollerweiterungen untersucht, die eine optimierte Ressourcenvergabe zum Ziel haben. Abschließend wird im Abschnitt 5.8 die Implementierung des Protokolls beschrieben, bevor Meßergebnisse für den Aufbau von Sessions vorgestellt werden.

5.1 Eigenschaften

NRP ([DFR96], [FiDe96]) wurde für den Aufbau von Sessions in CINEMA entwickelt. NRP führt die Dienstgüteaushandlung und Ressourcenreservierung für einen Flußgraphen durch unter Berücksichtigung von Dienstgüteanforderungen, Stromrelationen und Formatbeschränkungen im Flußgraphen sowie der Ressourcenverfügbarkeit auf Endsystemen und im Netzwerk.

NRP ermöglicht die unabhängige Spezifikation der Dienstgüte an den Flußgraphensenken. Zugleich erlaubt es die Verwendung von variablen Filtern in Flußgraphen, wodurch eine entkoppelte Dienstgüteaushandlung für Flußgraphenteile ermöglicht wird. NRP realisiert eine Aushandlung sowohl für medienspezifische Parameter als auch in Bezug auf Verzögerung.

NRP ist für eine breite Klasse von Flußgraphentopologien anwendbar, die einem sogenannten Zweizonenmodell entsprechen (siehe Abb. 22). Solche Graphen bestehen aus zwei Teilen, einer Mischer-Zone, die Quellen umfaßt, die über zwischengelagerte Komponenten eine zentrale Komponente speisen, und einer Multicast-Zone, die Ströme über zwischengelagerte Kompo-

nenten an Senken verteilt. Die Mischer-Zone zeichnet sich dadurch aus, daß sie keine Multicast-Links zuläßt, während in der Multicast-Zone solche Links erlaubt sind, dafür aber keine Mischer vorgesehen sind (die Ströme nicht verteilen, sondern zusammenführen).

Das Modell deckt Multicast-Szenarien zur Verteilung von Information an (mehrere) Senken, Mischer-Szenarien zur Bereitstellung von Information aus mehreren Quellen für eine Senke sowie Konferenzszenarien zum Austausch von Information zwischen mehreren Quellen und Senken ab. NRP ist auf Flußgraphen unabhängig von der Verteilung der Komponenten auf Rechnern anwendbar. Dies ermöglicht die Berücksichtigung von Szenarien, in denen mehrere Komponenten auf einem Rechner gegeben sind, z. B. eine Quelle und eine Senke für einen Endbenutzer.

Das Design von NRP ist medienunabhängig, d. h. es unterstützt die Aushandlung für medienspezifische Parameter beliebiger Medien. Die gleichzeitige Aushandlung mehrerer Parameter wird unterstützt, sie erfordert jedoch die Verzahnung dieser Parameter in Bezug auf ihre Dienstgüteabstufung. Für einen Parameter, z. B. Bildgröße, ist die Stufung trivial durch die abnehmende Größe gegeben. Für zwei Parameter, z. B. Bildgröße und Rate ist eine Stufung von (Größe, Rate)-Wertepaaren erforderlich. Die Stufung wird von NRP als gegeben vorausgesetzt, wobei eine beliebige Stufung monoton abnehmender Wertetupel gewählt werden kann.

Das Design von NRP ist dezentral gestaltet. Es gibt keine Einheit, die einen globalen Zustand des Flußgraphen verwaltet. Entscheidungen zur Dienstgüteberechnung und Ressourcenreservierung werden verteilt vorgenommen. Jeder Rechner muß dabei lediglich die eigene Ressourcenverfügbarkeit beachten. Information, die zwischen benachbarten (d. h. durch den Flußgraphen verbundenen) Rechnern ausgetauscht wird, enthält lediglich Angaben zur unterstützbaren Dienstgüte und nicht ressourcenbezogene Angaben.

Eine dezentrale Lösung vermeidet die Abhängigkeit von einem zentralen Koordinator, der aufgrund des zentralen Abgleichs und der nötigen Kommunikation mit allen Teilen des Flußgraphen bei größeren Graphen eine zusätzliche Verzögerung bedingt [RDF97]. In [RDF97] wird ferner gezeigt, daß NRP nicht auf beliebige Flußgraphen erweiterbar ist, ohne eine zentrale Koordinationseinheit vorzusehen. Unter Verwendung einer solchen Einheit läßt sich eine solche

Erweiterung vornehmen, die in CINEMA in Form eines zweiten Protokolls XNRP realisiert wurde ([Fied98]).

NRP ist in mehreren Versionen entwickelt worden. Die Grundversion, die in Abschnitt 5.3 beschrieben wird, stellt einen konsistenten Session-Aufbau sicher und ist in der Lage, Dienstgüteanforderungen und funktionale Einschränkungen flußgraphenweit miteinander optimal abzugleichen. Für den Fall knapper Endsystemressourcen beinhaltet diese Version keine Vorkehrungen, die eine optimierte Zuteilung von Ressourcen ermöglichen. Sie ist daher vor allem für Szenarien geeignet, in denen gemeinsam benutzte Systemressourcen so dimensioniert sind, daß keine Ressourcenkonflikte bei ihrer Belegung auftreten.

Für den Fall knapper Ressourcen werden zwei Protokollerweiterungen beschrieben. In der ersten wird die Vermeidung von Ressourcenkonflikten angestrebt, indem der Session-Aufbau für eine berechnete minimale Dienstgüte versucht wird. In der zweiten Erweiterung wird NRP um einen Mechanismus ergänzt, der eine optimierte Koordination der Ressourcenvergabe auf den Endsystemen sicherstellt. Die erweiterten Protokolle werden in den Abschnitten 5.6 und 5.7 beschrieben.

In den folgenden Abschnitten wird das Design von NRP ausführlich dargestellt. Um die Beschreibung zu vereinfachen, werden ohne Beschränkung der Allgemeinheit einige Annahmen gemacht. So wird die Aushandlung lediglich für einen medienspezifischen Parameter (Bildgröße) beschrieben. Außerdem wird angenommen, daß alle Komponenten lediglich einen Ausgangs-Port besitzen und daß nur Mischer mehrere Eingangs-Ports haben können.[11] Schließlich wird der Begriff Parameter oder Dienstgüte in der Regel auf medienspezifische Parameter bzw. Dienstgüte bezogen, außer wenn es explizit anders erwähnt wird.

5.2 Protokollarchitektur

Als ein Reservierungsprotokoll der Anwendungsebene kann NRP in die Ressourcenmanagement-Struktur eingeordnet werden, die in Kapitel 3 vorgestellt wurde. Entsprechend dieser

[11] Allgemeinere Komponenten sind unterstützbar. Als Plausibilitätsargument sei erwähnt, daß komplexere Komponenten aus elementaren Komponenten zusammengesetzt werden können. Z. B. kann ein Mischer mit n Ausgängen aus einem Mischer mit einem Ausgang, einem Multicast-Link und n Filtern modelliert werden.

Struktur wird NRP durch verteilte Ressourcenmanagement-Agenten realisiert, die in dem Kontext von NRP als Protokollagenten (PA) bezeichnet werden. Konzeptionell ist jeder Komponente und jedem Link ein PA[12] zugeordnet, wobei die PA entsprechend der Flußgraphentopologie miteinander verbunden sind (Abb. 21).

Jeder PA nimmt zwei Aufgaben wahr. Zum einen interagiert er zur Ressourcenreservierung mit der zugeordneten Komponente bzw. dem zugeordneten Link. Zum anderen tauschen die PA untereinander Dienstgüteinformation über erfolgte Reservierungen der Komponenten bzw. Links aus. Durch die Propagierung dieser Information entlang des Flußgraphen werden die Koordinationsaufgaben zur Dienstgüteaushandlung und Ressourcenreservierung erbracht.

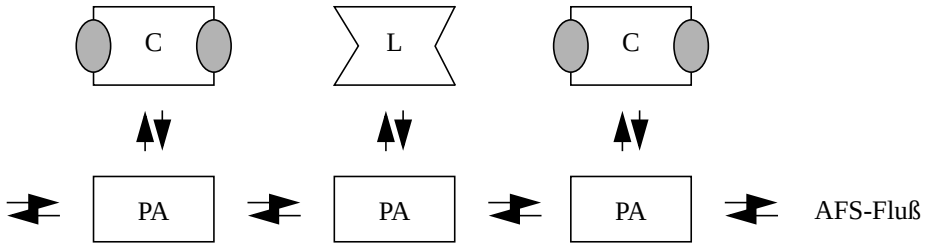


Abb. 21: *NRP Protokollagenten*

NRP ermöglicht für Komponenten und Links eine autonome Sicht bezüglich Reservierung. Diese haben lediglich eine Anzahl generischer Methoden zu unterstützen, die durch die jeweiligen PA aufgerufen werden. Eine Komponente oder ein Link interagiert mit seinem PA, nicht jedoch mit anderen Komponenten oder Links. Zur Unterstützung der verschiedenen Protokollphasen (siehe nächste Abschnitte) werden verschiedene Methoden (*Reserve*, *Relax*, etc.) bereitgestellt.

Die PA tauschen untereinander Informationen in sogenannten *Application FlowSpecs* (AFS) aus. Ein AFS enthält Information über medienspezifische Parameter sowie über die zu berechnende „Ende-zu-Ende“-Verzögerung. Zu jedem medienspezifischen Parameter ist ein Wertebereich enthalten, der die möglichen Werte anzeigt. Die PA benutzen diese Angaben für Reservie-

[12] Genauer wird ein PA für einen entfernten Link oder für einen lokalen Multicast-Link benötigt. Lokale Unicast-Links verändern keine AFS und belegen keine Ressourcen (siehe 3.3.2).

rungsaufrufe an Komponenten oder Links, wobei Wertebereiche (z. B. wegen Formatbeschränkungen) durch diese eingeschränkt werden können. PA der Senken berücksichtigen zusätzlich die Dienstgüteanforderungen des Klienten (A-QoS).

NRP ist für die Propagierung von AFS zwischen verbundenen Komponenten und Links zuständig. Da NRP Dienstgüteanforderungen in den AFS miterfaßt, kann NRP somit als der Aushandlungs-“Kitt“ zwischen Klienten, Komponenten und Links angesehen werden, der die topologieorientierte AFS-Propagierung in sich verbirgt.

5.3 Protokoll-Design

5.3.1 Übersicht

NRP wird in drei Phasen ausgeführt (Abb. 22). In jeder Phase werden AFS entlang des Flußgraphen zwischen Quellen und Senken propagiert, entweder von den Quellen zu den Senken (flußabwärts) oder von den Senken zu den Quellen (flußaufwärts). Während jeder Phase werden Komponenten und Links mit AFS aufgerufen. Komponenten und Links interpretieren diese Information, manipulieren sie und liefern sie zurück an NRP zur weiteren Propagierung.

Wenn AFS flußabwärts propagiert werden, erhalten Komponenten und Links ein AFS für jeden Eingangs-Port und liefern als Ergebnis ein AFS für den Ausgangs-Port[13]. Während der Propagierung flußaufwärts wird von NRP für den Ausgangs-Port ein AFS geliefert und als Ergebnis ein AFS für jeden Eingangs-Port erhalten.

In der *ersten Phase* (Reserve) von NRP wird Information über die Fähigkeiten von Komponenten und Links zur Dienstgüteunterstützung von den Quellen an die Senken propagiert. Die Information erfaßt funktionale Einschränkungen der Komponenten, wobei NRP durch eine gleichzeitige Reservierung sicherstellt, daß diese Information auf gesicherten Ressourcen beruht. Die Phase beginnt an den Quellen durch eine Generierung initialer AFS, die die Formatbeschränkungen der Quellen erfaßt. NRP propagiert die AFS zu flußabwärts liegenden Komponenten und Links, wobei jedes dieser Elemente seine (funktionalen oder ressourcenbedingten)

[13] Einem Link sind die Ports zugeordnet, die durch ihn verbunden werden.

Einschränkungen in den AFS erfassen kann. Wenn ein AFS bei einer Senke anlangt, wird es zusammen mit der spezifizierten Dienstgütevorgabe (A-QoS) an die Senke weitergereicht.

Die *zweite Phase* (Request) dient dem Abgleich der A-QoS der Senken mit den Parameterwertebereichen an den Komponenten-Ports. Als Ergebnis werden für die Quellen die Werte ermittelt, aus denen in der dritten Phase die Werte für alle anderen Ports eindeutig folgen und aufgrund derer die endgültige Reservierung vorgenommen wird. AFS werden von den Senken zu den Quellen propagiert, wobei Komponenten und Multicast-Links nach einem Aufruf Berechnungen intern durchführen und erhaltene AFS verändern. Die Phase endet, wenn die AFS die Quellen erreicht haben.

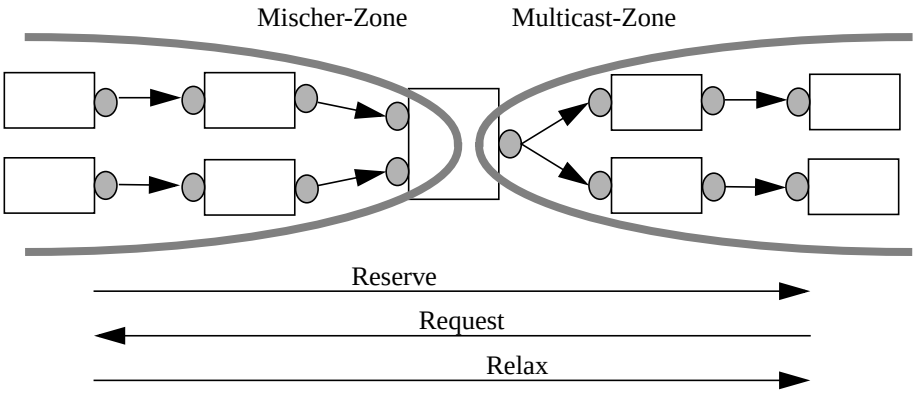


Abb. 22: *Protokollphasen*

Während der *dritten Phase* (Relax) werden die endgültigen Parameterwerte für die Komponenten-Ports eingestellt. Komponenten und Links können mit diesen Werten ihre endgültige Ressourcenreservierung vornehmen, indem sie eine eventuelle Überreservierung der ersten Phase relaxieren. Komponenten und Links werden nacheinander von den Quellen zu den Senken durchlaufen. Wenn die AFS die Senken erreicht haben, ist die Session aufgebaut.

Ein AFS, das durch NRP zur Propagierung benutzt wird, hat die Form:

```
AFS(    Generische Parameter
        Medienspezifische Parameter
                Param1: Wertebereich (val_1, val_2, ..., val_q)
                ...
                Param n: Wertebereich (val_1, ... )
).
```

Der generische Teil des AFS umfaßt Parameter zur Berechnung der Verzögerung. Ihre Behandlung wird in Abschnitt 5.5 beschrieben. Der zweite Teil enthält die medienspezifischen Parameter. Zu jedem Parameter ist ein Wertebereich sowie ein sogenannter *Downstream Request Value* (DRV) angegeben. Der Wertebereich enthält alle Werte, die der Parameter am Ort eines AFS im Flußgraphen (d. h. an einem Komponenten-Port) annehmen kann. Der DRV-Wert hält zusätzlich den Parameterwert fest, der dem maximalen Wert entspricht, der für flußabwärts liegende Senken gefordert wird. Im unten dargestellten Flußgraphen sind Beispiel-AFS abgebildet, wie sie in der zweiten Phase von NRP auftreten können. Ihre Form ergibt sich aus der Überlegung, welche Information am Port der Quelle benötigt wird, um die optimale Dienstgüte für die Senken einzustellen.

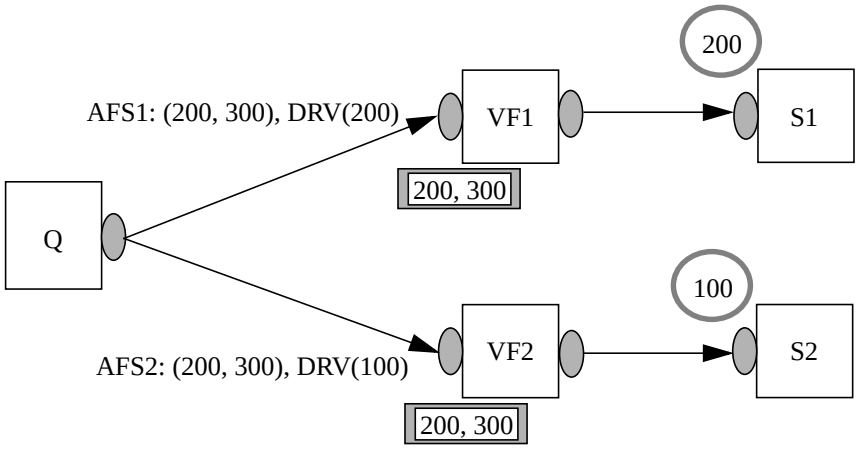


Abb. 23: *Verwendung von Wertebereichen und Downstream Request Values*

Es sei angenommen, daß an den Senken die Werte 200 für S1 und 100 für S2 gefordert werden. Am Eingangs-Port der variablen Filter VF1 und VF2 sei der Bereich (200, 300) unterstützbar. Wäre für den Quellen-Port lediglich diese Information bekannt, so wüßte dieser nicht, welcher Wert einzustellen ist. Erst durch die DRV-Werte für stromabwärts liegende Senken, kann erkannt werden, daß ein Wert über 200 nicht erforderlich ist. Auf der anderen Seite muß jeder variable Filter auch den vollen unterstützbaren Bereich an seinem Eingans-Port melden, nicht nur den DRV-Wert. Sei z. B. angenommen, daß an S2 der Wert 300 gefordert wird. Die Quelle kann diesen Wert nur einstellen, falls VF1 neben dem eigenen DRV-Wert von 200 anzeigt, daß er den Wert 300 an seinem Eingang unterstützen kann. Nur aufgrund dieser Anzeige kann für

die Quelle angenommen werden, daß beide flußabwärts nachfolgenden Komponenten diesen Wert unterstützen können.

Die Verwendung der DRV-Werte ist eine direkte Folge der Verwendung variabler Filter. Sie unterstützen die Dienstgüteaushandlung für heterogene Senken, ermöglichen aber auch, nicht übereinstimmende Formatbeschränkungen zu überbrücken. Wäre im obigen Beispiel die Quelle nur in der Lage einen Wert von 200 zu liefern und die Senke S1 nur in der Lage einen Wert von 100 zu empfangen, könnte durch Benutzung von DRV-Werten die nötige Konvertierung am Filter eingestellt werden.

5.3.2 Beispiel

Der Ablauf von NRP wird anhand des Flußgraphen in Abb. 24 erläutert. Dabei wird angenommen, daß der Parameter Bildgröße ausgehandelt werden soll, der über einen eindimensionalen Wert definiert wird.[14] Es wird angenommen, daß die Kamerakomponenten Cam1 und Cam2 Formatbeschränkungen unterliegen. Cam1 kann lediglich Bildgrößen im Bereich von (400..480) erzeugen, während Cam2 einen Wert im Bereich (320..560) liefern kann (d. h. die Werte 320, 400, 480, 560). Ferner soll der Mischer an seinen Eingängen die gleiche Bildgröße erfordern. An den Senken seien die Dienstgüteanforderungen (240..480) für Dsp1 bzw. (240..560) für Dsp2 spezifiziert. Vereinfachend wird die Betrachtung von Links ausgeklammert, außer wenn sie unerläßlich ist.

Die erste Phase beginnt mit dem Aufruf von Cam1 und Cam2 zur Ressourcenreservierung. Beide sollen in der Lage sein, die jeweilige Formatbeschränkung maximal zu unterstützen, daher liefern sie die Bereiche (400..480) bzw. (320..560) an NRP zurück. Diese werden in zwei AFS an den Mischer zur Reservierung propagiert. Dieser bildet zuerst die Schnittmenge (400..480) zwischen den Eingangs-Ports und leitet (aufgrund einer Gleichheits-Stromrelation) (400..480) auch für den Ausgangs-Port ab. Der Mischer soll maximal reservieren können, so daß er (400..480) an NRP zurückliefert. Der Bereich wird in zwei AFS an VF1 bzw. VF2 geliefert. Da diese variable Filter sind, sind sie in der Lage jeden Eingangswert auf einen niedrigeren Ausgangswert zu reduzieren. VF1 leitet daher den Bereich (80..480) für den Ausgans-Port ab

[14] Die Bildgröße soll alle Werte annehmen können, die Vielfache von 80 darstellen.

und versucht maximal zu reservieren.[15] Da dies gelingt, liefert VF1 (80..480) an NRP zurück. Für VF2 wird der gleiche Ablauf unterstellt.

Es sei angenommen, daß der Link zu Dsp1 nur eine Übertragung von Bildgrößen bis 320 zuläßt. Der Link liefert daher den reduzierten Bereich (80..320) an NRP zurück, der in einem AFS an Dsp1 propagiert wird. Dsp1 wird außerdem die A-QoS(240..480) übergeben. Dsp1 bildet die Schnittmenge (240..320) und versucht maximal zu reservieren. Dies sei möglich. Für Dsp2 sei angenommen, daß keine Reduktion im Link von VF2 nötig ist. Dsp2 wird mit dem Bereich (80..480) sowie der A-QoS(240..560) zur Reservierung aufgerufen. Dsp2 sei in der Lage, für den Schnittbereich (240..480) maximal zu reservieren.

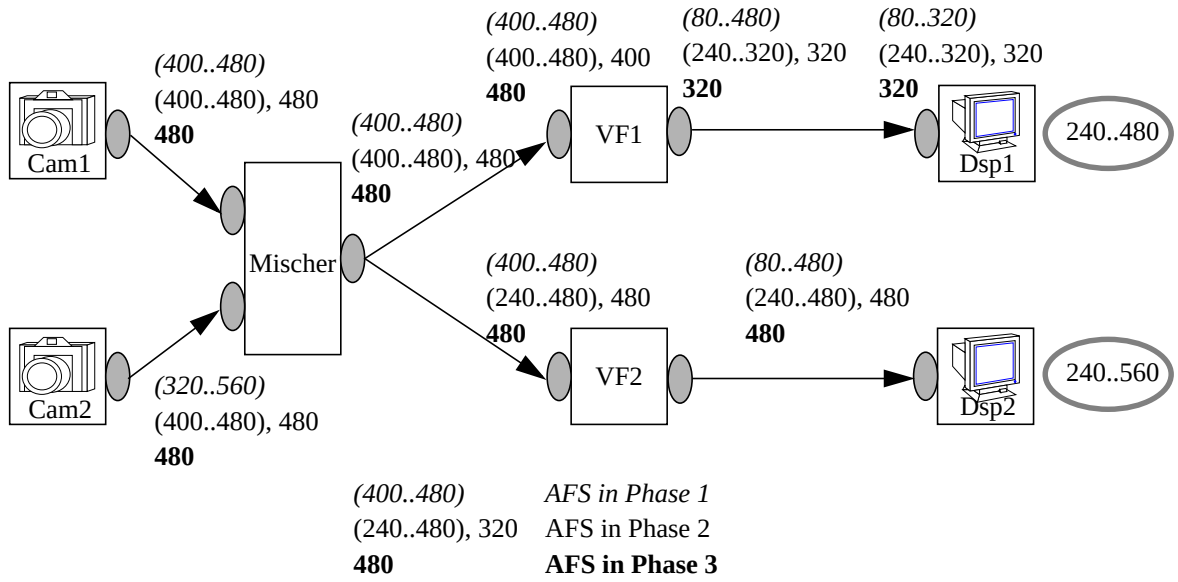


Abb. 24: *NRP-Aushandlungsbeispiel*

Die zweite Phase propagiert die AFS zu den Quellen. In dieser Phase werden neben Wertebereichen auch die DRV-Werte benötigt. NRP erhält von Dsp1 den Bereich (240..320) und einen DRV von 320. Diese Information wird an VF1 propagiert, welches ein neues AFS für seinen Eingangs-Port berechnet: (400..480), DRV(400). Der Bereich reflektiert die am Eingangs-Port

[15] Maximale Reservierung bedeutet für einen variablen Filter, daß er die ressourcenaufwendigste Konversion zwischen Eingangs- und Ausgangs-Port betrachten muß (siehe 5.3.4).

empfangbare Bildgröße (unter Berücksichtigung des in der ersten Phase ermittelten Wertebereichs), während der DRV-Wert dem flußabwärts maximal erwarteten Wert entspricht. Der Ablauf für den Zweig VF2-Dsp2 ist analog. Von Dsp2 erhält NRP (240..480), DRV(480). Dieses wird in einem AFS an VF2 gereicht, welches das neue AFS für seinen Eingangs-Port berechnet: (400..480), DRV(480). Sowohl VF1 als auch VF2 speichern die DRV-Werte, die sie an ihren Aufgangs-Ports erhalten erhaben: 320 für VF1 und 480 für VF2.

Die zwei AFS von VF1 und VF2 werden an den Multicast-Link gereicht, der sie in ein einziges AFS kombiniert. Im vorliegenden Fall ergibt dies (400..480), DRV(480), da der Bereich die möglichen Werte darstellt und 480 der maximalen Anforderung flußabwärts entspricht. Das AFS wird über NRP an den Mischer weitergereicht, der für seine zwei Eingangs-Ports zwei AFS mit unverändertem Inhalt an NRP liefert. Ein AFS wird an Cam1 weiterpropagiert, das andere an Cam2.

In der dritten Phase werden die ermittelten DRV-Werte für die endgültige Reservierung benutzt. Cam1 wird mit DRV(480) aufgerufen, was keine Reservierungsrelaxierung erfordert. Cam1 antwortet mit dem unveränderten DRV(480) an NRP, der an den Mischer propagiert wird. Cam2 wird mit DRV(480) aufgerufen, was eine entsprechende Reservierungsrelaxierung zur Folge hat. An NRP wird DRV(480) gereicht, welches den Wert an den Mischer propagiert. Der Mischer wird mit den zwei DRV-Werten aufgerufen. Der Mischer berechnet DRV(480) für den Ausgangs-Port, erkennt eine Relaxierung als nicht nötig und übergibt DRV(480) an NRP. NRP propagiert dies an VF1 und VF2.

VF1 erkennt an diesem Wert und dem in Phase 2 gespeicherten Ausgangs-Port-Wert von 320, daß es zur Laufzeit eine Konversion von 480 auf 320 zu leisten hat und fixiert dies intern. Außerdem relaxiert es die Ressourcenbelegung entsprechend. VF1 liefert DRV(320) an NRP, welches dies zu Dsp1 propagiert. Dsp1 stellt fest, daß keine Relaxierung nötig ist und meldet den Abschluß an NRP. Der Ablauf für den Zweig VF2-Dsp2 ist analog. VF2 stellt fest, daß es keine Konversion vornehmen muß (am Ausgang wird der in Phase 2 gespeicherte Wert von 480 erwartet), und gibt alle reservierten Ressourcen frei. Der Wert DRV(480) wird über NRP an Dsp2 propagiert, wo ebenfalls keine Relaxierung nötig ist. An dieser Stelle ist die Session für den Flußgraphen aufgebaut. Den Abschluß der dritten Phase teilen die Senken-PA dem Klienten mit.

5.3.3 Protokollspezifikation

NRP ermöglicht die Ressourcenreservierung für eine Anwendung durch die Propagierung von AFS zwischen den PA eines Flußgraphen. Von den PA aus ruft es Komponenten und Links auf, an die es AFS übergibt und durch die es neue AFS erhält. Von jeder Komponente und von jedem Link wird hierzu eine generische Schnittstelle erwartet, die drei Methoden beinhaltet:[16]

Reserve (aufgerufen während der ersten Protokollphase)

Request (aufgerufen während der zweiten Protokollphase)

Relax (aufgerufen während der dritten Protokollphase) .

Komponenten und Links führen bei einem entsprechendem Methodenaufruf bestimmte Funktionen durch, die für medienspezifische Parameter im folgenden definiert werden. Die Behandlung generischer Parameter wird in Abschnitt 5.5 erläutert.

Phase 1 (Reserve)

In dieser Phase wird eine Komponente über ihre *Reserve*-Methode aufgerufen. Sie erhält hierbei eine Menge von AFS, die für jeden ihrer Eingangs-Ports ein Eingangs-AFS (AFS_E) enthält. Als Ergebnis des Aufrufs wird für den Ausgangs-Port ein Ausgangs-AFS (AFS_A) geliefert, welches durch NRP flußabwärts propagiert wird. Zur Berechnung des AFS_A führen Komponenten folgende Schritte durch:

Eine *Quelle* wird mit einem leeren AFS aufgerufen, da sie erst ein initiales AFS_A als Ergebnis liefern muß. AFS_A enthält diejenigen Werte der Formatbeschränkung am Ausgangs-Port der Quelle, für die eine Reservierung von Ressourcen möglich ist:

(1) $WB_A = \{ a \in FB_A : \text{Reservierung für a ist möglich} \}$

Entsprechend der Diskussion in 5.3.4 genügt es hierzu, den höchsten Wert aus FB_A zu ermitteln, für den eine Reservierung möglich ist, da alle niedrigeren Werte einen niedrigeren (oder gleichbleibenden) Ressourcenbedarf bedingen.

Ein *fixer Filter* bildet zuerst die Schnittmenge aus dem erhaltenen Wertebereich WB_E und der am Port definierten Formatbeschränkung FB_E. Anschließend wird mit Hilfe der fixen Filterre-

[16] Eine weitere Methode *Free* wird benötigt, um die Beendigung einer Session anzuzeigen.

lation (siehe 2.3) der am Ausgangs-Port in Frage kommende Wertebereich ermittelt, aus dem alle Werte ausgeschlossen werden, die aufgrund der Ressourcenverfügbarkeit nicht unterstützt werden können. Formal ausgedrückt, wird folgendes berechnet:

(2) $WB'_E = WB_E \cap FB_E$

(3) $WB_A = WB'_E * F$

(4) $WB''_E = \{ e \in WB'_E$: Reservierung für e ist möglich $\}$

(5) $WB'_A = \{ a \in WB_A$: Reservierung für a ist möglich $\}$.[17]

Entsprechend der Diskussion von 5.3.4 genügt es für den letzten Schritt, die Werte aus WB_A (und implizit WB'_E) in absteigender Reihenfolge zu betrachten, wobei Werte aus WB_A entfernt werden, für die nicht genügend Ressourcen vorhanden sind. Wird der erste Wert erreicht, für den eine Ressourcenreservierung möglich ist, so gilt dies auch für alle niedrigeren Werte aus WB_A. Für den ermittelten besten Wert werden die benötigten Ressourcen reserviert. Der Aufruf des Filters wird abgeschlossen, indem ein AFS_A als Ergebnis zurückgeliefert wird, das den unterstützen Bereich WB'_A enthält.

Ein *Mischer* erhält ein AFS_E für jeden seiner k Eingangs-Ports. Für jeden Port wird zunächst der empfangene Wertebereich (WB_{Ei}, i=1..k) mit der entsprechenden Formatbeschränkung (FB_{Ei}) in Einklang gebracht. Anschließend werden die Mischrelationen (siehe 2.3) zwischen den Eingangs-Ports verwendet, um die Schnittmenge der Parameterwerte zu ermitteln, die für den gemeinsamen Bezugs-Port (hier als Port 1 angenommen) und für die restlichen Eingangs-Ports funktional (d.h. aufgrund der Formatbeschränkungen und Stromrelationen) möglich sind. Der Wertebereich für den Ausgangs-Port wird, analog zu oben, über die Filterrelation des Mischers ermittelt. Abschließend werden aus allen Wertebereichen die Werte ausgeschlossen, für die die Ressourcenverfügbarkeit nicht ausreicht.

(6) $WB'_{Ei} = WB_{Ei} \cap FB_{Ei}$, für i = 1..k

(7) $WB''_{E1} = WB'_{E1} \cap (WB'_{E2} / f_{2,1}) \cap \ldots \cap (WB'_{Ek} / f_{k,1})$

$WB''_{Ei} = WB''_{E1} * f_{i,1}$, für i = 2..k

(8) $WB_A = WB''_{E1} * F$

[17] Eine Division bzw. Multiplikation eines Wertebereichs mit einem Faktor soll eine entsprechende Disvision bzw. Multiplikation jedes Elements des Wertebereichs bedeuten.

(9) $WB''''_{Ei} = \{ e \in WB''_{Ei} : \text{Reservierung für e ist möglich} \}$

(10) $WB'_A = \{ a \in WB_A : \text{Reservierung für a ist möglich} \}$.

Wie für einen fixen Filter werden in (10) die Werte von WB_A (und implizit von WB_{Ei}) in absteigender Reihenfolge durchlaufen (siehe 5.3.4), bis ein aufgrund der Ressourcenverfügbarkeit unterstützbarer Wert erreicht ist, für den Ressourcen reserviert werden. Alle Werte darüber werden aus WB_A ausgeschlossen und der so ermittelte Bereich WB'_A wird als Ergebnis im AFS_A an NRP zurückgeliefert.

Ein *variabler Filter* erhält ein AFS_E für seinen Eingangs-Port. Der darin enthaltene Wertebereich WB_E wird zunächst mit der dazugehörigen Formatbeschränkung abgeglichen. Anschließend wird der funktional unterstützte Wertebereich am Ausgangs-Port ermittelt, indem aus der entsprechenden Formatbeschränkung alle Werte entfernt werden, die der variablen Filterrelation nicht genügen (siehe 2.3). Aus den Wertebereichen für den Eingangs- bzw. Ausgangs-Port werden ferner die Werte ausgeschlossen, für die eine Ressourcenreservierung nicht möglich ist.

(11) $WB'_E = WB_E \cap FB_E$

(12) $WB_A = \{ e \in FB_A : e <= \max(WB'_E) \}$

(13) $WB''_E = \{ e \in WB'_E : \text{Reservierung für e ist möglich} \}$

(14) $WB'_A = \{ a \in WB_A : a <= \max \{WB''_E\} \}$.

Zur Überprüfung der Ressourcenverfügbarkeit gemäß (13) werden alle Werte, die am Eingangs-Port aufgrund von WB'_E möglich sind, in absteigender Reihenfolge betrachtet. Zu jedem Wert ew wird der maximale Ressourcenbedarf (MaxRB) ermittelt, der durch eine beliebige Einstellung eines Werts aw aus WB_A (aw <= ew) bedingt werden kann (siehe 5.3.4 zur Motivation):

(15) $MaxRB(ew) = \max\{ RB(ew, aw): aw <= ew \text{ und } aw \in WB_A \}$.

Die Ressourcenreservierung wird zunächst für den höchsten Wert ew versucht. Ist die Reservierung nicht möglich, so wird ew aus WB'_E gestrichen. Um die variable Filterrelation nicht zu verletzten, werden gemäß (14) ebenso alle Werte aw aus WB'_A entfernt, die das neue Maximum der Werte in WB''_E übersteigen.

Anschließend wird für den neuen Maximalwert aus WB'_E eine Reservierung versucht. Der hierbei gemäß (15) zu verwendende Ressourcenbedarf kann aufgrund der Berechnungsweise

gegenüber dem ersten Versuch nur gleichbleiben oder abnehmen (siehe 5.3.4). Wiederholte Ressourcenreservierungen werden durchgeführt, bis eine von ihnen erfolgreich ist.

Der verbliebene Wertebereich WB''_E wird durch einen variablen Filter (als WB_{P1}) zur späteren Verwendung im Kontext des Eingangs-Ports gespeichert. Der am Ausgangs-Port unterstützte Wertebereich WB'_A wird im AFS_A als Ergebnis des *Reserve*-Aufrufs an NRP geliefert.

Eine *Senke* erhält neben einem AFS_E auch die Dienstgüteanforderung (A-QoS), die an dem Port spezifiziert wurde (siehe 2.5). Um den unterstützbaren Wertebereich an ihrem Eingangs-Port zu ermitteln, bildet sie zuerst die Schnittmenge der Wertebereiche aus dem AFS_E, der Formatbeschränkung für ihren Eingangs-Port und der A-QoS. Sie liefert als Ergebnis ein AFS für ihren Eingangs-Port zurück. Diese umfaßt den Wertebereich, für den (analog zu (1)) reserviert werden konnte und einen DRV-Wert, der auf das Maximum dieses Wertebereichs gesetzt ist:

(16) $WB_{Senke} = \{ e \in (WB_E \cap WB_{A-QoS} \cap FB_E) : \text{Reservierung für e ist möglich} \}$

(17) $DRV_{Senke} = \max \{WB_{Senke}\}$.

Die Arbeitsweise von Links wird ausführlich in Abschnitt 5.4 dargestellt. Aus Sicht von NRP kann ein Link vereinfacht als eine Komponente mit einem Eingangs- und einem order mehreren Ausgangs-Ports aufgefaßt werden, wobei der Link diese Ports mit den Komponenten gemeinsam hat, die er miteinander verbindet. Entsprechend der intern erbrachten Kommunikationsfunktion werden die Ports eines Links im folgenden als Sende- bzw. Empfangs-Ports unterschieden.

Wird ein *entfernter Link* mit einem AFS für seinen Sende-Port aufgerufen, so werden alle Werte aus dem entsprechenden Wertebereich ausgeschlossen, für die die intern aufgebaute Transport-Verbindung nicht ausreichend ist. Als Ergebnis wird für den Empfangs-Port des Links ein AFS erzeugt, das diesen eingeschränkten Wertebereich enthält. Handelt es sich um einen Multicast-Link, so wird ein AFS für jeden Empfangs-Port generiert. Lokale Links sind an der ersten Protokollphase nicht beteiligt, da sie keine Ressourcen reservieren (siehe 3.3.3).

Phase 2 (Request)

In dieser Phase wird eine Komponente über ihre *Request*-Methode aufgerufen. Sie erhält hierbei ein AFS_A für ihren Ausgangs-Port und berechnet als Ergebnis des Aufrufs eine Menge von

AFS. Diese Menge enthält für jeden Eingangs-Port ein Eingangs-AFS (AFS_E) und wird durch NRP flußaufwärts weiterpropagiert. Zur Berechnung der AFS_E führen die Komponenten folgende Funktionen aus:

Ein *fixer Filter* benutzt seine Filterrelation, um den Wertebereich WB_E bzw. den DRV-Wert DRV_E zu bestimmen:

(18) $WB_E = WB_A / F$

(19) $DRV_E = DRV_A / F$.

Ein *Mischer* verfährt analog, um die Einstellungen für seinen Eingangs-Port zu ermitteln, der über die Filterrelation mit dem Ausgangs-Port gekoppelt ist (siehe 2.3). Anschließend werden aus diesen Angaben die Wertebereiche bzw. DRV-Werte für die restlichen Eingangs-Ports ermittelt, indem die entsprechenden Mischrelationen eingesetzt werden:

(20) $WB_{E1} = WB_A / F$

(21) $DRV_{E1} = DRV_A / F$

(22) $WB_{Ei} = WB_{E1} * f_{i,1}$ für i = 2, 3, .., k

(23) $DRV_{Ei} = DRV_{E1} * f_{i,1}$ für i = 2, 3, .., k .

Ein *variabler Filter* muß, analog zu den anderen Komponentenarten, für den Eingangs-Port einen Wertebereich ermitteln, der dem erhaltenen Wertebereich WB_A am Ausgangs-Port entspricht. Um der variablen Filterrelation zu genügen, werden aus dem in der ersten Phase abgespeicherten Wertebereich WB_{P1} alle Werte entfernt, die unterhalb des Minimums von WB_A liegen:

(24) $WB_E = \{ e \in WB_{P1} : e >= \min(WB_A) \}$.

