

**Diplomarbeit-Nr. 1496**

**Aushandlung von Dienstgüte in CINEMA**

**Andreas Salamanis**

**1996**

Universität Stuttgart  
Fakultät Informatik

# Kurzfassung

Im Rahmen dieser Arbeit wurden die Protokolle NRP und XNRP in das Cinema-System integriert und die Eigenschaften der Implementierung untersucht. Das Cinema-System ist eine Plattform zur Entwicklung und Steuerung von verteilte Multimedia-Anwendungen. Die Aufgabe der genannten Protokolle ist es, die mögliche Dienstgüte des Systems auszuhandeln und die Reservierung der notwendigen Ressourcen durchzuführen. Bei der Aushandlung von Dienstgüte handelt es sich um einen Abgleich zwischen den QoS-Anforderung von Klienten, der Verfügbarkeit von Betriebsmitteln und der funktionalen Fähigkeit der verteilten Anwendung.

In einem ersten Schritt wurde eine Analyse des Cinema-Systems und der Protokolle durchgeführt. Anhand dieser Analyse wurden verschiedene Implementierungsmöglichkeiten ermittelt und verglichen. Das NRP Protokoll läuft verteilt ab und ließ sich ohne prinzipielle Probleme integrieren. Das XNRP Protokoll enthält eine zentrale Komponente die innerhalb des Systems bekannt sein muß. Diese Anforderung ließ sich erfüllen, indem die Komponente auf dem System plaziert wurde, welches das Protokoll startet.

In den Protokolldefinitionen sind für die Bearbeitung der Protokollaufgaben Protokollagenten vorgesehen. Diese Protokollagenten sind Programme die autonom Aufgaben bearbeiten können. Bei jedem Ablaufen der Protokolle können auf einem Rechner eine unterschiedliche Anzahl von Protokollagenten die Protokollaufgaben parallel bearbeiten. Als Alternative wurde die Bearbeitung der Protokollaufgaben mit nur einem Protokollagenten je Rechner untersucht. Diese Möglichkeit erwies sich als vorteilhaft, da sich Aufgaben wie die Reservierung von Ressourcen nicht parallel durchführen lassen. Weiterhin verringert sich der Verwaltungsaufwand da keine Synchronisation notwendig ist.

Bei der Untersuchung der Protokolleigenschaften zeigte sich eine problemlose Unterstützung verschiedener Medien wie Audio, Video oder Text. Beide Protokolle bieten die Möglichkeit neue Medien hinzuzufügen. Die Protokolle sind in der Lage eine maximale Dienstgüte auszuhandeln und die notwendigen Reservierungen durchzuführen. Für die benötigte Zeit ergab sich eine starke Abhängigkeit der Protokolle von dem Zeitbedarf der Reservierung von Ressourcen wie Speicher, CPU und Bandbreite. Bei der Verwendung der Protokolle in großen Systemen muß daher auf eine effiziente Realisierung der Reservierung von Ressourcen geachtet werden.

# Inhaltsverzeichnis

<b>Kurzfassung</b>	<b>1</b>
<b>Inhaltsverzeichnis</b>	<b>2</b>
<b>Abbildungsverzeichnis</b>	<b>4</b>
<b>Tabellenverzeichnis</b>	<b>5</b>
<b>1. Einführung</b>	<b>6</b>
1.1 Motivation	6
1.2 Aufgabenstellung	7
1.3 Überblick	7
<b>2. Das Cinema System</b>	<b>8</b>
2.1 Applikationsmodell	8
2.2 Cinema Dienste	10
2.2.1 <i>Cinema-Programmierschnittstelle</i>	11
2.2.2 <i>Konfigurationsmanagement</i>	12
2.2.3 <i>Synchronisation und Stromsteuerung</i>	12
2.2.4 <i>Ereignisverwaltung</i>	13
2.2.5 <i>Ressourcenmanagement</i>	13
2.3 Sessionaufbau	14
<b>3. Beschreibung von NRP und XNRP</b>	<b>16</b>
3.1 NRP	18
3.1.1 <i>Erste Phase, Reserve</i>	18
3.1.2 <i>Zweite Phase, Inform</i>	20
3.1.3 <i>Dritte Phase, Relax</i>	20
3.1.4 <i>Einschränkung der Konfigurationen bei NRP</i>	21
3.2 XNRP	22
3.2.1 <i>Erste Phase, Inform</i>	22
3.2.2 <i>Zweite Phase, Reserve</i>	23
3.2.3 <i>Gleichungslöser</i>	24
3.2.4 <i>Dritte Phase, Relax</i>	25
3.3 Zusammenfassung und Analyse der Protokolle	25
<b>4. Architektur des Cinema-Systems</b>	<b>28</b>
4.1 Organisation der Funktionen in Cinema	28
4.2 Thread Struktur	29

<b>5. Implementierung der Protokolle</b>	<b>.30</b>
5.1 Protokollagenten	.30
5.2 Schnittstelle zum Klient	.33
5.3 Kommunikation der Protokollthreads	.34
5.4 Aushandlungsmethoden der Komponenten und Links	.35
5.5 Gesamtablauf	.37
5.5.1 NRP	.38
5.5.2 XNRP	.39
<b>6. Messungen</b>	<b>.42</b>
6.1 Meßschema	.42
6.2 Die Systemumgebung	.43
6.3 Durchführung der Messungen	.43
6.3.1 Messungen einzelner Teile der Implementierung	.44
6.3.2 Die Abhängigkeit des Protokollablaufs von typischen Merkmalen eines Geflechts	.46
6.3.3 Skalierbarkeit der Protokolle	.48
6.3.4 Änderung des Zeitbedarfs bei parallelen Komponenten	.50
6.4 Zusammenfassung und Analyse der Meßergebnisse	.50
<b>7. Bewertung der Protokolle und der Implementierung</b>	<b>.53</b>
7.1 Definition der Anforderungen an die Protokolle	.53
7.2 Lösungskonzepte der Protokolle	.54
7.2.1 Zentrale Instanz	.54
7.2.2 Verteilter Ansatz	.55
7.3 Verwandte Arbeiten	.56
7.4 Bewertung	.57
7.4.1 Implementierung	.57
7.4.2 Protokolle	.58
7.4.3 Zusammenfassung	.59
7.4.4 Ausblick	.60
<b>Anhang A Installation</b>	<b>.61</b>
A.1 Dateien	.61
A.2 Umgebungsvariablen	.62
A.3 Definitionen für die Compilierung von Cinema mit NRP	.62
A.4 Definitionen in den Header-files	.62
A.5 Änderungen an Cinema für die Protokoll Einbindung	.62
A.6 Sonstige Änderungen an Cinema	.63
A.7 Einschränkungen des Cinema-Systems bei Benutzung der Protokolle	.65
A.8 Wichtige Anmerkung für Wartung oder Änderungen	.65
<b>Anhang B Literaturverzeichnis</b>	<b>.66</b>

# Abbildungsverzeichnis

Abbildung 2.1	Beispiel für einen Datenflußgraphen anhand einer Audio Konferenz . . . . .	9
Abbildung 2.2	Beispiel einer Uhren-Hierarchie . . . . .	10
Abbildung 2.3	Architektur des Cinema-Systems . . . . .	10
Abbildung 2.4	Beispielprogramm für eine Cinema-Programmierschnittstelle . . . . .	11
Abbildung 2.5	Durch das Beispielprogramm erzeugte Session . . . . .	11
Abbildung 2.6	Verteilung einer Topologie auf Threads . . . . .	12
Abbildung 2.7	QoS-Architektur von Cinema . . . . .	14
Abbildung 2.8	Beispiel für Abhängigkeiten der Datenströme in verteilten Multimedia-Systemen . . . . .	14
Abbildung 3.1	QoS-Architektur von Cinema 17	
Abbildung 3.2	Beispiel eines Geflechts, bei dem eine Aushandlung mit NRP nicht möglich ist! . . . . .	21
Abbildung 3.3	XNRP Architektur . . . . .	22
Abbildung 3.4	Propagieren der AFS durch eine Geflecht . . . . .	26
Abbildung 3.5	„Kritischer Pfad“ . . . . .	26
Abbildung 4.1	Aufteilung der Funktionen auf die Prozesse des Cinema-Systems . . . . .	28
Abbildung 4.2	Threadstruktur und Kommunikation des Cinema-Systems . . . . .	29
Abbildung 5.1	Verzögerungen bei sequentiellm Ablauf des Protokolls . . . . .	31
Abbildung 5.2	Erweiterung des cin Prozesses um einen Thread . . . . .	32
Abbildung 6.1	Szenario für die Messung des Zeitbedarfs für die Kommunikation . . . . .	44
Abbildung 6.2	Szenario für die Messung des Zeitbedarfs für den Ressourcenmanger des Transportsystems . . . . .	45
Abbildung 6.3	Szenario für die Messung des Zeitbedarfs für ein minimales Geflecht, Quelle-Senke . . . . .	45

Abbildung 6.4	Szenario für die Messung des Zeitbedarfs des Links einer Multicastverbindung . . . . .	46
Abbildung 6.5	Szenario für die Messung des Zeitbedarfs für Mixer mit einer unterschiedlichen Anzahl von Eingangsports . . . . .	47
Abbildung 6.6	Szenario für die Messung des Zeitbedarfs für den Ressourcenmanger des Transportsystems . . . . .	47
Abbildung 6.7	Szenario für die Messung des Zeitbedarfs einer Sequenz von Filtern . . . . .	48
Abbildung 6.8	Szenario für die Messung des Zeitbedarfs einer Sequenz von Mixern . . . . .	49
Abbildung 6.9	Messung des Zeitbedarfs für komplexe Geflechte . . . . .	50
Abbildung 7.1	Konzentration aller Informationen an einer zentralen Instanz. . . . .	54

## Tabellenverzeichnis

Messung 1	Zeitbedarf für den Transport einer Nachricht zwischen zwei Threads . . . . .	44
Messung 2	Zeitbedarf für die Kommunikation zwischen Applikation-Handler und Gleichungslöser . . . . .	44
Messung 3	Die Größe der Nachrichten . . . . .	44
Messung 4	Der Zeitbedarf des Ressourcenmanger des Transportsystems . . . . .	45
Messung 5	Der Zeitbedarf für die Initialisierung der Komponenten . . . . .	45
Messung 6	Ein minimales Geflecht, Quelle-Senke . . . . .	46
Messung 7	Der Zeitbedarf des Links bei Multicastverbindungen . . . . .	46
Messung 8	Der Zeitbedarf für Mixer mit einer unterschiedlichen Anzahl von Eingangsports . . . . .	47
Messung 9	Der Zeitbedarf bei verschiedener Größe der AFS . . . . .	48
Messung 10	Der Zeitbedarf der Berechnungen des Gleichungslöser . . . . .	48
Messung 11	Der Zeitbedarf für eine Sequenz von Filtern . . . . .	49
Messung 12	Der Zeitbedarf für eine Sequenz von Mixern . . . . .	49
Messung 13	Der Zeitbedarf bei parallelen Komponenten . . . . .	50

# 1. Einführung

## 1.1 Motivation

Netzwerke und Rechner mit „Multimedia-Fähigkeit“, also mit der Kapazität multimediale Datenströme zu verarbeiten, werden heute für private wie geschäftliche Anwendungen eingesetzt [8]. Der Einsatz von multimedialen Anwendungen für Netzwerke, besonders mittels verteilter Multimedia-Systeme [1], wird folglich immer interessanter. Verteilte Multimedia-Systeme, sind Systeme zur Übertragung und Bearbeitung von multimedialen Datenströmen in Rechnernetzwerken. (z.B. für Videokonferenzen oder Virtual-Reality Anwendungen)

Bei verteilten Multimedia-Systemen können die Teile einer Anwendung über eine große Anzahl von Rechnern verstreut sein. Für die Funktionsfähigkeit müssen die Zusammenhänge zwischen den Teilen beachtet werden. Zusammenhänge der QoS (Quality of Service) zwischen den Datenströmen, Formatbeschränkungen der Elemente (z.B. eine Kamera erzeugt Bilder nur mit einer bestimmten Größe) und Ressourcenbeschränkungen der Rechner (CPU, Speicher) und des Netzwerks (Bandbreite).

Die Ermittlung der maximal möglichen QoS, kann selbst bei kleinen Systemen, ein komplexes Problem sein. Zusätzlich können, durch wechselnde Ressourcenauslastung der Rechner und Netzwerke, Schwankungen der Wiedergabequalität auftreten, die von Benutzern als störend empfunden werden.

Schwankungen der Qualität können durch Systeme zur Reservierung von Ressourcen verhindert werden. Beispiele hierzu sind ATM [2] und RSVP [3], mit Reservierungsmöglichkeiten bei der Übertragung von Daten, oder Algorithmen für Garantien bei der Ausführung von Prozessen wie „Earliest Deadline First“ [1] und „Rate Monotonic“ [1]. Da reservierte Ressourcen anderen Anwendungen nicht zur Verfügung stehen und eventuell bezahlt werden müssen, möchte man nur soviel Ressourcen reservieren wie nötig sind, um die gewünschte QoS zu erhalten.

Für die Aushandlung der QoS und die Reservierung der Ressourcen wurden am IPVR<sup>1</sup> der Universität Stuttgart die Protokolle NRP (Network Reservation Protocol) [5] und XNRP (eXtended Network Reservation Protocol) [7] entwickelt.

Jedoch fehlten bisher Implementierungen, um die Leistungsfähigkeit dieser Protokolle unter Beweis zu stellen und die Möglichkeiten der Integration in ein reales System zu untersuchen.

---

1. Institut für Parallele und Verteilte Höchstleistungsrechner

## 1.2 Aufgabenstellung

In dieser Diplomarbeit sollen die Reservierungsprotokolle NRP [5] und XNRP [7] implementiert und validiert werden, welche die Bestimmung der möglichen QoS für eine verteilte Multimedia-Anwendung durchführen und die entsprechenden Ressourcen reservieren. Es soll dabei untersucht werden, auf welche Weise diese Protokolle in das Cinema-System [4] integriert werden können. Bei der Integration sollen die zu entwickelnden Protokollelemente soweit abgeschlossen sein, daß andere Teile des Cinema-Systems nur um die Bereitstellung von Informationen für die Protokolle erweitert werden müssen.

Die Protokolle sollen den Reservierungsschritt sowohl für Komponenten als auch für Links durchführen. Es müssen daher konkrete Konzepte entwickelt werden, damit lokale Ressourcen-Manager sowie Transport-/Netzwerkprotokolle mit Reservierungsmöglichkeiten (z.B. RSVP) in das Protokoll eingebunden werden können.

Leistungsmessungen sollen den effizienten Einsatz der Protokolle in Cinema belegen.

Die Arbeit soll in C++ unter dem Betriebssystem Sun Solaris 2 auf Workstations vom Typ Sun SparcStation 10 und 20 implementiert werden. Dabei sind die Richtlinien der Abteilung Verteilte Systeme zur Implementierung von Programmen zu beachten.

## 1.3 Überblick

In Kapitel 2 werden die Konzepte und die QoS („Quality of Service“)-Architektur des Cinema-Systems beschrieben die als Grundlage für die Beschreibung der Protokolle NRP und XNRP in Kapitel 3 dienen. In Kapitel 4 wird die Struktur und Organisation der Implementierung des Cinema-Systems untersucht. In Kapitel 5 erfolgt die Beschreibung der Realisierung und der Integration der Protokolle. In Kapitel 6 wird die Durchführung der Messungen beschrieben und in Kapitel 7 erfolgt die Zusammenfassung des Ergebnisses und eine Bewertung der Implementierung und der Protokolle.

## 2. Das Cinema System

Cinema („Configurable INtegratEd Multimedia Architecture“) ist eine Entwicklungsplattform für den effizienten Entwurf und die Steuerung verteilter Multimedia-Anwendungen. Es gehört zu den Middleware-Systemen und wurde an der Universität Stuttgart entwickelt. [4]

### 2.1 Applikationsmodell

Multimedia-Anwendungen, die mit Hilfe von Cinema realisiert werden, setzen sich aus einer Vielzahl von Einzelbausteinen zusammen, den sogenannten Komponenten. Innerhalb dieser Komponenten werden kontinuierliche Datenströme produziert und verarbeitet. In Cinema gibt es verschiedene Arten von Komponenten: Quellen, Senken, Filter und Mixer. Quellen erzeugen und Senken konsumieren Datenströme. Zwischen den Quellen und Senken können Mixer und Filter liegen. Mixer mischen mehrere Datenströme zu einem Datenstrom. Filter empfangen einen Datenstrom, verarbeitet diesen und leiten ihn weiter.

Komplexere Konstrukte können aus einfachen Komponenten zusammengesetzt werden. Somit läßt sich jede denkbare Anwendung aus den Grundkomponenten Quellen, Senken, Mixer und Filter herleiten.

Eine Komponente kommuniziert mit anderen Komponenten über Ports. Einer Komponente bleibt es verborgen, ob vom Port gelesene Daten von einer lokalen Komponente stammen oder von einer Komponente auf einem anderen Rechner erzeugt wurden. Man unterscheidet Ausgangsports und Eingangsports.

Quellen haben einen Ausgangsport, Senken einen Eingangsport, Filter einen Eingangs- und einen Ausgangsport, Mixer besitzen einen Ausgangsport und mehrere Eingangsports.

Ports werden in Cinema mit einem bestimmten Stromtyp assoziiert. Jeder Stromtyp definiert einen Satz von medienspezifischen Parametern, die charakteristisch für diesen Stromtyp sind und diesen näher beschreiben.

Die Ports der Komponenten sind über Links verbunden. Es dürfen allerdings nur Ports miteinander verbunden werden, wenn sie zueinander kompatible Stromtypen aufweisen. Dabei kann ein Ausgangsport mit mehreren Eingangsports verbunden werden.