Wie in 5.3.1 beschrieben wurde, entspricht der DRV-Wert der flußabwärts maximal gewünschten (medienspezifischen) Dienstgüte. Diese Semantik wird durch einen variablen Filter durch folgende Berechnung des DRV_E beibehalten:

(25) $DRV_E = \min\{ e \in WB_E : e >= DRV_A\}$.

Die Berechnung gewährleistet, daß der DRV-Wert sich nicht verändert, falls er im gemäß (21) berechneten Wertebereich WB_E liegt. Ist dies nicht der Fall, so wird der DRV-Wert so wenig wie möglich erhöht, indem der nächsthöhere Wert eingestellt wird, der in WB_E liegt. Der DRV-Wert

ist somit der minimal mögliche, der eine flußabwärts geforderte maximale Dienstgüte weiterhin ermöglicht.

Ein variabler Filter speichert den erhaltenen DRV-Wert DRV_A bzw. den erhaltenen Wertebereich WB_A (als DRV_{P2} bzw. WB_{P2}) zur späteren Verwendung ab.

Ein *Multicast-Link* wird mit einem AFS_{EP} für jeden seiner (k) Empfangs-Ports aufgerufen und liefert ein AFS_{SP} für den Sende-Port als Ergebnis zurück. Der Wertebereich WB_{SP} sowie der DRV-Wert DRV_{SP} werden wie folgt berechnet:

(26) $WB_{SP} = WB_{EP1} \cap \ldots \cap WB_{EPk}$

(27) $DRV_{SP} = \min\{e \in WB_{SP} : e >= DRV\_MAX_{EP}\}$ für $DRV\_MAX_{EP} <= \max\{WB_{SP}\}$

$\max\{WB_{SP}\}$ für $DRV\_MAX_{EP} > \max\{WB_{SP}\}$,

wobei:

$DRV\_MAX_{EP} = \max\{DRV_{EP1}, \ldots, DRV_{EPk}\}$.

Obige Berechnung stellt sicher, daß ein Wertebereich WB_{SP} flußaufwärts weiterpropagiert wird, der an allen Empfangs-Ports unterstützt werden kann. Die Berechnung des DRV-Werts stellt ferner sicher, daß dieser immer innerhalb des gültigen Wertebereichs liegt. Sofern möglich, wird er auf das Maximum der DRV-Werte der Empfangs-Ports gesetzt. Falls dieser Wert nicht in WB_{SP} liegt, dafür aber ein höherer, so wird der nächsthöhere Wert aus WB_{SP} verwendet, um flußabwärts jeden geforderten DRV-Wert zu unterstützen. Ist dies nicht möglich, da das Maximum der erhaltenen DRV-Werte oberhalb von WB_{SP} liegt, so wird dessen Maximum als DRV_{SP} bestimmt. Die flußabwärts unterstützbaren DRV-Werte werden in diesem Fall zwar reduziert, jedoch nur soweit wie (durch WB_{SP}) unbedingt erforderlich.

Ein *Unicast-Link* verändert ein erhaltenes AFS nicht, sondern liefert es unverändert an NRP zurück. Der Aufruf einer *Quelle* in der zweiten Phase markiert zugleich den Beginn der Relaxierungsphase von NRP. Entsprechend führt sie bereits die Relaxierungsfunktion der dritten Phase aus.

Phase 3 (Relax)

Die dritte Phase wird zur Relaxierung der Ressourcenreservierung verwendet. Hierzu wird eine Komponente über ihre *Relax*-Methode aufgerufen. Analog zur ersten Phase erhält sie ein AFS_E

für jeden ihrer Eingangs-Ports und berechnet ein AFS_A für ihren Ausgangs-Port, welches von NRP flußabwärts propagiert wird. In dieser Phase sind lediglich die DRV-Einträge der AFS von Bedeutung, so daß im folgenden nur ihre Behandlung beschrieben wird.

Wenn eine *Quelle* zur Relaxierung aufgerufen wird, stellt sie den erhaltenen DRV-Wert an ihrem Ausgangs-Port ein. Liegt dieser unterhalb des in der ersten Phase eingestellten Wertes, so nimmt sie eine entsprechende Relaxierung der Reservierung vor. Als Ergebnis des Aufrufs liefert es ein AFS_A zurück, das den DRV-Wert unverändert enthält.

Ein *fixer Filter* stellt an ihrem Eingangs-Port den erhaltenen DRV_E ein und berechnet für seinen Ausgangs-Port einen DRV-Wert, der der fixen Filterrelation genügt:

(28) $DRV_A = DRV_E * F$.

Ist dieser Wert niedriger als derjenige, für den in der ersten Phase Ressourcen reserviert wurden, so wird die Reservierung entsprechend relaxiert.

Ein *Mischer* stellt an seinen Eingangs-Ports die DRV-Werte ein, die er in den AFS_{Ei} (i = 1..k) erhält. Für den Ausgangs-Port berechnet er einen DRV-Wert mit Hilfe seiner Filterrelation:

(29) $DRV_A = DRV_{E1} * F$.

Analog zu oben wird anschließend eine Relaxierung der Reservierung durchgeführt, falls die eingestellten DRV-Werte dies ermöglichen.

Ein *variabler Filter* stellt an seinem Eingangs-Port den erhaltenen Wert DRV_E ein und berechnet den DRV-Wert für seinen Ausgangs-Port:

(30) $DRV_A = DRV_{P2}$, falls $DRV_E >= DRV_{P2}$

$\max \{ a \in WB_{P2} : a <= DRV_E \}$, sonst .

Diese Berechnung stellt sicher, daß der in der Phase 2 ermittelte DRV_{P2}-Wert für den Ausgangs-Port beibehalten wird, falls dies aufgrund der Filterrelation möglich ist. Ist dies nicht der Fall, so wird der DRV-Wert soweit abgesenkt, daß er, wie gefordert, den DRV-Wert am Eingangs-Port nicht übersteigt.

Sind die DRV-Werte an den Ports des Filters eingestellt, so sind Ressourcen nur für den entsprechenden Ressourcenbedarf erforderlich (siehe 5.3.4). Liegt dieser Ressourcenbedarf unterhalb

desjenigen, für den in der ersten Phase reserviert wurde, so kann die Reservierung entsprechend relaxiert werden.

Ein *entfernter Link* reduziert seine interne Reservierung entsprechend des erhaltenen DRV_E - Werts, falls dieser unterhalb des in der Phase 1 eingestellten Wertes liegt. Für den Ausgangs-Port wird der DRV-Wert des Eingangs unverändert übernommen. Bei einem Multicast-Link wird für jeden Empfangs-Port derselbe DRV-Wert als Ergebnis an NRP zurückgeliefert.

5.3.4 Maximale Reservierung

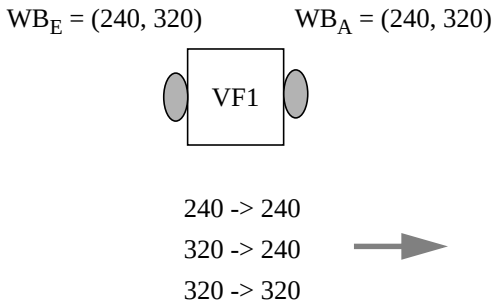
Komponenten und Links belegen Ressourcen, wenn sie im Verlauf der *Reserve*-Phase von NRP aufgerufen werden. Die hierbei übergebenen AFS beinhalten Wertebereiche, aufgrund derer eine Komponente bzw. ein Link den maximal erforderlichen Ressourcenbedarf ableitet. Hierbei muß für jede Ressource die höchste Anforderung berücksichtigt werden, die bei einer zulässigen Einstellung von Werten aus den erhaltenen Wertebereichen auftreten kann.

Für eine Komponente ohne variable Stromrelation (Quellen, Senken, fixe Filter, Mischer) legt die Einstellung eines Wertes an einem Port die Werte an den restlichen Ports ebenfalls fest. Sind an diesem Port Werte aus einem Wertebereich WB möglich, so ist der maximale Ressourcenbedarf wie folgt definiert:

(R1) $\quad MaxRB = \max\{\ RB(w):\ w \in WB\ \}$.[18]

Für einen Link ist eine analoge Beziehung definierbar. Für einen variablen Filter dagegen können die Parameterwerte am Eingangs- und am Ausgangs-Port teilweise unabhängig eingestellt werden. Ist an den Ports jeweils ein Wertebereich gegeben, so sind funktional alle Wertkombinationen möglich, die der variablen Stromrelation nicht widersprechen. Für das in Abb. 25 dargestellte Beispiel sind z. B. die Wertepaare (Eingang: 240, Ausgang: 240), (E: 320:, A: 240), (E: 320, A: 320) möglich. Der maximale Ressourcenbedarf ergibt sich aus der Betrachtung aller entsprechenden Einträge in der Ressourcentabelle.

[18] RB bzw. MaxRB stellen Tupel von n Ressourcenanforderungen für n Ressourcen dar. Die Bildung des Maximums für ein Tupel wird durch die Maximierung jedes Elements erreicht.



Ressourcenbedarfstabelle

Bildgröße Eingang	Bildgröße Ausgang	CPU	Puffer
. . .			
240	240	k. B.	k. B.
320	240	abc	xyz
320	320	k. B.	k. B.
. . .			

Abb. 25: *Ressourcenbedarf für variable Filter*

Im gegebenen Beispiel ist nur für die Einstellung (E: 320, A:240) eine Ressourcenbelegung nötig, da nur sie eine Konvertierung der Bildgröße bedingt. Die anderen Einstellungen bewirken keinen Ressourcenbedarf. Für den maximalen Ressourcenbedarf variabler Filter ist daher nicht (allein) die Höhe der einzelnen Parameterwerte ausschlaggebend. Vielmehr muß für einen eingestellten Eingangswert (z. B. 320) jede mögliche Konvertierung auf einen niedrigeren (bzw. gleichen) Wert (320 oder 240) betrachtet werden, um den maximal auftretenden Ressourcenbedarf ermitteln zu können:

(R2) $\quad MaxRB = \max\{ RB(ew, aw): aw <= ew \text{ und } ew \in WB_E \text{ und } aw \in WB_A \}$.

Für die Funktionsweise von NRP ist die Art der Berechnung des maximalen Ressourcenbedarfs zunächst unerheblich. Auf der anderen Seite weist die Belegung von Ressourcen durch Komponenten und Links bestimmte Eigenschaften auf, die folgende Vorgehensweise in CINEMA begründen.

Für Komponenten ohne variable Filterrelationen ist die Annahme plausibel, daß eine Erhöhung der zu unterstützenden Parameterwerte auch eine Erhöhung (oder zumindest keine Absenkung) des Ressourcenbedarfs nach sich zieht. Der Grund hierfür ist, daß sowohl eine höhere „Mediendatenrate" (z. B. Bildrate) als auch eine erhöhte „Mediendatengröße" (z. B. Bildgröße) einen erhöhten Verarbeitungsaufwand erwarten lassen. In CINEMA eingesetzte Komponenten hatten alle diese Eigenschaft. Zur Feststellung ihres maximalen Ressourcenbedarfs können diese daher den Ressourcenbedarf verwenden, der dem höchsten zu unterstützenden Parameterwert ent-

spricht:

(R3) MaxRB = RB(max(WB)) .

Ebenso kann für einen Link angenommen werden, daß der maximale Ressourcenbedarf für einen zu unterstützenden Wertebereich dann auftritt, wenn der höchste Wert aus diesem Bereich an den Ports des Links eingestellt wird. Der Grund hierfür ist, daß höhere medienspezifische Parameterwerte auch eine höhere Transportbandbreite erfordern. Zur Ableitung des Ressourcenbedarfs genügt daher eine Berechnung analog zu (R3).

Der Verarbeitungsaufwand variabler Filter, die für CINEMA eingesetzt wurden, hängt vor allem von zwei Faktoren ab. Zum einen beeinflußt die am Eingangs-Port ankommende „Medienlast" den Verarbeitungsaufwand des Filters, zum anderen hängt dieser von der Konvertierung innerhalb des Filters ab. Wird beides abgesenkt, so ist ein niedrigerer Verarbeitungsaufwand zu erwarten. Diese Eigenschaft erlaubt eine gegenüber (R2) weniger aufwendige Berechnungsweise des maximalen Ressourcenbedarfs:

(R4) MaxRB = max{ RB(max(WB_E), aw): aw <= max(WB_E) und aw $\in$ WB_A } .

Hierbei wird am Eingangs-Port der höchstmögliche Wert angenommen und der Ressourcenbedarf wird über alle möglichen Werte maximiert, die am Ausgangs-Port eingestellt werden können. Es ist leicht ersichtlich, daß die Betrachtung eines niedrigeren Eingangswertes sowohl einen verringerten Parameterwert am Eingang als auch eine verringerte Konversion bewirken.

5.3.5 Diskussion

Die drei Phasen von NRP sorgen dafür, daß Einschränkungen eines Flußgraphenelements zu jedem anderen Element im Flußgraphen propagiert werden (siehe Abb. 26).

In der ersten Phase von NRP werden Wertebereiche in AFS von den Quellen zu den Senken propagiert. Anhand der Vorgehensweise in 5.3.3 ist zu erkennen, daß Komponenten für ihre Ports alle Werte ausschließen, die aufgrund von Formatbeschränkungen nicht unterstützbar sind (siehe Beziehungen (1), (6), (11), (12), (16) in 5.3.3). Ferner werden alle Werte ausgeschlossen, die aufgrund fehlender Ressourcen nicht in Frage kommen (1), (4), (5), (9), (10), (13), (16).

Ebenso wird durch die Berücksichtigung der Mischrelationen (7) bzw. der Filterrelationen (3), (8), (12) erreicht, daß der für den Ausgangs-Port einer Komponente berechnete Wertebereich in keinem Widerspruch zu den Wertebereichen der Eingangs-Ports steht. Vielmehr können für jeden Wert des Ausgangs-Ports entsprechende Werte an den Eingangs-Ports eingestellt werden, die weder Formatbeschränkungen noch die Ressourcenverfügbarkeit noch die Stromrelationen der Komponente verletzen.

Da obige Aussage für alle in der ersten Phase durchlaufenen Komponenten (und Links) gilt, kann der in dieser Phase an einem (beliebigen) Komponenten-Port eingestellte Wertebereich aufgrund der flußaufwärts berechneten Wertebereiche unter Einhaltung aller oben erwähnten Randbedingungen unterstützt werden. Genauer ausgedrückt, sind zu jedem Wert aus dem betrachteten Bereich entsprechende Werte an den flußaufwärts liegenden Ports einstellbar. Beispielsweise wird im Szenario des Abschnitts 5.3.2 für den Ausgang des Mischers der Wertebereich (400..480) ermittelt, aus dem jeder Wert auch flußaufwärts unterstützt werden kann. Dagegen ist beispielsweise der Wert 560 nicht erfaßt, weil dieser an Cam1 nicht unterstützt werden kann.

In der zweiten Phase werden Wertebereiche in AFS von den Senken zu den Quellen propagiert. Durch die Berücksichtigung der A-QoS wird zunächst gewährleistet, daß an den Senken-Ports nicht gewünschte Werte ausgeschlossen werden (16). Durch die Berücksichtigung der Misch- bzw. der Filterrelationen ((22) bzw. (18), (20), (24)) sowie der Multicast-Eigenschaft entsprechender Links (26), wird erreicht, daß der für einen (beliebigen) Port einer Komponente berechneten Wertebereich nicht im Gegensatz zu den flußabwärts eingestellten Wertebereichen steht. Für jeden Wert aus dem betrachteten Bereich sind somit entsprechende Werte an den flußabwärts liegenden Ports einstellbar, ohne daß dabei Formatbeschränkungen, Ressourcenverfügbarkeit, Stromrelationen oder A-QoS verletzt werden. In Abschnitt 5.3.2 wird beispielsweise für den Ausgang des Mischers der Wertebereich (400..480) berechnet, da zu jedem Wert aus diesem Bereich flußabwärts entsprechende Werte möglich sind. Zu dem Wert 480 kann z. B. der Wert 320 am Ausgang von VF1 bzw. Eingang von Dsp1 sowie der Wert 480 am Ausgang von VF2 bzw. Eingang von Dsp2 eingestellt werden.

Die besondere Form von Flußgraphen, die dem Zweizonenmodell entsprechen (siehe 5.1), bedingt, daß ein Komponenten-Port existiert, der die zwei Zonen miteinander koppelt. In Abb.

24 stellt der Ausgang des Mischers diesen Port dar. Dieser „Koppel"-Port zeichnet sich dadurch aus, daß er (als einziger) von allen Quellen flußabwärts liegt, während er von allen Senken aus gesehen flußaufwärts angesiedelt ist.[19] Entsprechend obiger Aussagen wird an diesem Port in der ersten bzw. zweiten Protokollphase ein Wertebereich eingestellt, der sowohl flußaufwärts als auch flußabwärts und somit im gesamten Flußgraphen unterstützt werden kann. In Abb. 24 entspricht dies dem Wertebereich (400..480) an dem Mischer-Ausgang.

Der in der zweiten Phase betrachtete DRV-Wert liegt an jedem Komponenten-Port innerhalb des dort berechneten Wertebereichs ((17), (19), (21), (23), (25), (27)). Für den erwähnten Koppel-Port bedeutet dies, daß der ermittelte DRV-Wert einen Wert darstellt, der sowohl flußaufwärts als auch flußabwärts unterstützt werden kann. Im Szenario des Abschnitts 5.3.2 ist ein entsprechender DRV-Wert von 480 am Mischer-Ausgang eingestellt.

Der DRV-Wert am Koppel-Port wird verwendet, um die DRV-Werte in der Mischer- bzw. in der Multicast-Zone einzustellen. Ersteres wird durch die Propagierung des DRV-Werts vom Koppel-Port in Richtung zu den Quellen in der zweiten Phase vorgenommen. Dabei wird durch die Beachtung der Misch- bzw. Filterrelationen ((23) bzw. (19), (21), (25)) sichergestellt, daß die eingestellten DRV-Werte keine dieser Relationen verletzen.[20] In Abschnitt 5.3.2 wird auf diese Weise für alle Ports, die in der Mischer-Zone liegen, der DRV-Wert von 480 eingestellt.

Die Propagierung des DRV-Werts des Koppel-Ports in Richtung zu den Quellen wird in der dritten Phase vorgenommen. Dabei wird durch die Beachtung der Filterrelationen (28), (29), (30) sichergestellt, daß die eingestellten Werte diese Relationen nicht verletzen. In Abschnitt 5.3.2 wird beispielsweise der Wert 320 für den Ausgang von VF1 bzw. den Eingang von Dsp1 eingestellt, während für VF2 und Dsp2 der Wert 480 eingestellt wird.

Abb. 26 faßt die oben beschriebene Ausbreitung von Dienstgüteinformation in den einzelnen Phasen von NRP schematisch zusammen. Dabei wird deutlich, daß die Einstellung der DRV-Werte für die Mischer-Zone bzw. Multicast-Zone in der zweiten bzw. dritten Protokollphase

---

[19] Besteht der Flußgraph nur aus einer Mischer- bzw. Multicast-Zone, so kann der (einzige) Senken-Port bzw. der (einzige) Quellen-Port als Koppel-Port angesehen werden, so daß alle Aussagen ihre Gültigkeit behalten.

[20] Dementsprechend werden die DRV-Werte an den Ports der Mischer-Zone in der dritten Phase nicht mehr verändert, da sie bereits alle Stromrelationen erfüllen.

aufgrund von Information aus allen Teilen des Flußgraphen erfolgt. Ferner verdeutlicht der Ablauf, daß für Flußgraphen, die dem Zweizonenmodell entsprechen, auf jeden Fall drei Phasen der AFS-Propagierung nötig sind, um Dienstgüteinformation einzelner Flußgraphenelemente im gesamten Flußgraphen zu verbreiten.

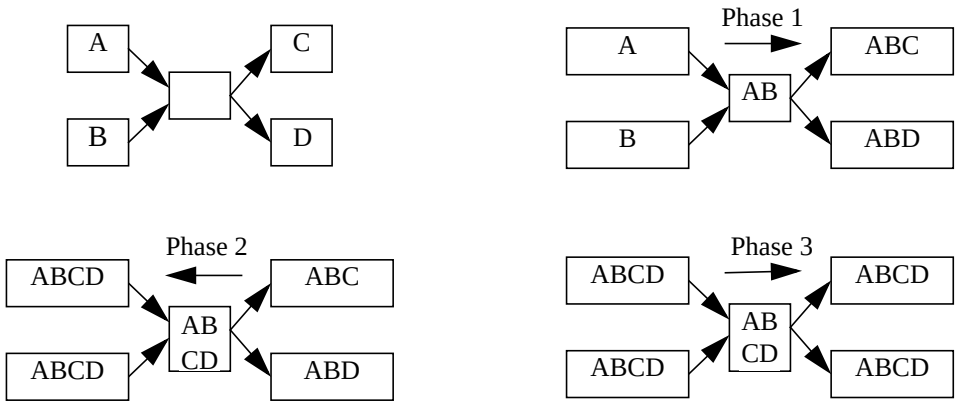


Abb. 26: *Ausbreitung der Dienstgüteinformation*

Die oben beschriebenen Eigenschaften von NRP stellen sicher, daß die an den Komponenten-Ports eingestellten DRV-Werte keine der Randbedingungen verletzen, die durch Formatbeschränkungen, Stromrelationen, Ressourcenverfügbarkeit oder A-QoS vorgegeben werden. NRP sorgt in diesem Sinne immer für einen konsistenten Session-Aufbau.

Darüber hinaus garantiert NRP, daß an den Senken-Ports die maximal möglichen Werte eingestellt werden, sofern keine Ressourcenkonflikte (d. h. keine Ausschlüsse von Werten aufgrund von Ressourcenknappheit) auftreten. Dies wird zum einen dadurch erreicht, daß Wertebereiche an allen Komponenten-Ports nur aufgrund notwendiger Randbedingungen eingeschränkt werden (siehe 5.3.3). Der für den Koppel-Port berechnete Wertebereich stellt somit den gesamten Wertebereich dar, der im gesamten Flußgraphen unterstützt werden kann. Zum anderen werden DRV-Werte, die in der zweiten Phase zunächst die maximal gewünschten Dienstgütewerte an den Senken darstellen, nur reduziert, falls sie nicht in den jeweiligen möglichen Wertebereichen an den Komponenten-Ports liegen (25), (27), (28), (30). Insbesondere faßt ein Multicast-Link die DRV-Werte seiner Empfangs-Ports im Sinne einer Maximierung zusammen (27), um möglichst den höchsten flußabwärts geforderten DRV-Wert zu ermöglichen.

NRP stellt ferner sicher, daß an jedem Port der minimale Wert eingestellt wird, der zum Erreichen der maximal möglichen Dienstgüte an den Senken nötig ist. Dies geschieht in der Erwartung, daß möglichst niedrige Parameterwerte an den Komponenten-Ports einen möglichst niedrigen Ressourcenbedarf insgesamt bewirken. Die minimale Wertewahl wird dadurch erreicht, daß ein DRV-Wert in der zweiten Phase durch variable Filter nur dann erhöht wird, falls Formatbeschränkungen am Eingang des Filters dies erfordern (25). Mischer und fixe Filter rechnen die DRV-Werte gemäß der festen Filterfaktoren um (19), (21). Multicast-Links fassen zwar die DRV-Werte im Sinne einer Maximierung zusammen, diese orientiert sich aber immer am maximalen DRV-Wert, der erhalten wurde (26), (27) (was zur Maximierung der Dienstgüte zumindest an einer Senke erforderlich ist). Der DRV-Wert wird nicht über dieses Maximum hinaus erhöht.

Der Fehlschlag eines Session-Aufbaus wird in der ersten oder zweiten Phase daran erkannt, daß der Wertebereich, der für irgendeinen Port ermittelt wird, leer ist. Wird der Fehlschlag in der Mischer-Zone bewirkt, so wird dies bereits in der ersten Phase erkannt. Auf diese Weise wird sichergestellt, daß (über die Senken) alle Quellen über den Fehlschlag in der zweiten Phase informiert werden. Ein Fehlschlag in der Multicast-Zone wird in der zweiten Phase festgestellt, so daß erneut alle Quellen hierüber informiert werden. Alle Quellen haben vor der dritten Phase Kenntnis vom Scheitern des Aufbaus, so daß sie in der dritten Phase die Freigabe aller Ressourcen veranlassen können.

Im Hinblick auf die in Kapitel 4 formulierten Anforderungen an die Ressourcenreservierung auf der Anwendungsebene (siehe 4.1) bzw. im Vergleich mit den entsprechenden bestehenden Ansätzen läßt sich NRP wie folgt bewerten:

Unterstützte Topologien: NRP ist auf beliebige Flußgraphentopologien anwendbar, die dem Zweizonenmodell entsprechen (siehe 4.1).

Komponenten-Verteilung: NRP ist bei beliebiger Verteilung von Komponenten eines Flußgraphen in einem verteilten System anwendbar.

Formatbeschränkungen: NRP unterstützt diese gemäß der Definition von Kapitel 2.

Stromrelationen: NRP unterstützt sowohl Mischrelationen für Mischer als auch fixe bzw. variable Stromrelationen für fixe Filter und Mischer bzw. für variable Filter.

Heterogene Empfänger: NRP erlaubt eine individuelle Spezifikation von A-QoS sowie eine entkoppelte Einstellung der Dienstgüte für einzelne Senken.

Unterstützte Parameter: NRP unterstützt eine Dienstgüteaushandlung für medienspezifische Parameter sowie in Bezug auf Verzögerung (siehe 5.5).

Wertebereiche: NRP erlaubt die Spezifikation von A-QoS in Form von Wertebereichen für medienspezifische Parameter.

Globale Reservierung: NRP reserviert Ressourcen sowohl zur Verarbeitung als auch zur Kommunikation. Die Reservierungen auf den Endsystemen und im Netzwerk werden exakt abgeglichen in dem Sinne, daß genau die Ressourcen reserviert werden, die für die eingestellten Dienstgütewerte an den Senken benötigt werden.

Lokale Koordination: NRP stellt die optimal möglichen Dienstgütewerte ein, sofern keine Ressourcenkonflikte auftreten. Ist dies nicht der Fall, so führt NRP keine Optimierung bezüglich der Zuteilung knapper Ressourcen an die Komponenten eines Endsystems durch.

Obige Liste zeigt, daß NRP die meisten der in 4.1 aufgestellten Anforderungen erfüllt. Als einzige Ausnahme beinhaltet NRP keine Vorkehrungen für eine optimierte Zuteilung knapper Endsystemressourcen, um eine möglichst hohe Dienstgüte zu ermöglichen. Möglichkeiten zur Erweiterung von NRP um entsprechende Mechanismen werden im Rahmen nachfolgender Abschnitte (5.6 bzw. 5.7) behandelt.

5.4 Kopplung mit Ressourcenreservierung der Transportebene

Im vorigen Abschnitt wurde beschrieben, daß entfernte Links zur Reservierung und Relaxierung aufgerufen werden. Solche Links müssen ein Transportsystem für den Datentransfer zwischen den Lokationen verbundener Komponenten benutzen. Da die Ressourcenreservierung für eine Transportverbindung mit Hilfe eines eigenen Protokolls durchgeführt wird, müssen Aufrufe an den Link entsprechend in Reservierungs- bzw. Relaxierungsaufrufe an das Transportsy-

stem umgesetzt werden. CINEMA sieht für einen Link eine Struktur bestehend aus mehreren Link-Objekten vor (Abb. 27).

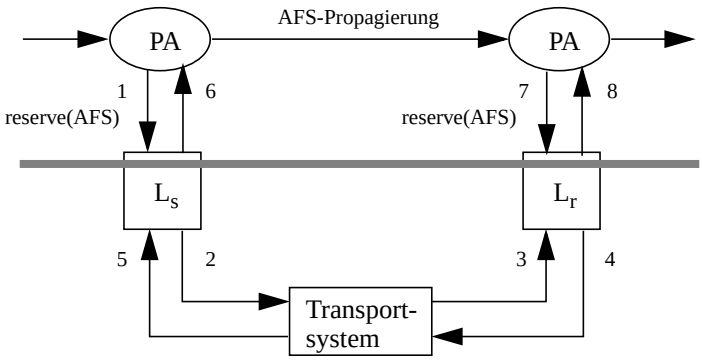


Abb. 27: *Link-Struktur und Ressourcenreservierung*

Für die Sendeseite gibt es ein senderseitiges Link-Objekt L_s, das eine Verbindung zu einem oder mehreren empfängerseitigen Link-Objekten L_r unterhält. Aus Sicht von NRP stellen sich beide Link-Objektarten wie Komponenten mit jeweils einem Eingangs- und einem Ausgangs-Port dar. Die Beschreibung des vorigen Abschnitts ist daher auch unter Einbeziehung der Link-Objekte gültig. Als einzige Erweiterung kommt hinzu, daß das senderseitige Link-Objekt für einen Multicast-Link nicht nur ein, sondern mehrere AFS (eines für jedes verbundene L_r) für den Ausgangs-Port erzeugt bzw. erhält.

In Abb. 27 ist die Umsetzung des *Reserve*-Aufrufs innerhalb des Links dargestellt. Da NRP in der ersten Phase von den Quellen zu den Senken verläuft, ruft NRP zunächst L_s mit einem AFS auf. Von diesem erhält es ein Antwort-AFS, das zu L_r propagiert wird. L_r liefert ebenfalls eine Antwort an NRP. NRP behandelt beide Seiten des Links in gleicher Weise. Um korrekt zu arbeiten, erwartet NRP, daß die Reservierung im Link sich in mindestens einem der Antwort-AFS widerspiegelt. Ansonsten ist NRP unabhängig vom Ablauf der Reservierung im Link.

In Abb. 27 ist ein senderseitiger Aufbau einer Transportverbindung dargestellt. Der Reservierungsaufruf an das senderseitige Link-Objekt (1) wird in eine Verbindungsanforderung an das Transportsystem (2) umgesetzt. Um die erforderliche Transportdienstgüte (T-QoS) zu erhalten, bildet L_s die medienspezifischen Parameterwerte des AFS auf entsprechende Transportparame-

terwerte ab. Hierzu benutzt L_s eine vorgegebene Ressourcenbedarfstabelle, die für jeden medienspezifischen Wert einen Eintrag für T-QoS enthält (siehe 3.1).

Der Aufbauwunsch wird der Empfängerseite über ein entsprechendes Dienstgüte-Primitiv (3) angezeigt. Die Reaktion des empfängerseitigen Link-Objekts kann zwei verschiedenen Schemata entsprechen. Im ersten Schema akzeptiert L_r den Verbindungsaufbau ohne die angezeigte T-QoS zu verändern. Zu einer genauen Antwort ist es nicht in der Lage, da diese von der Dienstgüte auf der Empfängerseite abhängt, die durch NRP erst ausgehandelt werden muß. Das Ergebnis des Verbindungsaufbaus wird L_s angezeigt (5). Hierbei wird erwartet, daß falls die ursprünglich geforderte Bandbreite nicht zur Verfügung steht, das Transportsystem die maximal mögliche Bandbreite anzeigt. Aufgrund dieser Information kann L_s (mit Hilfe der Abbildungstabelle) die medienspezifischen Parameterwerte ausschließen, für die die Bandbreite nicht ausreicht. Anschließend übergibt L_s den möglicherweise reduzierten medienspezifischen Wertebereich an NRP (6).

In der dritten Phase wird der Link zur Relaxierung aufgerufen. Hierbei ist der endgültige medienspezifische Wert bekannt. Zu diesem kann L_s die entsprechende T-QoS ableiten und mit Hilfe eines Primitivs das Transportsystem zur Relaxierung aufrufen. Nach deren Beendigung, meldet sich L_s mit dem unveränderten AFS bei NRP. Wenn L_r zur Relaxierung aufgerufen wird, braucht es keine Aktion durchzuführen. Es gibt das AFS unverändert an NRP zurück.

Das zweite Schema benötigt keine Relaxierungs-Primitive des Transportsystems. Statt dessen wird der Verbindungsaufbau über die NRP-Phasen hinweg durchgeführt. Dazu antwortet L_s in der Reservierungsphase an NRP, bevor es den Verbindungsaufbau anstößt. Wenn ein Verbindungswunsch bei L_r angezeigt wird, wird dieser nicht sofort, sondern erst in der dritten Phase bestätigt. Wenn L_r durch NRP zur Reservierung aufgerufen wird, kann es aufgrund der erhaltenen T-QoS die medienspezifischen Werte ermitteln, für die reserviert werden konnte. Diese übergibt L_r in seiner Antwort an NRP. In der Relaxierungsphase wird beim Aufruf von L_s das AFS nicht verändert. Wenn L_r aufgerufen wird, kann es aufgrund des erhaltenen AFS die endgültige T-QoS berechnen. Mit der T-QoS wird der Verbindungsaufbau aus der ersten Phase beendet.

Das zweite Schema kann zwar ohne Relaxierungs-Primitive auskommen, es bedingt allerdings eine längere Zeit für den Aufbau einer Transportverbindung. Diese kann bei strikten zeitlichen Anforderungen an den Abschluß einer Signalisierung im Netzwerk möglicheweise nicht zugelassen werden. Für diesen Fall ist das erste Schema das einzig mögliche. Das Schema wird von Reservierungsprotokollen wie ST-II ([DeBe95]) unterstützt, von anderen Protokollen, z. B. der ATM-Signalisierung dagegen nicht.

Aus obiger Darstellung ergeben sich zwei Anforderungen an die Reservierung auf Transportebene. Zum einen sollte die Reservierung die maximal mögliche T-QoS anzeigen, falls die geforderte T-QoS nicht erbracht werden kann. Zum anderen soll sie Relaxierungs-Primitive vorsehen, die nach einem erfolgten Verbindungsaufbau aufgerufen werden können. Allerdings wird NRP von diesen Einzelheiten nicht beeinflußt, da ein Link so gestaltet werden kann, daß es die Reservierung für eine Transportverbindung mehrmals (z. B. auch für den minimalen medienspezifischen Parameterwert) probieren kann, um eine Reservierung durchzuführen. Falls eine Relaxierung durch das Transportsystem nicht möglich ist, kann der Link die Relaxierung durch Abbau der alten und Aufbau einer neuen Verbindung mit reduzierter T-QoS nachbilden.

5.5 Berechnung der Verzögerung

Die bisherige Beschreibung hat sich auf die Bestimmung der medienspezifischen Parameterwerte an den Komponenten-Ports beschränkt. Die Behandlung der Verzögerung in CINEMA ist zweistufig möglich. In einer ersten Stufe ist es möglich, für jede Senke die maximale Verzögerung zwischen den Quellen und der Senke anzugeben. In einer zweiten Stufe kann unter gewissen Voraussetzungen überschüssige Verzögerung der Senken an Komponenten und Links verteilt werden. Als überschüssige Verzögerung wird der Anteil bezeichnet, um den die eingestellte Verzögerung unter der maximal zulässigen Verzögerung liegt. Im Rahmen dieser Arbeit wird die erste Stufe der Unterstützung beschrieben, für die andere Stufe wird auf [Fied98] verwiesen.

Um die “Ende-zu-Ende”-Verzögerung zwischen einer Quelle und einer Senke zu bestimmen, müssen alle Verzögerungsanteile aufaddiert werden, die durch dazwischenliegende Komponenten und Links verursacht werden. Diese Anteile sind abhängig von der tatsächlich durchgeführten Reservierung, genauer von den medienspezifischen Parameterwerten, für die reserviert wurde. Belegt z. B. eine Komponente eine CPU mit einer bestimmten Aufrufrate, so wird durch

das „Rate-Monotonic“-Verfahren eine maximale Verzögerung garantiert, die nicht über der Aufrufperiode liegt (siehe 3.3.2).

Die CPU-Aufrufperiode ist durch die Einstellung der endgültigen medienspezifischen Parameter (z. B. Bildrate) ableitbar. Die Verzögerung kann daher nicht vor der Relax-Phase von NRP für jede Komponente und Link eindeutig ermittelt werden. Um die Ende-zu-Ende Verzögerung für die einzelnen Senken zu bestimmen, enthält ein AFS (im Teil für generische Parameter) einen Eintrag „akkumulierte Verzögerung" (*AccDelay*). Dieser wird in der *Relax*-Phase von den Komponenten und Links verwendet, um ihren Verzögerungsanteil zu erfassen.

Zu Beginn der *Relax*-Phase sind alle *AccDelay*-Einträge auf Null gesetzt. Wird eine Komponente aufgerufen, so ermittelt sie in Interaktion mit dem CPU-Manager, welche Verzögerung sie maximal erfahren kann. Handelt es sich bei der Komponente um einen Mischer, so berechnet sie davor das Maximum der für die Eingangs-Ports enthaltenen *AccDelay*-Werte. Dieses Maximum stellt die größte Verzögerung zu den flußaufwärts liegenden Senken dar. Die Komponente addiert den eigenen Anteil zu diesem Maximum und trägt die Summe als neuen *AccDelay*-Wert in das AFS ein.

Wird ein Link aufgerufen, so erwartet dieser bei der Relaxierung der Reservierung im Transportsystem eine Angabe über die Verzögerung der relaxierten Verbindung. Alle früher erwähnten Reservierungsprotokolle sind in der Lage die maximale Transportverzögerung anzugeben. Der Link addiert diesen Anteil an das *AccDelay* des erhaltenen AFS. Handelt es sich um einen Multicast-Link, so kann durch das Transportsystem für jeden Empfangs-Port ein eigener Verzögerungswert angegeben werden. Da der Link für jeden Empfangs-Port ein eigenes AFS zur Verfügung stellt, ist die individuelle Erfassung dieser Werte sichergestellt.

Sind die AFS bei den Senken angelangt, rechnen diese ihren Verzögerungsanteil hinzu und stellen NRP die *AccDelay*-Werte zur Verfügung. Für jede Senke ist somit die “Ende-zu-Ende”-Verzögerung zu den Senken vorhanden. Das Verhalten der Mischer erzwingt, daß diese sich auf alle Quellen bezieht. Eine individuelle Unterscheidung ist aber prinzipiell möglich. Hierzu müßte im AFS ein *AccDelay*-Eintrag für jede Quelle mitgeführt werden. Mischer müßten die *AccDelay*-Werte nicht im Sinne einer Maximierung zusammenfassen, ansonsten wäre ihre Behandlung durch Komponenten und Links gleich.

Am Ende der *Relax*-Phase sind alle Senken über die jeweilige Verzögerung informiert. Wurde im Rahmen der Dienstgütespezifikation an den Senken jeweils eine maximale “Ende-zu-Ende”-Verzögerung vorgegeben, so können die PA der Senken feststellen, ob diese verletzt wurde. Falls ja, wird dies dem Klienten angezeigt, der daraufhin die Möglichkeit hat, die Session zu beenden. NRP baut die Session nicht automatisch ab, da die berechnete “Ende-zu-Ende”-Verzögerung die initiale Information ist, die ein Klient über die zu einem gegebenen Zeitpunkt erzielbare Verzögerung im verteilten System erhält.

5.6 Erweiterung zur Vermeidung von Ressourcenkonflikten

5.6.1 Problemstellung

Das im Abschnitt 5.2 vorgestellte Protokoll stellt den korrekten Aufbau von Sessions sicher. Eine Optimalität in Bezug auf die eingestellte Dienstgüte garantiert es, falls im Laufe der Reservierung (in der ersten Phase) keine Ressourcenkonflikte auftreten. Bei Ressourcenkonflikten beinhaltet es keine Mechanismen, um die Aufteilung der Ressourcen zu optimieren.