Links erlauben eine Abstraktion des zwischen zwei Komponenten real vorhandenen Transport-

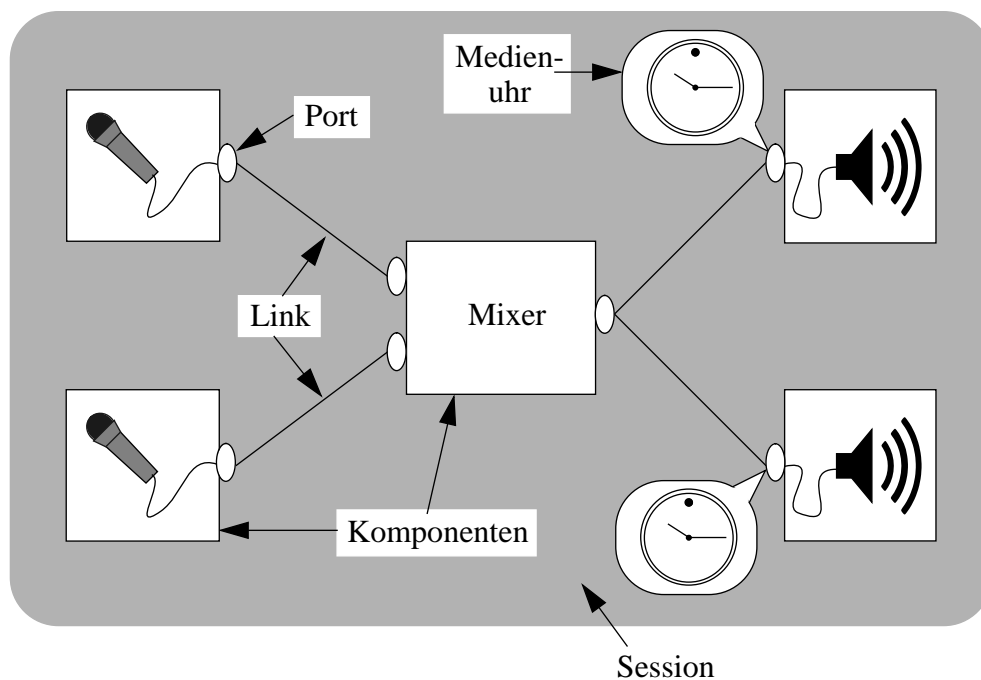
systems. Dies kann bei lokal kommunizierenden Komponenten ein gemeinsamer Speicherbereich oder im entfernten Fall das verwendete Übertragungsmedium (z. B. Ethernet usw.) sein.

Über das Konzept von Komponenten und Links lassen sich beliebige, auf die Problemstellung zugeschnittene Anwendungstopologien aufbauen.

Eine Session wird durch Angabe einer Menge von Quellen und Senken, die über ein beliebiges Netzwerk von zwischengelagerten Komponenten verbunden sein können, definiert. Die Session kann somit einen kompletten Datenflußgraphen oder nur Teile davon umfassen.

Der Datenflußgraph kann dynamisch zur Laufzeit verändert werden, d. h. es können Komponenten hinzugefügt bzw. entfernt werden. Abbildung 2.1 zeigt ein Beispiel für einen möglichen Datenflußgraphen.

Ströme verschiedenen Typs (z.B. Video und Audio) werden separat voneinander übertragen. Bei Strömen mit einem logischen Zusammenhang (z.B. Bild und Ton eines Films), ist hierbei eine Synchronisation notwendig. Darüber hinaus sollte der zeitliche Verlauf eines Datenstromes steuerbar sein. Das heißt ein Datenstrom soll schneller bzw. langsamer übertragen oder sogar rückwärts abgespielt werden können. Zu diesem Zweck wurde in Cinema die Abstraktion Medienuhr und das Uhren-Hierarchie-Konzept eingeführt.

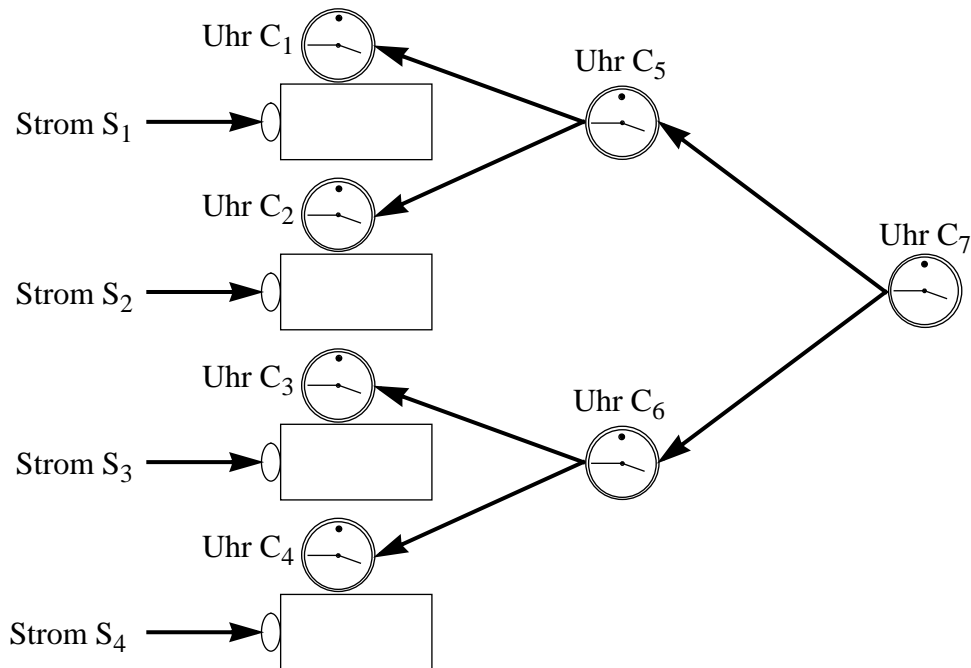


Beispiel für einen Datenflußgraphen anhand einer Audio Konferenz

Abbildung 2.1

Medienuhren dienen zur Steuerung und Synchronisation von Datenströmen. Diese Uhren sind mit Quellen- und Senkenkomponenten verbunden und erlauben die Funktionen: Uhr starten, Uhr anhalten und Tickrate ändern. Entsprechend wird das Abspielen eines Datenstroms gestartet bzw. gestoppt oder in einer anderen Geschwindigkeit abgespielt. Die Tickrate kann auch negative Werte annehmen. Dadurch ist es möglich, den Datenfluß rückwärts abspielen zu lassen.

Durch das Verknüpfen einer oder mehrerer Medienuhren mit einer übergeordneten Mediuhr, läßt sich eine Uhren-Hierarchie aufbauen. Diese Uhren-Hierarchie ermöglicht die Synchronisation mehrerer Datenströme durch eine übergeordnete Instanz.

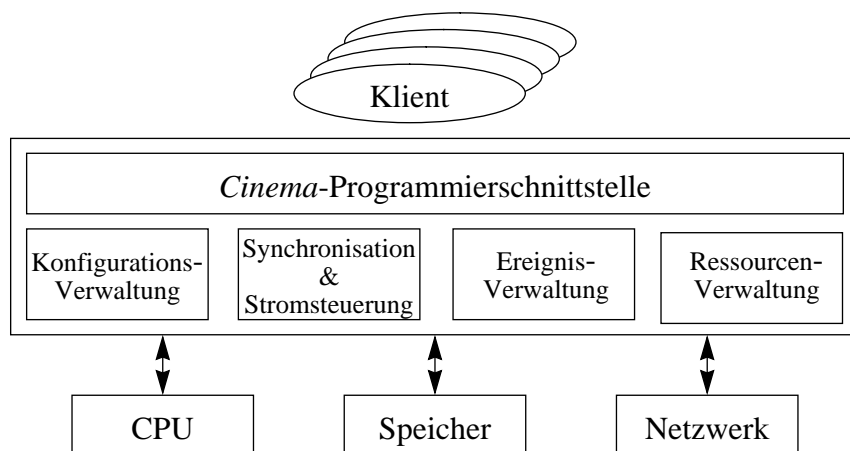


Beispiel einer Uhren-Hierarchie

Abbildung 2.2

## 2.2 Cinema Dienste

Ein Klient ist eine Applikation, die Dienste von Cinema zur Übertragung und Synchronisation von Datenströmen in Anspruch nimmt. Eine API („Application Programming Interface“) stellt die Programmierschnittstelle zwischen dem Cinema-System und einem Klient dar. Über diese Dienstschnittstelle kann auf die von Cinema bereitgestellten Managementbereiche und -funktionen zugegriffen werden. Dies sind: Konfiguration- und Ressourcenmanagement, Synchronisation und Steuerung, sowie die Ereignisverwaltung.



Architektur des Cinema-Systems

Abbildung 2.3

## 2.2.1 Cinema-Programmierschnittstelle

Der Klient kann mit Hilfe der objektorientierten Programmiersprache C++ die zur Verfügung gestellten Funktionen nutzen und einen Datenflußgraphen zur Bearbeitung von multimedialen Datenströme erzeugen und starten. Der Aufbau und das Verändern von Datenflußgraphen ist dynamisch, während der Laufzeit möglich.