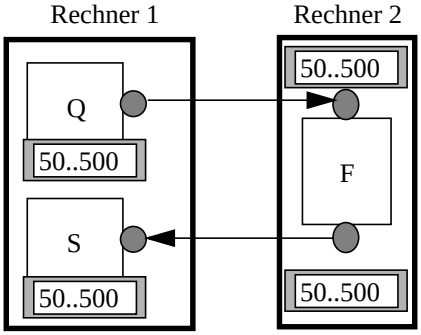


Abb. 28: *Konkurrenz bei Belegung lokaler Ressourcen*

Obiger Flußgraph verdeutlicht das Problem. Es soll eine Session für einen Flußgraphen aufgebaut werden, der Komponenten auf zwei Rechnern umfaßt: Quelle und Senke auf R1 und einen Filter auf R2. Dargestellt sind auch die Formatbeschränkungen, aufgrund derer NRP in der ersten Phase Ressourcen reserviert. Zuerst werden Ressourcen für die Quelle belegt, dann für den Filter und zuletzt für die Senke. Es sei angenommen, daß die Quelle für den maximalen

Parameterwert von 500 reservieren kann, so daß sie dabei zugleich die Ressourcen von R1 erschöpft. Wenn die Senke eine Belegung versucht, sind keine Ressourcen vorhanden und es kann keine Session aufgebaut werden kann. Würde die Quelle ihre Reservierung entsprechend eines niedrigeren Parameterwerts (z. B. 50) relaxieren, so hätte die Senke evtl. auch für diesen Wert reservieren können. Um dies zu erreichen, ist ein Verfahren nötig, das bei Ressourcenkonflikten Absenkungen von Parameterwerten auf einem Rechner vornimmt.

Das Problem läßt sich auch in Bezug auf das Netzwerk formulieren. Ist die Bandbreite zwischen R1 und R2 begrenzt, so ist evtl. eine Absenkung der Parameterwerte für die Links Q-F und F-S nötig, um auf diese Weise für beide Links eine Reservierung durchführen zu können.

Zur Behandlung von Ressourcenkonflikten wurden in CINEMA zwei Ansätze untersucht. Die Vermeidung von Ressourcenkonflikten wird im ersten Ansatz angestrebt, indem ein Session-Aufbau für eine minimal mögliche Dienstgüte an den Senken versucht wird (siehe unten). Der zweite Ansatz beinhaltet dagegen einen Mechanismus zur Auflösung von Ressourcenkonflikten, um unter gewissen Voraussetzungen eine möglichst optimale Dienstgüte zu erzielen (5.7).

5.6.2 Reservierung für minimale Dienstgüte

5.6.2.1 Konzept

Eine Möglichkeit Ressourcenkonflikte zu vermeiden, besteht darin, nach einem fehlgeschlagenen Session-Aufbau, die Dienstgüteanforderungen zu reduzieren, um implizit auch den damit verbundenen Ressourcenbedarf abzusenken. Diese Vorgehensweise wurde früher für den im Abschnitt 4.3.3 beschriebenen Ansatz vorgeschlagen, allerdings nur für Flußgraphen, die keine variablen Filter enthalten.

In solchen Flußgraphen ist die Dienstgütebestimmung dadurch vereinfacht, daß die Parameterwerte aller Ports miteinander fest gekoppelt sind. Wird der Wert an einem Port verändert, so ändert sich der Wert an allen anderen Ports proportional (siehe 2.2). Dies ermöglicht ein Verfahren, das die möglichen Werte an einem (Kontroll-)Port in absteigender Reihenfolge durchläuft, zu jedem Wert entsprechende Werte an den anderen Ports einstellt und für den Flußgraphen eine Reservierung versucht. Der Vorteil des Verfahrens ist, daß es die optimale Dienstgüte

einstellen kann.[21] Als Nachteil steht dem entgegen, daß es bei n Parameterwerten bis zu n mal Reservierungen (auf Rechnern und im Netzwerk) „testet".

Für Flußgraphen mit variablen Filtern läßt sich im Prinzip ein ähnliches Verfahren entwickeln. Es beruht auf der Beobachtung, daß falls die Dienstgütevorgaben an allen Senken (als Werte) fixiert sind, sich für alle anderen Ports die maximal nötigen Parameterwerte ableiten lassen, und zwar unabhängig von der gegebenen Ressourcenlage. Dies leistet NRP, wenn es nicht zur Ressourcenreservierung, sondern nur zum funktionalen Abgleich eingesetzt wird. Hierzu muß NRP in der ersten Phase funktionale Einschränkungen und Dienstgüteanforderungen berücksichtigen, darf aber keine Ressourcenbelegungen bewirken (so daß keine Konflikte auftreten können). Am Ende der dritten Phase sind dann an allen Ports die minimal erforderlichen Parameterwerte eingestellt, die für die Wertvorgaben an den Senken nötig sind. Zu dieser Einstellung ist ein eindeutiger Ressourcenbedarf für den Flußgraphen gegeben, so daß durch eine anschließende Reservierung überprüfbar ist, ob eine ausreichende Ressourcenverfügbarkeit gegeben ist.

Die Tatsache, daß variable Filter die Einstellungen an den Senken voneinander entkoppeln können, bedingt, daß falls für jeden Senken-Port k Werte gegeben sind, insgesamt n^k Einstellungen möglich sind. Um den Ansatz anwendbar zu machen, ist es nötig, die Zahl der zu testenden Dienstgütevorgaben an den Senken-Ports sinnvoll einzuschränken. Dies ist allerdings ohne Kenntnis der Ressourcenverfügbarkeit im allgemeinen nicht systematisch möglich, da sich diese unterschiedlich auf eine optimale Parametereinstellung an den einzelnen Senken auswirken kann. Ohne Berücksichtigung der Ressourcenverfügbarkeit läßt sich jedoch eine Extremeinstellung als Dienstgütevorgabe berechnen, nämlich die minimal mögliche Dienstgüte unter Berücksichtigung funktionaler Einschränkungen.

Der Gesamtablauf des Session-Aufbaus kann dann in zwei Schritten durchgeführten werden. In einem ersten Schritt wird NRP ausgeführt, um eine möglichst hohe Dienstgüte an den Senken zu erzielen. Kann NRP wegen mangelnder Ressourcen keine Session aufbauen, so wird anschließend die minimal zulässige Dienstgüte an den Senken berechnet und eine Reservierung hierfür versucht. Entspricht der minimalen Dienstgüteeinstellung auch ein minimaler Ressour-

[21] Sofern sich die Ressourcenlage zwischen zwei Reservierungen nicht ändert.

cenbedarf, so wird eine Session aufgebaut, sofern dies überhaupt möglich ist. Das für den zweiten Schritt benötigte Protokoll (NRP-Min) wird im folgenden beschrieben.

5.6.2.2 Das NRP-Min-Protokoll

Wird NRP ohne Ressourcenreservierung in der ersten Phase durchgeführt, so stellt es an den Senken die funktional maximal möglichen Parameterwerte ein. Durch eine geringe Änderung der zweiten Protokollphase ist NRP in der Lage, an den Senken die funktional minimal möglichen Werte einzustellen. Diese müssen nicht gleich den minimal geforderten Werten sein, da Formatbeschränkungen im Flußgraphen die Verwendung bestimmter (z. B. der minimal geforderten) Werte verhindern können.

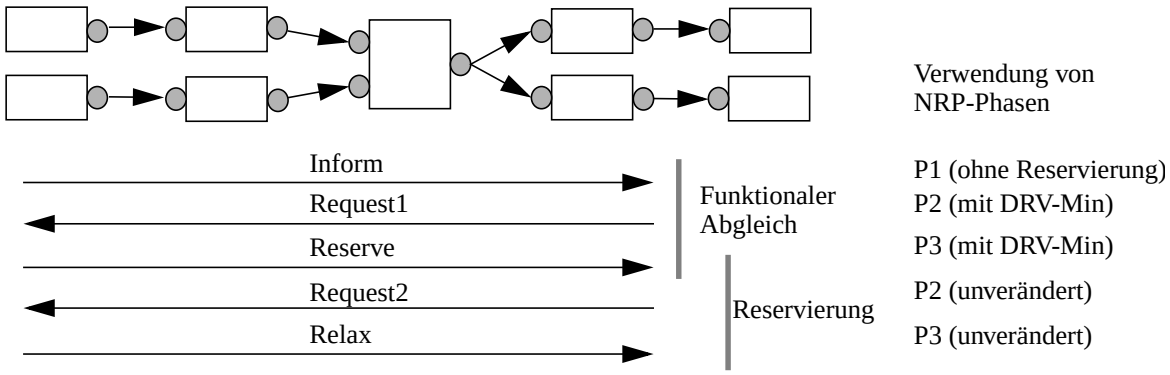


Abb. 29: *Das NRP-Min Protokoll*

Hierzu wird in den AFS statt des DRV-Werts ein sogenannter Min-DRV-Wert verwendet, der von den Senken in der zweiten Phase auf den minimalen Wert des jeweiligen Wertebereichs gesetzt wird. Die Behandlung der Min-DRV-Werte während der zweiten Phase unterscheidet sich ansonsten nicht von der Behandlung der DRV-Werte. Insbesondere werden sie durch Komponenten und Links nur dann erhöht, falls unbedingt erforderlich (siehe 5.3). Am Ende der zweiten Phase verfügen die Quellen über Min-DRV-Werte, die sie zur Reservierung verwenden müssen, falls an den Senken die minimal mögliche Dienstgüte eingestellt werden soll. Für diese feste Einstellung kann ein Session-Aufbau mit NRP versucht werden.

Die Umsetzung des Verfahrens in ein Protokoll (NRP-Min) erfordert die zweifache Verwendung von NRP: einmal zur Ableitung der Min-DRV-Werte und einmal zur Durchführung der Reservierung für diese Werte (Abb. 29). Der funktionale Abgleich stellt die endgültigen Min-DRV-Werte (wie die DRV-Werte) für die Quellen am Ende der zweiten Phase bereit. Ab hier kann die dritte Phase des ersten NRP-Durchlaufs zusammen mit der ersten Phase des Durchlaufs zur Reservierung durchgeführt werden, da an allen Komponenten und Links die endgültigen DRV-Werte eingestellt werden, bevor reserviert wird. Insgesamt werden so fünf Phasen benötigt.

Die erste und die dritte Phase unterscheiden sich nur darin, daß während der dritten Phase gemäß der Min-DRV-Werte reserviert wird, während in der ersten Phase keine Reservierung stattfindet. In der zweiten Phase werden die Berechnungen in Bezug auf die Werte DRV-Min statt DRV durchgeführt. Ansonsten sind die verwendeten Phasen gegenüber den NRP-Phasen unverändert.

5.6.3 Diskussion

Das NRP-Min Protokoll hat den Vorteil, daß es Parameterwerte an den Komponenten-Ports unabhängig von einer Ressourcenlage einstellen kann. Enthalten die an den spezifizierten A-QoS lediglich Werte (und keine Wertebereiche) stellt das Protokoll an allen Komponenten-Ports die minimalen Werte ein, die die geforderten Dienstgüten an den Senken ermöglichen. Sind an den Senken Wertebereiche vorgegeben, so findet das Protokoll die funktional minimal möglichen Werte an allen Komponenten-Ports einschließlich der Senken.

Unter der Annahme, daß die Einstellung minimaler Port-Werte einen minimalen Ressourcenbedarf bewirkt, wird anschließend ein Session-Aufbau mit minimalem Ressourcenbedarf versucht. Diese Annahme ist in Bezug auf die Netzwerklast erfüllt, da minimale Port-Werte an den Links auch zu einer minimalen Bandbreite im Netzwerk führen (siehe 5.3.4). In Bezug auf die Endsysteme ist dies nicht notwendigerweise gegeben, da der Ressourcenbedarf variabler Filter nicht dann minimal sein muß, wenn seine Port-Werte minimiert sind (siehe 5.3.4). Für andere Flußgraphenelemente (Komponenten ohne variable Stromrelationen, Link-Objekte) kann andererseits eine Reduktion des Ressourcenbedarfs in der Regel angenommen werden. NRP-Min

minimiert den Ressourcenbedarf, falls dieser zweite Effekt überwiegt. Dies ist insbesondere für Flußgraphen ohne variable Filter sichergestellt.

5.7 Erweiterung zur Auflösung lokaler Ressourcenkonflikte

5.7.1 Konzept

Das zuvor beschriebene Protokoll versucht Ressourcenkonflikte durch Dienstgüteanforderungen zu vermeiden, die im voraus (d. h. vor einer Ressourcenreservierung) minimiert werden. Ein alternativer Ansatz besteht darin, die Dienstgüteparameter nur bei Bedarf und im erforderlichen Ausmaß zu reduzieren. Im folgenden wird eine Erweiterung von NRP vorgestellt, die auf Flußgraphen ohne variable Filter angewandt werden kann und unter gewissen Voraussetzungen eine optimale Dienstgüte für die Senken des Flußgraphen ermöglicht.

Die Grundidee für die Erweiterung besteht darin, potentielle Ressourcenkonflikte auf den Endsystemen auszuschließen, bevor eine Reservierung mit Hilfe von NRP durchgeführt wird. Um dies zu erreichen, werden auf jedem Endsystem alle Werte an den Komponenten-Ports entfernt, die zu lokalen Ressourcenkonflikten führen können. Wird anschließend NRP ausgeführt, so kommen Ressourcenkonflikte auf den Endsystemen nicht mehr vor und es sind keine Reduzierungen medienspezifischer Parameterwerte wegen knapper Endsystemressourcen nötig.

Sofern der Ausschluß von Parameterwerten an den Komponenten-Ports zur Auflösung lokaler Ressourcenkonflikte optimal durchgeführt werden kann und im Netzwerk keine Ressourcenkonflikte auftreten, wird durch NRP die bestmögliche Dienstgüte für die Senken eingestellt. Dies ist auf die entsprechende Eigenschaft von NRP zurückzuführen, die bestmögliche Dienstgüteeinstellungen für den Fall garantiert, daß keine der benötigten Ressourcen einer Kapazitätsknappheit unterliegt (siehe 5.3.5). Diese angestrebten Eigenschaften haben den im folgenden vorgestellten Ansatz begründet.

Der Ansatz wird in mehreren Schritten beschrieben. Zunächst wird das Konzept zum Ausschluß der Parameterwerte dargestellt, die zu einem Ressourcenkonflikt auf einem Endsystem führen können. Anschließend wird die Einbettung des Verfahrens in ein Protokoll aufgezeigt, das NRP

miteinbezieht. Darauf aufbauend werden die Eigenschaften des vorgestellten Protokolls erläutert.

Der vorgestellte Ansatz ist auf beliebige Flußgraphen anwendbar, die dem Zweizonenmodell entsprechen (siehe 5.1) und keine variablen Filter enthalten. Wie nachfolgend deutlich wird werden ferner fixe Relationen nur im Sinne von Gleichheitsbeziehungen behandelt, d.h. alle Filter- und Mischfaktoren werden als gleich eins vorausgesetzt.

5.7.2 Lokale Koordination

Im Mittelpunkt des Verfahrens steht auf jedem Endsystem ein sogenannter lokaler Koordinator (LK), der die Ressourcenbelegung auf dem Endsystem zwischen den lokalen Komponenten und Link-Objekten koordiniert (Abb. 30). Der LK erhält von Komponenten und Links des Endsystems Meldungen über die möglichen Parameterwerte an ihren Ports sowie den damit verbundenen lokalen Ressourcenbedarf (siehe unten). Stellt der LK bei einer neuen Anmeldung fest, daß der Gesamtressourcenbedarf nicht erfüllt werden kann, senkt der LK für alle lokalen Elemente die Port-Werte ab, bis ihr Ressourcenbedarf die verfügbaren Ressourcen nicht überschreitet.

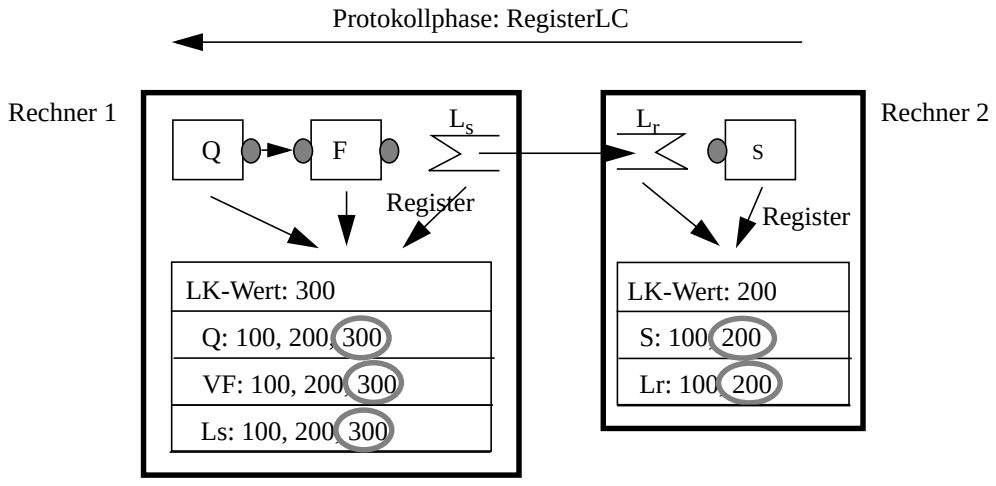


Abb. 30: *Lokale Koordination*

Die Anmeldung der Elemente auf einem Endsystem erfolgt im Verlauf einer Protokollphase, die Komponenten und Links in Richtung von den Senken zu den Quellen durchläuft. Die Propagierung der AFS findet in dieser Phase analog zu der Vorgehensweise in der zweiten Phase von NRP statt (siehe 5.3.3). Insbesondere werden die A-QoS an den Senken in den AFS erfaßt und es finden die gleichen Umrechnungen der Wertebereiche zwischen Ausgangs- und Eingangs-Ports statt. Ferner werden die Werte ausgeschlossen, die aufgrund von Formatbeschränkungen nicht zulässig sind. Anders jedoch als in der zweiten Phase von NRP, stoßen Komponenten und Link-Objekte auch eine Reservierung von Endsystemressourcen an, die auf jedem Endsystem wie folgt koordiniert wird.

Hat eine Komponente (bzw. ein Link-Objekt) für ihre Ports den jeweiligen funktional zulässigen Wertebereich ermittelt, so übergibt die Komponente (bzw. das Link-Objekt) einen entsprechenden Wertebereich (M-Bereich) sowie einen Zeiger auf die eigene Ressourcenbedarfstabelle an den LK. Für Komponenten ohne variable Stromrelationen (Quellen, Senken, fixe Filter, Mischer) genügt die Übergabe eines Wertebereichs für einen Port, da an den anderen Ports derselbe Wert einzustellen ist.[22]

Der LK unterhält eine Kontrollgröße (LK-Wert). Diese gibt die zu einem Zeitpunkt gültige Obergrenze für Parameterwerte an den Ports gemeldeter Elemente vor. Wird der LK-Wert reduziert, so wird für jedes Element der größte Wert aus seinem M-Bereich angenommen, der kleiner oder gleich dem LK-Wert ist. Zu jedem Zeitpunkt legt somit der LK-Wert aktuelle Port-Werte für alle gemeldeten Elemente fest. Der LK kann den Ressourcenbedarf jedes einzelnen Elements aus der Ressourcenbedarfstabelle entnehmen. In Interaktion mit den lokalen Ressourcenmanagern kann der LK daher zu einem aktuellen LK-Wert feststellen, ob der Gesamtressourcenbedarf gemeldeter Elemente durch die Endsystemressourcen erbracht werden kann.

Wenn sich ein neues Element anmeldet, ermittelt der LK den höchstmöglichen LK-Wert, der für die gemeldeten Elemente einen erfüllbaren Ressourcenbedarf bedingt. Hierzu wird der LK-Wert solange gesenkt, bis ein solcher Wert erreicht ist, wobei ursprünglich, d. h. bis zur ersten Absenkung wegen eines Ressourcenkonflikts, ein LK-Wert von unendlich angenommen wird.

[22] Alle Filter- und Mischfaktoren werden als gleich eins vorausgesetzt.

Am Ende der Protokollphase sind alle Komponenten- und Link-Objekte durchlaufen, so daß auf jedem Endsystem ein LK-Wert eingestellt ist, der für alle lokalen Komponenten und Link-Objekte eine ausreichende Ressourcenverfügbarkeit anzeigt. Ferner sind die entsprechend benötigten Port-Werte der Komponenten und Link-Objekte ermittelt. Für das in Abb. 30 dargestellte Szenario wurde beispielsweise für den Rechner 2 ein LK-Wert von 200 eingestellt, ein Wert, der für die Senke bzw. das Empfangs-Link-Objekt die Einstellung des gleichen Wertes zuläßt. Auf Rechner 1 wurde ein LK-Wert von 300 ermittelt, der wiederum den Wert 300 für die Quelle, den Filter und das Sende-Link-Objekt ermöglicht.

LK-Werte, die über dem eingestellten Wert liegen, sind auf einem Endsystem nicht unterstützbar. Dies wird dadurch überprüft, daß alle (in Frage kommenden) höheren LK-Werte, im Hinblick auf eine ausreichende Ressourcenverfügbarkeit in absteigender Reihenfolge betrachtet werden. Der ermittelte LK-Wert ist somit der höchste Wert, für den eine ausreichende Ressourcenverfügbarkeit für alle lokalen Komponenten und Link-Objekte gegeben ist. Auf der anderen Seite bedingen niedrigere Werte an Komponenten-Ports auch einen niedrigeren (oder höchstens gleichbleibenden) Ressourcenbedarf. Für Komponenten ohne variable Stromrelationen bzw. für Link-Objekte wurde dies in 5.3.4 motiviert.

Durch den ermittelten LK-Wert wird für jedes Element ein Wertebereich abgegrenzt, aus dem jeder Wert für das Element einen erfüllbaren Ressourcenbedarf bedingt. In Abb. 30 sind beispielsweise für die Quelle die Werte (100, 200, 300) in diesem Sinn unterstützbar. Werden im nachfolgenden Protokollphasen nur Werte aus diesen Bereichen verwendet, so können auf den Endsystemen keine Ressourcenkonflikte auftreten.

5.7.3 Das NRP-LK-Protokoll

Mit Hilfe der oben beschriebenen Protokollphase wird eine koordinierte Auflösung von Ressourcenkonflikten auf Endsystemen erreicht. Wird im Anschluß daran NRP ausgeführt, so wird eine Reservierung sowohl auf Endsystemen als auch im Netzwerk vorgenommen. Werden hierbei lediglich die Wertebereiche berücksichtigt, die im Verlauf der *RegisterLC*-Phase abgeleitet wurden, so zeigt die Diskussion in 5.7.5, daß Vorteile bei der Bereitstellung der Dienstgüte erzielbar sind. Zusätzlich stellt die Anwendung von NRP sicher, daß in jedem Fall, d. h. auch wenn Ressourcenkonflikte im Netzwerk auftreten, ein konsistenter Session-Aufbau erfolgt

(siehe 5.3.5). Die Zusammenfassung der *RegisterLC*-Phase mit den drei Phasen, die NRP entsprechen, führt zu dem folgenden, als NRP-LK bezeichneten Protokoll.

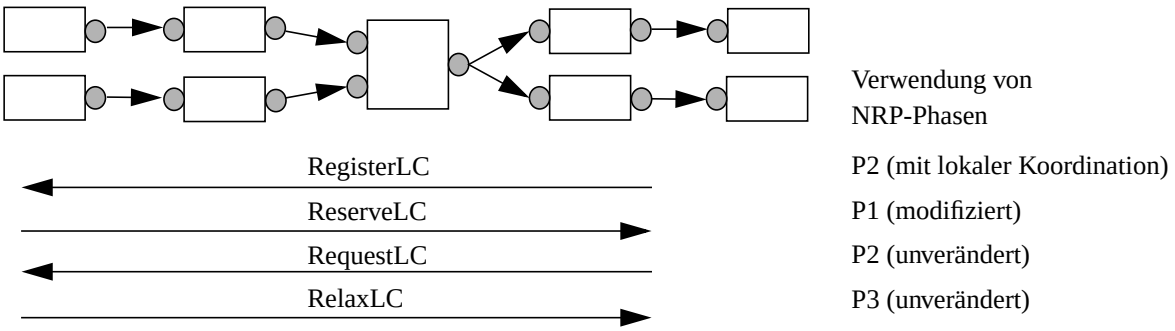


Abb. 31: *Phasen des NRP-LK Protokolls*

In der ersten Phase (*RegisterLC*) werden AFS von den Senken zu den Quellen propagiert. Die Wertebereiche in den AFS werden mit den an den Senken geforderten A-QoS-Bereichen initialisiert. Die Elemente des Flußgraphen werden nacheinander zur Registrierung bei dem LK aufgerufen, wobei jedes Element seinen M-Bereich unter Berücksichtigung der Formatbeschränkungen ableitet und bei dem LK über eine *LCInform*-Methode anmeldet. Der LK stellt den höchsten LK-Wert fest, der durch die aktuelle Ressourcenlage möglich ist und reserviert für jedes angemeldete Element Ressourcen entsprechend des berechneten LK-Werts.

In der zweiten Phase (*ReserveLC*) werden AFS von den Quellen zu den Senken propagiert, um eingestellte Port-Werte flußabwärts abzugleichen. Jedes Element ruft den LK mit einer *LCQuery*-Methode auf, wobei der LK erst antwortet, wenn die erste Phase auf dem Rechner beendet ist (d. h. wenn sich alle Elemente angemeldet haben[23]). Dem Element wird der eingestellte LK-Wert mitgeteilt, so daß es seine Port-Werte entsprechend einstellen kann. Ferner werden dem Element Zugriffsobjekte („Handler") auf die Reservierungen der ersten Phase übergeben, mit denen es im Verlauf späterer Phasen Relaxierungen vornehmen kann. Link-Objekte bauen benötigte Transportverbindungen auf, d. h. sie reservieren Ressourcen auch im Netzwerk.

[23] Der LK wird durch das Konfigurationsmanagement über die Zahl der lokalen Elemente informiert.

In der dritten Phase (*RequestLC*) werden AFS von den Senken zu den Quellen propagiert, um eingestellte Port-Werte flußaufwärts abzugleichen. Die Phase entspricht exakt der zweiten Phase von NRP. In Analogie zu NRP sind am Ende der *RequestLC*-Phase an den Quellen die endgültigen Werte bekannt, die ihren endgültigen Ressourcenbedarf bestimmen. Diese Werte werden in der letzten Phase (*RelaxLC*) in AFS von den Quellen zu den Senken propagiert, wobei jedes Element die endgültigen Werte an seinen Ports einstellt und hierbei evtl. zuviel belegte Ressourcen freigibt.

5.7.4 Beispiel

Es sei der Flußgraph von Abb. 32 gegeben. An den Senken S1 und S2 sei die geforderte Diensgüte als (10..40) bzw. (10..50) spezifiziert. Alle Ports sollen dieselbe Formatbeschränkung (10..50) haben, so daß von deren Betrachtung abgesehen werden kann. Auf die Betrachtung der Link-Objekte wird vereinfachend ebenfalls verzichtet.

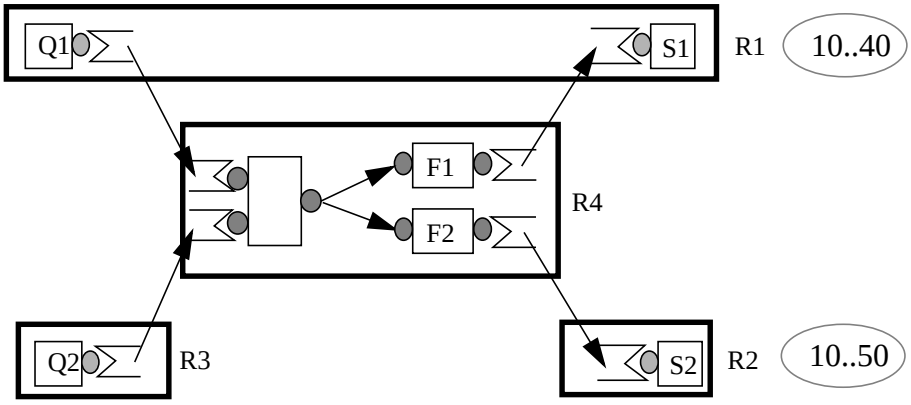


Abb. 32: *Beispielflußgraph*

Die erste Phase beginnt mit dem Aufruf der Senken zur Registrierung. Beide melden sich mit den M-Bereichen (10..40) bzw. (10..50) bei ihrem LK an. Die LK überprüfen die Ressourcenverfügbarkeit für den höchsten Wert (40 bzw. 50). S1 und S2 erhalten unveränderte Bereiche zurück, da hierbei kein Konflikt auftritt. Die Bereiche werden in zwei AFS an F1 und F2 propagiert. F1 meldet den Bereich (10..40) bei dem LK an. Der LK überprüft die Ressourcenver-

fügbarkeit für die maximale Einstellung (40) und stellt einen Ressourcenkonflikt fest, so daß der LK-Wert auf 30 reduziert wird. Für diesen Wert sollen auf R4 ausreichend Ressourcen vorhanden sein. F2 meldet analog den Bereich (10..50) an den LK. Der LK soll für den Wert 30 ausreichend Ressourcen feststellen, so daß der Bereich (10..30) an F2 zurückgegeben wird.

Die AFS von F1 und F2 werden durch den Multicast-Link miteinander geschnitten, so daß der Mischer mit dem Bereich (10..30) aufgerufen wird. Dieser meldet sich bei dem LK an, der entsprechende Ressourcenverfügbarkeit feststellt und den Bereich unverändert zurückliefert. Der Bereich wird in AFS an Q1 bzw. Q2 weiterpropagiert. Bei der Anmeldung von Q1 soll sich ein Konflikt auf R1 ergeben. Der LK-Wert auf R1 wird auf 20 reduziert, für welchen Fall die Ressourcen ausreichend sind. An Q1 wird daher der Bereich (10..20) geliefert. Q2 meldet sich beim LK von R3 mit dem Bereich (10..30) an und erhält ihn wegen ausreichender Ressourcen unverändert zurück.

In der zweiten Phase belegen die Komponenten und Links benötigte Ressourcen und propagieren ihre Wertebereiche flußabwärts. Um die Werte an ihren Ports zu ermitteln, ist es nötig, den LK-Wert von dem LK zu erfragen (dieser gibt Auskunft über die lokale Ressourcenverfügbarkeit nach Anmeldung aller lokalen Elemente), sowie die an Eingangs-Ports erhaltenen Wertebereiche zu beachten (diese geben Auskunft über die Unterstützung durch flußaufwärts gelegene Flußgraphenteile). Darüber hinaus erhalten die Elemente Zugriff auf die in der ersten Phase reservierten Ressourcen.[24]

Im gegebenen Beispiel erfragen Q1 und Q2 den gültigen LK-Wert (20 bzw. 30). Da die Wertebereiche an ihren Ports (10..20) bzw. (10..30) den jeweiligen LK-Wert nicht überschreiten, müssen sie nicht eingeschränkt werden. An die senderseitigen Link-Objekte auf R1 bzw. R3 werden die Bereiche (10..20) bzw. (10..30) propagiert. L_s auf R1 baut eine Verbindung zu R4 entsprechend dem Wert 20 auf. Dies sei möglich. L_s auf R3 und das entsprechende Link-Objekt auf R4 verfahren (für den Wert 30) analog.

An den Mischer werden für seine Ports die Bereiche (10..20) bzw. (10..30) übergeben, woraus er die Schnittmenge (10..20) bildet. An F1 bzw. F2 wird dieser Bereich weitergereicht. F1 und

[24] Das Erfragen des LK-Wertes sowie die Übernahme der Ressourcen-”Handler” findet für alle Elemente in gleicher Weise statt und wird im folgenden nicht mehr einzeln erwähnt.

F2 leitet für den jeweiligen Ausgangs-Port den Bereich (10..20) ab. Die senderseitigen Link-Objekte auf R4 bauen Verbindungen zu R1 und R2 auf, wobei erneut keine Ressourcenkonflikte im Netzwerk angenommen werden. An S1 bzw. S2 werden daher die Bereiche (10..20) bzw. (10..20) gereicht.

In der dritten Phase werden die Wertebereiche in AFS an die Quellen propagiert. Im Beispiel bewirkt dies lediglich für Q2 sowie den Link zwischen R3 und R4 eine Anpassung (von 30 auf 20). In der letzten Phase werden die maximalen Werte der Quellen flußabwärts propagiert und zur Ressourcenrelaxierung benutzt. Im Beispiel müssen alle Elemente außer Q1 und dem Link zwischen R1 und R4 eine Relaxierung durchführen, da für sie in der ersten bzw. der zweiten Phase (gemäß des Wertes 30) zuviel an Ressourcen reserviert wurde.

5.7.5 Diskussion

Entsprechend der Argumentation von 5.6.2 zeichnen sich Flußgraphen ohne variable Filter dadurch aus, daß die Festlegung des Parameterwerts an einem Komponenten-Port die Einstellung entsprechender Werte an allen anderen Ports bedingt. Sind, wie in 5.7.1 angenommen, alle Filter- und Mischfaktoren auf eins gesetzt, so muß an allen Komponenten-Ports derselbe Wert eingestellt werden.

Im Rahmen von NRP-LK werden zuletzt Phasen ausgeführt, die dem Ablauf von NRP entsprechen. Hierdurch wird erreicht, daß an den Ports nur Werte verwendet werden, die aufgrund der gegebenen fixen Stromrelationen und Formatbeschränkungen zulässig sind (siehe 5.3.3). Dies wiederum bedeutet, daß NRP-LK denselben Wert an allen Ports einstellt und daß der eingestellte Wert zur Schnittmenge aller Formatbeschränkungen gehört. Sei im folgenden dieser Wertebereich mit FWB bezeichnet.

Sofern keine variablen Filter auf einem Rechner R instanziiert sind, garantiert die lokale Koordination der ersten Phase, daß die höchste Obergrenze für die Werte ermittelt wird, die an den Ports aufgrund verfügbarer Ressourcen eingestellt werden können. Ist ein maximaler LK-Wert maxW(R) bestimmt, so kann an den Ports der Elemente auf R jeder Wert aus FWB unterstützt werden, der maxW nicht überschreitet. Ein höherer Wert aus FWB ist dagegen nicht unterstütz-

bar, da die lokale Koordination für den entsprechend höheren LK-Wert den Ressourcenbedarf exakt ermitteln und im Hinblick auf die verfügbaren Ressourcen "testen" kann (siehe 5.7.2).

Für einen Rechner R ergibt sich folgender Wertebereich, der aufgrund der Ressourcenverfügbarkeit auf R und der Formatbeschränkungen im ganzen Flußgraphen unterstützt werden kann:

$$WB(R) = \{\ e \in FWB : e <= maxW(R)\ \}\ .$$

Der maximale Wert, der durch alle Rechner i (i=1..r) aufgrund ihrer jeweiligen Ressourcen und der Formatbeschränkungen unterstützt werden kann, ist dann gegeben durch:

$$Q = \max(\ WB(R1) \cap \ldots \cap WB(Rr)\ )$$
$$= \{\ e \in FWB : e <= maxW\ \}\ ,|$$

wobei maxW = min(maxWB(R1), ... , maxWB(Rr)) .

Um zu zeigen, daß Q durch NRP-LK tatsächlich eingestellt wird, sei derjenige Rechner Rk betrachtet, dessen LK-Wert minimal ist, d. h. genau maxW entspricht. Für die Elemente auf Rk werden in der ersten Phase aus allen Wertebereichen die Werte ausgeschlossen, die über maxW liegen. NRP-LK kann daher in den weiteren Phasen nicht einen Wert an den Ports einstellen, der größer als maxW ist.

Auf der anderen Seite ist der Wert maxW am Ende der ersten Phase in den Wertebereichen aller Elemente (aller Rechner) enthalten. Somit ist eine Einstellung dieses Wertes in den restlichen Phasen möglich, sofern keine Ressourcenkonflikte im Netzwerk auftreten. In den Phasen zwei, drei und vier treten nämlich unter dieser Annahme überhaupt keine Ressourcenkonflikte auf. Entsprechend der Fähigkeit von NRP, für solch einen Fall die bestmöglichen Werte an den Senken einzustellen (siehe 5.3.5), ist NRP-LK in der Lage, an den Senken den optimalen Wert einzustellen. Dieser muß zum einen in FWB liegen und nicht höher als maxW sein. Er entspricht somit genau dem oben berechneten Wert Q.

Unter der Voraussetzung, daß im Netzwerk keine Ressourcenkonflikte auftreten, stellt NRP-LK somit sicher, daß an den Ports des Flußgraphen der bestmögliche Wert eingestellt wird.

Der Aufbau von Netzwerkverbindungen wird durch NRP-LK erst in der zweiten Phase vorgenommen. Für jede Verbindung wird somit erreicht, daß genau ein Aufbau- und evtl. ein Relaxierungsvorgang (in der letzten Phase) stattfinden muß. Würde die lokale Koordination der ersten

Phase die wiederholte Relaxierung für eine Netzwerkverbindung beinhalten, so müßten pro Verbindung bis zu k-1 Relaxierungsvorgänge stattfinden, falls der LK-Wert k Stufen durchlaufen kann. Um diesen zusätzlichen Aufwand zu vermeiden, wurde die beschriebene Vorgehensweise gewählt.

Auf der anderen Seite können benötigte Netzwerkverbindungen aufgebaut werden, bevor NRP-LK ausgeführt wird. Dies wird z. B. durch NRP erreicht, sofern während der *Reserve*-Phase von NRP nur Link-Objekte Ressourcen im Netzwerk reservieren. Wird anschließend NRP-LK durchgeführt, so stellt jedes Link-Objekt sicher, daß es nur einen Wertebereich an den LK meldet, der die Transportdienstgüte für die aufgebaute Verbindung widerspiegelt. Der Ablauf von NRP-LK ist ansonsten unverändert. Da spätere Ressourcenkonflikte im Netzwerk somit ausgeschlossen sind, ist NRP-LK in der Lage für die Ports des Flußgraphen den bestmöglichen Wert abzuleiten, der aufgrund begrenzter Endsystemressourcen UND der im voraus aufgebauten Netzwerkverbindungen möglich ist.

Obige Ergebnisse sind für den Fall gültig, daß NRP-LK auf Flußgraphen anwendet wird, die keine variablen Filter enthalten. Auf der anderen Seite sind die lokale Koordination und NRP-LK insgesamt in gleicher Weise anwendbar, falls variable Filter gegeben sind. Dabei sind jedoch wesentliche Einschränkungen zu beachten. Zum einen muß die Berechnung des Ressourcenbedarfs für einen variablen Filter im Sinne einer Maximierung gemäß 5.3.4 erfolgen. Die Folge hiervon ist, daß für einen LK-Wert W ein Bedarf ermittelt wird, der größer sein kann als derjenige, der bei der Einstellung der höchstmöglichen Werte (unterhalb von oder gleich W) an den Ports des Filters auftritt (siehe 5.3.4). Dies wiederum bedeutet, daß der durch den LK ermittelte bestmögliche LK-Wert nicht unbedingt die bestmögliche obere Grenze für die Port-Werte lokaler Elemente darstellt.

Zum anderen ermöglicht die Betrachtung variabler Filter eine entkoppelte Einstellung von Port-Werte in unterschiedlichen Teilen des Flußgraphen, insbesondere auch an den Senken-Ports. Eine Aussage über die Güte von NRP-LK muß sich daher nicht nur auf einen einzigen Wert beziehen (der an allen Ports gleichermaßen eingestellt wird), sondern auf die Gesamtheit der Werte, die für die Senken-Ports bestimmt wird. Die Formulierung einer solchen Aussage steht gegenwärtig aus.[25]

5.8 Implementierung

Das NRP-Protokoll wurde als Teil der CINEMA-Systemplattform realisiert. Die zugrundeliegende Systemumgebung umfaßte dabei eine Anzahl von SunSparc10-Rechnern, die unter Solaris 2.4 betrieben wurden und über ein Ethernet-Netzwerk miteinander verbunden waren. Zur Verfügung stand ferner eine Implementierung des RSVP-Protokolls [ZDES93], welches zum Aufbau von Netzwerkverbindungen benutzt wurde.