Abbildung 2.4 zeigt ein Codefragment, das ein Beispiel für die Cinema-Programmierschnittstelle darstellt.

```
1   comp_camera = COMPONENT(„camera“, Rechner1);
2   comp_display = COMPONENT(„display“, Rechner2);
3   link(comp_camera->port(„video“),
4       comp_display->port(„video“));
5   comp_display->associate(clock);
6   create_session(comp_camera->port(„video“),
7   comp_display->port(„video“));
8   clock->start();
```

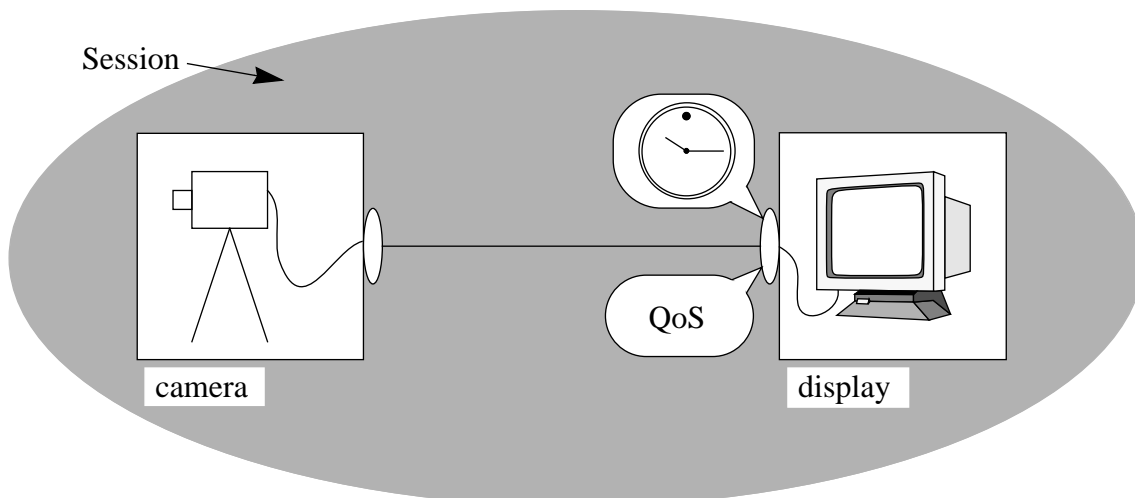
Beispielsprogramm für eine Cinema-Programmierschnittstelle

Abbildung 2.4

In den Zeilen 1 und 2 werden die Komponentenobjekte erzeugt. Dabei handelt es sich um eine Quellenkomponente namens *camera* und um eine Senkenkomponente, die den Namen *display* erhält. Die Parameterwerte bestimmen, welche Komponentenklasse und Lokation gewünscht wird. In Zeile 3 und 4 werden die Ports der Komponentenobjekte über einen Link miteinander verbunden. Nur Ports vom gleichen Typ (hier *video*) dürfen miteinander verbunden werden. Dadurch wird sichergestellt, daß die verbundenen Ports identische QoS unterstützen.

In Zeile 5 wird der Senkenkomponente eine Uhr zugeordnet. Diese steuert die Übertragungsgeschwindigkeit des Datenstroms.

In den Zeilen 6 und 7 erfolgt die Definition der Cinema-Session. Durch das Starten der Uhr in Zeile 8 wird mit der Übertragung und Verarbeitung des Videostroms begonnen. Abbildung 2.5 zeigt die durch das Codefragment definierte Anwendungstopologie.



Durch das Beispielsprogramm erzeugte Session

Abbildung 2.5

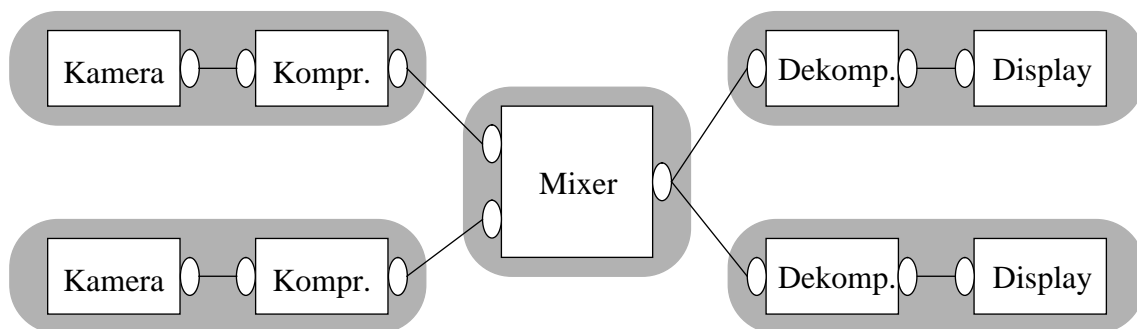
## 2.2.2 Konfigurationsmanagement

Eine verteilte Multimedia-Anwendung, die mit Hilfe von Cinema aufgebaut wurde, besteht aus einer Menge von Komponenten, die über Kommunikationskanäle miteinander verbunden sind. Das Konfigurationsmanagement [16] überträgt dieses Geflecht in eine ablauffähige Instanziierung. Das ganze Komponentengeflecht wird dabei auf die verschiedenen Rechner verteilt, erst daraufhin ist der Datenflußgraph ablauffähig. Komponenten können auch von mehreren Klienten gemeinsam benutzt werden. Dieser Komponententyp („Shared Component“) wird ebenfalls von dem Konfigurationsmanagement verwaltet. Darüber hinaus stellt das Konfigurationsmanagement Informationen für die anderen Managementbereiche des Cinema-Systems zur Verfügung, sowohl über die Topologie als auch über die Komponenten selbst.

Die Komponenten werden auf den zugewiesenen Rechnern innerhalb der Cinema-Prozesse ausgeführt. Es werden Threads verwendet, da mittels diesem Konzept Nebenläufigkeit mit geringem Prozeßwechsel-Aufwand möglich ist. Darüber hinaus ist eine effiziente Kommunikation möglich, da die Threads innerhalb eines Cinema-Prozesses den gleichen Adressraum belegen.

Nicht jede Komponente hat ihren eigenen Thread. So würde mit der Verteilung von je einer Komponente auf einen Thread zwar die maximal mögliche Nebenläufigkeit erreicht werden, die hohe Zahl von Threads würde jedoch eine größere Ende-zu-Ende-Verzögerung bedeuten.

Deshalb wird in Cinema versucht, die Anzahl der Threads entlang eines Pfades klein zu halten. Dazu werden mehrere Komponenten auf einem Pfad zu einem Thread zusammengefaßt. Somit wartet nur die erste Komponente innerhalb eines Threads auf Datenströme. Die darauffolgenden Komponenten im selben Thread konsumieren die Dateneinheiten vollständig und ohne weitere Zeitverzögerung. Es entsteht dadurch eine Bearbeitungspipeline, bei der nur die erste Komponente innerhalb eines Threads Daten konsumieren kann, die von Komponenten aus anderen Threads oder von anderen Rechnerknoten stammen. Abbildung 2.6 zeigt eine solche Topologie.



Verteilung einer Topologie auf Threads

Abbildung 2.6

## 2.2.3 Synchronisation und Stromsteuerung

Cinema bietet die Abstraktion Mediuhr und das Uhren-Hierarchie-Konzept an. Dabei werden die einzelnen Dateneinheiten eines Datenstroms mit Zeitmarken versehen, die eine Verwaltung des Datenstroms durch die Uhren ermöglichen.

Der Managementbereich „Synchronisation und Stromsteuerung“ besitzt Informationen über den Status der Uhren, die gewählte Tickrate und ob Uhren angehalten bzw. welche Uhren gerade aktiv sind. Desweiteren wird von dieser Instanz die Zuordnung der Uhren zu den Komponenten und ihre Einbindung in die Uhren-Hierarchie verwaltet.

Die Synchronisationsdienstleistungen werden durch geeignete Synchronisationsprotokolle erbracht. Entsprechend der großen Zahl an vorhandenen Protokollen ermöglicht die Cinema-Architektur die Integration unterschiedlicher Synchronisationsprotokollen. Je nach Anwendungsszenario kann ein Protokoll ausgewählt und verwendet werden [14][15].

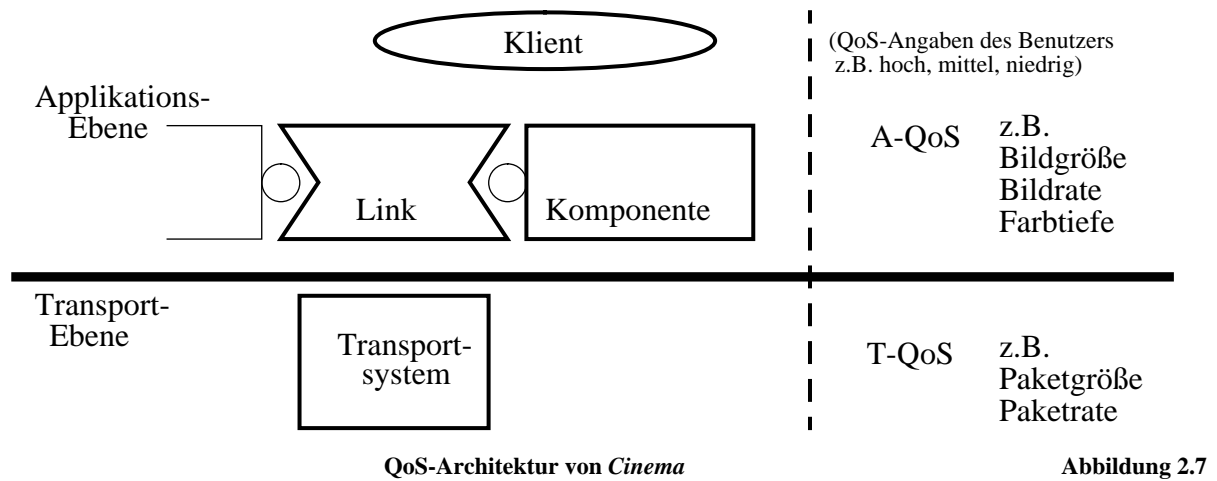
## **2.2.4 Ereignisverwaltung**

Objekte kommunizieren in Cinema über das Versenden von Nachrichten. Erreicht eine Nachricht das für sie bestimmte Objekt, so wird in diesem Objekt ein Ereignis ausgelöst, d. h. eine Callback-Methode aufgerufen. Jedes Objekt in Cinema kann Ereignisse auslösen oder auf Ereignisse reagieren. Die Ereignisverwaltung ist dabei die zentrale Instanz. Objekte teilen ihr mit, daß sie Interesse an bestimmten Ereignissen haben und übermitteln ihr eine Adresse der dazugehörigen Callback-Funktion. Diese wird beim Eintreten des entsprechenden Ereignisses aufgerufen.

## **2.2.5 Ressourcenmanagement**

Die Einhaltung der QoS muß durch eine Verwaltung, Reservierung und Zuteilung der Ressourcen sichergestellt werden. Für die einzelnen Betriebsmittel übernehmen Ressourcenmanager diese Aufgabe. Diese müssen den benötigten Ressourcenumfang reservieren und zur Laufzeit zuteilen. In Cinema ist bisher nur der Ressourcenmanager für das Transportsystem [13] integriert. Die Integration der Ressourcenmanager für CPU und Speicher ist vorgesehen.

Innerhalb von Cinema unterscheidet man die QoS der Klient-, Applikations- und Transportschicht. T-QoS sind QoS-Angaben des Transportsystems. Für das Transport- und Kommunikationssystem wird eine T-QoS benötigt, die beispielsweise die Paketgröße oder die Paketverlustrate beschreibt. Diese Parameter sind jedoch nicht für die Übernahme in das Cinema-System geeignet. Cinema selbst ist auf der Applikationsebene angesiedelt. Hier werden QoS Angaben der Applikationsebene verwendet. Diese A-QoS orientieren sich an den Eigenschaften und Typen der zu übertragenden und zu verarbeitenden Datenströme. So kann eine Applikation die gewünschte Qualität eines Videos angeben, in dem sie z. B. dessen Bildgröße, -breite und Farbtiefe spezifiziert. Weiterhin können Komponenten Einschränkungen aufweisen, die sich auf medienspezifische Parameter beziehen. Beispielsweise ist es möglich, daß eine MPEG-Videokomponente nur Bilder bis zu einer bestimmten Größe verarbeiten kann. Über der Applikationsebene kann es noch eine weitere, höhere Abstraktionsebene geben. Eine Anwendung kann zur Bestimmung der vom Benutzer verlangten QoS andere Parameter als z. B. Bildgröße und Bildrate anbieten. Beispielsweise könnte sie für eine Videokonferenz die Auswahl auf „hohe“, „mittlere“ und „niedrige“ Qualität beschränken. Die Angabe solcher Parameter hängt stark von der Art der Anwendung ab. Die Anwendung muß eine Abbildung dieser Parameter auf die A-QoS der Applikationsebene durchführen.

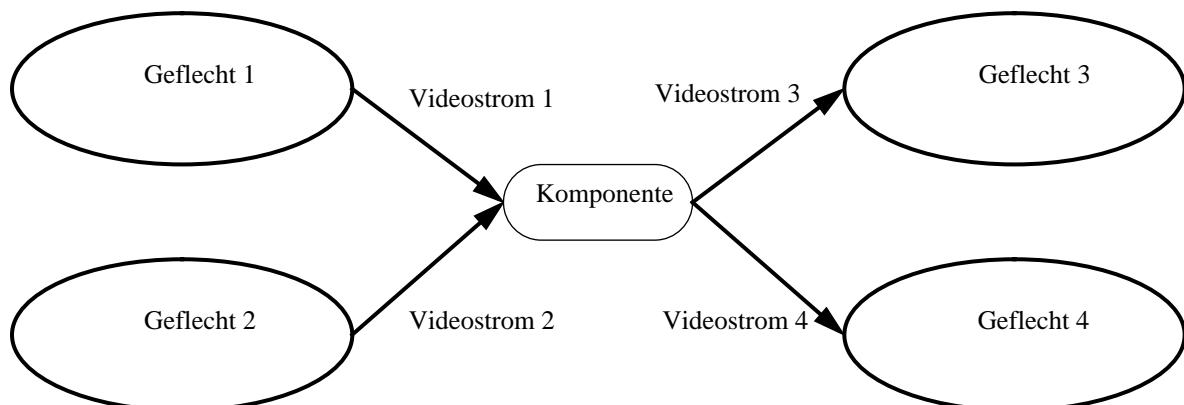


## 2.3 Sessionaufbau

Eine Session besteht aus Komponenten, die über ein Rechnernetz verteilt und durch Links verbunden sind. Die Anzahl der Komponenten innerhalb einer Session ist theoretisch nicht beschränkt. Bei Anwendungsgebieten wie Videokonferenzen, Bildverarbeitung oder virtuelle Realität kann man eine Anzahl von etwa einhundert annehmen [1].

Bevor die Verarbeitung der Daten anlaufen kann, müssen sich alle beteiligten Elemente auf die verwendete QoS einigen. Eine Komponente darf nur Daten erzeugen, die von nachfolgenden Links und Komponenten verarbeitet werden können. Relationen zwischen den Eingangsports und Ausgangsports der Komponenten und das Aufeinandertreffen oder Aufspalten der Datenströme an Mixern und Multicast-Verbindungen, kann dabei zu Abhängigkeiten zwischen sonst unabhängigen Teilen des Systems führen.

In Abbildung 2.8 ist ein System zur Verarbeitung von Videodaten zu sehen, das aus vier Teilgeflechten besteht, die über eine einzelne Komponente verbunden sind.



**Beispiel für Abhängigkeiten der Datenströme in verteilten Multimedia-Systemen**

**Abbildung 2.8**

Die Komponente empfängt Datenströme von den Geflechten 1 und 2, und sendet Datenströme an die Geflechte 3 und 4.

Als Beispiel werden hier Abhängigkeiten der Bildgrößen angenommen:

- Die Komponente erwartet an ihren Eingängen eine bestimmten Relation,  $\text{Bildgröße (Videostrom 1)} < \text{Bildgröße (Videostrom 2)}$
- Die Komponente sendet nur mit einer einheitlichen Größe,  $\text{Bildgröße (Videostrom 3)} = \text{Bildgröße (Videostrom 4)}$
- Der Ausgang der Komponente hat eine feste Relation zum Eingang,  $\text{Bildgröße (Videostrom 2)} = 2 * \text{Bildgröße (Videostrom 3)}$

Diese drei Bedingungen beschreiben eine vollständige Abhängigkeit der Bildgröße zwischen allen vier Links und damit zwischen allen vier Geflechten. In einem realen Fall können noch eine Vielzahl von Parametern hinzukommen, z.B. Bildrate und Komprimierungsfaktor. Hinzu kommen Formatbeschränkungen der Komponenten selbst (z.B. kann eine Komponente nur Videobilder in bestimmten Größen verarbeiten) und Beschränkungen durch die benötigten Ressourcen (z.B. Bandbreite des Transportsystems). Die Aushandlung der QoS umfaßt also A-QoS der Applikationsebene und T-QoS der Ressourcenmanager.

Ein Protokoll zur Aushandlung der QoS und Reservierung der Ressourcen muß zwischen den Wünschen des Benutzers (oder einer Applikation) und den Möglichkeiten des Systems gegebenenfalls Kompromisse schließen. So kann eine Hochlastsituation z.B. dazu führen, daß die angeforderten Ressourcen einfach nicht zur Verfügung gestellt werden können. Daraufhin muß das Protokoll im Rahmen der vorgegebenen Formatbeschränkungen die QoS anpassen und für eine geringere Wiedergabequalität Ressourcen reservieren. Ist jedoch auch mit der niedrigsten Qualität eine Reservierung nicht möglich, so scheitert die Aushandlung.

In Cinema wurde die QoS bisher manuell aus den Formatbeschränkungen der Komponenten, d.h. ohne Einbeziehung der Ressourcen ermittelt. Die notwendigen Einstellungen an den Komponenten wurden ebenfalls manuell durchgeführt. Dies bedeutet aber bereits bei kleinen Systemen einen großen Zeitaufwand und eine hohe Fehleranfälligkeit. Noch aufwendiger ist es, wenn eine Aushandlung und Reservierung der benötigten Ressourcen erfolgen soll.