Die verfügbare Implementierung von NRP stellt eine vollständige Realisierung eines Reservierungsprotokolls auf der Anwendungsebene dar (siehe 4.1). Um ihre wesentlichen Eigenschaften zu verdeutlichen, wird im folgenden die Implementationsstruktur vorgestellt, gefolgt von Meßergebnissen und einer zusammenfassenden Leistungsbewertung.

5.8.1 Implementierungsstruktur

Entsprechend der in 3.2 beschriebenen Struktur des Ressourcenmanagements wird NRP durch sogenannte NRP-Agenten realisiert, wobei auf jedem Rechner ein NRP-Agent gegeben ist (Abb. 33). Ein NRP-Agent ist zum einen für die Interaktion mit den Komponenten und Link-Objekten zuständig, die auf dem Rechner instanziiert sind. Mit NRP-Agenten verbundener Rechner tauscht er ferner AFS aus, um die Ressourcenreservierung global zu koordinieren.

Der Aufbau einer Session wird in CINEMA durch eine Sessionkontrolle koordiniert. Letztere ist Teil des CINEMA-Ressourcenmanagements und realisiert die Schnittstelle, über die ein Klient den Aufbau einer Session auslösen kann (siehe 2.5). Sie ist in CINEMA durch eine zentrale Einheit realisiert und kann auf einem beliebigen Rechner untergebracht sein. Fordert ein Klient den Aufbau einer Session an, so sendet die Sessionkontrolle entsprechende Nachrichten an alle NRP-Agenten, die für Rechner zuständig sind, auf denen mindestens eine Quellkomponente instanziiert ist. Über den Aufruf dieser Komponenten stoßen die NRP-Agenten das NRP-Protokoll an (siehe 5.3). Am Ende der NRP-Phasen sind die NRP-Agenten, die mit Senken interagie-

[25] Ohne Beweis wird angemerkt, daß eine solche Aussage für Flußgraphen möglich ist, die variable Filter enthalten, dafür aber keine Formatbeschränkungen für Komponenten-Ports zulassen. In diesem Fall ermöglicht die lokale Koordination die Einstellung von Werten an den Senken des Flußgraphen, die nicht alle um eine Stufe erhöht werden können, ohne daß es auf irgendeinem Rechner zu einem Ressourcenkonflikt kommt.

ren, dafür verantwortlich, daß sie den Erfolg oder Mißerfolg des Session-Aufbaus an die Sessionkontrolle mitteilen.

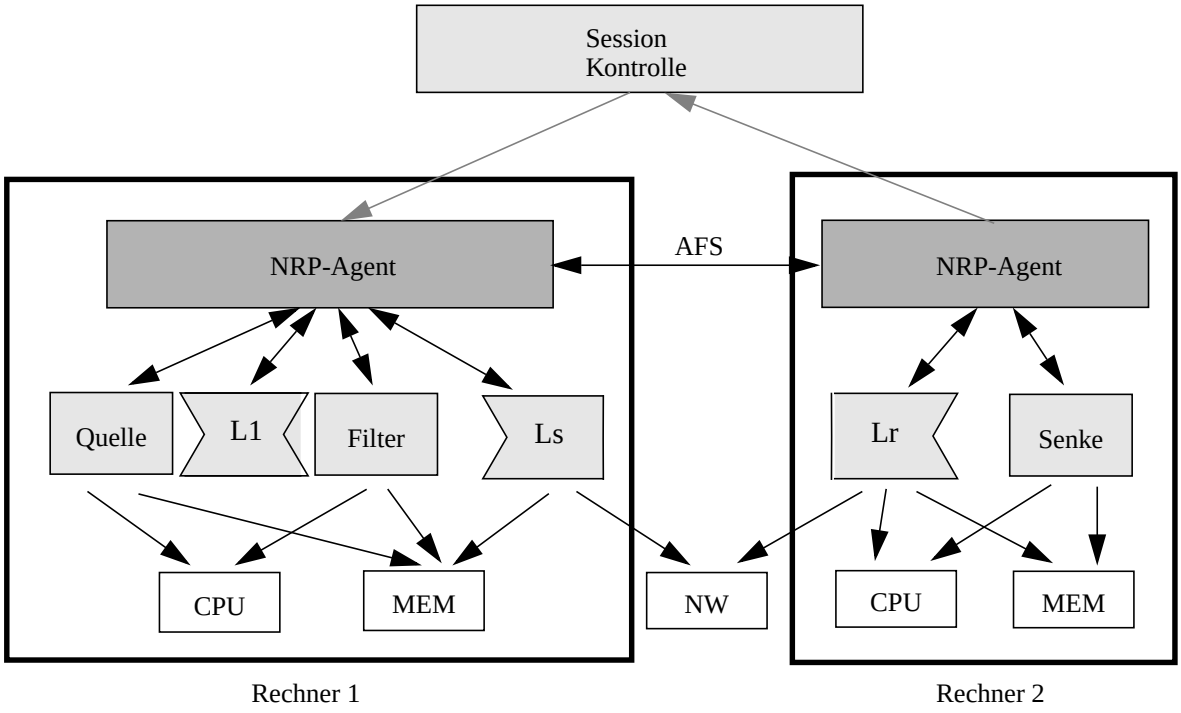


Abb. 33: *Implementierungsstruktur*

Die Implementierung der NRP-Agenten wurde in Anlehnung an die Struktur der Protokollagenten objektorientiert durchgeführt [Sala97]. Auf jedem Endsystem enthält der NRP-Agent die PA-Objekte der lokalen Flußgraphenelemente. Jedem Flußgraphenelement entspricht im Agenten ein PA-Objekt, das durch das Element während der Konfigurationsphase von CINEMA instanziiert wird. Hierbei werden jedem PA die Bezeichner der PA-Objekte mitgeteilt, mit denen er entsprechend der Flußgraphentopologie im Verlauf von NRP kommunizieren muß, wobei ein Bezeichner einen PA sowohl lokal als auch global (über einen Rechnernamen) identifiziert.

Jeder NRP-Agent enthält eine AFS-Warteschlange und einen AFS-Verteiler (Abb. 34). Generiert ein Element ein AFS, so wird es über dessen PA in die AFS-Warteschlange gestellt, wobei der Ziel-PA mitangegeben wird. Der Verteiler liest ein AFS aus der Warteschlange aus und ruft damit den Ziel-PA auf. Der PA stellt anhand der AFS die Protokollphase fest und aktiviert die

entsprechende Methode (*Reserve*, etc.) des zugeordneten Elements. Stellt der Verteiler fest, daß ein AFS an einen entfernten PA zu liefern ist, so sendet es das AFS an den Verteiler (bzw. die AFS-Warteschlange) des entsprechenden Rechners.

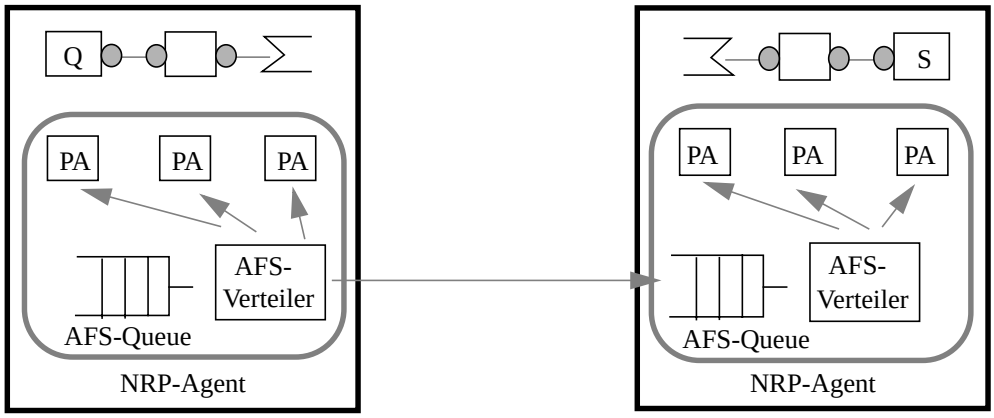


Abb. 34: *Struktur der NRP-Agenten*

Obige Struktur hat den Vorteil, daß die NRP-Agenten keine zentrale topologieorientierte Information bereithalten müssen. Der Zusammenhang zwischen den PA ist vielmehr durch die Verbindung jedes PA mit seinen Nachbarn gegeben, wobei dieser Zusammenhang durch den Konfigurationsdienst von CINEMA direkt aus der Topologie abgeleitet werden kann. Der NRP-Agent eines Rechners läuft in einem einzigen Thread ab, so daß beim Verteilen der AFS keine Thread-Wechsel nötig sind und ein entsprechender Zeitverlust vermieden werden kann.

5.8.2 Leistungsbewertung

Die Leistungsfähigkeit von NRP ist über die Zeit meßbar, die für einen Session-Aufbau benötigt wird. Um Aussagen zum Verhalten von NRP in vielfältigen Szenarien zu erhalten, wurden verschiedene Flußgraphen (mit bis zu 60 Komponenten in unterschiedlichen Konfigurationen) getestet. Die hierbei erzielten Meßergebnisse werden nachfolgend vorgestellt.

Die Dauer eines Session-Aufbaus setzt sich aus Verzögerungen zusammen, die durch verschiedene Teile einer multimedialen Systemplattform bedingt werden. Zum einen sind darin die

Anteile enthalten, die durch das NRP-Protokoll eingeführt werden:
(a) Zeit zum Verteilen der AFS auf Endsystemen
(b) Zeit zur Propagierung der AFS zwischen Rechnern
(c) Zeit zur AFS-Manipulation in Komponenten und Link-Objekten.

Jeder dieser Anteile deckt Verzögerungen ab, die in den drei Phasen von NRP entstehen. Insbesondere umfaßt die Verzögerung aufgrund der AFS-Manipulation Zeiten zum Abgleich der Wertebereiche erhaltener AFS mit Formatbeschränkungen, zur Umrechnung von Wertebereichen zwischen Eingangs- und Ausgangs-Ports sowie zur Ermittlung des Ressourcenbedarfs zu gegebenen Parameterwerten an Ports von Komponenten bzw. Link-Objekten (siehe 5.3.3).

Darüber hinaus entstehen Verzögerungen durch die Reservierung bzw. Relaxierung von Ressourcen auf den Endsystemen sowie im Netzwerk. So benötigt eine „typische“ CINEMA-Komponente bei gegebenem Ressourcenbedarf Interaktionen zur:
(d) Reservierung von CPU-Zeit durch den CPU-Manager
(e) Belegung von Puffer durch den Speichermanager.

Ein Link-Objekt interagiert entsprechend mit einem Netzwerkprotokoll, welches eine Ressourcenreservierung im Netzwerk bewirkt. Dementsprechend ist die Verzögerung zu berücksichtigen für:
(f) Aufbau und Anpassung von Netzwerkverbindungen .

Um den Einfluß obiger Beiträge zur Gesamtverzögerung zu ermitteln, wurden zunächst Messungen im Hinblick auf jede der erwähnten Verzögerungsquellen isoliert durchgeführt. In einem zweiten Schritt wurden die Einzelanteile aggregiert und mit Messungen verglichen, die die Gesamtverzögerung für den Aufbau einer Session betrafen.

Zeitaufwand für Teilschritte des Session-Aufbaus

Um den Aufwand für die *lokale Verteilung von AFS* auf einem Endsystem abzuschätzen, wurden Flußgraphen mit bis zu 20 linear verbundenen Komponenten betrachtet, die auf einem Rechner instanziiert waren. Gemessen wurde die Zeit für den Aufbau einer Session, wobei Komponenten keinerlei Berechnungen durchführten, wenn sie entsprechend der Protokollpha-

sen über ihre Methoden aufgerufen wurden. Abhängig von der Größe des Flußgraphen wurden folgende Zeiten gemessen:

Anzahl der Komponenten	Gemessene Dauer (3 Phasen)
4	2 msec
8	3 msec
16	7 msec
20	9 msec

Tabelle 1 : Zeitaufwand für lokale AFS-Verteilung

Die *entfernte Propagierung von AFS* wurde zwischen zwei Komponenten gemessen, die auf zwei Rechnern instanziiert waren. Der entsprechende Wert betrug 2-2.5 msec. Die Gültigkeit dieses Wertes wurde auch anhand längerer Ketten von Komponenten festgestellt, wobei erneut jede Komponente auf einem anderen Rechner instanziiert war. Abhängig von der Zahl der Komponenten, d. h. der Zahl der AFS-Propagierungen zwischen zwei Rechnern (in drei Protokollphasen), wurde folgendes gemessen:

Anzahl der Rechner-”Hops”	Gemessene Dauer (3 Phasen)
1	8 msec
2	14 msec
3	21 msec
4	29 msec

Tabelle 2 : Zeitaufwand für entfernte AFS-Verteilung

Die Dauer der *AFS-Manipulation* in Komponenten und Link-Objekten wurde für verschiedene Komponententypen (Quellen, Filter, Mischer bzw. Senken) ermittelt. Dabei wurde angenommen, daß fünf Parameterwerte in den Wertebereichen der AFS bzw. der Formatbeschränkungen

gegeben sind. Dementsprechend wurde fünf mal ein Ressourcenbedarf aus der Ressourcenbedarfstabelle abgeleitet. Folgende Werte wurden ermittelt:

Komponenten-typ	Zeitaufwand Phase 1	Zeitaufwand Phase 2	Zeitaufwand Phase 3	Zeitaufwand Gesamt
Quelle	1 msec	0.5 msec	0.5 msec	2 msec
Senke	“	“	“	“
Filter	“	“	“	“
Mischer1	“	“	“	“
Mischer4	2 msec	2.5 msec	2 msec	6.5 msec
Mischer8	3.5 msec	4 msec	3 msec	10.5 msec
Mischer10	4.5 msec	6 msec	4 msec	14.5 msec
Mischer20	9.5 msec	11 msec	9.5 msec	30 msec

Tabelle 3 : Zeitaufwand für AFS-Manipulation in Komponenten

Bei obigen Messungen ist zu beachten, daß der Aufwand für einen Mischer sich in Abhängigkeit von der Zahl der betrachteten Eingangs-Ports erhöht, da letztere unmittelbar die Zahl der zu berücksichtigenden Formatbeschränkungen und Mischrelationen bestimmt.

Der Aufwand *lokaler Ressourcenmanager* zur Reservierung von Ressourcen auf einem Endsystem wurde in Bezug auf die CPU sowie den Hauptspeicher wie folgt ermittelt:

Ressourcenmanager	Gemessene Dauer Reservierung	Gemessene Dauer Relaxierung
CPU-Manager	0.16 msec	< 0.16 msec
Speichermanager	0.03 msec	< 0.03 msec

Tabelle 4 : Zeitaufwand zur Belegung von Ressourcen

Obige Werte gelten für einen einzigen Reservierungsaufruf. Werden wiederholte Reservierungen, z. B. für abnehmende Parameterwerte, versucht, so multipliziert sich der Aufwand entsprechend. Bei fünf durchlaufenen Parameterwerten liegt der Aufwand zur Reservierung aber immer noch im Bereich von ca. 1 msec.

Für den Aufbau von *Netzwerkverbindungen* in entfernten Links wurde in CINEMA eine Implementierung des RSVP-Protokolls eingesetzt. Messungen haben folgende Zeiten für den Aufbau einer Verbindung ergeben.

	Gemessene Dauer Verbindungsaufbau	Gemessene Dauer Relaxierung
RSVP-Verbindung	10-13 msec	10-13 msec

Tabelle 5 : Zeitaufwand zum Aufbau von Netzwerkverbindungen

Die Zeit für die Relaxierung einer Reservierung bleibt gleich, da hierfür RSVP denselben Mechanismus wie für die Reservierung verwendet. Der maximale, angegebene Wert trat bei dem Aufbau einer RSVP-Verbindung auf. Wurden dagegen viele Verbindungen hintereinander aufgebaut, so wurde für die Aufbaudauer pro Verbindung ein abnehmender Wert festgestellt. So wurden oberhalb von 8 Verbindungen Aufbauzeiten von ca. 10 msec gemessen.

Obige Messungen zeigen einen unterschiedlichen Einfluß der einzelnen Mechanismen auf die Gesamtverzögerung. So bewirkt insbesondere die entfernte Propagierung von AFS einen wesentlich höheren Zeitaufwand als die lokale Verteilung der AFS, sofern pro Rechner wenige Komponenten instanziiert sind. Ebenso liegt der Zeitaufwand zur Reservierung lokaler Ressourcen wesentlich niedriger als derjenige, der für den Aufbau einer Netztwerkverbindung erforderlich ist.

Zeitaufwand zum Aufbau vollständiger Sessions

Um die Gesamtdauer für den Aufbau einer Session mit Hilfe von NRP zu ermitteln, wurden Messungen unter Einbeziehung der oben erwähnten Teilschritte durchgeführt. Insbesondere wurde der Aufbau von RSVP-Verbindungen im Rahmen des Session-Aufbaus mitberücksichtigt. Dagegen konnte der Einfluß der Reservierung lokaler Ressourcen nicht erfaßt werden, da zu dem Zeitpunkt der Messungen der CPU- bzw. der Speichermanager nicht in die CINEMA-Plattform integriert waren. Die erzielten Meßwerte für den Session-Aufbau sowie der für die Ressourcenmanager isoliert ermittelte Aufwand lassen jedoch den Schluß zu, daß hierdurch die erzielten Ergebnisse nicht wesentlich beeinflußt werden (siehe unten).

Abb. 35: *Meßszenario 1*

Für die Messungen wurden drei Szenarien betrachtet. Im *ersten Szenario* wurde die Session-Aufbaudauer für den in Abb. 35 dargestellten Flußgraphen evaluiert.

Session-Aufbau	Erwartete Dauer	Gemessene Dauer
Szenario 1	38 msec	44 msec

Tabelle 6 : Dauer des Session-Aufbaus für Szenario 1

Obige Tabelle enthält zum einen die mit Hilfe der zuvor angegebenen Einzelbeiträge berechnete Dauer des Session-Aufbaus. Hierzu wird folgende Formel verwendet:

$$V_{erw} = 3 \text{ x AFS-Prop} + \text{AFS-M(Q)} + \text{AFS-M(S)} + \text{RSVP-Reserve} + \text{RSVP-Relax} + 4 \text{ x CPU-Reserve} + 4 \text{ x Mem-Reserve} + \text{AFS-LocalProp}$$

Dabei kennzeichnet AFS-Prop die Dauer einer (einfachen) Propagierung eines AFS zwischen zwei Rechnern, AFS-M() den Aufwand für die Manipulation der AFS in der Quelle bzw. der Senke, während RSVP-Reserve bzw. RSVP-Relax die Dauer für den Aufbau einer Verbindung bzw. deren Anpassung angeben. Enthalten sind ferner die Anteile, die zur Belegung von CPU-Zeit sowie Speicher nötig sind, wobei eine vierfache Berücksichtigung wegen der zwei Komponenten sowie der zwei Link-Objekte erforderlich ist. Schließlich ist ein Term (AFS-LocalProp) zur Erfassung der Zeit zur lokalen Verteilung der AFS enthalten.

Im berechneten Wert der Tabelle sind die Anteile CPU-Reserve sowie Mem-Reserve nicht berücksichtigt, da sie auch bei der Messung nicht anfielen. Da sie, entsprechend der Messungen für den CPU- bzw. den Speichemanager, Zeiten von maximal 4*1 msec bedingt hätten, beeinflußt dies nur unwesentlich die aufgezeigten Ergebnisse.

Der Vergleich des gemessenen Wertes von 44 msec mit dem berechneten Wert (38 msec) läßt eine gute Übereinstimmung erkennen. Ferner wird deutlich, daß der Session-Aufbau insgesamt

wenig Zeit erfordert und der Hauptteil dieser Zeit (26 msec) durch den Aufbau bzw. die Anpassung der RSVP-Verbindung bedingt wird.

Das *zweite Meßszenario* bezieht sich auf eine lineare Kette von Komponenten, in der zwischen einer Quelle und einer Senke bis zu 16 zwischengelagerte Filter gegeben sein können (Abb. 36). Die Komponenten wurden maximal verteilt, d. h. je zwei benachbarte Komponenten waren auf unterschiedlichen Rechnern untergebracht.[26]

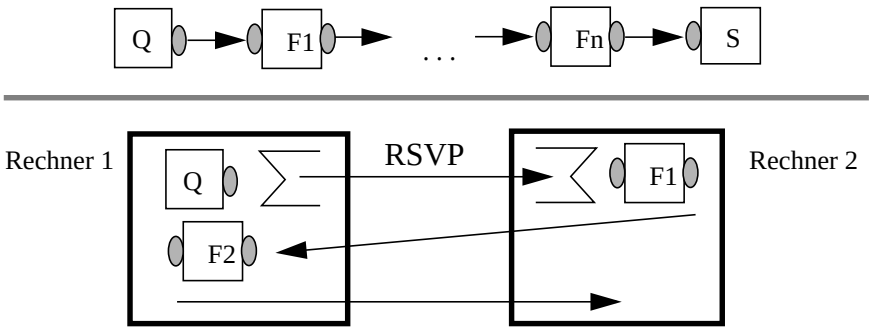


Abb. 36: *Meßszenario 2*

Die erwartete Verzögerung für dieses Szenario läßt sich in Abhängigkeit von der Zahl der zwischengeschalteten Filter wie folgt berechnen:

$$V_{erw} = 3 \times (n+1) \times \text{AFS-Prop} + \text{AFS-M(Q)} + \text{AFS-M(S)} + n \times \text{AFS-M(F)} +$$
$$+ (n+1) \times \text{RSVP-Reserve} + (n+1) \times \text{RSVP-Relax} +$$
$$+ (3 \times n + 4) \times \text{CPU-Reserve} + (3 \times n + 4) \times \text{Mem-Reserve} + \text{AFS-LocalProp} .$$

Die Faktoren vor CPU-Reserve bzw. Mem-Reserve berücksichtigen dabei, daß n+2 Komponenten sowie 2 x n + 2 Link-Objekte gegeben sind. Analog zum ersten Szenario wurden die Werte für CPU-Reserve sowie Mem-Reserve vernachlässigt, da sie bei den Messungen nicht anfielen.

[26] Real wurden lediglich zwei Rechner verwendet. Gegenüber dem Fall, daß (n+2) Rechner verwendet werden, hatte dies keinen Einfluß, da zu jedem Zeitpunkt auf einem Rechner höchstens ein AFS bearbeitet wurde.

Eine Berechnung dieser Anteile ergibt, daß sie in allen Fällen einen Beitrag von weniger als 10 Prozent zur Gesamtverzögerung geleistet hätten.

Session-Aufbau	Erwartete Dauer	Gemessene Dauer
n = 1	73 msec	74 msec
2	107 msec	109 msec
4	179 msec	171 msec
8	321 msec	353 msec
16	605 msec	690 msec

Tabelle 7 : Dauer des Session-Aufbaus für Szenario 2

Obige Tabelle zeigt eine gute Übereinstimmung zwischen den erwarteten und den tatsächlich gemessenen Werten, wobei die maximale Abweichung weniger als 15 Prozent beträgt und lediglich für das umfangreichste Szenario mit 16 Filtern auftrat.

Das *dritte Szenario* wurde so gewählt, daß der Einfluß von Mischfunktionen sowie eines Multicast-Links berücksichtigt werden konnte. Der dabei betrachtete Flußgraph bestand aus n Quellen, die mit den n Eingangs-Ports eines Mischers verbunden waren, sowie n Senken, die über einen Multicast-Link an den Ausgang des Mischers angeschlossen waren. Quellen, Mischer sowie Senken waren auf jeweils einem anderen Rechner instanziiert.

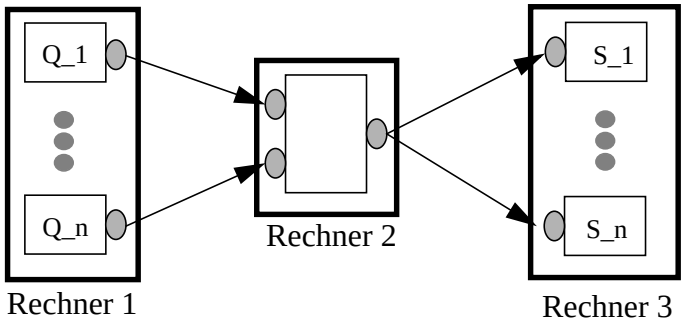


Abb. 37: *Meßszenario 3*

Im Unterschied zu den vorangegangenen Szenarien konnte in diesem Fall lediglich eine Obergrenze als Schätzung berechnet werden, da einige der Teilschritte bei dem Session-Aufbau

nebenläufig stattfinden konnten. Hierzu gehörte insbesondere die Propagierung der AFS zwischen Rechner 2 und 3 in allen drei Protokollphasen. Folgende Formel berücksichtigt alle auftretenden Verzögerungsanteile, als seien sie serialisiert ausgeführt:

$$V_{OG} = 6 \times N \times \text{AFS-Prop} + N \times \text{AFS-M(Q)} + \text{AFS-M(Mischer\_mit\_N\_Eingängen)} + N \times \text{AFS-M(S)} + 2 \times N \times \text{RSVP-Reserve} + 2 \times N \times \text{RSVP-Relax} + (5 \times N + 2) \times \text{CPU-Reserve} + (5 \times N + 2) \times \text{Mem-Reserve} + \text{AFS-LocalProp.}$$

Session-Aufbau	Berechnete Dauer	Gemessene Dauer
N = 1	73 msec	74 msec
4	290 msec	293 msec
8	482 msec	370 msec
10	605 msec	480 msec
20	1210 msec	630 msec

Tabelle 8 : Dauer des Session-Aufbaus für Szenario 3

Bei der Berechnung wurden die Anteile CPU-Reserve bzw. Mem-Reserve erneut vernachlässigt. Wie in den früheren Fällen ergibt eine Berechnung dieser Anteile, daß sie weniger als 10 Prozent der Gesamtverzögerung ausgemacht hätten, so daß sie die aufgezeigten Ergebnisse qualitativ nicht wesentlich beeinflussen.

Der Vergleich berechneter und gemessener Werte zeigt eine gute Übereinstimmung bis zu einem Wert N von vier. Für höhere Werte ergibt sich eine Diskrepanz, die teilweise durch die parallele Ausführung der AFS-Propagierung zwischen Rechner 2 und 3 erklärt werden kann. Wird dieser Anteil in der Formel vernachlässigt, so reduziert sich der berechnete Wert z. B. für N=8 um 60 msec (= 3 x 8 x AFS_Prop), für N = 10 um 75 msec sowie für N = 20 um 150 msec. Für N = 8 und N = 10 erreichen so berechnete Werte (412 msec bzw. 530 msec) eine ähnliche Größenordnung wie die gemessenen Werte. Für N = 20 bleibt dagegen eine erhebliche Abweichung (1060 msec gegenüber 630 msec) bestehen, die im Rahmen der Messungen nicht genauer geklärt werden konnte.

5.8.3 Diskussion

Die vorgestellten Ergebnisse zeigen, daß die Sessionaufbauzeiten, die durch NRP impliziert werden, für eine breite Klasse von Flußgraphen gering sind. Dies gilt sowohl für “lange” Flußgraphen, die bis zu 18 hintereinander geschaltete Komponenten enthalten, als auch für Mischerszenarien, in denen bis zu 20 Quellen und ebensoviele Senken über einen Mischer miteinander verbunden sind.

In allen Fällen war die durch die lokale Verbreitung der AFS auf Endsystemen bedingte Verzögerung vernachlässigbar. Die Propagierung von AFS zwischen den Rechnern fiel wesentlich stärker ins Gewicht und bewirkte in den betrachteten Szenarien bis zu 25 Prozent der festgestellten Verzögerung. Die Anteile für die AFS-Manipulation in den Komponenten und Link-Objekten waren mit weniger als 10 Prozent der Gesamtverzögerung vergleichsweise unerheblich. Den Hauptanteil an der Gesamtverzögerung machte jedoch die Verwendung von RSVP-Verbindungen aus, die pro Verbindung einen Aufwand im Bereich 20 bis 26 msec bedingt hat.

Insgesamt zeigen die Messungen, daß NRP für unterschiedlichste Flußgraphen und Flußgraphengrößen sehr gut skalierbar ist. Dadurch ist NRP für eine Vielzahl von Szenarien anwendbar einschließlich Telekonferenzen und Informationsverteilung sowie komplexer Medienverarbeitungsszenarien (z. B. Bildverarbeitung, “Virtual Reality”, “TV-Studio Processing”). Für Szenarien der Informationsverteilung an viele Hunderte von Teilnehmern ist NRP dagegen nicht skalierbar. Für diese Fälle ist ein Ansatz vorzuziehen, wie er durch RSVP ([ZDE93]) ermöglicht wird. Allerdings wird hierdurch die wesentliche Fähigkeit von NRP aufgegeben, Sessions für Flußgraphen aufzubauen, die zwischengelagerte Verarbeitung in Form von Filtern und Mischern in flexibel konfigurierbaren Flußgraphen verwenden.

Die Sessionaufbauzeit für die abgeleiteten Protokolle (NRP-Min, NRP-LK) läßt sich leicht anhand der entsprechenden Zeit für NRP abschätzen, da sie proportional zu der Zahl benötigter Protokollphasen ist. Für das NRP-Min Protokoll ist sichergestellt, daß jedes Element genau einmal eine Ressourcenreservierung versucht, so daß die Zahl möglicher Parameterwerte in den AFS keine Rolle spielt. Das NRP-LK-Protokoll verhält sich in dieser Hinsicht wie NRP, da es (im schlimmsten Fall) Ressourcenbelegungen für alle möglichen Parameterwerte durchprobieren muß.

5.9 Zusammenfassung

Für den Aufbau von Sessions mit garantierter Dienstgüte werden Protokolle zur Dienstgüteaushandlung und Ressourcenreservierung auf der Anwendungsebene benötigt. Diese müssen Dienstgüteanforderungen mit den funktionalen Einschränkungen eines Flußgraphen sowie der gegebenen Ressourcenverfügbarkeit im Netzwerk und auf Endsystemen in Einklang bringen.

Das NRP-Protokoll, das in diesem Kapitel vorgestellt wurde, erfüllt diese Anforderungen für Flußgraphen, die einem allgemeinen Zweizonenmodell entsprechen und somit die meisten Graphtypen abdecken, die in Multimedia-Anwendungen vorkommen. NRP ist unabhängig von der Beschaffenheit eines Flußgraphen. Es unterstützt insbesondere zwischengelagerte Verarbeitung mit Hilfe unterschiedlicher Komponententypen und erlaubt die Verankerung dieser Verarbeitung auf zwischengelagerten Rechnern. NRP ist unabhängig von der Art verwendeter Medientypen.

Anders als bisherige Ansätze der Anwendungsebene erlaubt NRP die unabhängige Spezifikation von Dienstgüte an den einzelnen Senken und unterstützt eine unabhängige Dienstgüteeinstellung für sie, indem es variable Filter in die Aushandlung einschließt. NRP ist unabhängig von spezifischen Reservierungsprotokollen der Transport- bzw. Netzwerkebene. Durch die Bereitstellung von Link-Objekten kann NRP mit beliebigen solchen Protokollen integriert ausgeführt werden.

NRP bestimmt die Einstellung sowohl der medienspezifischen Dienstgüte als auch der “Ende-zu-Ende”-Verzögerung. Letztere wird zwischen Quellen und Senken durch Erfassen sämtlicher Verzögerungsbeiträge berechnet, die durch Verarbeitung oder Kommunikation entstehen. Für die medienspezifische Dienstgüteeinstellung ist NRP optimal, sofern bei der Belegung von Ressourcen, die durch verschiedene Teile des Flußgraphen benutzt werden, keine Konflikte entstehen.

Zur optimierten Behandlung von Ressourcenkonflikten kann NRP in zwei Richtungen erweitert werden. Das NRP-Min Protokoll sorgt durch eine Minimierung der einzustellenden Dienstgüte dafür, daß ein Session-Aufbau mit einem verringerten Ressourcenbedarf durchgeführt wird. Es ist auf Flußgraphen mit allen erwähnten Randbedingungen anwendbar und wird sinnvoller-

weise im Anschluß an einen Versuch verwendet, eine Session für eine nichtminimale Dienstgüte mit Hilfe von NRP aufzubauen.

Das NRP-LK Protokoll schließt eine lokale Koordination der Ressourcenvergabe in die Aushandlung mit ein. Es liefert optimale Sessions, unter der Voraussetzung, daß keine Ressourcenkonflikte im Netzwerk auftreten. Wie gezeigt wurde, ist die lokale Koordination auch dann von Nutzen, wenn diese Annahme nicht zutrifft, sofern benötigte Netzwerkerbindungen (z. B. mit Hilfe von NRP) vor der Ausführung von NRP-LK aufgebaut werden. NRP-LK hat (gegenwärtig) den Nachteil, daß Aussagen über seine Verfahrensgüte nur möglich sind, sofern es auf Flußgraphen ohne variable Filtern angewandt wird.

Die Implementierung von NRP zeigt, daß ein Session-Aufbau mit Hilfe der erwähnten Protokolle eine geringe Zeit in Anspruch nimmt. Diese Aussage ist in einem weiten Bereich unterschiedlicher Flußgraphen gültig. Die Protokolle sind daher für eine Vielzahl von Szenarien multimedialer Kommunikation und Verarbeitung anwendbar.

6 Task-Plazierung für verteilte Multimedia-Anwendungen

Die Lokation von Komponenten verteilter Multimedia-Anwendungen muß nicht im voraus gegeben sein. Eine Wahl der Lokation kann vielmehr davon abhängen, auf welchen Rechnern Komponenten funktional oder durch ausreichende Ressourcen unterstützbar sind und welche Dienstgüte durch eine entsprechende Plazierung erreicht werden kann. Um dies zu berücksichtigen, sind Verfahren nötig, die Eigenschaften sowohl des konfigurierten Flußgraphen als auch des zur Verfügung stehenden verteilten Systems erfassen und eine geeignete Allokation der Komponenten bewirken. In diesem Kapitel werden wesentliche Aspekte beschrieben, die bei dem Entwurf solcher Verfahren von Bedeutung sind und es wird ein Plazierungsproblem für Multimedia-Flußgraphen formuliert, das diese Aspekte geeignet berücksichtigt.

Im Abschnitt 6.1 wird ein Grundmodell eingeführt, das die Elemente und das Ziel der Plazierung von Flußgraphen beschreibt. Im Abschnitt 6.2 werden verschiedene Schritte unterschieden, die zum Erreichen einer dienstgüteoptimierten Plazierung nötig sind. Anschließend wird untersucht, in welcher Form Information über Ressourcenbedarf und Ressourcenverfügbarkeit für ein Plazierungsverfahren bereitgestellt werden kann. Das Plazierungsproblem wird im Abschnitt 6.3 für ein Szenario formuliert, das die Bereitstellung dieser Information zuläßt und die Grundlage für das im Kapitel 7 beschriebene Plazierungsverfahren liefert. Im Abschnitt 6.4 werden verwandte Plazierungsprobleme dargestellt und gegen das formulierte Problem abgegrenzt. Abschließend wird das Plazierungsproblem in eine Taxonomie eingeordnet, die für frühere Plazierungsverfahren entwickelt wurde.

6.1 Grundlagen der Plazierung

6.1.1 Elemente der Plazierung

Die zwei wesentlichen Elemente des Plazierungsproblems sind ein spezifizierter Flußgraph und ein verteiltes System, innerhalb dessen der Flußgraph plaziert werden soll. Der *Flußgraph* ist ein gerichteter, azyklischer Graph, der Komponenten beinhaltet, die über ihre Ports durch Links verbunden sind (siehe Kapitel 2). Das *verteilte System* besteht aus einer Anzahl von Rechnern, die über ein Netzwerk miteinander kommunizieren können.

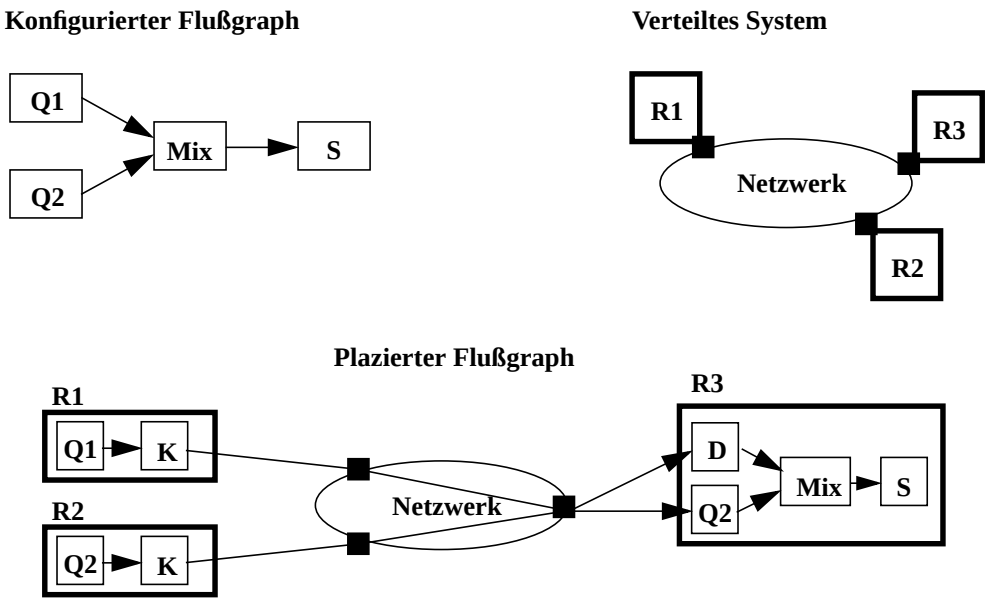


Abb. 38: *Elemente des Plazierungsproblems*

Eine Plazierung des Flußgraphen im verteilten System ist durch die Zuordnung jeder Komponente zu einem Rechner definiert. Der *spezifizierte Flußgraph* ist von dem plazierten Flußgraphen zu unterscheiden. Ersterer ist der durch den Klienten festgelegte Flußgraph, der alle Komponenten enthält, die für die Semantik der Stromverarbeitung wichtig sind. Er legt jedoch nicht die Elemente fest, die für die Kommunikation zwischen verteilten Komponenten nötig sind, da die Orte benötigter Kommunikation vor der Plazierung nicht bekannt sind.

Der *plazierte Flußgraph* legt die Lokation der Komponenten fest und bestimmt, zwischen welchen Komponenten entfernte Links vorzusehen sind. Ferner ergänzt er den Flußgraphen um Komponenten zur Kompression/Dekompression (K/D) von Datenströmen, die vor bzw. nach einem entfernten Link eingefügt werden können. Von einem Klienten wird erwartet, daß er im spezifizierten Flußgraphen (evtl. zu jedem Link) die Art der zu verwendenden Kompression angibt, so daß bei Bedarf die entsprechenden K/D-Komponenten eindeutig bestimmt werden können.

6.1.2 Einschränkungen

Die Plazierbarkeit eines Flußgraphen unterliegt einer Reihe von Einschränkungen. So kann das Fehlen geeigneter Systemunterstützung (z. B. Hardware-Unterstützung zur Bildverarbeitung) oder plattformspezifischer Software-Implementierungen die Verwendung einzelner Komponenten auf einem Rechner verhindern. Um solche, im folgenden als *funktionale Einschränkungen* bezeichnete Aspekte zu berücksichtigen, ist zu jedem Rechner Information über die jeweils unterstützten Komponenten erforderlich.