Für Aushandlungen und Reservierungen auf der Transportebene existieren bereits Lösungen wie ATM [2] und RSVP [3]. Und mit Algorithmen wie „Earliest Deadline First“ [1] und „Rate Monotonic“ [1], lassen sich Reservierungen der Rechnerressourcen realisieren. Dagegen fehlte bisher eine entsprechende Lösung für die Applikationsebene. Auf dieser Ebene sollen nun die Protokolle NRP und XNRP die Aushandlungen durchführen.

### 3. Beschreibung von NRP und XNRP

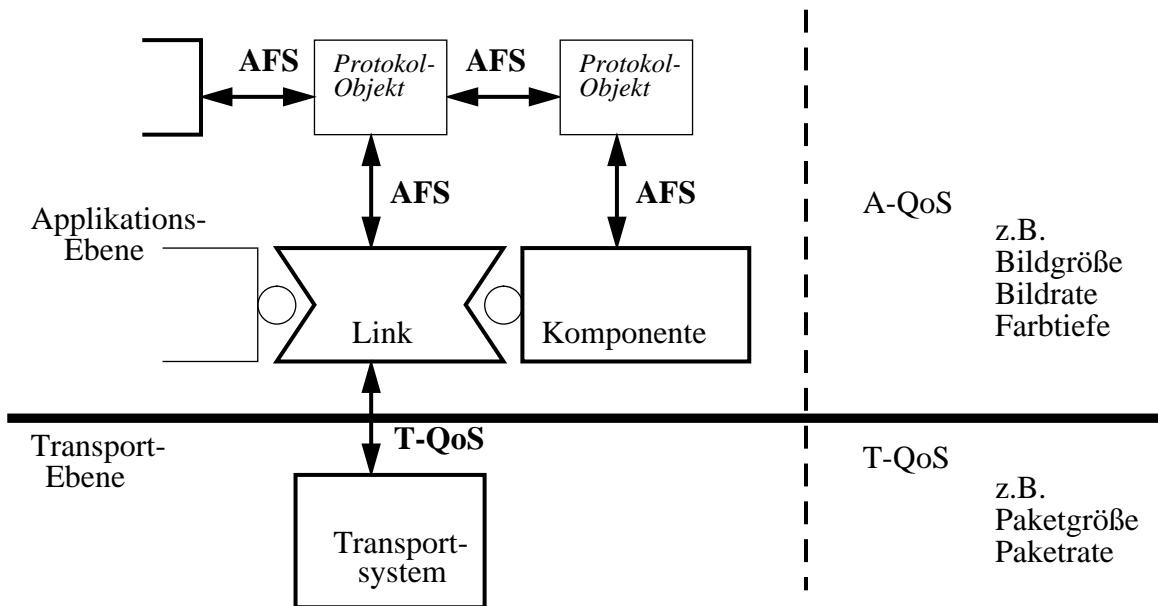
NRP und XNRP sind zwei Protokolle für die Aushandlung der QoS und die Reservierung von Ressourcen in verteilten Multimedia-Systemen. Beide Protokolle wurden am IPVR (Institut für Parallele und Verteilte Höchstleistungsrechner) der Universität Stuttgart entwickelt [5][7]. Dieses Kapitel beschreibt die Struktur und Funktionsweise beider Protokolle.

Beide Protokollmodelle beruhen auf dem in Cinema verwendeten Applikationsmodell (Abschnitt 2.1) und der QoS-Architektur (Abbildung 2.7). Bei beiden Protokollen werden AFS („Application FlowSpec“) entlang des Flußgraphen einer Konfiguration propagiert (jeweils nur „in“ oder „gegen“ die Stromrichtung) und von den Komponenten und Links bearbeitet.

Die Steuerung des Protokollablaufs erfolgt durch Protokollagenten. Jeder Komponente und jedem Link des Geflechts wird je ein Protokollagent zugeordnet, der die AFS für dieses Element empfängt, Methoden des Elements für die Bearbeitung der AFS aufruft und die resultierenden AFS weiter versendet. Die Komponenten und Links führen die Bearbeitung der AFS selbständig durch und besitzen eine einheitliche Schnittstelle für die Protokollagenten. Dadurch sind diese unabhängig davon ob ihnen eine Komponente oder ein Link zugeordnet ist und sie sind unabhängig vom Typ der bearbeiteten Medien.

Die AFS enthält medienspezifische Informationen über die gewünschte und die realisierbare A-QoS der Applikationsebene. Diese können in Form von Mengen mit diskreten oder kontinuierlichen A-QoS-Bereichen angegeben werden. Bei der Angabe einer diskreten Menge enthält die AFS eine Anzahl von A-QoS und die ausgehandelte QoS muß einer A-QoS dieser Menge entsprechen. Dagegen enthält die AFS bei der Angabe eines kontinuierlichen Bereichs nur zwei A-QoS, zwischen denen die ausgehandelte QoS liegen muß.

Die Aushandlung der QoS erfolgt nicht immer über die A-QoS. So werden für das Transport-system z.B. T-QoS-Angaben benötigt. Die Abbildung der A-QoS auf die T-QoS der Transportebene ist innerhalb der Links gekapselt.



QoS-Architektur von *Cinema*

Abbildung 3.1

Wenn ein Link eine AFS von seinem Protokollagenten mit dem Auftrag erhält, eine Reservierung durchzuführen, so führt er eine Abbildung der enthaltenen A-QoS auf die entsprechenden T-QoS des Transportsystems durch und übergibt diese dem Transportsystem. Das Transportsystem führt eine Reservierung entsprechend den Anforderungen der T-QoS durch und gibt das Ergebnis als T-QoS an den Link zurück. Der Link führt wiederum eine Abbildung der T-QoS auf die entsprechenden A-QoS durch und übergibt diese innerhalb einer AFS seinem ihm zugeordneten Protokollagenten. Bei entfernten Verbindungen ist zu Beachten, daß ein Link aus zwei Teilen besteht. Je ein Teil befindet sich auf einem der Rechner, die der Link verbindet. Die Informationen für eine Reservierung können je nach Transportsystem am Sender, am Empfänger oder auch auf beiden Seiten benötigt werden. Um die Transparenz zu erhalten, wird beiden Teilen ein Protokollagent zugeordnet und die AFS damit beiden Seiten übergeben. Diese Regelung erhält die Transparenz, weil die Teile von den Protokollagenten wie zwei unabhängige, aufeinanderfolgende Elemente des Geflechts betrachtet werden und der Link (bzw. das Transportsystem) entscheidet, auf welcher Seite die Reservierung erfolgen soll.

Eine Komponente kann direkt mit den medienspezifischen und generischen Parametern der A-QoS arbeiten, jedoch ist für die Reservierung der Rechnerressourcen ebenfalls eine Abbildung notwendig. Um eine Speicher-Reservierung durchführen zu können, ermittelt die Komponente die Schnittmenge aus der A-QoS der AFS und den Formatbeschränkungen des Ports, für den sie die AFS erhalten hat. (Eventuell müssen noch Relationen zu der A-QoS anderer Ports beachtet werden.) Dann führt sie eine Abbildung auf die ressourcenspezifische QoS-Angaben des Speichers durch und gibt diese Werte an den Ressourcenmanager für den Speicher weiter. Der Ressourcenmanager führt anhand dieser Werte die Reservierung durch und gibt als Ergebnis die Größe des tatsächlich reservierten Speichers (eventuell weniger als gewünscht) zurück. Diesen Wert wandelt die Komponente wieder in eine A-QoS um und übergibt ihn in einer AFS dem zugehörigen Protokollagenten.

Die Beschreibung der einzelnen Protokolle erfolgt anhand der in Cinema unterschiedenen Komponententypen:

- *Quelle*: produziert einen Datenstrom.
- *Senke*: konsumiert einen Datenstrom.
- *variabler Filter*: skaliert einen Datenstrom, wobei der Skalierungsfaktor vom Ergebnis der Aushandlung abhängig ist, aber nur kleiner oder gleich 1 sein kann. (Die QoS kann nur reduziert werden, nicht erhöht.) Im weiteren Text werden „variable Filter“ nur mit „Filter“ bezeichnet.
- *fester Filter*: skaliert einen Datenstrom mit einem vorgegebenen (vom Protokoll nicht veränderbaren) Skalierungsfaktor.
- *Mixer*: mischt mehrere Datenströme zu einem.

Beide Protokolle lassen sich in drei Phasen aufteilen. Die erste Phase läuft stromabwärts und es werden AFS, entlang der Links, von den Quellen zu den Senken propagiert. Die zweite Phase läuft stromaufwärts und es werden AFS, entlang der Links, von den Senken zu den Quellen propagiert. Die dritte Phase läuft wieder stromabwärts.

Prinzipiell wird in jeder Phase, an Komponenten und Links eine zu dieser Phase gehörende Methode aufgerufen und eine AFS übergeben. An den Eingangsports der *Senken* wird vor dem Start der Protokolle der Bereich vorgegeben, in dem die ausgehandelte QoS liegen muß.