Die funktionale Unterstützung für eine bestimmte Komponente kann auf verschiedenen Rechnern unterschiedlich sein. Beispielsweise könnte ein Mischer auf einem Rechner eine bestimmte Bildgröße akzeptieren, die auf anderen Rechnern nicht vorgesehen ist. *Formatbeschränkungen*, die die Unterstützung von Parameterwerten durch Komponenten beschreiben (siehe Kapitel 2), sind für eine Komponente erst im Kontext eines Rechners eindeutig. Zu jedem Rechner sind für jede unterstützte Komponente eigene Formatbeschränkungen anzunehmen.

Plazierte Komponenten und Links benötigen Ressourcen auf den jeweiligen Rechnern. Da die Ressourcenverfügbarkeit der Rechner begrenzt ist, muß sichergestellt sein, daß die Plazierung einen nicht zu hohen Ressourcenbedarf bewirkt. Hierzu ist auf jedem Rechner die Möglichkeit anzunehmen, eine Beschreibung *begrenzter Ressourcenkapazitäten* zu erhalten.

Entfernte Links benötigen, zusätzlich zu Ressourcen auf Endsystemen, Kommunikationsressourcen im Netzwerk. Für die Plazierung eines Flußgraphen wird im Rahmen dieser Arbeit angenommen, daß zwischen zwei (miteinander kommunizierenden) Rechnern ein Übertragungskanal mit einer bekannten Gesamtbandbreite vorausgesetzt werden kann. Die Bandbreite kann dabei durch eine beliebige Anzahl von Links verwendet werden, die Komponenten auf den zwei Rechnern miteinander verbinden. Solche Kanäle werden beispielsweise durch das ISDN-System bereitgestellt [Stall88]. Diese haben eine feste Bandbreite und können bei Bedarf aufgebaut werden, wobei letzteres durch ein geeignetes Design des Netzwerks mit hoher Wahrscheinlichkeit garantiert werden kann.

Im Rahmen dieser Arbeit werden ferner lediglich Flußgraphen betrachtet, die keine variablen Filter enthalten. In der Konsequenz ist eine entkoppelte Dienstgüteeinstellung an den Senken

nicht möglich. Entsprechend kann die Dienstgütespezifikation für einen Flußgraphen an einem einzigen, beliebig wählbaren Komponenten-Port gegeben sein.

6.1.3 Ziel der Plazierung

Ein Plazierungsverfahren wird benötigt, um eine Session für einen vom Klienten spezifizierten Flußgraphen innerhalb eines ebenfalls vom Klienten abgegrenzten verteilten Systems aufzubauen. Ziel ist es, für den Flußgraphen eine mögliche Plazierung zu finden, die den funktionalen und ressourcenbedingten Einschränkungen des verteilten Systems nicht widerspricht. Ferner wird eine Plazierung angestrebt, die den „Nutzen“ einer Session möglichst maximiert.

Der Nutzen einer Session wird über die erzielbare Dienstgüte definiert. Im Kapitel 2 wurde eine Dienstgüte beschrieben, die vom Klienten für die Senken eines Flußgraphen spezifiziert werden kann und aus einer Anforderung an medienspezifische und generische Dienstgüte besteht. Dementsprechend sind Plazierungsverfahren denkbar, die sich an der medienspezifischen und/oder der generischen Dienstgüte orientieren. Im Rahmen dieser Arbeit wird ein Verfahren vorgestellt, das eine Optimierung in Bezug auf eine medienspezifische Dienstgüte anstrebt. Für die Plazierung werden keine Vorgaben an die Verzögerung gemacht, da a-priori Information über die erzielbare Netzwerkverzögerung zwischen Rechnern nicht allgemein vorausgesetzt werden kann. Die Berechnung der „Ende-zu-Ende“-Verzögerung für die Senken ist vielmehr erst im Anschluß an die Plazierung durchführbar.[27]

6.2 Eigenschaften des Plazierungsproblems

6.2.1 Schritte der Plazierung

Die Dienstgüteberechnung für Flußgraphen ohne variable Filter ist dadurch vereinfacht, daß die Parametereinstellungen an allen Ports miteinander gekoppelt sind. Wird der Wert an einem Port verändert, so ändert sich der Wert an allen anderen Ports entsprechend. Dies ermöglicht ein Ver-

[27] Ein Klient kann durch die Auswahl der Rechner, die für die Plazierung in Frage kommen, sein „Wissen“ über Verzögerung zwischen Rechnern benutzen. Z. B. könnte er nur Rechner in einem LAN auswählen, um die Verzögerung voraussichtlich gering zu halten.

fahren, das die möglichen Dienstgütewerte an einem ausgewählten „Leit“-Port durchläuft, und zu jedem Wert entsprechende Werte an den anderen Ports einstellt. Verfügt man über ein Plazierungsverfahren, das für einen Flußgraphen mit vorgegebenen Port-Werten die Plazierbarkeit in einem verteilten System entscheiden kann, so kann man die höchste Dienstgüte ermitteln, für die eine Plazierung möglich ist.

Eine Plazierung läßt sich somit durch die Wiederholung zweier Teilschritte durchführen. Für eine Dienstgütevorgabe am Leit-Port werden in einem ersten Schritt die Port-Werte im gesamten Flußgraphen berechnet und es werden für jede Komponente die Rechner ermittelt, die die Komponente mit der gegebenen Einstellung der Port-Werte unterstützen können. In einem zweiten Schritt muß durch die Anwendung eines Plazierungsverfahrens entschieden werden, ob eine Plazierung für den so „vorbereiteten“ Flußgraphen möglich ist.

6.2.2 Einstellung der Port-Werte

Der erste Schritt läßt sich konzeptionell durch das Einsammeln und die Auswertung von Information über den Flußgraphen und über die Unterstützung der Komponenten des Flußgraphen im verteilten System realisieren. Die Information über den Flußgraphen wird vom Klienten geliefert und umfaßt die Struktur des Flußgraphen einschließlich der darin vorkommenden Stromrelationen sowie des geforderten Dienstgütewertebereichs. Die Information muß im allgemeinen Fall an einer zentralen Stelle verfügbar gemacht werden, da der Flußgraph lediglich als Strukturinformation ohne Plazierung vorliegt.

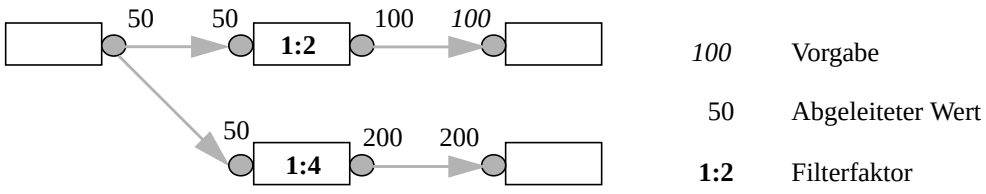


Abb. 39: *Einstellung der Port-Werte*

Mit diesem Wissen kann zu jedem Dienstgütewert die Parametereinstellung für alle Komponenten-Ports vorgenommen werden. Sind die Misch- und Filterfaktoren (siehe 2.3) alle auf eins

gesetzt, so muß an jedem Port derselbe Wert eingestellt werden. Allgemein können die Port-Werte einfach durch Propagierung des vorgegebene Dienstgütewerts unter Berücksichtigung der Filterfaktoren berechnet werden.

Vom Klienten wird eine Menge von Rechnern spezifiziert, die für die Plazierung betrachtet werden sollen. Zu jedem Rechner ist eine Beschreibung der unterstützten Komponenten sowie der dazugehörigen Formatbeschränkungen erforderlich. Da diese Information statisch ist, kann sie in einer beliebigen Informationsquelle, z. B. in einem Verzeichnisdienst, gegeben sein. Für eine gegebene Einstellung der Port-Werte kann somit entschieden werden, welche Rechner für die Plazierung welcher Komponenten in Frage kommen.

6.2.3 Plazierung mit vorgegebener Dienstgüte

Ausgangspunkt für die Bestimmung einer Plazierung ist ein Flußgraph mit eingestellten Port-Werten und der Angabe möglicher Plazierungsrechner zu jeder Komponente. Für den Entwurf eines Plazierungsverfahrens ist von Bedeutung, welche ressourcenbezogene Information für die Berechnung einer Plazierung benutzt werden kann. Aufgrund dieser Information muß das Verfahren eine Plazierung finden, die verfügbare Ressourcenkapazitäten auf den Rechnern nicht übersteigt.

Durch die Vorgabe der Werte an den Ports einer Komponente ist deren Ressourcenbedarf auf jedem Rechner aus der entsprechenden Ressourcenbedarfstabelle ableitbar. Wichtig für ein Plazierungsverfahren ist, daß die Plazierung sich nicht auf die unabhängige Plazierung einzelner Komponenten reduzieren läßt. Erstens bedingt eine Plazierung verbundener Komponenten auf unterschiedlichen Rechnern einen zusätzlichen, kommunikationsbedingten Ressourcenbedarf. Dieser umfaßt zum einen den Bedarf der benötigten Link-Objekte, zum anderen den Bedarf von K/D-Komponenten, die zusätzlich in den Flußgraphen eingefügt werden müssen.

Zweitens kann der Ressourcenbedarf von Komponenten und Link-Objekten, die auf einem Rechner plaziert werden, durch die Art der Thread-Bildung beeinflußt werden. Wie im Abschnitt 3.3.4 beschrieben wurde, kann in CINEMA eine Kette von Komponenten in einem Thread zusammengefaßt werden mit der Folge, daß weniger Speicher benötigt wird. Im Rah-

men anderer Systeme wurden weitere Zusammenfassungen linearer Komponentenketten beschrieben, die auch den CPU-Bedarf absenken können.[28]

Eine Formulierung des Plazierungsproblems sollte sich daher nicht nur auf die Kenntnis des Ressourcenbedarfs einzelner Komponenten auf den verfügbaren Rechnern stützen, sondern eine Thread-orientierte Beschreibung des Ressourcenbedarfs erlauben sowie den kommunikationsbedingten Ressourcenbedarf miteinbeziehen können. Da beides von einer Plazierung der Komponenten abhängt, muß sich eine Formulierung des Problems (zumindest teilweise) auf angenommene Plazierungen eines Flußgraphen stützen. Eine solche Formulierung wird im folgenden im Kontext bestimmter Szenarien vorgestellt.

6.3 Das Host-Satellite-Problem

6.3.1 Das Host-Satellite Modell

Das Host-Satellite-Modell (HS-Modell; [Bokh88]) setzt die Verwendung bestimmter Flußgraphentypen und ein bestimmtes Szenario für die Verteilung der Flußgraphen-Komponenten voraus. Es läßt sternförmige Flußgraphen zu, deren Komponenten innerhalb eines verteilten Systems plaziert werden, das Rechner gemäß eines HS-Schemas enthält.

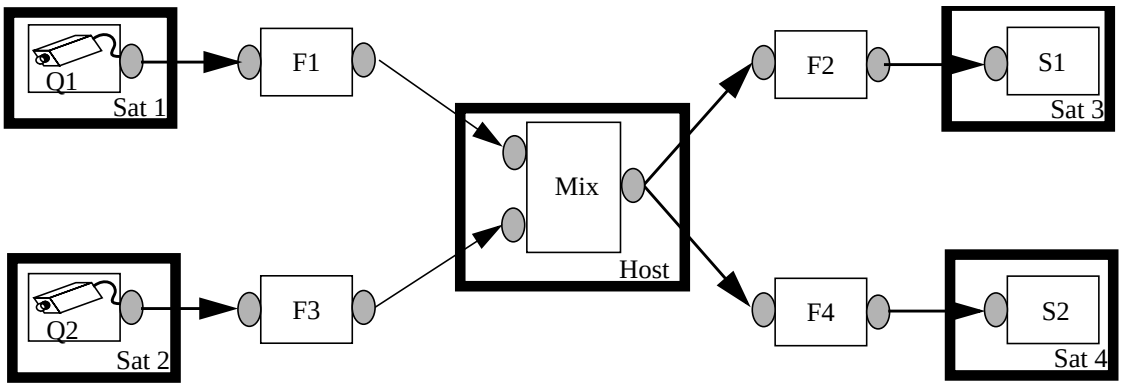


Abb. 40: *Sternförmiger Flußgraph in einem Host-Satellite-System*

[28] In [YMGH96] werden die Verarbeitungsfunktionen aufeinanderfolgender Komponenten in einer Verarbeitungsschleife zusammengezogen, um für jedes Pixel nur einen Speicherzugriff von der CPU aus zu benötigen.

Ein *sternförmiger Flußgraph* setzt sich aus einer beliebigen Zahl linearer Ketten von Komponenten zusammen, die in einer zentralen Komponente zusammenlaufen. Eine Kette stellt dabei eine lineare Sequenz von Komponenten dar, die entweder mit einer Quelle oder einer Senke beginnt und ansonsten nur Filter und Unicast-Links umfaßt (Abb. 40).

Das Modell ist dem Zweizonenmodell des Kapitels 5 ähnlich, da es Komponentenketten zuläßt, die Quellen über zwischengelagerte Komponenten mit einer zentralen Komponente (einem Mischer) verbinden, von der aus (über Komponentenketten) eine Anzahl von Senken anschließbar ist. Das Zweizonenmodell ist jedoch allgemeiner, da es für jede Zone nicht nur eine Menge von Ketten, sondern eine beliebige Baumtopologie ermöglicht. Anders als für das Zweizonenmodell, sind ferner bestimmte, feste Plazierungsvorgaben von Komponenten definiert.

Das Host-Satellite-Modell unterscheidet zwei Arten von Rechnern. Auf *Satelliten* sind die „freien Enden“ der Komponentenketten, d. h. Quellen oder Senken eines Flußgraphen fest verankert. Eine andere Plazierung dieser Komponenten ist nicht möglich. Die ursprüngliche Definition des HS-Modells in [Bokh88] sieht pro Satelliten genau eine Komponentenkette und entsprechend eine Quelle bzw. Senke vor. Dies verallgemeinernd wird im folgenden von einer beliebigen Zahl von Ketten (und damit Quellen und Senken) auf einem Satelliten ausgegangen.

Auf dem *Host* ist die zentrale Komponente des Flußgraphen fest verankert. Jede Kette verbindet eine Quelle oder Senke mit der zentralen Komponente, wobei sie sich über einen Satelliten-Rechner und den Host-Rechner erstrecken kann. Eine Kette kann nicht über mehrere Satelliten verlaufen oder mehrfach zwischen dem Satelliten und dem Host wechseln. Allerdings kann der Host mit einem der Satelliten identisch sein.

Das HS-Modell läßt Freiheitsgrade in Bezug auf die Plazierung der Komponenten einer jeden Kette zu. Jede Kette kann durch einen Schnitt in zwei Teile geteilt werden, von denen einer auf dem Host, der andere auf dem Satelliten plaziert wird (siehe Abb. 41). Wird jede Kette geteilt, so ist eine Plazierung des gesamten Flußgraphen gegeben. Das Finden einer solchen Plazierung mit bestimmten Eigenschaften läßt sich somit auf das Finden entsprechender Kettenschnitte reduzieren.

Im folgenden wird obiges Modell zur Formulierung des HS-Plazierungsproblems verwendet. Zur Formulierung eines erweiterten Problems (EHS-Problem) wird das Modell in leicht modi-

fizierter Form betrachtet, die keine vorgegebene Verankerung der Komponentenketten auf den Satelliten erfordert (siehe 6.3.3).

6.3.2 Bestimmung des Ressourcenbedarfs

Die Struktur des HS-Modells erlaubt eine besondere Betrachtung von Schnitten von Komponentenketten. Ein Schnitt legt für jede Kette fest, welche Komponenten auf dem Host bzw. dem Satelliten zu plazieren sind. Der kommunikationsbedingte Ressourcenbedarf ist daher eindeutig für jeden Schnitt bestimmbar. Ferner sind für jede Kette die Komponenten bekannt, die für eine Thread-Bildung auf dem Host bzw. auf dem Satelliten in Frage kommen. Bei der Thread-Bildung wird wie in 3.3.4 angenommen, daß der Mischer in einem eigenen Thread gegeben ist und die Komponenten jeweils einer Kette in einem einzigen Thread zusammengezogen werden können.

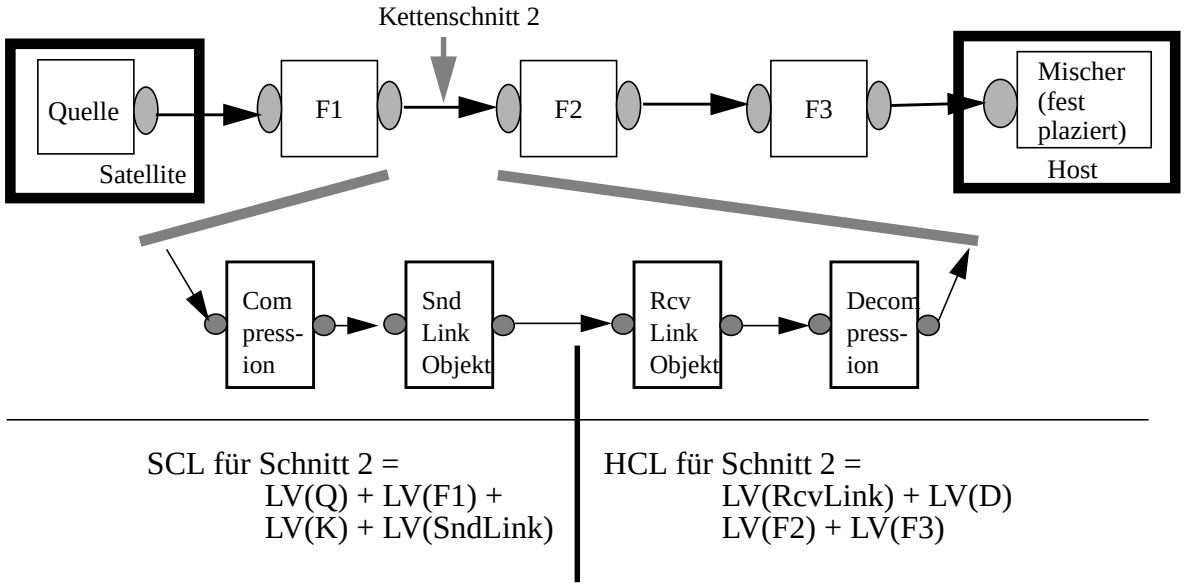


Abb. 41: *Bestimmung des Ressourcenbedarfs für einen Kettenschnitt*

Zu jedem Schnitt j einer Komponentenkette i läßt sich eine Schnittlast $SL_{i,j}$ angeben, die die durch den Kettenschnitt bedingte Last SatL(i, j) auf dem Satelliten sowie die Last HostL(i, j) auf dem Host beinhaltet:

(1) $SL_{i,j} = (SatL(i, j), HostL(i, j))$.

Sowohl SatL(i, j) als auch HostL(i, j) stellen r-elementige Vektoren dar, wobei jedes Element eine Ressourcenanforderung bezüglich einer bestimmten Ressource beschreibt:[29]

(2) $SatL(i, j) = (SatL_1(i,j), ..., SatL_r(i, j))$

(3) $HostL(i, j) = (HostL_1(i,j), ..., HostL_r(i, j))$.

SatL(i, j) wird durch die Aggregierung der Ressourcenanforderungen der Kettenelemente berechnet, die durch den Schnitt j dem Satelliten zugewiesen werden:

(4) $Sat(i, j) = LV(E_{i, 1}) \oplus \ldots \oplus LV(E_{i, j}) \oplus LV(SatCom_{i, j})$.

Dabei bezeichnet $LV(E_{i,e})$ den r-elementigen Lastvektor des Elementes e und $LV(SatCom_{i,j})$ den (r-elementigen) kommunikationsbedingten Ressourcenbedarf, der auf dem Satelliten aufgrund einer K/D-Komponente und des Link-Objekts auftreten kann (Abb. 41).

Die Aggregierung der Anforderungen in Bezug auf eine Ressource hängt von der betrachteten Ressource ab. In Bezug auf die CPU genügt es, die Einzelanforderungen der Elemente der Kette (einschließlich des Bedarfs der K/D-Komponenten und des Link-Objekts) zu addieren (siehe 3.3.4). Der Bedarf an Bandbreite ergibt sich aus dem entsprechenden Bedarf des Link-Objekts, da nur dieses Daten über Rechnergrenzen transportiert. In Bezug auf Speicher muß der Bedarf entsprechend der in 3.3.4 beschriebenen Vorgehensweise ermittelt werden, um eine Überbelegung zu vermeiden.

Die Berechnung für die Last auf dem Host findet analog statt, wobei zur Aggregierung des Ressourcenbedarfs die dem Host zugewiesenen Elemente sowie der kommunikationsbedingte Ressourcenbedarf auf dem Host zu berücksichtigen sind:

(5) $Host(i, j) = (E_{i, j + 1}) \oplus \ldots \oplus LV(E_{i, m}) \oplus LV(HostCom_{i, j})$.

Als Beispiel sei der in Abb. 41dargestellte Kettenschnitt betrachtet. In diesem Fall berechnet sich SatL(i, 2) aus den Lastvektoren der Quelle, des Filters, der eingefügten Komprimierkomponente sowie des senderseitigen Link-Objekts. Entsprechend wird die Last HostL(i, 2) aus den Lastvektoren für das empfangsseitige Link-Objekt, die Komponente zur Dekompression sowie die Filter F2 und F3 berechnet. Der Lastvektor des Mischers wird dagegen nicht erfaßt, da dieser

[29] Z. B. Bandbreite in bit/sec, Anteil an CPU-Kapazität in Prozent, Speichergröße in Bytes.

auf dem Host fest plaziert ist und sein Ressourcenbedarf unabhängig von der Plazierung durch Abzug von den freien Ressourcenkapazitäten des Hosts berücksichtigt werden kann.

6.3.3 Formulierung des Plazierungsproblems

Die Plazierung von Komponenten im HS-Modell wird durch die Verfügbarkeit von Ressourcen verwendeter Rechner beschränkt. Jede Ressource der Satelliten oder des Hosts hat eine begrenzte verfügbare Kapazität, die durch die Summe des Ressourcenbedarfs lokal plazierter Komponenten und Link-Objekte nicht überschritten werden darf. Die verfügbare Kapazität einer Ressource ist bei dem entsprechenden Ressourcenmanager abfragbar und kann somit für die Bestimmung einer Plazierung verwendet werden.[30] Sind die verfügbaren Ressourcenkapazitäten gegeben und ist ferner der Ressourcenbedarf für jeden Kettenschnitt bekannt, so läßt sich das Problem der Plazierung wie folgt betrachten.

Es sei ein verteiltes System gegeben, das aus s Satelliten-Rechnern und einem Host-Rechner besteht. Jeder dieser Rechner verfüge über r Ressourcen, deren Kapazität er zur Verarbeitung und Kommunikation in multimedialen Flußgraphen zur Verfügung stellen kann. Ein Rechner kann dabei eigene, rechnerspezifische Ressourcen verwenden. Für jeden Satelliten (q=1..s) bzw. für den Host sei durch SatC(q) bzw. HostC der r-elementige Vektor bezeichnet, der die verfügbaren Kapazitäten der r Ressourcen erfaßt:

(6) $SatC_q = (SatC_{q,1}, ..., SatC_{q,r})$, für q = 1..s

(7) $HostC = (HostC_1, ... , HostC_r)$.

Es sei ferner ein sternförmiger Flußgraph mit k Komponentenketten gegeben, die alle an einen Mischer angeschlossen sind (Abb. 42). Der Mischer sei auf dem Host fest plaziert. Ferner sei zu jeder Kette i (i=1..k) ein Satellit SatId(i) zugeordnet, auf dem das „freie“ Ende (d. h. die Quelle bzw. die Senke) der Kette verankert ist.

---

[30] Da bei der Benutzung einer Ressource ein gewisses „Overhead“ auftreten kann, stimmt die verfügbare Ressourcenkapazität im allgemeinen nicht mit der tatsächlich belegbaren Kapazität überein. Es wird hier angenommen, das dieser Anteil bei der Berechnung der Kapazität durch die Ressourcenmanager ausreichend (aber nicht notwendigerweise absolut) genau abgeschätzt wird.

Zu jeder Kette i (i=1..k) wird angenommen, daß sie aus m+1 Komponenten besteht, so daß sie m Schnitte $S_{i,j}$ (j=1..m) ermöglicht. Zu jedem Schnitt $S_{i,j}$ sei die Last auf dem Satelliten SatId(i) sowie auf dem Host gemäß des vorigen Abschnitts berechnet:

(8) $SatL(i, j) = (SatL_1(i, j), ..., SatL_r(i, j))$

(9) $HostL(i, j) = (HostL_1(i ,j), ..., HostL_r(i, j))$.

Mit Hilfe der Kapazitätswerte der Ressourcen aus (6) bzw. (7) sei für jeden Schnitt $S_{i,j}$ (j=1..m) einer Kette i die normierte Last für den Satelliten SatId(i) bzw. den Host wie folgt definiert:

(10) $SatRL(i, j) = (SatL_1(i, j) / SatC_{SatId(i),1} , ... , SatL_r(i, j) / SatC_{SatId(i),r})$

(11) $HostRL(i, j) = (HostL_1(i, j) / HostC_1 , ... , HostL_r(i, j) / HostC_r)$.

Die normierte Last gibt für jede Ressource den relativen Anteil an der verfügbaren Kapazität an, der für einen Schnitt j einer Kette i auf dem Satelliten bzw. dem Host beansprucht wird.

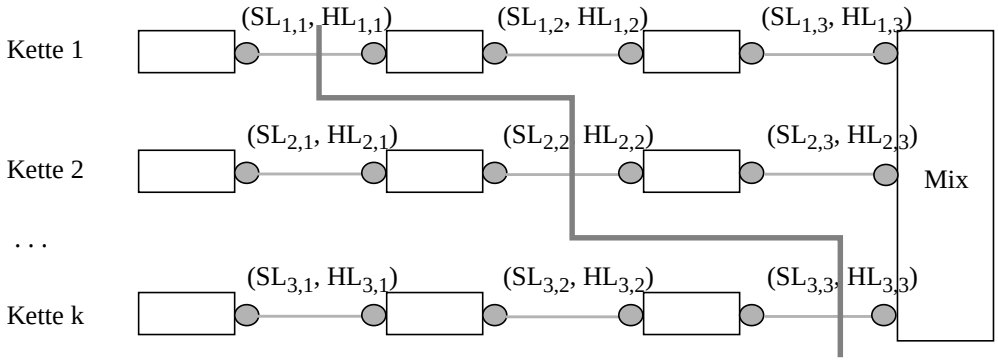


Abb. 42: *Lastvektoren und Schnittbildung zur Plazierung eines Flußgraphen*

Eine Plazierung P eines Flußgraphen in einem gegebenen Host-Satellite-System wird durch eine Menge von Kettenschnitten spezifiziert, wobei zu jeder Kette i (i=1..k) ein Schnitt vorausgesetzt wird:

(12) $P = \{S_{1,j1}, ... , S_{k,jk} \}$.

Eine Plazierung bedingt für jeden Satelliten q (q=1..s) bzw. für den Host eine (normierte) Gesamtlast:

(13) $SatRL_q(P) = \sum_{i \in Kq} SatRL_{i, j(P)}$

(14) $$HostRL(P) = \sum_{i=1}^{k} HostRL_{i,\,j(P)} \quad .$$

Dabei wird mit K_q die Menge der Ketten bezeichnet, die auf dem Satelliten q verankert sind, während j(P) den Schnitt der Kette i gemäß der Plazierung P bezeichnet.

Ist eine Plazierung P gegeben, so wird für jeden Satelliten q (q=1..s) sowie den Host die maximale Ressourcenbeanspruchung („Maximum Resource Usage“, MRU) wie folgt definiert:

(15) $MRU\text{-}S_q(P) = \max(SatRL_{q,1}(P), \ldots , SatRL_{q,r}(P))$

(16) $MRU\text{-}H(P) = \max(HostRL_1(P), \ldots , HostRL_r(P)) .$

Ein solcher Wert stellt die höchste relative Ressourcenanforderung auf einem Satelliten bzw. auf dem Host dar. Für das gesamte verteilte System läßt sich ein globaler MRU-Wert ableiten:

(17) $MRU(P) = \max(MRU\text{-}H(P), MRU\text{-}S_1(P), \ldots , MRU\text{-}S_s(P)) .$

Jede Plazierung P eines Flußgraphen innerhalb eines gegebenen verteilten Systems bedingt einen solchen Wert, der zur Bewertung der Plazierung verwendet werden kann. Eine Plazierung mit einer MRU <= 1 stellt eine mögliche Plazierung dar, da sie für keine Ressource eine Überschreitung verfügbarer Kapazität bedingt. Gute Plazierungsverfahren zeichnen sich ferner dadurch aus, daß sie auch bei geringer Ressourcenverfügbarkeit (im Vergleich zu der geforderten Last) noch eine Lösung finden, sofern es eine Lösung gibt.

Ein Verfahren, das die Plazierung mit minimaler MRU bestimmt, findet immer eine mögliche Lösung, sofern es eine gibt. Ferner ist das Maß der MRU unabhängig von der absoluten Größe verfügbarer Kapazitäten und es eignet sich dazu, die relative Güte von Plazierungsverfahren zu beschreiben. Ein Verfahren, das Plazierungen mit niedrigerer MRU liefert, wird eine Plazierung evtl. auch dann noch finden, wenn ein anderes Verfahren, das eine höhere MRU bedingt, keine Plazierung mehr bereitstellen kann. Liefert ein Verfahren in aller Regel eine MRU, die sehr nahe an der minimal möglichen MRU liegt, so wird das Verfahren eine Plazierung finden, sofern eine möglich ist, außer in den seltenen Fällen, in denen der Ressourcenbedarf der verfügbaren Kapazität praktisch gleich ist.

Diese Überlegungen begründen folgende Formulierung des Plazierungsproblems:
Sind ein sternförmiger Flußgraph sowie ein verteiltes System gemäß des oben beschriebenen Host-Satellite-Modells gegeben, so besteht das HS-Problem der Plazierung darin, eine Plazierung P_{opt} zu finden, die eine minimale MRU bedingt:
(18) $MRU(P_{opt}) <= MRU(P)$ *, für alle möglichen Plazierungen P.*

Das HS-Problem wird (zu dem EHS-Problem) erweitert, falls Komponentenketten zwar auf dem Host verankert sind, jedoch nicht (unbedingt) auf den Satelliten. Die Problemdarstellung ist weiterhin mit Hilfe von Plazierungsketten beschreibbar, für die zur Lösung des Problems eine optimale Schnittmenge gesucht wird (siehe Abb. 42). Hierzu muß eine (im Problemgraphen verwendete) Plazierungskette sowohl sämtliche Verankerungen einer Komponentenkette auf Satelliten als auch die entsprechenden Schnittalternativen beinhalten.

Ist eine Kette auf s Satelliten plazierbar und weist sie m Schnitte auf, so wird bei dem EHS-Problem eine Plazierungskette durch m*s Schnittmöglichkeiten dargestellt. Zu jedem Schnitt ist dabei weiterhin ein (SatL, HostL)-Vektor angegeben. Hinzu kommt eine Zuordnung zu einem Satelliten, da jeder Schnitt für eine Kette sowohl eine Zweiteilung zwischen dem Host und einem Satelliten als auch die Wahl des Satelliten bedingen muß. Die Problemformulierung ist ansonsten analog zu der Formulierung des HS-Problems.

Obige Problemformulierungen wurden im Kontext von Lastbeschreibungen auf Endsystemen aufgestellt. Begrenzte CPU- oder Speicherressourcen können damit genauso berücksichtigt werden wie eine begrenzte Bandbreite am Netzwerkzugang eines Rechners. Ferner kann die bekannte Bandbreite eines Übertragungskanals zwischen einem Satelliten und dem Host miteinbezogen werden, indem diese als eine Begrenzung der Kommunikation am Netzwerkzugang des Satelliten aufgefaßt wird. Dies ist möglich, da sämtliche Kommunikation von oder zu dem Satelliten durch den (einen) Host empfangen bzw. generiert wird.

Allgemein lassen die Problemformulierungen Lastangaben in Bezug auf sehr unterschiedliche Ressourcen zu. Z. B. könnte eine Lastbeschreibung sich auch auf die nötige Bandbreite auf einem Kommunikationsbus beziehen, der von allen Komponentenketten benutzt wird. Das Finden der Plazierung mit minimaler MRU würde dann auch die Last auf dem Bus in die Minimierung einschließen. Nicht zuletzt wäre mit jedem Schnitt eine Kostenangabe assoziierbar, wobei

die Problemformulierung dann eine mögliche Plazierung mit Gesamtkosten unter einer vorgegebenen Kostengrenze anstreben würde. Diese Aspekte werden im Rahmen dieser Arbeit nicht weiterverfolgt, sollten aber im Rahmen zukünftiger Überlegungen in Betracht gezogen werden.

6.3.4 Ablaufmodell

Die vorangegangene Problemformulierung stützt sich auf Information über den benötigten Ressourcenbedarf für Komponenten und Link-Objekte sowie über verfügbare Ressourcenkapazitäten im verteilten System. Erstere ist für Multimedia-Flußgraphen (indirekt) aus statischen Tabellen gegeben, letztere kann durch Interaktion mit den Ressourcenmanagern im verteilten System zu einem beliebigen Zeitpunkt abgefragt werden.

Aufgrund dieser Information ist eine Vorgehensweise möglich, bei der die Berechnung einer Plazierung von der Belegung der Ressourcen durch Komponenten und Links getrennt ist. Ihre Motivation stützt sich auf die Erwartung, daß eine globale Berücksichtigung der zu einem bestimmten Zeitpunkt verfügbaren Ressourcenkapazitäten eine optimierte Plazierung ermöglicht. Isolierte Plazierungen für einen Rechner berücksichtigen nicht die (ressourcenbezogenen) Fähigkeiten anderer Rechner (z. B. Komponenten gemäß der höchsten Dienstgüte zu unterstützen) und können aufgrund der variabel einstellbaren Dienstgüte nicht absehen, welcher Ressourcenbedarf durch plazierte Komponenten entsteht.

Die Vorgehensweise beruht ferner darauf, daß die von Ressourcenmanagern bereitgestellte Information ausreichend genau die tatsächlich belegbaren Ressourcen beschreibt. Dies erfordert insbesondere auch, daß die Ressourcenlage zwischen dem Feststellen verfügbarer Kapazitäten und der tatsächlichen Plazierung eines Flußgraphen relativ konstant bleibt.[31] Die (als selten angenommenen) Fälle ungenauer Plazierungsinformation werden dadurch abgefangen, daß im Anschluß an die Plazierung eines Flußgraphen eine Ressourcenreservierung durchgeführt wird.

---

[31] Diese Annahme ist für Systeme erfüllt, die die Plazierung eines Flußgraphen abschließen, bevor sie mit der nächsten Plazierung beginnen. Sind nebenläufig Plazierungen mehrerer Flußgraphen möglich, so ist für die Annahme die Wahrscheinlichkeit gleichzeitiger Plazierungsanforderungen entscheidend.

Der Gesamtablauf einer Plazierung ist dann in folgende Schritte zerlegbar. Zuerst wird (zu jedem Dienstgütewert) Information über den Ressourcenbedarf von Kettenschnitten sowie über die verfügbaren Ressourcenkapazitäten eingesammelt. Die bestmögliche Plazierung wird berechnet und es wird eine entsprechende Plazierung des Flußgraphen im verteilten System instanziiert. Abschließend wird die Belegung der Ressourcen durch (plazierte) Komponenten und Links mit Hilfe von NRP durchgeführt. Dieser Ablauf garantiert, daß bei genauer Plazierungsinformation der Flußgraph gemäß der maximal möglichen Dienstgüte Ressourcen belegen kann. In jedem Fall ist sichergestellt, daß ein konsistentes Ergebnis des Session-Aufbaus erreicht wird.

6.4 Einordnung des Host-Satellite Problems

6.4.1 Taxonomie

In [Gosc91] wurde eine Klassifikation von Verfahren zur Lastbalancierung eingeführt, die statische und dynamische Verfahren unterscheidet. Als statisch werden Verfahren bezeichnet, die eine Plazierung aller Module vor Beginn der Ausführung finden, keine Migration von Modulen zwischen Rechnern während der Laufzeit erlauben und bei der Berechnung einer Plazierung Information über Ressourcenbedarf und Ressourcenverfügbarkeit voraussetzen. Als dynamisch werden Verfahren bezeichnet, die wenig a-priori Information über den Ressourcenbedarf der Module bzw. über die Ressourcenverfügbarkeit im verteilten System voraussetzen. Plazierungen werden vielmehr über eine Ausbalancierung der Last in verschiedenen Rechnern durchgeführt und sie können evtl. zur Laufzeit durch Migration von Modulen korrigiert werden.

Das Ablaufmodell zur Lösung des Plazierungsproblems für Multimedia-Flußgraphen läßt sich in die Gruppe statischer Verfahren einordnen. Zum einen muß die Plazierung eines Flußgraphen vor dessen Benutzung festgelegt sein, da Multimedia-Kommunikation einen kontinuierlichen Datenfluß erfordert, der durch eine Migration von Modulen nicht gestört werden kann. Zum anderen steht für eine Berechnung Information zur Verfügung, aufgrund derer eine sinnvolle Plazierung berechnet werden kann.

Letzteres hängt damit zusammen, daß Komponenten und Links Ressourcen vor Benutzung des Flußgraphen reservieren, um eine gewünschte Dienstgüte garantieren zu können. Ressourcen

werden gemäß eines im voraus bestimmten Ressourcenbedarfs belegt, wobei bei der Berechnung des Ressourcenbedarfs Aspekte der Kommunikation und Thread-Bildung berücksichtigt werden können. Auf der anderen Seite sind verfügbare Ressourcen auf den Rechnern abfragbar, so daß diese Information zur Berechnung einer Plazierung verwendet werden kann. Die Reservierung von Ressourcen stellt schließlich sicher, daß die Benutzung einer Ressource durch verschiedene Einheiten (z. B. verschiedene Elemente eines Flußgraphen) sich nicht gegenseitig beeinflußt.

6.4.2 Verwandte Plazierungsprobleme

Das Problem der Plazierung von Multimedia-Flußgraphen ist mit früher betrachteten Problemen der Plazierung paralleler Programme in Multiprozessorumgebungen bzw. verteilten Systemen verwandt (siehe [NoTh93] und Referenzen unten). Ausgangspunkt dieser Probleme sind gleichzeitig laufende Programmodule, die während der Verarbeitung miteinander kommunizieren können. Die Problemdarstellung stützt sich auf statische Programmgraphen, die sich aus Modulen (Graphknoten) zusammensetzen, die über Links miteinander verbunden sind ([YWPS95]). Wie Komponenten dienen Module zur Durchführung bestimmter Verarbeitungsfunktionen, während Links zur Kommunikation zwischen Modulen verwendet werden. Ferner wird ein Systemgraph vorausgesetzt, der die Ausführungslokationen (Prozessoren oder verteilte Rechner) und ihre Konnektivität erfaßt. Module und Links sind mit Last- und/oder Kostenangaben assoziiert, die sie bei einer Plazierung aufweisen können.

Eine Reihe von Unterschieden bestehen gegenüber dem hier eingeführten HS-Problem. Allgemein berücksichtigen die meisten Ansätze nur bestimmte Ressourcentypen zur Entscheidung einer Plazierung. Eine Klasse von Ansätzen reduziert das Plazierungsproblem auf das Finden einer Plazierung mit minimalen Gesamtkosten ohne jegliche Beachtung ressourcenbedingter Einschränkungen ([KLZ97], [LeSh97], [BiEl95], [Bill94], [LLK92], [Fern89], [Lo88], [Tows86], [Ston77]). Eine zweite Gruppe nimmt für jedes Modul eine CPU-Lastangabe an und sucht nach einer Plazierung mit balancierter Ressourcenlast auf den einzelnen Prozessoren ([OlMa95], [IqBo95], [HaLi92], [NiHa91], [Bokh88]). Weitere Ressourcen (Speicher, Netzwerkzugang) werden nicht betrachtet. Wiederum andere Ansätze verfolgen eine Kostenoptimierung und zugleich eine Balancierung der CPU-Last ([SBAM96]), [YaSa93], [BNG92], [KiPa90], [Efe82]).

Wenige Ansätze sind beschrieben worden, die begrenzte Kapazitäten mehrerer Ressourcentypen in Betracht ziehen. In [YWPS95] wird ein Ansatz beschrieben, der u. a. CPU-Last, Speicherbedarf sowie Kommunikationsbandbreite beachtet. Allerdings werden zum Finden einer Plazierung „Branch-and-Bound“ sowie „Simmulated-Annealing“-Techniken eingesetzt, die beide exponentielle Komplexität bei der Berechnung einer Plazierung aufweisen. Der Ansatz von [WoMo93] berücksichtigt die CPU-Last sowie eine Kommunikationslast auf einem gemeinsam benutzten Kommunikationsbus, weitere Ressourcen werden nicht berücksichtigt.

Ein wesentliches Merkmal des HS-Problems besteht darin, daß die Ressourcenlast für die Komponentenketten schnittorientiert berechnet wird, d. h. unter Berücksichtigung von Kommunikation und Thread-Bildung. Letztere wird in Ansätzen zur Plazierung von Programmgraphen nicht beachtet. In einigen Ansätzen (z. B. [IqBo95], [WoMo93], [HaLi92]) wird ein Modell angenommen, bei dem für die Dauer der Kommunikation zwischen zwei Rechnern der Kommunikationsbus und die beteiligten CPUs gleichermaßen blockiert sind. Auf diese Weise läßt sich der Einfluß der Kommunikation auf die CPU-Last mitbetrachten. Für Fälle, in denen kommunikationsbedingter Ressourcenbedarf nicht nur aufgrund der Übertragung im Netz, sondern auch unabhängig hiervon (z. B. für Kompression/Dekompression) entsteht, ist das Modell nicht ausreichend.

Einige der erwähnten Verfahren sind für Programmgraphen allgemeiner Topologie entwickelt worden und verwenden unterschiedliche Heuristiken zur Lösung des jeweiligen Plazierungsproblems (siehe [NoTh93]). Exakte oder approximative Verfahren sind im Kontext eingeschränkter Plazierungsszenarien betrachtet worden, so z. B. zur Plazierung einer Modulkette auf einer Kette von Rechnern ([OlMa95], [IqBo95], [HaLi92], [Bokh88]).

Das Host-Satellite-Problem wurde im Kontext von Kettenpartitionierungen zwischen einem Host und mehreren Satelliten in [Bokh88] eingeführt. Der Ansatz ging von der Annahme aus, daß auf jedem Satelliten eine einzige Kette verankert ist und daß nur die CPU-Last zu betrachten sei. Als Ziel wurde das Finden der Kettenschnitte verfolgt, die die Last auf der Flaschenhals-CPU minimiert, wobei die Flaschenhals-CPU diejenige CPU ist, die nach erfolgter Plazierung die höchste Last aufweist. Für diesen Fall wurde eine exakte Lösung geliefert, die später durch schnellere Verfahren (exakter oder approximativer Art) ergänzt wurde ([IqBo95], [HaLi92], [NiHa91]).

Das HS-Problem, das im Abschnitt 6.3.3 definiert wurde, geht über die frühere Problembetrachtung hinaus. Erstens ermöglicht es die Betrachtung beliebig vieler Ressourcen auf den Rechnern, indem es für Kettenschnitte mehrdimensionale Lastvektoren zuläßt. Entsprechend sind verfügbare Kapazitäten der Rechner über mehrdimensionale Beschreibungen angebbar. Zweitens läßt die Formulierung die Verwendung beliebig vieler Ketten pro Satelliten zu. Dies schließt insbesondere Flußgraphen ein, die von einem Satelliten aus Daten bereitstellen und auf demselben Satelliten (bearbeitete) Daten zur Präsentation empfangen. Für das so definierte HS-Problem wurden in der Vergangenheit ebensowenig Lösungen bereitgestellt, wie für das erweiterte HS-Problem, bei dem Komponentenketten nicht im voraus einzelnen Satelliten zugeordnet sind.

7 Das FPP-Protokoll

Um die Plazierung von Flußgraphen gemäß des im vorigen Kapitel eingeführten Host-Satellite-Modells vorzunehmen, wird ein Verfahren benötigt, das Mechanismen zum Einsammeln von Plazierungsinformation und einen Algorithmus zur Berechnung einer dienstgüteoptimierten Plazierung bereitstellt sowie eine Instanziierung von Komponenten und Links gemäß der Plazierung veranlaßt. Ein solches Verfahren wurde innerhalb von CINEMA in Form des „Flowgraph Placement Protocol“ (FPP) entwickelt, das in diesem Kapitel beschrieben wird. Im nächsten Abschnitt wird eine Übersicht über die wichtigsten Merkmale von FPP gegeben. Anschließend werden im Abschnitt 7.2 die zugrundeliegende Protokollarchitektur sowie der Protokollablauf vorgestellt. Insbesondere werden die Einheiten identifiziert, die für die Ausführung der Plazierungsalgorithmen verantwortlich sind. Verschiedene Algorithmen zur Lösung des formulierten (einfachen und erweiterten) Host-Satellite-Problems werden im Abschnitt 7.3 untersucht und hinsichtlich Lösungsgüte und Berechnungsdauer miteinander verglichen.

7.1 Eigenschaften

Das FPP-Protokoll wurde für sternförmige Flußgraphen entwickelt, die in einem verteilten System bestehend aus Satelliten- und Host-Rechnern plaziert werden sollen. Das Protokoll unterstützt eine Plazierung in Abhängigkeit von der gewünschten Dienstgüte, den Formatbeschränkungen der Komponenten sowie den verfügbaren Ressourcen auf den einzelnen Rechnern. Das Protokoll erwartet als Anfangsinformation die Spezifikation eines zu plazierenden Flußgraphen sowie der zu verwendenden Satelliten- und Host-Rechner. Es läßt außerdem eine Spezifikation medienspezifischer Dienstgüte an einem Komponenten-Port des Flußgraphen zu. Eine entkoppelte Dienstgüteeinstellung für die einzelnen Senken ist nicht möglich, da der verwendete Plazierungsalgorithmus keine variablen Filter berücksichtigt.

Das Protokoll wird zur Plazierung eines Flußgraphen verwendet, wobei Information über benötigte und verfügbare Ressourcen der Rechner verarbeitet wird. FPP nimmt die Plazierung der Komponenten vor. Eine Instanziierung plazierter Komponenten und Link-Objekte sowie die anschließende Belegung von Ressourcen werden nicht durch FPP, sondern durch andere Mechanismen in CINEMA vorgenommen. Für die (Code-) Instanziierung wird das Konfigura-

tionsmanagement von CINEMA benutzt, für die Ressourcenreservierung wird das NRP-Protokoll eingesetzt. Die Trennung von Plazierung und Ressourcenreservierung läßt die Verwendung von FPP auch in Fällen zu, in denen die Plazierungsinformation zwar als gute Schätzung, aber nicht notwendigerweise als exakt vorausgesetzt werden kann.

FPP macht keine Annahmen über die Art des Ressourcenbedarfs von Komponenten und Links auf Rechnern. Eine Komponente kann auf jedem für sie in Frage kommenden Rechner eine rechnerspezifische Menge von Ressourcen belegen. Genauso kann die benötigte Kapazität einer bestimmten Ressource auf unterschiedlichen Rechnern beliebig unterschiedlich sein, so daß insbesondere unterschiedliche Implementierungen für eine Komponente verwendet werden können. Das FPP-Protokoll ist in der Lage, die im Kapitel 6 geforderte schnittorientierte Beschreibung des Ressourcenbedarfs bereitzustellen und zu verwenden.

Das im Kapitel 6 formulierte Host-Satellite-Problem impliziert die Verwendung eines festgelegten Host-Rechners. Das FPP-Protokoll beinhaltet die mehrfache Lösung des HS-Problems in Bezug auf verschiedene Hosts. Insofern stellt das FPP-Protokoll eine Lösung für ein allgemeineres Problem dar, das den zusätzlichen Freiheitsgrad beinhaltet, einen von mehreren Hosts zur Plazierung auszuwählen.

7.2 Protokollarchitektur und Protokollablauf

Das Design von FPP stützt sich auf eine Menge sogenannter Plazierungskoordinatoren. Auf jedem beteiligten Rechner (Host oder Satellit) ist ein *lokaler Koordinator* (LK) gegeben, der Berechnungen in Bezug auf Komponenten durchführt, die auf dem Rechner plaziert werden können. Zusätzlich wird ein *zentraler Koordinator* (ZK) verwendet, der für eine Session auf einem beliebigen Rechner zur Verfügung gestellt werden kann (siehe Abb. 43).

Der ZK realisiert zum einen die Dienstschnittstelle zum Klienten hin. Er ist somit der Punkt, an dem der Klient einen Flußgraphen und die gewünschte Dienstgüte bereitstellt. Der ZK ist ferner die Einheit, die die verschiedenen Aufgaben der LK miteinander koordiniert und letztlich über die endgültige Wahl einer Plazierung entscheidet.

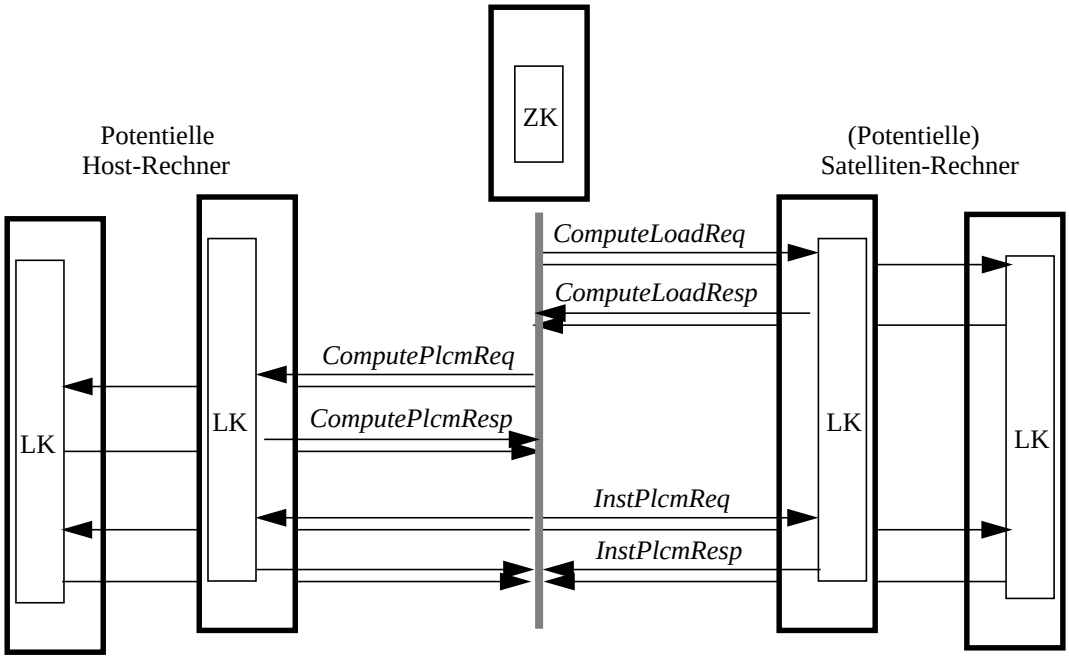


Abb. 43: *Protokollablauf*

Die LK der Rechner erledigen auf Aufforderung durch den ZK verschiedene Aufgaben zur Berechnung der Plazierung. Der LK eines Satelliten stellt eine Methode (*ComputeLoad*) zur Berechnung des Ressourcenbedarfs für alle Schnitte der Komponentenketten bereit, die auf dem Satelliten verankert werden können. Der LK eines Hosts realisiert eine weitere Methode (*ComputePlacement*) zur Berechnung der bestmöglichen Plazierung unter Verwendung des Hosts. Ferner stellen alle LK eine Methode bereit, über die nach Festlegung der Plazierung die Instanziierung der Komponenten und Link-Objekte angestoßen werden kann.

Der Ablauf des Protokolls ist wie folgt. Nach Erhalt der Flußgraphen- und Dienstgütespezifikation vom Klienten stellt der ZK für jeden Satelliten eine Nachricht (*ComputeLoadRequest*) zusammen, in der zum einen die Ketten (mit ihren Komponentenbezeichnern) beschrieben sind, die auf dem Satelliten (potentiell) zu verankern sind. Für jede Kette ist außerdem der Wertebereich angegeben, aus dem die Anfangskomponente (d. h. Quelle oder Senke) der Kette Werte annehmen kann[32]. Die Nachricht hat folgende Struktur:

[32] Die Wertebereiche werden vom ZK aus dem spezifizierten Dienstgütewertebereich mit Hilfe der Filterfaktoren im Flußgraphen abgeleitet.

ComputeLoadRequest(

Kette 1: Komponente 1, Komponente 2, ..., Komponente m
Wertebereich(val_1, ..., val_q)

...

Kette n: Komponente 1, Komponente 2, ..., Komponente m
Wertebereich(val_1, ..., val_q)

).[33]

Nach Erhalt einer solchen Nachricht berechnet der LK eines Satelliten für jede Kette den Ressourcenbedarf für jeden möglichen Kettenschnitt. Die Berechnung wird für jeden Parameterwert durchgeführt, der am Anfang der Kette eingestellt werden kann. Für jeden Kettenschnitt berücksichtigt der LK den Ressourcenbedarf der auf dem Satelliten zu plazierenden Komponenten einschließlich der evtl. einzufügenden Komponente zur Kompression/Dekompression und des Link-Objekts, das zur entfernten Kommunikation eingesetzt wird. Die Berechnung des Ressourcenbedarfs zu einem Schnitt stützt sich auf die Ressourcenbedarfstabellen der Komponenten und des Link-Objekts (siehe 6.3.2). Bei k Ketten, m Schnitten pro Kette und q möglichen Parameterwerten, muß der LK insgesamt k*q*m Berechnungen durchführen. Der Ressourcenbedarf für alle möglichen Parameterwerte bzw. alle möglichen Kettenschnitte wird vom LK in einer Nachricht (*ComputeLoadResponse*) an den ZK gesandt:

ComputeLoadResponse(

Kette 1: val 1: $SatL_{1,1}$, ..., $SatL_{1,m}$
...
val q: $SatL_{1,1}$, ..., $SatL_{1,m}$

...

Kette n: ...

).

Der ZK empfängt die Nachrichten aller LK und aggregiert sie in eine neue Nachricht (*ComputePlacementRequest*), mit der alle Host-LK aufgerufen werden. Jeder Host-LK ergänzt die

[33] Analog zur Darstellung von NRP, bezieht sich die Beschreibung des FPP-Protokolls auf einen einzigen medienspezifischen Parameter. Entsprechend umfassen angegebene Wertebereiche diskrete Werte dieses Parameters.

Beschreibung für jeden Kettenschnitt um den Ressourcenbedarf, der auf dem Host abgeleitet wird ($HostL_{i,j}$). Für jeden Dienstgütewert ist auf diese Weise zu jeder Kette die gesamte Information vorhanden, die im Abschnitt 6.3.3 als Grundlage des Host-Satellite-Problems definiert wurde:

Kette i: Schnitt 1: ($SatL_{i,1}$, $HostL_{i,1}$), ..., Schnitt m: ($SatL_{i,m}$, $HostL_{i,m}$) .

Jeder Host ist mit dieser Information in der Lage, die Plazierung mit minimal erforderlicher Ressourcenbeanspruchung (MRU) zu berechnen. Jeder Host ermittelt die Plazierung, die eine MRU hat, die 100 % nicht überschreitet und die einem möglichst hohen Dienstgütewert entspricht. Hierzu kann der Host die Plazierungen zu allen Dienstgütewerten berechnen und die beste Plazierung (mit einer MRU <= 1) auswählen.[34] Diese Plazierung (d. h. die Schnitte aller Ketten) wird zusammen mit dem dazugehörigen Dienstgütewert als Ergebnis dem ZK mitgeteilt:

ComputePlacementResponse(
best_val: $S_{1,j1}$, ... , $S_{k,jk}$
).

Nach Erhalt aller Antworten von den Hosts kann der ZK entscheiden, welcher Host die Plazierung mit der bestmöglichen Dienstgüte ermöglicht. Mit der ausgewählten Plazierung ruft der ZK die LK der Satelliten und des ausgewählten Hosts auf, um die Instanziierung des Flußgraphen anzustoßen:

InstantiatePlacementRequest(
Kette 1: Komponente 1, ..., Komponente j1
...
Kette n: Komponente 1, ..., Komponent jn
).

[34] Da angenommen werden kann, daß der Ressourcenbedarf mit abnehmender Dienstgüte ebenfalls abnimmt, kann der Host den gesuchten Dienstgütewert durch ein Intervallhalbierungsverfahren finden, das auf den Dienstgütewertebereich angewandt wird. Die Zahl der Berechnungen ist dann durch $\lceil (\log_2 q) \rceil$ begrenzt.

Der LK eines Rechners nutzt Funktionen des Konfigurationsmanagements, um die Instanziierung der Komponenten und der Link-Objekte durchzuführen. Die Beendigung der Instanziierung wird dem ZK über eine *InstantiatePlacementResponse*-Nachricht mitgeteilt. Nach Empfang dieser Nachricht von allen LK ist der ZK informiert, daß der Flußgraph auf allen beteiligten Rechnern instanziiert ist. Der ZK kann anschließend den Aufbau einer (Reservierungs-) Session mit Hilfe von NRP durchführen.

7.3 Lösung des Host-Satellite-Problems

In der Vergangenheit wurden Lösungen für das Host-Satellite-Problem beschrieben, die von einer eindimensionalen Lastbeschreibung ausgehen und genau eine Komponentenkette pro Satelliten erlauben. In diesem Abschnitt werden zwei neu entwickelte Verfahren vorgestellt, die das Problem ohne diese Einschränkungen lösen. Das erste Verfahren liefert eine exakte Lösung für den Fall, daß benötigte sowie verfügbare Kapazität einer beliebigen Ressource durch ganze Zahlen in einem begrenzten Wertebereich (0..V-1) gegeben sind. Anschließend wird ein heuristisches Verfahren vorgestellt, das diese Einschränkung nicht aufweist. Beide Verfahren werden hinsichtlich der Verarbeitungsdauer und der Güte der erzielten Lösung miteinander verglichen. Die Anwendung der Heuristik auf das erweiterte Host-Satellite-Problem wird anschließend behandelt.

7.3.1 Exaktes Verfahren

7.3.1.1 Lösung für den Fall eines Satelliten

Es sei zunächst angenommen, daß außer dem Host ein einziger Satellit gegeben ist, so daß alle Ketten auf demselben Satelliten verankert sind. Die Problemdarstellung ist in Form eines k-fach geschichteten Graphen gegeben, wobei jede Schicht eine Komponentenkette mit ihren möglichen m Schnitten repräsentiert. In Abb. 39 ist ein solcher Graph dargestellt. Eine Plazierung entspricht einem Pfad, der die zwei Punkte S(tart) und E(nde) miteinander verbindet und hierbei jede Kette genau einmal durchschneidet.

Die Kanten des Graphen sind mit mehrdimensionalen Gewichten versehen. Für den Fall eines Satelliten repräsentieren die (2*r-dimensionalen) Gewichte die Lastbeschreibungen der Ketten-

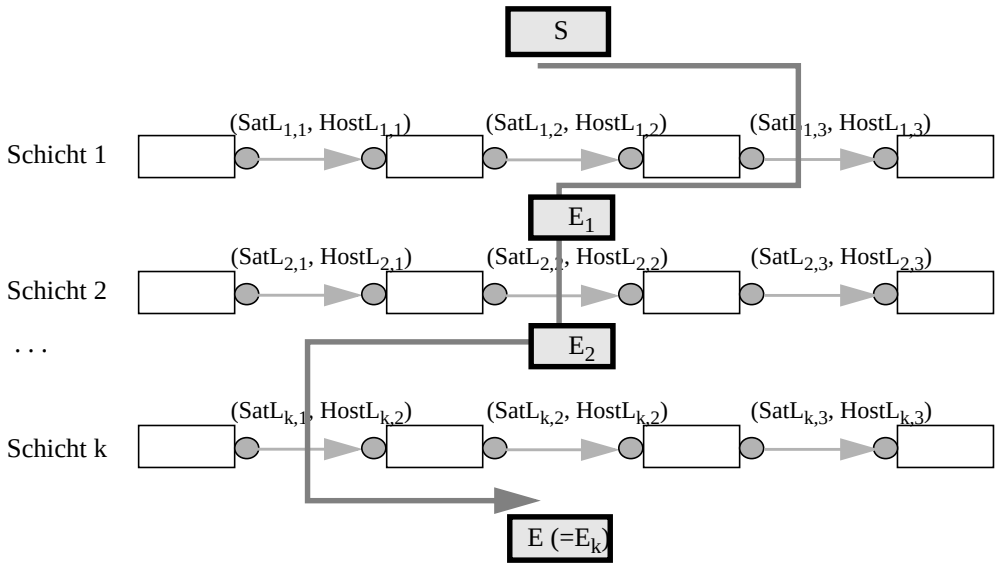


Abb. 44: *Plazierungspfad für einen geschichteten Graphen*

schnitte für den Satelliten $SatL_{i,j}$ und den Host $HostL_{i,j}$ (siehe 6.3.3). Die Lastangaben werden als in Bezug auf die entsprechenden Ressourcenkapazitäten normiert vorausgesetzt, wobei sie ganzzahlige Werte zwischen 0 und einem Maximalwert V-1 annehmen können.[35]

Für jeden Pfad ist ein Lastvektor (L-Vektor) angebbar, der sich durch Vektoraddition der Gewichte der geschnittenen Kanten ergibt. Ein L-Vektor besteht aus einem Teilvektor für den Satelliten (SatL), der die Last für den Satelliten beschreibt, sowie einem Teilvektor für den Host (HostL), der die aufsummierte Last für die Hostressourcen erfaßt. Ist ein Pfad P gegeben, so wird die dazugehörige MRU durch das maximale Element des L-Vektors von P definiert:

(1) $MRU(P)= \max(LV(P)) = \max(SatL_1, ... , SatL_r, HostL_1, ... ,HostL_r)$.

Entsprechend obiger Betrachtung wird zur Lösung des Plazierungsproblems ein Pfad gesucht, der S mit E verbindet und eine minimale MRU aufweist.

Die im folgenden vorgestellte Lösungsidee zielt auf das Finden eines Pfades zwischen S und E zu jedem möglichen L-Vektor ab. Wird z. B. eine einzige Ressource für den Satelliten bzw. den Host angenommen und werden zur Lastbeschreibung Werte zwischen 0 und V-1=1 zugelassen, so genügt es, zu jedem denkbaren L-Vektor ((0, 0), (0, 1), (1, 0) bzw. (1,1)) einen Pfad zu fin-

[35] Ein Wert von V-1 entspricht einer Lastanforderung, die exakt der verfügbaren Ressourcenkapazität gleich ist.

den, sofern es einen solchen gibt. Ist dieses Wissen gegeben, so kann der L-Vektor mit minimaler MRU (gemäß (1)) sowie der hierzu entsprechende Pfad als Lösung des Plazierungsproblems bestimmt werden.

Um solche Information über L-Vektoren sowie die dazugehörigen Pfade zu erfassen, wird im folgenden zu jeder Schicht i eine sogenannte Schichttabelle betrachtet:

(2) $SchichtT_i$ = { (L, Flag(L), Schnitt(L)):

$L = (L_1, ..., L_{2r})$ mit $L_1, ... , L_{2r} = 0..V-1$

Flag(L) = <u>true</u>, falls ein Pfad von S nach E_i existiert mit LV(P) = L

<u>false</u>, sonst

Schnitt(L) = Schnitt der Schicht i, der LV(P) = L ermöglicht

}.

Dabei wird zu jedem 2*r-dimensionalen Vektor L festgehalten, ob ein Pfad existiert, der die Ketten 1 bis i durchläuft und einen L-Vektor gleich L aufweist. Sofern solche Pfade existieren, wird zu dem L-Vektor ein Schnitt der Schicht i festgehalten, der durch irgendeinen dieser Pfade durchlaufen wird.

Wie weiter unten gezeigt wird, läßt sich ein Lösungspfad berechnen, sofern die Schichttabellen aller Schichten i=1..k bekannt sind. Um die Schichttabelle einer Schicht i zu ermitteln, wird zu jedem Schnitt j einer Kette i (j=1..m) eine weitere, als Schnittabelle bezeichnete Struktur verwendet:

(3) $SchnittT_{i,j}$ = { (L, Flag(L)):

$L = (L_1, ..., L_{2r})$ mit $L_1, ... , L_{2r} = 0..V-1$

Flag(L) = <u>true</u>, falls ein Pfad von S nach E_i existiert mit LV(P) = L,

der die Schicht i im Schnitt j durchläuft

<u>false</u>, sonst

}.

Mit Hilfe obiger Strukturen läßt sich eine Prozedur zur Berechnung eines optimalen Pfades beschreiben (siehe Abb. 45, 46). Hierbei lassen sich zwei Phasen unterscheiden. In einer ersten Phase werden die Schichttabellen berechnet, während in der zweiten Phase ein optimaler Pfad zwischen S und E mit Hilfe dieser Tabellen ermittelt wird.

Die *Berechnung der Schichtabellen* erfolgt in k Iterationen (siehe Abb. 45). Dabei wird in der Iteration i (i=1..k) die Schichttabelle $SchichtT_i$ mit Hilfe der in der Iteration (i-1) berechneten Tabelle $SchichtT_{i-1}$ ermittelt. Zur Berechnung der Tabelle der ersten Schicht wird eine Anfangstabelle (der Schicht „0“) vorausgesetzt, in welcher die Flags zu den L-Vektoren zurückgesetzt sind, außer für den Nullvektor, zu dem das Flag auf „true“ gesetzt ist.

Ist eine Schichttabelle der Schicht i gegeben, so lassen sich für die Schicht i+1 zunächst die m Schnittabellen $SchnittT_{i+1,j}$ (j=1..m) berechnen. Zu jeder solchen Tabelle werden hierzu sämtliche möglichen L-Vektoren durchlaufen (siehe Funktion1 in Abb. 46). Zu jedem L-Vektor L wird überprüft, ob ein Pfad von S nach E_{i+1} existiert, der die Schicht i+1 im Schnitt j durchschneidet. Hierzu genügt es festzustellen, ob ein Pfad von S nach E_i existiert, der einen L-Vektor start_L aufweist, der genau um den Anteil „kleiner“ ist, der durch das Gewicht $SL_{i,j}$ des Schnittes j der Kette i gegeben ist. Da die Existenz eines solchen Pfades in der Schichttabelle $SchichtT_i$ zu jedem möglichen start_L-Vektor festgehalten ist, ist diese Überprufung ohne weiteres möglich.

Sind die Schnittabellen der Schicht i+1 gegeben, so läßt sich die Schichttabelle $SchichtT_{i+1}$ gemäß Funktion2 (Abb. 46) berechnen. Hierbei werden alle möglichen L-Vektoren der Schichttabelle durchlaufen. Zu einem L-Vektor wird in $SchichtT_{i+1}$ das Flag gesetzt, sofern das Flag in einer der Schnittabellen $SchnittT_{i+1,j}$ zu dem L-Vektor L gesetzt war. Dies entspricht der Semantik aus (2), da ein Pfad von S zu E_{i+1} genau dann existiert, sofern ein Pfad von S zu E_{i+1} existiert, der irgendeinen der m Schnitte der Kette i+1 durchläuft. Bei der Berechnung wird ferner zu jedem L-Vektor, zu dem ein Pfad im obigen Sinne existiert, der Schnitt der Kette abgespeichert, der auf einem der möglichen Pfade liegt.

Nach der k. Iteration sind alle Schichttabellen berechnet. Anhand der Schichtabelle der Schicht k läßt sich der L-Vektor L_{opt} bestimmen, zu dem ein Pfad von S nach E existiert und der einen minimalen MRU-Wert MRU_{opt} aufweist. Hierzu werden erneut alle L-Vektoren der Tabelle durchlaufen und zu jedem L-Vektor, zu dem ein Pfad als möglich angezeigt wird, wird die entsprechende MRU gemäß (1) berechnet.

Zu L_{opt} wird in der zweiten Phase des Verfahrens ein *optimaler Pfad* P_{opt} abgeleitet. Hierzu werden die Schichttabellen in umgekehrter Reihenfolge in k Iterationen durchlaufen. In der

ersten Iteration wird aus der Tabelle SchichtT_k zu L_{opt} der abgespeicherte Schnitt ausgelesen. Dieser Schnitt liegt auf einem optimalen Pfad, daher kann er zur Festlegung des Teilpfades von E_{k-1} zu E_k verwendet werden.

Um den Pfad von S zu E_{k-1} zu finden, wird der L-Vektor abgeleitet, der in der Schichttabelle SchichtT_{k-1} betrachtet werden muß, um P_{opt} zu ermöglichen. Dieser Vektor ist durch Abzug des Gewichts des Schnittes der Kette k von dem ermittelten LK-Vektor L_{opt} zu erhalten. Mit diesem Vektor läßt sich aus SchichtT_{k-1} der Schnitt bestimmen, der den Teilpfad von E_{k-2} zu E_{k-1} festlegt.

Wird obige Vorgehensweise k mal angewandt, so sind alle Teilpfade (E_1, E_2), ..., (E_{k-1}, E_k) bestimmbar. Zusammengefaßt bilden diese einen optimalen Pfad P_{opt}, der in der Schichttabelle der Schicht k den ermittelten minimal möglichen MRU-Wert aufweist.

Komplexität des Verfahrens

Die Lösungsberechnung beinhaltet die k-fache Wiederholung der Berechnungen für eine Schicht. Für jede Schicht werden m Schnittabellen ermittelt. Hierzu sind insgesamt $m*V^{2r}$ (2*r-dimensionale Vektor-) Subtraktionen und Flag-Überprüfungen nötig. Für die Berechnung einer Schichttabelle werden für jeden der V^{2r} L-Vektoren m Flags überprüft, insgesamt sind also $m*V^{2r}$ Operationen nötig. Für eine Schicht ergibt sich insgesamt eine Berechnungskomplexität der Ordnung $O(r*m*V^{2r})$. Für k Schichten ist die Komplexität durch $O(k*r*m*V^{2r})$ gegeben. Das Auffinden des Pfads minimaler Länge erfordert insgesamt k*r Operationen, so daß die Berechnungskomplexität durch $O(k*r*m*V^{2r})$ dominiert wird.

Der Speicherbedarf für das Verfahren wird durch die gleichzeitige Bereithaltung der k Schichttabellen bestimmt, die jeweils V^{2r} -Einträge beinhalten. Auf einmal wird nur eine Schnittabelle benötigt, da ihre Flags in die nächste zu berechnende Schichttabelle übernommen werden können, bevor die nächste Schnittabelle berechnet wird.

Eingabe: k Ketten K_i, $1 \leq i \leq k$, jede mit m Schnittlasten
$SL_{i,j} = (SatL_{i,j}, HostL_{i,j})$, $1 \leq j \leq m$
Lastelemente mit ganzahligen Werten zwischen 0..V-1

Ergebnis: Optimale Plazierung P_{opt} gegeben als Menge von Schnitten $\{ j_1, ..., j_k \}$
Optimale Maximale Ressourcenbeanspruchung MRU_{opt}

Prozedur1 {

```
/* Initialisierung der Anfangsschichttabelle (Schicht 0) */
SchichtT_0 = Init()

/* Iterative Berechnung der Schichttabellen der Ketten i=1..k */
for(i=1; i<= k; i++) {
        /* Berechnung der Schnittabellen der Schnitte j=1..m der Kette i */
        for(j=1; j<=m; j++)
                SchnittT_i,j = Funktion1( SchichtT_i-1, SL_i,j )
        /* Berechnung der Schichttabelle der Kette i
        SchichtT_i = Funktion2( SchnittT_i,1, ..., SchnittT_i,m )
}

/* Finden eines L-Vektors in der k. Schichttabelle mit gesetztem Flag und minimaler MRU */
    MRUopt = UNENDLICH
    Lopt = null
    for( L = (L_1=0, ..., L_2*r=0); L <= (L_1=V-1, ... , L_2*r=V-1); L++ ) {
        if (SchichtT_k(L).Flag == true) {
                MRU = max( L_1, ..., L_2*r)
                if (MRU < MRUopt) {
            MRUopt = MRU
            Lopt = L
                }
        }
    }

    /* Finden eines Pfades P mit LV(P) = Lopt */
    L = Lopt
    for(i=k; i>=1; i--) {
        j_i = SchichtT_i(L).Schnitt
        L = L - SL_i,j
    }

}
```

Abb. 45: *Prozedur zur Berechnung eines Pfades mit minimaler MRU (1)*

```
neueSchnittTab Funktion1( SchichtT_{i-1}, SL_{i,j} ) {
        /* neue Schnittabelle mit nicht gesetzten Flags (= false) */
        neueTab = new SchnittTab()

        for( L = (L_1=0, ..., L_{2*r}=0); L <= (L_1=V-1, ... , L_{2*r}=V-1); L++ ) {
                start_L = L - SL_{i,j}
                if ( start_L >= 0 )
                        neueTab(L).Flag = SchichtT_{i-1}(start_L).Flag
        }
        return neueTab
}


neueSchichtTab Funktion2( SchnittT_{i,1}, ... , SchnittT_{i,m} ) {
        /* neue Schichttabelle mit nicht gesetzten Flags (=false) */
        neueTab = new SchichtTab()

        for( L = (L_1=0, ..., L_{2*r}=0); L <= (L_1=V-1, ... , L_{2*r}=V-1); L++ ) {
                for(j=1; j<=m; j++)
                        if (SchnittT_{i,j}(L).Flag == true) {
                                neueTab(L).Flag = true
                                neueTab(L).Schnitt = j;
                                break
                        }
        }
        return neueTab
}
```

Abb. 46: *Hilfsfunktionen zur Berechnung von Schnitt- und Schichttabellen (1)*

7.3.1.2 Lösung für den Fall mehrerer Satelliten

Sind mehrere Satelliten gegeben, so kann das eingeführte Verfahren ohne wesentliche Erhöhung der Berechnungskomplexität erweitert werden. Hierzu werden die Komponentenketten zunächst satellitenweise gruppiert und die Komponentengruppen in einer beliebigen Sequenz angeordnet (Abb. 47). Für jede Kettengruppe ist im Prinzip das Verfahren vom vorigen Abschnitt anwendbar. Um den Einfluß der Pfadwahl über die Kettengruppen hinweg zu betrachten, wird die Tabellenstruktur erweitert und es werden die Kettengruppen nacheinander in der vorgegebenen Sequenz betrachtet.

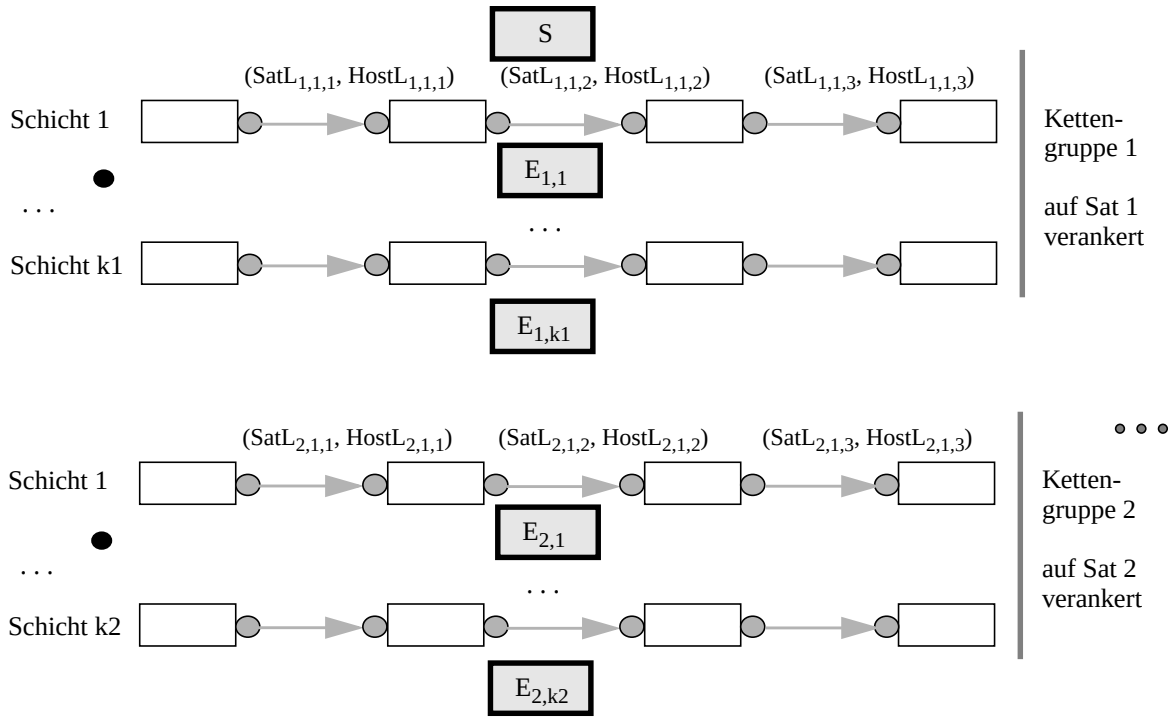


Abb. 47: *Gruppierung von Ketten*

In einer Schicht- bzw. Schnittabelle einer Kette i der Gruppe g ($SchichtT_{g,i}$ bzw. $SchnittT_{g,i,j}$) stellt, ähnlich wie im vorigen Abschnitt, ein L-Vektor L eine (2*r-elementige) Lastbeschreibung für den Satelliten Sat_g sowie den Host dar. Das Flag, das zu einem L-Vektor gehört, zeigt an, ob ein Pfad von S zu $E_{g,i}$ existiert, der den betrachteten L-Vektor aufweist. Hierbei ist zu beachten, daß ein solcher Pfad alle Ketten der Gruppen 1..(g-1) sowie die Ketten 1..i der Gruppe g durchläuft. Die dabei entstehende Last auf dem Satelliten Sat_g sowie auf dem Host wird durch den L-Vektor wiedergegeben, nicht jedoch die Last auf den Satelliten 1 bis (g-1).

Zwischen S und $E_{g,i}$ können verschiedene Pfade gegeben sein, die die gleiche Last auf Sat_g bzw. dem Host bewirken (und somit den gleichen L-Vektor aufweisen). Auf der anderen Seite können diese Pfade auf den Satelliten 1 bis (g-1) unterschiedliche Lasten bedingen, die zu einer unterschiedlichen MRU auf diesen Satelliten führen können. Unter diesen Pfaden existiert einer, der eine minimale MRU auf den Satelliten 1 bis (g-1) bewirkt.