## 3.1 NRP

NRP besteht aus den Phasen: „Reserve“, „Inform“ und „Relax“. Die AFS enthält A-QoS und einen DRV („Downstream Requested Value“), dessen Wert am Ende der ersten Phase von den Senken bestimmt und gesetzt wird. Der DRV-Wert informiert die stromaufwärts liegenden Komponenten und Links über die von der Senke erwartete A-QoS. Damit können diese ihre Reservierungen minimieren, auch wenn eine größere A-QoS möglich wäre (siehe Abschnitt 3.1.2).

### 3.1.1 Erste Phase, Reserve

In dieser Phase werden Reservierungen durchgeführt und Informationen über die mögliche QoS stromabwärts propagiert. Generell erhält eine Komponente eine AFS für jeden ihrer Eingangsports. Aus diesen entfernt sie die QoS, die der entsprechende Eingangsport nicht unterstützt, führt eine Reservierung durch und berechnet die AFS für den Ausgangsport. Ausnahmen sind Quellen und Senken, die nur einen Ausgangs- bzw. Eingangsport besitzen, sowie Links.

- *Quelle*: Eine Quelle führt eine Reservierung durch und berechnet aus den reservierten Ressourcen und ihren Formatbeschränkungen die A-QoS, die sie an ihrem Ausgangsport erzeugen kann. Mit den so berechneten A-QoS erzeugt sie eine AFS.

- *fester Filter*: Ein fester Filter berechnet aus der erhaltenen AFS, nach dem er die nicht unterstützte entfernt hat, eine neue AFS mittels seines Skalierungsfaktor. Beispielsweise können die Bildgrößen 1000\*1000 und 800\*800 eines Videostroms, die innerhalb einer AFS spezifiziert wurden, entsprechend einem Skalierungsfaktor wie 0,5, zu 500\*500 und 400\*400 transformiert werden. Der feste Filter führt eine Reservierung der Ressourcen durch und entfernt die A-QoS, die nicht mit den verfügbaren Ressourcen realisierbar sind. Das Ergebnis ist eine AFS, in der die A-QoS enthalten sind, die am Ausgangsport des festen Filters unterstützt werden können.
- *Mixer*: Nach dem Entfernen der von dem jeweiligen Eingangsport nicht unterstützten A-QoS, werden die AFS der Eingänge zu einer AFS zusammengefaßt. Diese hat eine feste Relation zur AFS des Ausgangs (entsprechend einem festen Filter). Benötigt der Mixer eine bestimmte Relation zwischen den A-QoS der Eingänge (z.B. der Bildgrößen bei Videodaten), so kann die AFS eines Eingangs ausgewählt werden. Durch Entfernen der A-QoS, für die keine entsprechende A-QoS an den anderen Eingängen vorhanden ist (entsprechend der verlangten Relationen zwischen den Eingängen), erhält man die gewünschte AFS. Die weiteren Schritte, Reservierung der Ressourcen und das entfernen der nicht unterstützten A-QoS, entsprechen einem festen Filter.
- *Filter*: Die vom Eingangsport nicht unterstützte A-QoS wird aus der AFS entfernt. Danach wird eine Reservierung durchgeführt und die A-QoS entfernt, für die keine ausreichenden Ressourcen vorhanden waren. Eine Kopie dieser AFS wird vom Filter gespeichert. Als letztes wird die A-QoS hinzugefügt, die der Filter durch skalieren der A-QoS des Eingangs erzeugen kann.
- *Link*: Ein Link führt eine Abbildung der in der AFS enthaltenen A-QoS nach T-QoS durch. Die Art der Abbildung ist vom Typ des Transportsystems abhängig, das von dem jeweiligen Link repräsentiert wird. Die T-QoS werden dem Transportsystem übergeben und das Transportsystem versucht eine Reservierung mit der größten T-QoS durchzuführen. Die mit den reservierten Ressourcen realisierbaren T-QoS gibt das Transportsystem als Ergebnis an den Link zurück. Der Link bildet die T-QoS wieder in A-QoS ab.
- *Senke*: Nach dem Entfernen der von den Eingangsports nicht unterstützten A-QoS, entfernen die Senken auch noch die A-QoS, die nicht im (von einem Benutzer oder einer Applikation) vorgegebenen Bereich liegt. Von der Senke wird eine Reservierung durchgeführt und die A-QoS entfernt, die mit den verfügbaren Ressourcen nicht unterstützt werden können. Die größte (noch enthaltene) A-QoS wird als DRV in die AFS eingetrag, d.h. diese A-QoS wird von der Senke gewünscht und unterstützt.

Mit Erreichen der Senken ist die erste Phase Reserve abgeschlossen. Die AFS an den Senken enthalten nur die QoS, die:

- Von den stromaufwärts gelegenen Komponenten und Links unterstützt wird.
- Innerhalb des (vom Benutzer oder einer Applikation) gewünschten Bereichs liegt.

### 3.1.2 Zweite Phase, Inform

In dieser Phase werden die stromaufwärts gelegenen Komponenten und Links über die stromabwärts ausgehandelte QoS informiert. Generell erhält eine Komponente eine AFS für ihren Ausgangsport und erzeugt eine AFS für jeden Eingangsport. Ein Link an einem Ausgangsport erhält für jede Verbindung an diesen Port eine AFS.

- *Senke*: In dieser Phase haben die Senken keine eigentliche Aufgabe. Die von ihnen in der ersten Phase Reserve erzeugte AFS ist die „Start“-AFS für diese Phase.
- *Link*: Die Links an Ausgangsports mit mehrfachen Verbindungen (Multicast) empfangen für jede der Verbindungen eine AFS. Aus den so empfangenen QoS-Bereichen bilden sie die Schnittmenge und tragen diese, zusammen mit dem Maxima der verschiedenen DRV, in eine neue AFS ein. Andere Links haben in dieser Phase keine Aufgabe.
- *Filter*: Die neue AFS für den Eingangsport wird aus der (in der ersten Phase Reserve) gespeicherten AFS erzeugt. Es werden die A-QoS entfernt die kleiner als die A-QoS der AFS für den Ausgangsport sind. Als DRV wird der DRV aus der AFS für den Ausgangsport verwendet. Ist die kleinste QoS größer als dieser DRV, wird diese als DRV verwendet. Die für den Ausgangsport erhaltene AFS wird gespeichert.
- *Mixer*: Die erhaltene AFS wird, entsprechend einer festen Relation zu einem der Eingangsports, skaliert (die A-QoS und auch der DRV). Wenn eine identische A-QoS für alle Eingangsports verlangt wird, muß die AFS lediglich weiter kopiert werden. Werden unterschiedliche A-QoS für die Eingangsports verlangt, so muß dagegen die AFS für jeden Eingangsports entsprechend skaliert werden.
- *fester Filter*: Ein fester Filter verhält sich wie ein Mixer mit einem Eingangsport.
- *Quelle*: Der DRV Wert wird zur Initialisierung verwendet, alle anderen QoS-Werte werden aus der AFS entfernt.

Vom gesamten Geflecht unterstützte QoS-Werte sind jetzt an den Quelle bekannt.

### 3.1.3 Dritte Phase, Relax

In dieser Phase wird der ausgehandelte QoS-Wert stromabwärts propagiert. Komponenten und Links führen ihre Initialisierungsmethoden aus und reduzieren ihre Reservierungen auf das benötigte Maß.

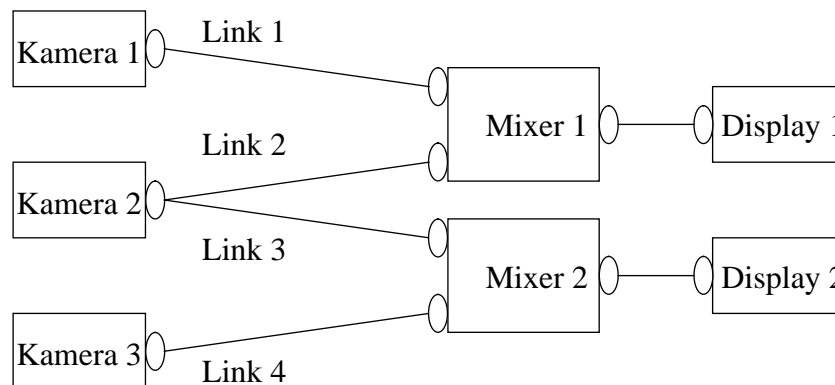
- *Quelle*: Quellen geben die reservierten Ressourcen bis auf das benötigte Minimum frei und liefern die „Start“-AFS mit den ausgehandelten A-QoS.
- *fester Filter*: Ein fester Filter gibt nicht benötigte Ressourcen frei und skaliert die AFS entsprechend seinem Skalierungsfaktor.
- *Mixer*: Ein Mixer verhält sich entsprechend einem festen Filter. In dieser Phase erhält ein Mixer für seine Eingangsports nur konsistente A-QoS, d.h. die A-QoS für die verschiedenen Eingangsports erfüllen die benötigten Relationen.

- *Filter*: Ist die A-QoS der erhaltenen AFS größer oder gleich dem DRV der gespeicherten AFS, so gibt ein Filter eine AFS mit dem DRV der gespeicherten AFS weiter. Sonst gibt er die A-QoS der erhaltenen AFS weiter. Seinen internen Skalierungsfaktor setzt er entsprechend der Relation zwischen erhaltener und weitergegebener A-QoS.
- *Link*: Ein Link verhält sich weitgehend wie in der ersten Phase Reserve, jedoch wird vom Transportsystem eine Anpassung der Ressourcen durchgeführt (und keine Reservierung).
- *Senke*: Die Senken geben (anhand der erhaltenen QoS) nicht benötigte Ressourcen frei und signalisieren dem Protokoll das Ende der Aushandlung.