Um bei der Betrachtung eines L-Vektors L in einer Tabelle der Gruppe g die Last auf vorangegangenen Satelliten 1..(g-1) zu berücksichtigen, wird zu jedem L-Vektor neben dem bisherigen Flag eine weitere Größe erfaßt. Diese wird als minimal erforderliche Last vorangegangener Satelliten (minLS) bezeichnet. Sie gibt die minimale MRU auf den Satelliten 1 bis (g-1) an, die für irgendeinen der Pfade auftritt, die S mit $E_{g,i}$ verbinden und dabei den L-Vektor L (für Sat_g sowie den Host) bedingen:

(4) $SchichtT_{g,i}$ = { (L, Flag(L), minLS(L), Schnitt(L)):
L = (L_1, ..., L_{2r}) mit L_1, ... , L_{2r} = 0..V-1
Flag(L) = <u>true</u>, falls ein Pfad von S nach $E_{g,i}$ existiert mit LV(P) = L
<u>false</u>, sonst
minLS(L) = niedrigste erforderliche MRU auf Satelliten 1 bis g-1
Schnitt(L) = Schnitt der Schicht i, der LV(P) = L und minLS(L) ermöglicht
}

(5) $SchnittT_{g,i,j}$ = { (L, Flag(L), minLS(L)):
L = (L_1, ..., L_{2r}) mit L_1, ... , L_{2r} = 0..V-1
Flag(L) = <u>true</u>, falls ein Pfad von S nach $E_{g,i}$ existiert mit LV(P) = L, der die $Schicht_{g,i}$ im Schnitt j schneidet
<u>false</u>, sonst
minLS(L) = niedrigste erforderliche MRU auf Satelliten 1 bis g-1
}.

Die Prozedur zur Lösung des Plazierungsproblems betrachtet nacheinander die Kettengruppen in der vorgegebenen Sequenz (siehe Abb. 49). Die Berechnung für jede Kettengruppe basiert dabei auf einer initialen Tabelle (I-Tabelle). Am Ende der Berechnungen für eine Kettengruppe wird die Schichttabelle der letzten Kette der Gruppe (Endtabelle E-T) als Basis für die Berechnung der I-Tabelle der nächsten Kettengruppe verwendet (siehe unten). Die I-Tabelle für die erste Kettengruppe wird initialisiert, indem für den Nullvektor in der Tabelle das Flag gesetzt wird, für alle anderen Vektoren wird das Flag nicht gesetzt. Der minLS-Wert (für den Nullvektor) wird auf 0 gesetzt, da es keine Last auf vorangegangenen Satelliten zu berücksichtigen gibt.

Innerhalb einer Kettengruppe g läuft das Verfahren weitgehend wie im vorigen Abschnitt ab. Um die m Schnittabellen der Schicht i+1 zu berechnen, wird die Schichttabelle der Schicht i verwendet (siehe Funktion5 in Abb. 51). Die Schnittabellen können wiederum zur Ermittlung der Schichttabelle der Schicht i+1 eingesetzt werden (Funktion6 in Abb. 51). Zu einem L-Vektor L der Schichttabelle wird das Flag gesetzt, falls das Flag in einer der Schnittabellen der Schicht gesetzt ist. Als minLS-Wert wird der kleinste minLS-Wert genommen, der in den Schnittabellen der Schicht i+1 vorkommt. Auf diese Weise wird erreicht, daß zu jedem L-Vektor angezeigt wird, ob ein Pfad von S zu $E_{g,i+1}$ existiert, und falls ja, welches der minLS-Wert zu diesem Pfad ist.

Aus der Schichttabelle der letzten Kette (Endtabelle) der Gruppe g wird die I-Tabelle für die Gruppe g+1 abgeleitet (siehe Funktion3 in Abb. 50). Ein Vektor der I-Tabelle beinhaltet keinen SatL-Anteil, der die Last auf dem Satelliten (g+1) beschreibt, da dieser (vor Betrachten der Ketten der Gruppe) gleich Null ist.[36] Jeder Vektor der I-Tabelle beschreibt vielmehr eine Lastsituation auf dem Host, die durch die vorangegangenen Berechnungen für die Kettengruppen 1 bis g gegeben sein könnte. Ob dies der Fall ist, wird aus der Endtabelle der Gruppe g ermittelt.

(6) $I\text{-}Tab_{g+1}$ = { (HL, Flag(HL), minLS(HL)):

HL = $(HL_1, ..., HL_r)$ mit $HL_1, ... , HL_r = 0..V\text{-}1$

Flag(HL) = <u>true</u>, falls ein Pfad von S nach $E_{g,kg}$ existiert mit

HostL(P) = HL

<u>false</u>, sonst

minLS(HL) = niedrigste erforderliche MRU auf Satelliten 1 bis g }.

Um zu überprüfen, ob für den Host in der Endtabelle der Gruppe g das Flag zu einem vorgegebenem HostL-Vektor HL gesetzt ist, müssen alle V^r L-Vektoren durchlaufen werden, die im HostL-Anteil mit HL übereinstimmen und ansonsten einen beliebigen SatL-Anteil enthalten.[37] Ist zu einem dieser L-Vektoren das Flag gesetzt, so gibt es zum HostL-Vektor HL in der I-Tabelle wenigstens einen Pfad. Zum L-Vektor in der I-Tabelle mit dem betrachteten HostL-Vektor HL wird in diesem Fall das Flag gesetzt, ansonsten nicht.

---

[36] Mit Null ist hier der r-dimensionale Nullvektor gemeint.

[37] Es gibt V^r P-Vektoren mit vorgegebenem H-Anteil und frei wählbarem S-Anteil.

Wird zu einem HostL-Vektor in der I-Tabelle das Flag gesetzt, muß ferner der zugehörige minLS-Eintrag berechnet werden. Ein minLS-Eintrag in der Endtabelle erfaßt die maximale Last auf den Satelliten 1 bis g-1, ein minLS-Eintrag in der I-Tabelle (der Kettengruppe g+1) muß dagegen die maximale Last auf den Satelliten 1 bis g erfassen. Um diese veränderte Semantik zu berücksichtigen, müssen zu jedem HostL-Vektor der I-Tabelle, zu dem ein Flag gesetzt wurde, die korrespondieren V^r L-Vektoren der Endtabelle der Gruppe g betrachtet werden (siehe Funktion3 in Abb. 50).

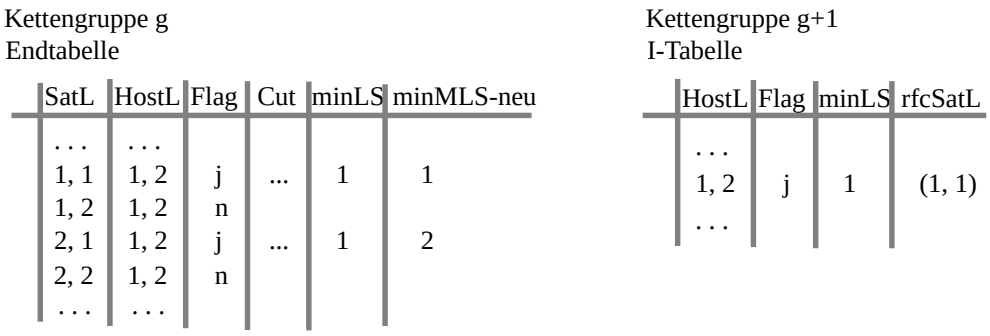


Abb. 48: *Ausschnitt aus der Berechnung einer I-Tabelle*

Als Beispiel seien die Tabellen in Abb. 48 betrachtet. Zu dem HostL-Vektor (1,2) der I-Tabelle der Gruppe g+1 korrespondieren (im Ausschnitt) die L-Vektoren (SatL=(1,1), HostL(1,2)), (SatL=(1,2), HostL(1,2)), (SatL=(2,1), HostL(1,2)), (SatL=(2,2), HostL(1,2)). Zu zweien dieser Vektoren sei das Flag gesetzt, die restlichen zwei bleiben daher unberücksichtigt. Der jeweilige minLS-Wert gibt die minimal erforderliche MRU auf den Satelliten 1 bis g-1 an. Die Berechnung der minLS-Werte für die Satelliten 1 bis g ist wie folgt.

Für den L-Vektor (SatL=(1,1), HostL(1,2)) beträgt der ursprüngliche minLS-Wert 1. Dieser ändert sich nicht, da keines der Elemente des SatL-Vektors höher als eins ist. Für den L-Vektor (SatL=(2,1), HostL(1,2)) ergibt sich der neue minLS-Wert zu 2, da dies durch eines der Elemente des SatL-Anteils erfordert wird. In die I-Tabelle wird zu dem HostL-Vektor (1,2) der minLS-Wert von 1 übernommen, da er der niedrigstmögliche ist. Ferner wird (in rfcSatL) festgehalten, für welche SatL-Last auf dem Satelliten der Gruppe g der berechnete minLS-Wert auftritt. Diese Referenz wird bei der Ermittlung des optimalen Pfades benötigt (siehe unten).

Die I-Tabelle der Schicht g+1 erlaubt die Initialisierung der Anfangstabelle für die Berechnungen innerhalb der Kettengruppe g+1 (siehe Funktion4 in Abb. 50). Die Berechnungen ergeben gemäß der Prozedur2 sämtliche Schichttabellen der Gruppe g+1, somit insbesondere auch die Endtabelle der Gruppe, aus der wiederum die I-Tabelle der nächsten Kettengruppe berechnet werden kann.

Durch Wiederholung obiger Vorgehensweise lassen sich die I- bzw. Schichttabellen aller Kettengruppen berechnen. In der Endtabelle der letzten Kettengruppe s sind zu jedem L-Vektor die MRU des Hosts sowie die MRU des Satelliten der letzten Gruppe ableitbar. Ferner ist der minLS-Wert bekannt, der die minimale erforderliche MRU auf den Satelliten 1 bis (s-1) angibt. Zu jedem L-Vektor läßt sich daher die globale MRU berechnen. Aus der Endtabelle der Gruppe s kann somit der L-Vektor L_{opt} mit der minimalen (globalen) MRU MRU_{opt} bestimmt werden.

Um den Pfad minimaler Länge zu rekonstruieren, ist es wie im vorigen Abschnitt erforderlich, im Laufe der Berechnungen innerhalb einer Kettengruppe in jeder Schichttabelle zu jedem relevanten L-Vektor den ausgewählten Schnitt der Schicht festzuhalten. Um die Rekonstruktion über die Kettengruppen hinweg zu ermöglichen, ist es ferner erforderlich, zu jedem HostL-Vektor der I-Tabelle der Gruppe g+1 einen entsprechenden L-Vektor der Endtabelle der Gruppe g festzuhalten. Letzterer ist derjenige L-Vektor, dessen minLS-Eintrag für die Berechnung in die I-Tabelle übernommen wurde (sieh oben).

Die Rekonstruktion des Pfads minimaler Länge läuft wie folgt ab. In der Endtabelle der letzten Kettengruppe ist der L-Vektor mit der MRU_{opt} bestimmt. Zu diesem L-Vektor kann wie im vorigen Abschnitt der Teilpfad ermittelt werden, der die Schnitte der betrachteten (hier letzten) Kettengruppe festlegt. Ist der L-Vektor für die Schichttabelle der ersten Kette ermittelt, so kann der entsprechende Schnitt der Kette sowie dessen Gewicht ermittelt werden. Durch Subtraktion des Schnittgewichts vom L-Vektor wird ein HostL-Vektor der I-Tabelle der Gruppe bestimmt, zu dem ein entsprechender L-Vektor der Endtabelle der vorangehenden (hier vorletzten) Kettengruppe abgespeichert ist. Mit dessen Hilfe kann die Prozedur zur Ermittlung des Teilpfads für die (vorletzte) Kettengruppe wiederholt werden. Das Verfahren ist fortsetzbar, bis alle Kettengruppen einschließlich der ersten durchlaufen sind. An diesem Punkt sind sämtliche Teile des Pfades mit minimaler Länge ermittelt.

Eingabe: s Satelliten Sat_g, g=1..s, mit je
k_g verankerten Ketten $K_{g,i}$, i=1..k_g , jede Kette mit m Schnittlasten

$SL_{g,i,j} = (SatL_{g,i,j}, HostL_{g,i,j})$, $1 \leq j \leq m$
Lastelemente mit ganzahligen Werten zwischen 0..V-1

Ergebnis: Optimale Plazierung P_{opt} gegeben als Menge von Schnitten { j_1, ... , j_k }
Optimale Maximale Ressourcenbeanspruchung MRU_{opt}

Prozedur2 {

```
/* Initialisierung der Anfangstabelle (gegeben als Endtabelle der Gruppe „0“) */
E-T_0 = Init()

/* Iterative Betrachtung der g Kettengruppen */
for(g=1; g<=s; g++) {

    /* Berechnung der I-Tabelle der Gruppe g und Initialisierung der Anfangstabelle */
    I-T_g = Funktion3(E-T_g-1)
    SchichtT_g,0 = Funktion4(I-T_g)

    /* Iterative Berechnung der Schichttabellen der Ketten i=1..k_g der Gruppe g*/
    for(i=1; i<= k_g; i++) {
        /* Berechnung der Schnittabellen der Schnitte j=1..m der Kette i */
        for(j=1; j<=m; j++)
                SchnittT_g,i,j = Funktion5( SchichtT_g, i-1 , SL_g,i,j )

         /* Berechnung der Schichttabelle der Kette i */
         SchichtT_i = Funktion6( SchnittT_g,i,1, ..., SchnittT_g,i,m )
    }
}

/* Finden eines L-Vektors Lopt in der k. Schichttabelle */
/* mit gesetztem Flag und minimaler MRUopt */
analog zu Vorgehensweise in Prozedur1 in Abb. 45

/* Finden eines Pfades P mit LV(P) = Lopt */
L = Lopt
for(g=s; g>=1; g--) {
    for(i=k_g; i>=1; i--) {
         j_g,i = SchichtT_g,i(L).Schnitt
         L = L - SL_g,i,j
    }

    HostL = (L_r+1, ..., L_2r)
    SatL = I-T(HostL).rfcSatL
    L = (SatL, HostL)
}
```

}

Abb. 49: *Prozedur zur Berechnung eines Pfades mit minimaler MRU (2)*

```
neueI-Tab Funktion3( E-T_g-1) {
        /* neue I-Tabelle mit nicht gesetzten Flags (= false) */
        neueTab = new I-Tab()

        for( HostL = (HL_1=0, ..., HL_r=0); HostL <= (HL_1=V-1, ... , HL_r=V-1); HostL++ ) {
                /* Berechnung der Flags und MinSL-Werte */
                minLS = UNENDLICH
                rfcSatL = null
                for( SatL = (SL_1=0, ..., SL_r=0); SatL <= (SL_1=V-1, ... , SL_r=V-1); SatL++ ) {
                        L = (SatL, HostL)

                        if (E-T_g-1(L).Flag == true) {
                                MRU_1 = max (SL_1, ..., SL_r, HL_1, ..., HL_r)
                                MRU = max(MRU_1, E-T_g-1(L).minLS)

                                if (MRU < minLS) {
                                        minLS = MRU
                                        rfcSatL = SatL
                                }
                        }
                }

                if (minLS < UNENDLICH) {
                        neueTab(HostL).Flag = true
                        neueTab(HostL).rfcSatL = rfcSatL
                        neueTab(HostL).minLS = minLS
                }
        }
        return neueTab
}

neueSchichtTab Funktion4(I-Tab_g) {
        /* neue Schichttabelle mit nicht gesetzten Flags (= false) */
        neueTab = new SchichtTab()

        for( HostL = (HL_1=0, ..., HL_r=0); HostL <= (HL_1=V-1, ... , HL_r=V-1); HostL++ ) {
                L = (0, ..., 0, HL_1, ..., HL_r)

                if ( I-Tab_g(L).Flag == true) {
                        neueTab(L).Flag == true
                        neueTab(L).minLS = InitTab(L).minLS
                }
        }
        return neueTab
}
```

Abb. 50: *Hilfsfunktionen zur Berechnung von Schnitt- und Schichttabellen (2)*

```
neue SchnittTab Funktion5( SchichtT_{g, i-1}, SL_{g,i,j} ) {
    /* neue Schnittabelle mit gelöschten Flags (= false) */
    neueTab = new SchnittTab()

    for( L = (L_1=0, ..., L_{2*r}=0); L <= (L_1=V-1, ... , L_{2*r}=V-1); L++ ) {
        start_L = L - SL_{g,i,j}
        if ( start_L >= 0 ) {
            neueTab(L).Flag = SchichtT_{g,i-1}(start_L).Flag
            neueTab(L).minLS = SchichtT_{g,i-1}(start_L).minLS
        }
    }
    return neueTab
}


neueSchichtTab Funktion6( SchnittT_{g,i,1}, ... , SchnittT_{g,i,m} ) {
    /* neue Schichttabelle mit gelöschten Flags (=false) */
    neueTab = new SchichtTab()

    for( L = (L_1=0, ..., L_{2*r}=0); L <= (L_1=V-1, ... , L_{2*r}=V-1); L++ ) {
        for(j=1; j<=m; j++) {
            minLS = UNENDLICH
            if ( (SchnittT_{g,i,j}(L).Flag == true) AND (SchnittT_{g,i,j}(L).minLS < minLS) {
                neueTab(L).Flag = true
                neueTab(L).minLS = SchnittT_{g,i,j}(L).minLS
                neueTab(L).Schnitt = j;
            }
        }
    }
    return neueTab
}
```

Abb. 51: *Hilfsfunktionen zur Berechnung von Schnitt- und Schichttabellen (3)*

Komplexität des Verfahrens

Die Lösungsberechnung beinhaltet wie im vorigen Abschnitt die k-fache Wiederholung der Berechnungen für eine Schicht, die jeweils durch eine Komplexität von $O(k*r*m*V^{2r})$ dominiert wird.[38] Hinzu kommt die Berechnung einer I-Tabelle für jede Kettengruppe, die das Betrachten aller HostL-Vektoren in der I-Tabelle bzw. der entsprechenden L-Vektoren in der

[38] Bei der Berechnung einer Schichttabelle benötigt die Flag-Bestimmung zu jedem L-Vektor m Vergleiche, genauso die Berechnung des minLS-Wertes.

Endtabelle bedingt. Für jede I-Tabelle sind somit $O(r*V^{2r})$ Operationen nötig. Da es höchstens k Kettengruppen geben kann, ändert dies die Ordnung der oben angegebenen Komplexität nicht. Die Rekonstruktion des Pfades minimaler Länge erfordert k*r Operationen, so daß die Ordnung der Berechnungskomplexität des Verfahrens insgesamt durch $O(k*r*m*V^{2r})$ gegeben ist.

7.3.2 Heuristisches Verfahren

Das vorgestellte exakte Verfahren weist eine Berechnungskomplexität auf, die exponentiell mit der Zahl r der betrachteten Ressourcen zunimmt. Die Meßergebnisse im nächsten Abschnitt zeigen, daß die Berechnungsdauer nur für den Fall einer betrachteten Ressource (auf jedem Satelliten und dem Host) befriedigend ist. Um auch Szenarien abzudecken, in denen mehrere Ressourcen pro Rechner zu berücksichtigen sind, wurde ein weiteres, heuristisches Verfahren betrachtet, das einen neuen, als SQueeze bezeichneten Algorithmus beinhaltet.

Die Grundidee von SQ besteht darin, für alle k Ketten initiale Schnitte gemäß einer „greedy“ - Vorgehensweise zu bestimmen und anschließend Schritt für Schritt die Wahl der Kettenschnitte zu optimieren. Letzteres geschieht, indem für jede Kette nach und nach „schlechte“ Schnitte ausgeschlossen werden, bis für jede Kette nur noch ein möglicher Schnitt übrigbleibt (siehe Abb. 52).

Der Algorithmus betrachtet die Ketten in einer beliebigen, dann aber feste Reihenfolge. Im *ersten Schritt* werden initiale Schnitte für die Ketten bestimmt. Die Wahl eines Schnitts für die Kette i beruht auf der Lastsituation auf den Satelliten und dem Host, die durch die initialen Schnitte der Ketten 1 bis i-1 verursacht wurde. Von den m möglichen Schnitten der Kette i, wird derjenige Schnitt ausgewählt, dessen hinzukommende Last auf dem Satelliten ($SatL_{i,j}$) bzw. auf dem Host ($HostL_{i,j}$) eine minimale MRU auf den Satelliten SatId(i) und dem Host bewirkt.

Der *Optimierungsteil* von SQ wird in mehreren Iterationen durchgeführt. In jeder Iteration werden die Ketten in der vorgegebenen Reihenfolge durchlaufen. Die Lastsituation auf den Satelliten und dem Host vor Beginn der Iteration i+1 wird durch die Lastsituation am Ende der Iteration i definiert. Vor Beginn der ersten Iteration wird die Lastsituation betrachtet, die durch die initialen Schnitte aller Ketten gegeben ist.

Zu Beginn jeder Iteration hat jede Kette i einen aktuellen Schnitt (minSchnitt_i), der im Verlauf der vorangegangen Iteration bestimmt wurde. Vor der ersten Iteration ist dies der initiale

Eingabe: k Ketten K_i, $1 \le i \le k$, jede mit
m Schnittlasten $(SatL_{i,j}, HostL_{i,j})$ $1 \le j \le m$
Lastelemente sind auf Werte zwischen 0 und 1 normalisiert
SatId(i)
s Satelliten S_q $1 \le q \le s$
1 Host H

Ergebnis: Plazierung P gegeben als Menge von Schnitten $\{ S_{1,j1}, \ldots, S_{k,jk} \}$
Maximale Ressourcenbeanspruchung MRU

Schritt 0: Initialisierung der Ressourcenbelegungen auf den Rechnern
für alle S_q: AktuelleLast AL(q) = 0
für H: AL(H) = 0

Schritt 1: Berechnung der initialien Schnitte der Ketten

```
for( i = 1; i <= k; i ++) {
    minMRU_i = UNENDLICH;
    for( j = 1; j <= m; j++) {
        AL( SatId(i) ) = AL( SatId(i) ) + SatL_i,j
        AL( H ) = AL( H ) + HostL_i,j
        bestimme neueMRU
        if (neueMRU < minMRU_i)
            minSchnitt_i = j
            minMRU_i = neueMRU
        AL( SatId(i) ) = AL( SatId(i) ) - SatL_i,j
        AL( H ) = AL( H ) - HostL_i,j
    }
    AL( SatId(i) ) = AL( SatId(i) ) + SatL_i,minSchnitt_i
    AL( H ) = AL( H ) + HostL_i,minSchnitt_i
}
```

Schritt 2: Iterative Optimierung der Schnittauswahl

```
for( iter = 1; iter <= m-1; iter++ ) {
    for( i = 1; i <= k; i++ ) {
        AL( SatId(i) ) = AL( SatId(i) ) - SatL_i,minSchnitt_i
        AL( H ) = AL( H ) - HostL_i,minSchnitt_i
        bestimme (minSchnitt_i, minMRU_i, maxSchnitt_i, maxMRU_i) (in Analogie zu S1)
    }
    markiere maxSchnitt_i für Kette i als gelöscht
    AL( SatId(i) ) = AL( SatId(i) ) + SatL_i,minSchnitt_i
    AL( H ) = AL( H ) + HostL_i,minSchnitt_i
}
```

Schritt 3: Zusammenstellung der Ergebnisse

```
for( i = 1; i <= k; i++)
    S_i,ji = minSchnitt_i
MRU = minMRU_k (nach letzter Iteration von Schitt 2)
```

Abb. 52: *Der SQ-Algorithmus*

Schnitt. Im Laufe einer Iteration werden zu einer Kette zwei Schnitte bestimmt: der „beste“ Schnitt, der (bei Austausch gegen den aktuellen Schnitt) die minimale MRU auf den Satelliten und dem Host bewirkt, sowie der „schlechteste“ Schnitt, der (bei Austausch gegen den aktuellen Schnitt) die maximale MRU bewirkt. Der schlechteste Schnitt wird für die Kette gelöscht, d. h. er wird im Verlauf folgender Iterationen nicht mehr als ein möglicher Schnitt berücksichtigt. Der aktuelle Schnitt wird gegen den besten Schnitt ausgetauscht. Die Last auf dem Satelliten der Kette und dem Host wird entsprechend neu berechnet.

Die Berechnungskomplexität des Verfahrens wird durch den iterativen Teil bestimmt. Während jeder Iteration wird für jede Kette ein Schnitt gelöscht. Da es m Schnitte gibt, werden m-1 Iterationen durchgeführt. Während einer Iteration werden für jede Kette höchstens m Schnitte betrachtet, wobei jede Betrachtung r Additionen und Vergleiche beinhaltet. Für eine Iteration ergibt sich eine Komplexität der Ordnung O(r*k*m), für alle Iterationen ergibt sich die Komplexität zu $O(r*k*m^2)$.

7.3.3 Bewertung der Verfahren

Die eingeführten Algorithmen (Exakt, SQ) wurden durch Messung der jeweils erzielbaren MRU für eine Vielzahl von stochastisch erzeugten Problemfällen evaluiert. Zusätzlich wurde ein drittes Verfahren (RCut) bewertet, das für jede Kette zufällig einen Schnitt bestimmt. Da das exakte Verfahren die optimale Lösung und damit auch die minimal erreichbare MRU liefert, wurden seine MRU-Ergebnisse als Vergleichsbasis herangezogen. Da die Berechnungen für das exakte Verfahren nur für den Fall einer betrachteten Ressource pro Rechner in vertretbarer Zeit durchzuführen waren, wurde das exakte Verfahren nur unter dieser Annahme benutzt. Für die anderen Fälle (mit zwei oder drei Ressourcen pro Rechner) wurden die Ergebnisse der Heuristiken relativ zum MRU-Ergebnis des SQ-Verfahrens betrachtet.[39] Alle Messungen wurden auf SunSparc20-Rechnern durchgeführt, die mit 50 MHz getaktet waren.

In Bezug auf die Wahl der Lastwerte für die Kettenschnitte wurden zwei Fälle unterschieden. Im ersten Fall („Random Component Load“) wurden für jede Komponente einer Kette die geforderten Lastwerte (für die Satelliten- bzw. Host-Ressourcen) als Zufallswerte aus einem Bereich (minL, maxL) ausgewählt. Die Lastvektoren für einen Kettenschnitt wurden durch

[39] In einfachen Fällen konnte die optimale MRU durch Enumeration aller Möglichkeiten bestimmt werden.

Addition der Lasten der Komponenten berechnet, die durch den Schnitt dem Satelliten bzw. dem Host zugewiesen werden. Die Last, die durch Kommunikation zwischen Satelliten und Host verursacht wird, wurde nicht berücksichtigt. Dies ist für Szenarien möglich, in denen die kommunikationsbedingte Ressourcenlast für alle Schnitte einer Kette gleich ist.[40] In diesem Fall ist sie von der Wahl der Kettenschnitte unabhängig und kann durch Subtraktion von der verfügbaren Ressourcenkapazität vor Bestimmung der Kettenschnitte (d. h. vor Anwendung eines der Berechnungsverfahren) berücksichtigt werden.

Im zweiten Fall („Random Cut Load“) wurden die Lastwerte unmittelbar für die Kettenschnitte durch zufällige Auswahl im Bereich (minL, maxL) bestimmt. Die benötigte Bandbreite wurde für jeden Schnitt auf beiden Seiten, d. h. für den Satelliten und den Host, auf den gleichen Wert festgelegt. Die Lastwerte für andere Ressourcen (z. B. CPU) wurden dagegen (für jeden Schnitt) unabhängig für den Satelliten und den Host ausgewählt.

Die verfügbare Kapazität auf den Satelliten wurde für jede Ressource auf einen Zufallswert im Bereich (minC, maxC) festgelegt. Für jeden Satelliten wurde dieselbe Anzahl von verankerten Ketten angenommen und es wurde derselbe Wertebereich zur Festlegung der Ressourcenkapazitäten verwendet. Da der Host alle Komponentenketten zu unterstützen hat, wurden seine Ressourcenkapazitäten entsprechend der Gesamtzahl der Satelliten aus dem Bereich (s*minC, s*maxC) ausgewählt, wobei s die Anzahl der verwendeten Satelliten angibt. Diese Wahl sorgt (im Mittel über viele Messungen hinweg) dafür, daß pro Kette sowohl auf den Satelliten als auch auf dem Host eine vergleichbare Ressourcenkapazität vorhanden ist, so daß die Ressource mit der maximalen Ressourcenbeanspruchung auf jedem Rechner mit näherungsweise vergleichbarer Wahrscheinlichkeit lokalisiert wird.

Im folgenden werden Ergebnisse zu verschiedenen Meßszenarien vorgestellt. Die Meßszenarien unterscheiden sich durch die Zahl betrachteter Ressourcen (eine oder drei) und eine unterschiedliche Berechnung der Schnittlasten („Random Component Load“, „Random Cut Load“). Für jedes Meßszenario wurden Messungen für verschiedene Problemgrößen vorgenommen, wobei die Problemgröße durch die Zahl der betrachteten Satelliten (2, 4, 8, 16) und die Zahl der Schnitte pro Kette (2, 4, 8) definiert wird. Für jede Problemgröße wurde eine Meßserie für 300

[40] Dies ist dann gegeben, falls alle Filterfaktoren der Filterkomponenten auf eins gesetzt sind.

Problemfälle mit zufällig vorgegeben Lastvektoren und Ressourcenkapazitäten durchgeführt. Für jeden Problemfall wurde für jedes Verfahren die erzielbare MRU relativ zu der exakt bestimmten MRU berechnet. Als Gütemaß für ein Verfahren wurde die mittlere relative MRU für eine Meßserie betrachtet. Für die im folgenden vorgestellten Ergebnisse wurden einheitlich zwei Ketten pro Satelliten vorausgesetzt sowie die Wertebereiche (minL=100, maxL=300) bzw. (minC=100, maxC=300) zugrundegelegt.

Anzahl Schnitte		Anzahl Satelliten					Anzahl Satelliten					Anzahl Satelliten			
		2	**4**	**8**	**16**		**2**	**4**	**8**	**16**		**2**	**4**	**8**	**16**
2	Exact	100	100	100	100		100	100	100	100		297	773	1102	2241
	RCut	134	146	153	160		119	120	121	123		0	0	0	0
	Sq	*101*	*100*	*100*	*100*		*100*	*100*	*100*	*100*		*0*	*0*	*0*	*1*
4	Exact	100	100	100	100		100	100	100	100		539	1012	1788	2379
	RCut	153	182	221	242		142	146	149	156		0	0	0	0
	Sq	*102*	*102*	*101*	*101*		*100*	*100*	*100*	*100*		*0*	*0*	*1*	*2*
8	Exact	100	100	100	100		100	100	100	100		798	1482	2683	3772
	RCut	164	189	236	249		162	170	173	175		0	0	0	0
	Sq	*104*	*103*	*101*	*101*		*100*	*100*	*100*	*100*		*0*	*1*	*2*	*4*

Durchschnittliche relative MRU („Random Component Load“)

Durchschn. relative MRU („Random Cut Load“)

Berechnungsdauer (in msec)

Abb. 53: *Vergleich der Meßergebnisse (1 Ressource pro Computer)*

In Abb. 53 sind die Meßergebnisse für den Fall einer Ressource auf dem Satelliten bzw. dem Host angegeben. Darin sind die durchschnittlichen relativen MRU in Prozent der MRU aufgeführt, die für das exakte Verfahren benötigt wird. Zusätzlich sind die Berechnungsdauern für die einzelnen Verfahren angegeben. Die Ergebnisse zeigen, daß die durch den SQ-Algorithmus erzielbare MRU für die extremste betrachtete Problemgröße (8 Schnitte pro Kette, 16 Ketten insgesamt) im Durchschnitt 4 % über der optimal erzielbaren MRU liegt. Für kleinere Problemszenarien liegt sie näher beim Optimum. Für das „RandomCutLoad“-Szenario ist die Abweichung unter einem Prozent für alle betrachteten Problemgrößen. Die Güte des SQ-Verfahrens ist deutlich besser als diejenige des RCut-Verfahrens.

Die Berechnungsdauer wurde für alle Algorithmen für den schlimmsten Fall bestimmt. Aus der Beschreibung des exakten und des SQ-Algorithmus ist ersichtlich, daß die Berechnungsdauer unabhängig von der Wahl der Werte für einen Problemfall ist. Für das RCut-Verfahren gilt dies ebenfalls. Für die Berechnungen mit Hilfe des exakten Algorithmus wurden ganzzahlige Last- bzw. Kapazitätswerte im Bereich von 0 bis 99 verwendet. Die gemessenen Zeiten sind für dieses Verfahren unter dieser Voraussetzung zu betrachten. Die Ergebnisse zeigen, daß das exakte Verfahren eine um Größenordnungen höhere Berechnungsdauer impliziert als die heuristischen Verfahren. Von diesen bedingt das RCut-Verfahren die geringere Berechnungsdauer, liefert aber auch die schlechtesten MRU-Ergebnisse.

Anzahl Schnitte		Anzahl Satelliten					Anzahl Satelliten					Anzahl Satelliten			
		2	**4**	**8**	**16**		**2**	**4**	**8**	**16**		**2**	**4**	**8**	**16**
2	Enum	100	100	100	X		100	100	100	X		0	14	3113	X
	RCut	129	137	146	154		115	116	117	117		0	0	0	0
	Sq	*101*	*101*	*100*	*100*		*100*	*100*	*100*	*100*		*0*	*0*	*1*	*2*
4	Enum	100	100	X	X		100	100	X	X		8	2895	X	X
	RCut	143	156	170	185		130	135	137	137		0	0	0	1
	Sq	*103*	*100*	*100*	*100*		*101*	*100*	*100*	*100*		*0*	*1*	*2*	*4*
8	Enum	100	X	X	X		100	X	X	X		114	X	X	X
	RCut	149	160	179	193		148	149	153	156		0	0	1	1
	Sq	*103*	*100*	*100*	*100*		*101*	*100*	*100*	*100*		*1*	*3*	*5*	*10*

Durchschnittliche relative MRU („Random Component Load“)

Durchschn. relative MRU („Random Cut Load“)

Berechnungsdauer (in msec)

Abb. 54: *Vergleich der Meßergebnisse (3 Ressourcen pro Computer)*

Die Ergebnisse für den Fall von drei betrachteten Ressourcen pro Rechner sind in Abb. 54 dargestellt. Das zuvor betrachtete exakte Verfahren wurde nicht mehr eingesetzt, da es bereits für den Fall von zwei Ressourcen eine Berechnungsdauer zumindest im Bereich von (vielen) Minuten bedingte. In einfachen Fällen wurde per Enumeration aller Schnittkombinationen die optimale Lösung bestimmt. In diesen Fällen zeigen die Messungen erneut, daß der SQ-Algorithmus Ergebnisse liefert, die weniger als 3% vom Optimum abweichen. In allen Fällen war der SQ-Algorithmus dem RCut-Ansatz deutlich überlegen. Er bedingte zudem eine Berechnungsdauer, die weniger als 10 msec für alle betrachteten Fälle betrug.

Messungen wurden für weitere Szenarien mit bis zu vier Ressourcen pro Computer und erweiterten Wertebereichen zur Auswahl der Last- und Kapazitätswerte vorgenommen. In allen Fällen konnten die oben getroffenen Aussagen qualitativ und quantitativ (mit unerheblichen Abweichungen) bestätigt werden.

7.3.4 Anwendung auf das erweiterte Host-Satellite-Problem

Für den Fall, daß die Komponentenketten nicht im voraus auf Satelliten verankert sind, enthält eine in der Problemdarstellung gegebene Plazierungskette alle Plazierungsalternativen, die für die entsprechende Komponentenkette möglich sind. Kann eine Komponentenkette auf einem von s Satelliten verankert werden und hat sie m mögliche Schnitte, so sind insgesamt s*m Alternativen gegeben. In der Problemdarstellung sind für die Kette entsprechend s*m mögliche Schnitte enthalten.

Das vorgestellte exakte Verfahren ist im Prinzip auf diesen Fall übertragbar. Allerdings können die Ketten nicht im voraus satellitenweise gruppiert werden, da unterschiedliche Schnitte unterschiedliche Zuordnungen zu Satelliten bedingen. Bei der Bildung der Schnittabellen einer Kette müssen daher Auswirkungen auf die Ressourcenlage des Hosts und aller Satelliten angenommen werden, auf denen die Kette plaziert werden kann. Um diese einzufangen, muß eine Schnittabelle (infolgedessen auch eine Schichttabelle) insgesamt $V^{(s+1)*r}$ Einträge haben. Zu ihrer Berechnung wäre eine Komplexität der Ordnung $o(V^{(s+1)*r})$ gegeben, was in der Praxis selbst für den kleinsten Problemfall (1 Ressource pro Computer, 2 Satelliten) zu unvertretbaren Berechnungsdauern in der Größenordnung von Stunden führt.

Der SQ-Algorithmus kann unverändert auf das Problem angewandt werden. Da nun s*m alternative Schnitte für eine Kette zu betrachten sind, erhöht sich der Aufwand für die Berechnung eines besten oder schlechtesten Schnitts in der Initialisierungs- bzw. der Iterationsphase. Da die Schnittanzahl quadratisch in die Berechnungskomplexität eingeht (siehe 7.3.2), erhöht sie sich um den Faktor s^2 zu $o(r*k*s^2*m^2)$. Die Berechnungsdauern aus Abb. 53 und Abb. 54 können unter Berücksichtigung dieses Faktors für das erweiterte Problem übernommen werden. Eine vorgegebene maximale Berechnungsdauer von 1 sec würde nach wie vor die Anwendung des Verfahrens für die Problemfälle von 7.3.3 erlauben mit bis zu 32 Komponentenketten, die je 4 Komponenten pro Kette enthalten und die beliebig auf 16 Satelliten verankert werden können.

In Abb. 55 sind die Meßergebnisse dargestellt, die die Güte des SQ-Algorithmus in Bezug auf das erweiterte HS-Problem beschreiben. Den Messungen liegen die Last- und Kapazitätsvorgaben des vorigen Abschnitts zugrunde. In den Meßreihen (von je 300 Messungen) wurde die Zahl verwendbarer Satelliten variiert. In Bezug auf die Zahl der Ketten wurden zwei Fälle unterschieden. In einem Fall war die Zahl der Ketten gleich der Zahl der Satelliten, im zweiten Fall war sie doppelt so hoch, um eine dichte „Packung“ der Ketten auf die Satelliten zu erzielen. Als Vergleichsbasis für die Messungen wurden die MRU-Zahlen verwendet, die per Enumeration berechnet wurden.

3 Ressourcen / Computer, MinL = 100, MaxL = 300, MinC = 100, MaxC = 300

Anzahl Schnitte	Anzahl Sat. (= Anz. Ketten)		Anzahl Sat. (= 1/2 Anz. K.)	
	2	**4**	**2**	**4**
2	105	108	105	108
4	104	106	105	X

Durchschnittliche relative MRU
(„Random Component Load“)

Anzahl Schnitte	Anzahl Sat. (= Anz. Ketten)		Anzahl Sat. (=1/2 Anz. K.)	
	2	**4**	**2**	**4**
2	103	106	106	106
4	104	107	107	X

Durchschnittliche relative MRU
(„Random Cut Load“)

Abb. 55: *Vergleich der Meßergebnisse (3 Ressourcen pro Computer)*

Die Ergebnisse zeigen, daß die durchschnittliche Abweichung vom Optimum gegenüber dem Host-Satellite-Problem mit fester Verankerung auf den Satelliten zunimmt. Jedoch ist diese mit maximal 8 % immer noch eng begrenzt. Die Referenzwerte für die optimale MRU konnten per Enumeration nur für (gegenüber 7.4.3) stärker eingeschränkte Problemfälle berechnet werden. Dementsprechend sind die Aussagen über die Güte von SQ nur für kleinere Problemgrößen abgesichert.

7.4 Diskussion

Die Evaluierung des SQ-Algorithmus für das HS-Problem zeigt, daß eine Berechnung einer Plazierung für einen vorgegebenen Dienstgütewert selbst für größere Flußgraphen in wenigen

Millisekunden möglich ist. Die dabei erzielte Lösungsgüte, die in Form der erzielten MRU gegeben ist, weicht im Durchschnitt nur unwesentlich von dem optimal erzielbaren Wert ab. Dementsprechend ist der SQ-Algorithmus, außer in Fällen äußerster Ressourcenknappheit, in der Lage, eine Plazierung zu finden, sofern es eine Plazierung gibt. Die geringe Rechendauer ermöglicht ferner die mehrfache Anwendung des Algorithmus für verschiedene Dienstgütewerte. Sind z. B. 8 (2^3) Dienstgütestufen möglich, so liegt die erforderliche Rechendauer für alle betrachteten Problemfälle unter 30 (3*10) msec. Die Anwendung des exakten Verfahrens bringt im Vergleich zu SQ wenig Gewinn, erfordert aber ein mehrfaches an Rechenzeit. Sein Nutzen bleibt daher auf eine Verwendung als Vergleichsbasis für den SQ-Algorithmus beschränkt.

Wird der SQ-Algorithmus auf das verallgemeinerte Host-Satellite Problem angewandt, so nimmt die erforderliche Rechenzeit im Quadrat der für eine Kette verfügbaren Satelliten zu. Für einen weiten Bereich der betrachteten Problemfälle liegt sie unterhalb einer Sekunde. Eine wiederholte Anwendung des Verfahrens für mehrere Dienstgütestufen ist insofern auch für die umfangreichsten Problemfälle in wenigen Sekunden durchführbar. Die Bewertung von SQ im Vergleich zu einem optimalen Verfahren zeigt eine Verschlechterung gegenüber dem Fall vorgegebener Zuordnung der Ketten zu den Satelliten, wobei die durchschnittliche Abweichung vom Optimum auch in diesem Fall nicht mehr als 8% beträgt.

Das Finden einer Plazierung erfordert die Verwendung eines Plazierungsalgorithmus durch die Hosts. Das FPP-Protokoll ist nicht von einem bestimmten Algorithmus abhängig. Genauso kann jeder Host einen eigenen Algorithmus verwenden. Durch seinen Ablauf ermöglicht FPP die gleichzeitige Lösung mehrerer Host-Satellite-Probleme, wobei die Durchführung der Berechnungen maximal verteilt ist. So sind die Koordinatoren der Satelliten für die Berechnung der satellitenseitigen Lastvektoren zuständig sowie für ihre Normierung bezüglich der lokal abgefragten verfügbaren Kapazitäten. Der Koordinator eines potentiellen Hosts führt die Plazierungsberechnung (mit Hilfe von SQ) in Bezug auf das Host-Sattelite-Problem durch, das den eigenen Host involviert. Dies stellt insbesondere sicher, daß FPP hinsichtlich erforderlicher Rechenzeit unabhängig von der Zahl potentieller Hosts ist.

FPP wird unabhängig von einer Ressourcenreservierung durchgeführt. Dies vermeidet die Verzögerung, die durch eine Belegung der Ressourcen durch Komponenten und Links für die Dauer der Plazierung entstehen würde. Sollte eine Plazierung nicht wie berechnet durchgeführt

werden können, weil entweder die Plazierungsinformation ungenau war oder sich während der Plazierungsberechnung verändert hat, so kann in zweifacher Weise darauf reagiert werden. Eine Reaktion besteht darin, die Plazierung beizubehalten und die Dienstgüte entsprechend der vorgefundenen Ressourcen zu belegen. Kann dies nicht erfolgen, so kann als zweite Reaktion eine erneute Plazierung versucht werden. Konnte die erste Plazierung nicht stattfinden, weil ein Host nicht ausreichend Ressourcen zur Verfügung hatte, so muß der ZK lediglich den Host bestimmen, der die nächstbeste Dienstgüte gemeldet hatte. Eine Neuberechnung der Plazierung ist nicht nötig. Konnte dagegen ein Satellit nicht wie erwartet benutzt werden, so muß eine Neuberechnung (evtl. unter Ausschluß des Satelliten) durchgeführt werden. Eine neue Plazierung kann im Prinzip berechnet werden, bis durch Ausschluß kein (unbedingt benötigter) Host oder Satellit mehr zur Verfügung steht.

8 Zusammenfassung und Ausblick

Die Integration von Audio und Video in rechnerbasierte Anwendungen führt zu einer Reihe von Anforderungen, die nicht durch Mechanismen gelöst werden können, die für herkömmliche Daten wie Text und Graphik entwickelt wurden. Sie stellen zeitabhängige Medien dar und können hohe Datenraten bedingen, die die Verfügbarkeit entsprechender Ressourcen erfordert. Insbesondere ist ihre Präsentation unter Einhaltung einer Dienstgüte sicherzustellen, die sowohl medienspezische Eigenschaften als auch eine begrenzte Verzögerung zwischen ihrer Erzeugung und Darstellung erfaßt.

Im Rahmen dieser Abhandlung wurden Konzepte und Verfahren vorgestellt, die die Bereitstellung von Dienstgüte für Multimedia-Anwendungen ermöglichen. Sie wurden als Teil der multimedialen Systemplattform CINEMA entwickelt, die den Entwurf und die Benutzung von Anwendungen durch multimediale Systemdienste unterstützt. Zur Bereitstellung von Dienstgüte wurden zwei Dienste realisiert, ein Dienst zur dienstgüteoptimierten Plazierung einer Anwendung sowie ein Dienst zum Aufbau einer Reservierungssession zur Erbringung einer gleichbleibenden Dienstgüte während der Benutzung einer Anwendung.

Multimedia-Anwendungen wurden in Form von Flußgraphen betrachtet. Diese realisieren Verarbeitungstopologien, die aus Quellen und Senken, zwischengelagerten Komponenten sowie Kommunikations-Links bestehen. Zur Berücksichtigung funktionaler Einschränkungen von Komponenten wurden Vorgaben in Form von Formatbeschränkungen sowie Stromrelationen eingeführt. Insbesondere wurden variable Filter vorgesehen, die eine entkoppelte Bereitstellung von Dienstgüte in verschiedenen Teilen eines Flußgraphen ermöglichen.

Das Konzept einer Session wurde als zentrale Abstraktion zur Benutzung der Systemdienste definiert. Sie stellt die Einheit zur Plazierung bzw. Ressourcenallokation für einen Flußgraphen dar. Anforderungen an eine Session werden durch einen Klienten spezifiziert, während die Instanziierung des Flußgraphen gemäß der Anforderungen durch die multimedialen Systemdienste durchgeführt wird.

Der Entwicklung der Systemdienste wurde eine Dienstgütearchitektur zugrundegelegt, die verschiedene Abstraktionsebenen der Dienstgütespezifikation unterscheidet. Auf der Systemebene

setzen die Dienste Mechanismen voraus, die die Reservierung einzelner Ressourcen auf Endsystemen sowie im Netzwerk zulassen. Auf der Anwendungsebene realisieren sie Protokolle, die medienspezische und generische Parameter erfassen und eine Task-Plazierung bzw. Ressourcenreservierung für die Elemente eines Flußgraphen durchführen.

Das im Rahmen dieser Arbeit vorgestellte NRP-Protokoll bildet die Kernfunktionalität des Dienstes zur Ressourcenreservierung auf der Anwendungsebene. Indem es auf Flußgraphen anwendbar ist, die einem Zweizonenmodell entsprechen, ist es in vielfältigen Szenarien einsetzbar. Hierzu gehören insbesondere Szenarien mit Verarbeitung und Kommunikation multimedialer Daten, die in Multicast-, Misch- bzw. Konferenzkonfigurationen vorkommen können. Die Komponenten des Flußgraphen können dabei beliebig auf den Rechnern eines verteilten Systems plaziert sein.

Durch die Unterstützung variabler Filter ist NRP in der Lage das Konzept „heterogener Empfänger“, welches früher für Netzwerkprotokolle realisiert wurde, auf die Anwendungsebene zu übertragen. Dementsprechend läßt es individuelle Dienstgütevorgaben an den Senken eines Flußgraphen zu. Für jede Senke erlaubt es die Spezifikation eines Wertebereichs für einen medienspezifischen Parameter bzw. einer oberen Grenze für die „Ende-zu-Ende“-Verzögerung, die sich zwischen Erzeugung und Präsentation multimedialer Information ergibt.

NRP führt die Ressourcenreservierung für einen Flußgraphen unter Beachtung der Ressourcenverfügbarkeit auf Endsystemen und im Netzwerk sowie der Formatbeschränkungen und Stromrelationen der Flußgraphenkomponenten durch. Die Berücksichtigung dieser Aspekte unterscheidet NRP von früheren Ansätzen, die weder heterogene Empfänger noch funktionale Einschränkungen betrachtet und häufig weitgehende Einschränkungen in Bezug auf Gestalt oder Verteilung eines Flußgraphen vorausgesetzt haben.

NRP stellt sicher, daß eine Session aufgebaut wird, die keine der erwähnten Randbedingungen verletzt. Es wurde nachgewiesen, daß NRP an den Senken eines Flußgraphen die bestmögliche medienspezifische Dienstgüte einstellt, sofern keine Ressourcenkonflikte auftreten. Zur optimierten Behandlung von Ressourcenkonflikten wurden zwei Erweiterungen von NRP beschrieben. Zur Minimierung des Ressourcenbedarfs wurde das NRP-Min-Protokoll definiert, welches eine Ressourcenreservierung durchführt, die der minimal möglichen Dienstgüte an den Senken

entspricht. Die Ausführung von NRP-Min ist für den Fall sinnvoll, daß eine Session mit nicht-minimaler Dienstgüte mit Hilfe von NRP nicht aufgebaut werden kann.

Um eine Minimierung der Dienstgüte zu vermeiden, wurde eine graduelle Absenkung der Dienstgüte durch eine lokale Koordination der Ressourcenvergabe auf Endsystemen realisiert. Das NRP-LK-Protokoll integriert diese Koordination und ist dadurch in der Lage, Ressourcenkonflikte auf Endsystemen optimal aufzulösen, sofern im Flußgraphen keine variablen Filter enthalten sind. Werden benötigte Netzwerkverbindungen für einen Flußgraphen im voraus mit Hilfe von NRP aufgebaut, so garantiert NRP-LK die Einstellung der bestmöglichen Dienstgüte, die aufgrund der Netzwerkverbindungen und der Ressourcenverfügbarkeit der Endsysteme möglich ist.

Die Implementierung des NRP-Protokolls zeigt, daß ein Session-Aufbau mit Hilfe der entwikkelten Protokolle für eine Vielzahl betrachteter Flußgraphen eine geringe Zeit in Anspruch nimmt. Die Evaluierung des mit NRP verbundenen Aufwands zeigt ferner, daß der höchste Anteil an dieser Verzögerung durch den Aufbau von Netzwerkverbindungen bedingt wird. Der Entwurf von NRP stellt sicher, daß dieser Aufwand höchstens zweimal auftritt, unabhängig von der Zahl der Stufen, die für eine medienspezifische Dienstgüte in Frage kommen.

Das Problem der dienstgüteoptimierten Plazierung eines multimedialen Flußgraphen in einem verteilten System wurde erstmals im Rahmen dieser Arbeit ausführlich betrachtet. Zu dessen Definition wurden Anforderungen identifiziert, die spezifisch für multimediale Flußgraphen sind und im Rahmen früherer Plazierungsprobleme nicht erfaßt wurden. Eine Anforderung ergibt sich daraus, daß Komponenten zur Kompression/Dekompression plazierungsabhängig eingefügt werden müssen. Eine weitere Anforderung ist durch die Notwendigkeit gegeben, den Ressourcenbedarf von Flußgraphen anhand von Verarbeitungs-Threads zu ermitteln, die mehrere Komponenten umfassen können. Schließlich muß für eine geeignete Plazierung die Verfügbarkeit verschiedener Ressourcen auf Endsystemen sowie im Netzwerk beachtet werden.

Um diesen Anforderungen zu genügen, wurde das Plazierungsproblem im Kontext eines Host-Satellite-Modells betrachtet, das sternförmige Flußgraphen zuläßt, deren Komponentenketten auf Satelliten- bzw. einem Host-Rechner zu plazieren sind. Das Modell wurde zur Definition eines entsprechenden Host-Satellite-Plazierungsproblems verwendet, wobei als Ziel das Finden

einer Plazierung angestrebt wurde, die eine möglichst hohe Dienstgüte ermöglicht. Es wurde gezeigt, daß dieses Ziel durch das Finden der Plazierung erreicht werden kann, die die maximal erforderliche Beanspruchung aller benötigten Ressourcen minimiert.

Als Kernbestandteil des Systemdienstes zur Task-Plazierung wurde das FPP-Protokoll entwikkelt, welches das Host-Satellite-Plazierungsproblem löst. Es wird durch Plazierungskoordinatoren auf den Rechnern des verteilten Systems ausgeführt, die Information über den Ressourcenbedarf von Komponentenketten auf den Rechnern bereitstellen können. Auf Host-Rechnern sind Koordinatoren vorgesehen, die die Plazierungsinformation zum Bestimmen der bestmöglichen Plazierung verwenden. Hierzu können sie einen von zwei im Rahmen dieser Arbeit entwickelten Verfahren einsetzen.

Ein exaktes Verfahren kann verwendet werden, sofern das Plazierungsproblem als ein ganzzahliges Problem mit Zahlen aus einem beschränkten Wertebereich betrachtet wird. Ein zweites, heuristisches Verfahren weist diese Einschränkung nicht auf. Beide Verfahren wurden anhand einer Vielzahl von Problemfällen unterschiedlicher Größe evaluiert. Dabei erforderte das exakte Verfahren einen sehr hohen Zeitaufwand, während der Aufwand für das heuristische Verfahren nicht über wenige Millisekunden hinausging. Hinsichtlich der Güte der gefundenen Plazierung war das heuristische Verfahren nur unwesentlich schlechter als das exakte Verfahren.

Die Plazierung eines Flußgraphen wurde unabhängig von einer Ressourcenreservierung betrachtet. Der Grund hierfür besteht darin, daß die Information, die zur Auswahl einer Plazierung verwendet wird, zwar als gute Schätzung, nicht jedoch als absolut genaue Beschreibung der Ressourcenverfügbarkeit im verteilten System vorausgesetzt werden kann. Im Anschluß an eine Plazierung eines Flußgraphen durch das FPP-Protokoll ist daher eine Instanziierung des Flußgraphen durch ein Protokoll wie NRP erforderlich, welches benötigte Ressourcen reserviert. Auf diese Weise ist sichergestellt, daß in jedem Fall eine Dienstgüte eingestellt wird, für die Ressourcen in ausreichendem Maß vorhanden sind.

Mit der Integration der beschriebenen Protokolle und Verfahren in multimediale Systemdienste wurde eine Funktionalität bereitgestellt, mit der Klienten die Plazierung und Instanziierung eines Flußgraphen gemäß Dienstgütevorgaben auf eine einfache Art bewerkstelligen können.

Damit sind für einen Kernbereich multimedialer Systemunterstützung Lösungen verfügbar, die für ein breites Spektrum multimedialer Anwendungen eingesetzt werden können.

Eine Reihe von Aspekten der Task-Plazierung bzw. der Ressourcenreservierung sollten im Rahmen zukünftiger Arbeiten untersucht werden. Als eine direkte Weiterführung dieser Arbeit sollte beispielsweise die Frage geklärt werden, inwieweit die lokale Koordination, die als Teil des NRP-LK-Protokolls definiert wurde, eine Optimierung der Ressourcenvergabe für Flußgraphen erreichen kann, die variable Filter enthalten.

Die vorgestellten Lösungen sind von einem Flußgraphen als der zu betrachtenden Einheit zur Ressourcenallokation ausgegangen. Um dynamische Änderungen des Flußgraphen, die z. B. die Hinzunahme neuer Quellen oder Senken beinhalten, durch Ressourcenreservierung zu unterstützen, sind ergänzende Konzepte erforderlich, die eine geeignete Aufteilung verfügbarer Ressourcen bewirken. Dies gilt auch für den Fall, daß mehrere Flußgraphen nacheinander instanziiert werden, deren Dienstgüte nicht unabhängig voneinander eingestellt werden soll.

Die beschriebenen Protokolle zur Ressourcenreservierung haben die Berechnung der „Ende-zu-Ende“-Verzögerung integriert. Die Behandlung weiterer Parameter wie z. B. Jitter oder Datenverlust blieb dagegen unberücksichtigt. Ihre Integration erfordert, analog zur Behandlung der Verzögerung, eine genaue Festlegung einer jeweiligen „Ende-zu-Ende“-Semantik, die alle entstehenden Anteile berücksichtigt und eine entsprechende Aggregation für einen Flußgraphen ermöglicht. Die Definition einer solchen Semantik ist gegenwärtig nicht gegeben.

Die dargestellten Lösungen zur Task-Plazierung wurden im Hinblick auf das Host-Satellite-Plazierungsproblem entwickelt. Weitergehende Verfahren sind erforderlich, falls die hierbei gemachten Annahmen über die Art der zu plazierenden Flußgraphen (sternförmige Flußgraphen ohne variable Filter) und des benutzten verteilten Systems (Satelliten- und Host-Rechner) gelockert werden. Die Lösung des erweiterten Host-Satellite-Problems in dieser Arbeit zeigt hierbei eine mögliche Vorgehensweise auf.

Wie bei der Durchführung einer Ressourcenreservierung könnten bei der Plazierung eines Flußgraphen weitere Randbedingungen formuliert werden. Insbesondere sollte die Berücksichtigung finanzieller Kosten erwogen werden, um eine möglichst kostengünstige Verarbeitung bzw. Kommunikation multimedialer Daten zu erreichen.

Anhang A. Literaturverzeichnis

[AhLi97] G. Ahanger, T. D. C. Little. A System for Customized News Delivery from Video Archives. *IEEE International Conference on Multimedia Computing and Systems*, Ottawa, Canada, 6 1996.

[AMZ95] E. Amir, S. McCanne, H. Zhang. An Application Video Level Gateway. *Third ACM International Multimedia Conference*, San Francisco, 11 1995.

[Ande93] D. P. Anderson. Metscheduling for Continuous Media. *ACM Transactions on Computer Systems*, Vol. 11, No. 3, pp. 226-252, 8 1993.

[ATM96a] The ATM Forum, Technical Committee: Native ATM Services: Semantic Description, Version 1.0, 2 1996.

[ATM96b] The ATM Forum, Technical Committee: Traffic Management Specification, Version 4.0, 4 1996.

[ATM94] The ATM Forum, Technical Committee: ATM User-Network Interface Specification, Version 3.1, 9 1994.

[ATWG90] D. Anderson, S.-Y. Tzou, R. Wahbe, R. Govindan, M. Andrews. Support for Continuous Media in the Dash System. *Tenth International Conference on Distributed Computing Systems*, Paris, France, 5 1990.

[Bart94] I. Barth. Extending the Rate Monotonic Scheduling Algorithm to Get Shorter Delays. *International Workshop on Advanced Teleservices and High-Speed Communication Architectures, Heidelberg, Germany*, S. 104–114, 9 1994.

[Bart96] I. Barth. Configuring Distributed Multimedia Applications Using CINEMA. *IEEE Workshop on Multimedia Software Development*, Berlin, 3 1996.

[Bart98] I. Barth. Management-Architektur für verteilte Multimedia-Anwendungen. Dissertation. *Universität Stuttgart, Fakultät Informatik*, 1998.

[BCS94] R. Braden, D. Clark, S. Shenker. Integrated Services in the Internet Architecture: An Overview. *RFC 1633*, 6 1994.

[BDF95] I. Barth, G. Dermler, W. Fiederer. Levels of Quality of Service in CINEMA. *GISI 95: Herausforderungen eines globalen Informationsverbundes für die Informatik*, Zürich, 9 1995.

[BDHR93] I. Barth, G. Dermler, T. Helbig, K. Rothermel, F. Sembach, T. Wahl. CINEMA: Eine konfigurierbare, integrierte Multimedia-Architektur. *GI/ITG-Arbeitstreffen Verteilte Multimedia Systeme, Stuttgart*, S. 33–47, 2 1993.

[Bett95] R. Bettati et al. Connection Establishment for Multi-Party Real-Time Communication. *Fifth Internationational Workshop on Network and Operating System Support for Digital Audio and Video, Durham, NH*, 4 1995.

[BHR95] I. Barth, T. Helbig, K. Rothermel. Implementierung multimedialer Systemdienste in CINEMA. *GI/ITG-Fachtagung Kommunikation in Verteilten Systemen KiVS'95, Chemnitz*, S. 173–187, 2 1995.

[BiEl95] A. Billionnet, S. Elloumi. An Algorithm for Finding the k-best Allocations of a Tree Structured Program. *Journal of Parallel and Distributed Computing*, 26, pp. 225-232, 1995.

[Bill94] A. Billionnet. Allocating Tree Structured Programs in a Distributed System with Uniform Communication Costs. *IEEE Transactions on Parallel and Distributed Systems*, Vol. 5, No. 4, 4 1994.

[BLMS96] H. Bryhni, H. Lovett, E. Maartmann-Moe, D. Solvoll, T. Sorenson. On-Demand Regional Television over the Internet. *Fourth ACM International Multimedia Conference*, Boston, Nov. 1996.

[BNG92] N.S. Bowen, C.N. Nikolau, A. Ghafoor. On the Assignment Problem of Arbitrary Process Systems to Heterogeneous Distributed Computer Systems. *IEEE Tranactions on Computers*, Vol. 41, No. 3, pp. 257-273, 1992.

[Bokh88] S. Bokhari. Partitioning Problems in Parallel, Pipelined, and Distributed Computing. *IEEE Transactions on Computers*, Vol. 37, No. 1, pp 48-57, Jan 1988.

[BoFr93] N. Borenstein, N. Freed. MIME (Multipurpose Internet Mail Extensions): Mechanism for Specifying and Describing the Format of Internet Message Bodies. *Internet RFC 152*, Sep 1993.

[BrSt97] T. Braun, H. Stüttgen. Die Internet-Architektur für integrierte Multimediakommunikation. *itti 4/97, Schwerpunktthema: Neue Kommunikationsarchitekturen*, 8 1997.

[BrZi96] T. Braun, M. Zitterbarth. Hochleistungskommunikation Band 2: Transportdienste und Protokolle. *Oldenbourg Verlag*, 1996.

[BuLi91] D. C. Bulterman, R. v. Liere. Multimedia Synchronisation and UNIX. *Second International Workshop on Network and Operating System Support for Digital Audio and Video*, 11 1991.

[CaCo96] A. Campbell, G. Coulson: A QoS Adaptive Transport System: Design, Implementation and Experience. *ACM Multimedia 96*, S. 117-127, Boston, 1996.

[CBRS93] G. Coulson, G. Blair, P. Robin, D. Shepherd. Extending the Chorus Micro-Kernel to support Continuous Media Applications. *Fourth Internationational Workshop on Network and Operating System Support for Digital Audio and Video, Lancaster, UK*, 1993.

[CSZ92] David D. Clark, Scott Shenker, Lixia Zhang. Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism. *SIGCOMM'92 Communications Architectures and Protocols*, S. 14–26, Boston 08 1992.

[DeBe95] L. Delgrossi, L. Berger. Internet Stream Protocol, Version 2, *RFC 1819*, 11 1995.

[DeIq98] G. Dermler, A. Iqbal. Task Allocation in Distributed Multimedia Systems based on the Host-Sattelite Model. *Fakultätsbericht 7/1998*, Universität Stuttgart, IPVR.

[DFBR95] G. Dermler, W. Fiederer, I. Barth, K. Rothermel. A Negotiation and Resource Reservation Protocol (NRP) for Distributed Multimedia Applications. *Fakultätsbericht 11/1995*, Universität Stuttgart, IPVR, 12 1995.

[DFR96] G. Dermler, W. Fiederer, I. Barth, K. Rothermel. A Negotiation and Resource Reservation Protocol (NRP) for Distributed Multimedia Applications. *IEEE International Conference on Multimedia Computing and Systems*, Hiroshima, Japan, 6 1996.

[DeFr93] G. Dermler, K. Froitzheim. JVTOS - A Reference Model for a New Multimedia Service. *Fourth IFIP Conference on High-Speed Networking, Liege, Belgium, 11 1993.*

[DGPR93] G. Dermler, T. Gutekunst, B. Plattner, F. Ruge, M. Weber. Constructing a Distributed Multimedia Joint-Viewing and Tele-Operation Service for Heterogeneous Workstation Environments. *Fourth Workshop on Future Trends in Distributed Systems,* Lisbon, Portugal, 9 1993.

[DGOP94] G. Dermler, T. Gutekunst, E. Ostrowski, N. Pires, T. Schmidt, H. Wolf. A Platform for Bridging Endsystem Heterogeneity. *International COST 237 Workshop on Multimedia Transport and Teleservices,* Vienna, Austria, 11 1994.

[DGOP94b] G. Dermler, T. Gutekunst, E. Ostrowski, N. Pires, T. Schmidt, M. Weber, H. Wolf, K. Froitzheim. Telecooperation by Using the Multimedia Teleservice JVTOS. *Conference Notes of NETWORLD+INTEROP ´94, Berlin, Germany, 6 1994*.

[DGOR94] G. Dermler, T. Gutekunst, E. Ostrowski, F. Ruge. Sharing Audio/Video Applications among Heterogeneous Platforms. *Fifth IEEE COMSOC International Workshop on Multimedia Communication*, Kyoto, Japan, 5 1994.

[DeKr75] F. DeRemer, H. Kron. Programming-in-the-large versus Programming-in-the-small. *Conference on Reliable Software*, pp. 114-121, 1975.

[DVV94] D. Deloddere, W. Verbiest, H. Verhille. Interactive Video on Demand. *IEEE Communications*, Vol. 32, No. 5, pp. 82-88, 5 1994.

[Efe82] K. Efe. Heuristic Models of Task Assignment Scheduling in Distributed Systems. *IEEE Computer*, pp. 50-56, June 1982.

[FDR97] W. Fiederer, G. Dermler, K. Rothermel. Dienstgüte-Management im Systemkonzept CINEMA. *itti 4/97, Schwerpunktthema: Neue Kommunikationsarchitekturen*, 8 1997.

[Fern89] D. Fernandez-Baca. Allocating Modules to Processors in a Distributed System. *IEEE Trans. on Software Engineering*, Vol. 15, No. 11, 11 1989.

[FiDe96] W. Fiederer, G. Dermler. A Protocol for Negotiation of Quality of Service in Distributed Multimedia Systems. *Fifth Open Workshop on High Speed Networks*, Paris, 03 1996.

[Fied98] W. Fiederer. Dienstgüte- und Ressourcenmanagement in verteilten Multimedia-Systemen. Vorgelegte Dissertation. *Universität Stuttgart, Fakultät Informatik*, 1998.

[FSVW96] M. Fry, A. Seneviratne, A. Vogel, V. Witana: Delivering QoS Controlled Continuous Media on the World Wide Web. *International Workshop on QoS ´96*, Hrsg. J. de Meer und A. Vogel, GMD-Studien Nr. 282, S. 115-124, Sankt Augustin 1996.

[Garg 95] Rahul Garg. Characterization of Video Traffic. Technical Report TR-95-007, *Department of Electrical Engineering and Computer Science, University of California at Berkeley*, 1995.

[Gosc91] A. Goscinski. Distributed Operating Systems: The Logical Design. *Addison-Wesley Publishing Company*, 1991.

[Hafi95] Abdelhakim Hafid. Hierarchical Negotiation for Distributed Multimedia Applications in A Multi-Domain Environment. *Second Workshop on Protocols for Multimedia Systems (PROMS)*, Salzburg, 10 1995.

[HaLi92] P. Hansen, K.W. Lih. Improved Algorithms for Partitioning Problems in Parallel, Pipelined and Distributed Computing. *IEEE Tranactions. on Computers*, Vol. 41, No. 6, pp. 769-771, 6 1992.

[Helb96] T. Helbig. Kommunikation und Synchronisation multimedialer Datenströme in verteilten Systemen. Dissertation. *Universität Stuttgart, Fakultät Informatik*, 1996.

[Herr91] Ralf Guido Herrtwich. Time Capsules: An Abstraction for Access to Continuous-Media Data. *The Journal of Real-Time Systems, Kluwer Academic Publishers*, S. 355–376, 3 1991.

[HuSc97] T. Hutschenreuther, A. Schill. Quality of Service Abstraktionen für verteilte Multimedia-Systeme. *itti 4/97, Schwerpunktthema: Neue Kommunikationsarchitekturen*, 8 1997.

[HWK96] J. Huang, Y. Wang, D. Kenchammana-Hosekote. Decentralized End-to-End Resource Management for Continous Multimedia. *Sixth International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Japan, 4 1996.

[HWVC97] J. Huang, Y. Wang, N.R. Vaidyanathan, F. Cao. GRMS: A Global Resource Management System for Distributed QoS and Criticality Support. *IEEE International Conference on Multimedia Computing and Systems (ICMCS)*, Ottawa, 06 1997.

[IETF96] Integrated Services Workgroup. Specification of Guaranteed Quality of Service, *Internet Draft*, 08 1996.

[IMA93] IMA. Multimedia System Services Version 1.0. *Verfügbar über FTP: ibminet.awdpa.ibm.com. Hewlett-Packart Comp.*, IBM Corp., SunSoft Inc., 7 1993.

[IqBo95] M.A. Iqbal, S.H. Bokhari. Efficient Algorithms for a Class of Partitioning Problems, *IEEE Transactions on Parallel & Distributed Systems*, vol. 6, no. 2, 2 1995.

[ISO93] ISO/IEC JTC1/SC2/WG11. Coding of Moving Pictures and Associated Audio for Digital Storage Media up to abou 1.5 Mbit/s (MEPG-1). *International Standard ISO/IEC IS 11172*, 1993.

[ISO94] ISO/IEC JTC1/SC2/WG11. Generic Coding of Moving Pictures and Associated Audio (MEPG-2). *International Standard ISO/IEC IS 13818*, 1994.

[GVKR95] D. J. Gemmel, H. M. Vin, D. D. Kandler, P. V. Rangan, L. Rowe. Multimedia storage servers: a tutorial. IEEE Computer, pp. 40-49, 5 1995.

[KHSM95] T. Käppner, F. Henkel, A. Schröer, M. Müller. Eine verteilte Entwicklungs- und Laufzeitumgebung für multimediale Anwendungen. *GI/ITG Fachtagung Kommunikation in Verteilten Systemen*, Chemnitz, 1995.

[KiNa97] K. Kim, K. Nahrstedt. QoS Translation and Admission Control for MPEG Video. *International Workshop on QoS '97,* New York, 5 1997.

[KiPa90] C.H. Lee, M. Kim, C.I. Park. An Efficient K-Way Graph Partitioning Algorithm for Task Allocation in Parallel Computing Systems. *IEEE Computer,* pp. 748-751, 1990.

[KLZ97] Y. Kopidakis, M. Lamari, V. Zissimopoulos. On the Task Assignment Problem: Two New Efficient Heuristic Algorithms. *Journal of Parallel and Distributed Computing*, Vol. 42, pp. 21-29, 1997.

[KrLe91] C. M. Krishna, Y. H. Lee. Real-Time Systems. *IEEE Computer Magazine,* 5 1991.

[LeSh97] C. H. Lee, K. G. Shin. Optimal Task Assignment in Homogeneous Networks. *IEEE Transactions on Parallel and Distributed Systems*, Vol. 8, No. 2, Feb 1997.

[LiLa73] C. L. Liu, J. W. Layland. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. *Journal of the ACM*, Vol. 20, No. 1, S. 46-61, 1973.

[Lind95] C. Lindblad. VuSystem Performance Measurements. *5th International Workshop on Network and Operating System Support for Digital Audio and Video, Durham, New Hampshire,,* 4 1995.

[LLK92] C. H. Lee, D. Lee, M. Kim. Optimal Task Assignment in Linear Array Networks. *IEEE Transactions on Computers,* Vol. 41, No. 7, pp. 877-880, 7 1992.

[LoSh93] P. Lougher, D. Shepherd. The Design of a Storage Server for Continuous Media. *Computer Journal,* Vol. 36, No. 1, pp. 32-40, 2 1993.

[LSD89] J. Lehoczky, L. Sha, Y. Ding. The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behaviour. *IEEE Real Time Systems Symposium*, 1989.

[Lo88] V.M. Lo. Heuristic Algorithms for Task Assignment in Distributed Systems. *IEEE Transactions on Computers*, Vol. 37, No. 11, pp. 1384-1397, 11 1988.

[Mode94] Cathérine Model. Speicherverwaltung für Multimedia im Cinema-Projekt. *Diplomarbeit*, Universität Stuttgart, 12 1994

[MST94] C. W. Mercer, S. Savage, H. Tokuda. Processor Capacity Reserves: Operating System Support for Multimedia Applications. *IEEE Int. Conference on Multimedia Computing and Systems*, Boston, USA, 5 1994.

[NaSm95] K. Nahrstedt, J. Smith. The QoS Broker. *IEEE Multimedia*, Vol. 2, No. 1, pp. 53-67, 1 1995.

[NaTo97] R. Nakatsu, N. Tosa. Towards the Realization of Interactive Movies - Inter Communication Theater: Concept and System. *IEEE International Conference on Multimedia Computing and Systems*, Ottawa, Canada, 6 1997.

[NiHa91] D.M. Nicol, D.R. O'Hallaron. Improved Algorithms for Mapping Pipelined and Parallel Computations. *IEEE Transactions on Computers*, Vol. 40, No. 3, pp. 295-305, 3 1991.

[NoTh93] M.G. Norman, P. Thanisch. Models of Machines and Computation for Mapping in Multicomputers. *ACM Computing Surveys*, Vol. 25, No. 3, 9 1993.

[OlMa95] B. Olstad, F. Manne. Efficient Partitioning of Sequences. *IEEE Transactions on Computers*, Vol. 44, No. 11, pp. 1322-1326, 11 1995.

[PREM96] ISO/IEC JTC1/SC24/N1594: Multimedia System Services, PREMO Part 3, International Standards Organisation, 1996.

[RBH94] K. Rothermel, I. Barth, T. Helbig. CINEMA - An Architecture for Distributed Multimedia Applications. *Architecture and Protocols for High-Speed Networks*, S. 253–271, Kluwer Academic Publishers, 1994.

[RDF97] K. Rothermel, G. Dermler, W. Fiederer. QoS Negotiation and Resource Reservation for Distributed Multimedia Applications. *IEEE International Conference on Multimedia Computing and Systems*, Ottawa, Canada, 6 1997.

[RoDe92] K. Rothermel, G. Dermler. Synchronization in Joint-Viewing Environments. *3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, S. 97–109, 11 1992.

[RoHe95] Kurt Rothermel, Tobias Helbig. An Adaptive Stream Synchronization Protocol. *5th International Workshop on Network and Operating System Support for Digital Audio and Video, Durham, New Hampshire, USA*, S. 189–202, 4 1995.

[RoHe96] Kurt Rothermel, Tobias Helbig. Clock Hierarchies: An Abstraction for Grouping and Controlling Media Streams. *IEEE Journal on Selected Areas in Communications - Synchronization Issues in Multimedia Communications*, 1996.

[Sala97] Andreas Salamanis. Aushandlung von Dienstgüte in Cinema. Diplomarbeit Nr. 1496, *Universität Stuttgart*, 06 1997.

[SBAM96] A. D. Stoyenko, J. Bosch, M. Aksit, T. Marlowe. Load Balanced Mapping of Distributed Objects to Minimize Network Communication, *Journal of Parallel and Distributed Computing*, No. 34, pp. 117-136, 1996.

[SCFJ95] Schulzrinne, Casner, Frederick, Jacobson. RTP: A Transport Protocol for Real-Time Applications. Internet Draft. 03 1995.

[SmPr97] K. Smith, R. Pretty. ATM RendeView: Multipoint Conferencing on ATM. *IEEE International Conference on Multimedia Computing and Systems*, Ottawa, Canada, 6 1997.

[Sren96] C. J. Sreenan. Resource Management System for a Broadband Multipoint Bridge. *IEEE International Conference on Multimedia Computing and Systems*, Hiroshima, Japan, June 1996.

[SSJH96] F. Samaria, H. Syfrig, A. Jones, A. Hopper. Enhancing Network Services through Multimedia Data Analysers. *Fourth ACM International Multimedia Conference*, Boston, Nov. 1996.

[Stall88] W. Stallings. Data and Computer Communications, Second Edition. *Macmillan Publishing Co.*, New York 1988.

[StNa95] R. Steinmetz, K. Nahrstedt. Multimedia: Computing, Communications and Applications. *Prentice Hall, Innovative Technology Series*, 1995.

[Ston77] H.S. Stone. Multiprocessor Scheduling with the Aid of Network Flow Algorithms. *IEEE Transactions on Software Engineering*, Vol. 3, No. 1, Jan 1997.

[Topo90] C. Topolcic. Experimental Internet Stream Protocol, Version 2 (ST-II). *RFC 1190*, 10 1990.

[Tows86] D. Towsley. Allocating Programs Containing Branches and Loops Within a Multiple Processor System. *IEEE Transactions on Software Engineering*, Vol. 12, No. 10, pp. 1018-1024, 10 1986.

[VKBG95] A. Vogel, B. Kerherve, G. v. Buchmann, J. Gecsei. Distributed Multimedia and QoS: A Survey. *IEEE Multimedia Maganzine*, 3 1995.

[Wils91] P. Wilson: Computer Supported Cooperative Work (CSCW): Origins, Concepts and Research Activities. *Computer Networks and ISDN Systems*, Vol. 23, 1991.

[Wolf96] L. Wolf: Resource Management for Distributed Multimedia Systems. *Kluwer Academic Publisher*, Boston/Dordrecht/London, 1996.

[WoMo93] C. Woodside, G. Monforton. Fast Allocation of Processes in Distributed and Parallel Systems. *IEEE Transactions on Parallel and Distributed Systems*, Vol. 4, No. 2, 2 1993.

[WWV94] H. Wittig, L. Wolf, C. Vogt. CPU Utilization of Multimedia Processes: The Heidelberg Predictor of Execution Times (HeiPOET) Measurement Tool. *Second International Workshop on Advanced Teleservices and High-Speed Communication Architectures, Heidelberg, Germany*, 9 1994.

[YaSa93] S.S. Yau, V.R. Satish. A Task Allocation Algorithm for Distributed Computing Systems. *IEEE Computer*, pp. 336-342, 1993.

[YMGH96] N. Yeadon, A. Mauthe, F. Garcia, D. Hutchison. QoS Filters: Addressing the Heterogeneity Gap. *European Workshop on Interactive Distributed Multimedia Systems and Services*, Berlin, Germany, 3 1996.

[YWPS95] S. Yalamanchili, L.T. Winkel, D. Perschbacher, B. Shenoy. Partitioning and Mapping in Embedded Multiprocessor Architectures in the Presence of Constraints. *Concurrency: Practice and Experience*, Vol. 7(3), pp. 167-189, 5 1995.

[ZDES93] L. Zhang, S. Deering, D. Estrin, S. Shenker, D. Zapalla. RSVP: A New Resource Reservation Protocol. *IEEE Network*, 9 1993.