Haben alle Senken das Ende der Aushandlung signalisiert, so ist die dritte Phase Relax und damit das Protokoll erfolgreich beendet.

### 3.1.4 Einschränkung der Konfigurationen bei NRP

Durch die Relationen zwischen den Eingangsports bei Mixern und den Verbindungen eines Multicast, können Szenarien auftreten die das NRP Protokoll nicht abdeckt. Bei dem in Abbil-



Beispiel eines Geflechts, bei dem eine Aushandlung mit NRP nicht möglich ist!

Abbildung 3.2

dung 3.2 gezeigten Geflecht ist eine Aushandlung mit NRP nicht möglich. Durch die Mixer entsteht eine Beziehung zwischen Link 1 und Link 4. In der ersten Phase Reserve erhalten Mixer 1 und Display 1 Informationen über die QoS des Link 1 und der Kamera 1. In der zweiten Phase Inform erhält Kamera 2 diese Informationen über Mixer 1. In der dritten Phase erhalten Mixer 2 und Display 2 diese Informationen. Jedoch erhalten weder der Link 4 noch die Kamera 3 Informationen über die QoS von Link 1 und Kamera 1. Entsprechend verhält es sich in umgekehrter Richtung. Weder Link 1 noch Kamera 1 erhalten Informationen über Link 4 und Kamera 4. Die Aushandlung der QoS für die Eingangsports der Mixer ist nicht vollständig. Der Grund dafür liegt in der Begrenzung von NRP auf drei Phasen, mit vier Phasen wäre eine Aushandlung in diesem Beispiel möglich. Jedoch läßt sich das Beispiel durch hinzufügen von Mixern (und entsprechenden Kameras und Displays) beliebig erweitern. Durch diese Erweiterung steigt die Zahl der benötigten Phasen ebenfalls beliebig.

NRP ist deshalb auf Geflechte beschränkt, die einem „Zwei-Zonen-Modell“ entsprechen, d.h. ein Geflecht muß in zwei Zonen aufgeteilt werden können:

- Die erste Zone in diesem Geflecht ist die „Mixer Zone“.
- Die zweite Zone ist die „Multicast-Zone“.

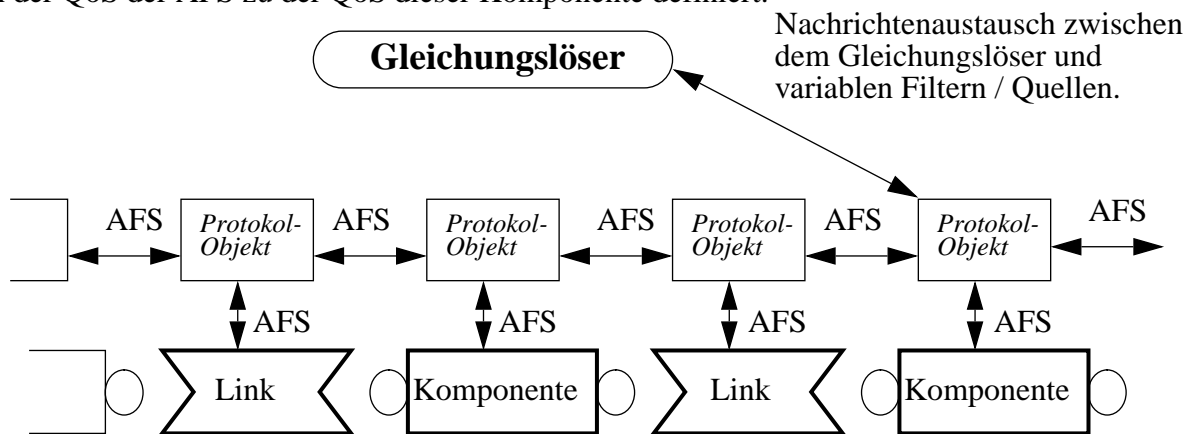
Vor einem Mixer dürfen keine Verzweigung (Multicast) von Strömen auftreten, nach einer Verzweigung darf wiederum kein Mixer sein. Für alle Geflechte, die dieser Einteilung entsprechen, kann NRP verwendet werden.

Dies ist keine zu starke Einschränkung, weil viele Anwendungen dieser Struktur entsprechen oder sich in eine solche überführen lassen (z.B. Konferenzsysteme) [1].

## 3.2 XNRP

XNRP besteht aus den Phasen: „Inform“, „Reserve“ und „Relax“. Nach dem Ende der zweiten Phase Reserve und vor dem Start der dritten Phase Relax, erfolgt zusätzlich der Aufruf des Gleichungslöser. Aus QoS und Stromrelationen errechnet dieser die optimale QoS für das Geflecht. Die notwendigen Informationen, QoS und Stromrelationen werden dem Gleichungslöser in der zweiten Phase Reserve von *Quellen*, *Mixern* und *Filtern* zugesendet.

Die AFS für XNRP enthält A-QoS, die ID (Identifikation) der nächsten stromaufwärts liegenden Komponente vom Typ *Quelle* oder *Filter* und einen Proportionalitätsfaktor, der die Relation der QoS der AFS zu der QoS dieser Komponente definiert.



XNRP Architektur

Abbildung 3.3

### 3.2.1 Erste Phase, Inform

In dieser Phase erhalten die Komponenten und Links Informationen über die Formatbeschränkungen der stromaufwärts gelegenen Komponenten und Links. Generell erhält eine Komponente eine AFS für jeden ihrer Eingangsports, aus denen sie eine AFS für ihren Ausgangsport berechnen.

- *Quelle*: Eine Quelle erzeugt aus ihren Formatbeschränkungen und ihrer ID eine neue AFS mit dem Proportionalitätsfaktor 1.
- *fester Filter*: Ein fester Filter bildet aus den A-QoS der erhaltenen AFS und den Formatbeschränkungen seines Eingangsports die Schnittmenge. Aus dieser berechnet er (mit seinem Skalierungsfaktor) die A-QoS und den Proportionalitätsfaktor der neuen AFS. Die ID übernimmt er aus der erhaltenen AFS.
- *Mixer*: Ein Mixer entfernt, entsprechend den Formatbeschränkungen und der Relationen der Eingangsports, die nicht unterstützte A-QoS aus der erhaltenen AFS. Aus der AFS eines Eingangsports, der eine feste Relation zum Ausgangsport besitzt, berechnet er die neue AFS. Dies geschieht entsprechend einem festen Filter. Die ID und die Proportionalitätsfaktoren der erhaltenen AFS werden von dem Mixer gespeichert.
- *Filter*: Entsprechend den Formatbeschränkungen seines Eingangsports entfernt ein variabler Filter die nicht unterstützte A-QoS aus der erhaltenen AFS und speichert diese. Dann berechnet er eine neue AFS für seinen Ausgangsport, indem er die A-QoS hinzufügt, die er durch Skalierung der A-QoS des Eingangsports erzeugen kann. In diese AFS trägt er seine eigene ID und einen Proportionalitätsfaktor von 1 ein.
- *Link*: Links haben keine Aufgabe in dieser Phase.
- *Senke*: Entsprechend ihren Formatbeschränkungen entfernt eine Senke die nicht unterstützte A-QoS aus der erhaltenen AFS. Zusätzlich entfernt sie die A-QoS, die nicht im, vom Klient definierten Bereich liegt und signalisiert das Ende der ersten Phase Inform.

Wenn alle Senken eine AFS erhalten haben, ist die Phase Inform beendet. Alle Komponenten besitzen jetzt Informationen über Beschränkungen der stromaufwärts liegenden Komponenten. Jeder Mixer besitzt jetzt die ID und jeder (variable) Filter noch den Proportionalitätsfaktor und die, mit dem Proportionalitätsfaktor skalierte A-QoS der nächsten stromaufwärts liegenden Komponente des Typs (variablen) Filter oder Quelle. Diese Informationen werden in der nächsten Phase benötigt.

### 3.2.2 Zweite Phase, Reserve

In dieser Phase erhalten die Komponenten und Links Informationen über QoS-Beschränkungen der stromabwärts liegenden Komponenten und Links. Die Komponenten und Links führen Reservierungen durch und (variable) Filter, Mixer sowie Quellen senden Informationen zum Gleichungslöser. Ein Link an einem Ausgangsport erhält für jede Verbindung zu diesem Port eine AFS.

- *Senke*: Eine Senke führt entsprechend der AFS aus der ersten Phase Inform eine Reservierung durch und entfernt A-QoS, für die keine ausreichenden Ressourcen reserviert wurden.
- *Link*: Ein Link erhält für eine Verbindung eine AFS und bildet die enthaltene A-QoS auf die T-QoS des Transportsystems ab. Die Art der Abbildung ist vom Typ des Transportsystems abhängig. Die T-QoS wird dem Transportsystem übergeben, das eine Reservierung durchführt, die nicht unterstützte T-QoS entfernt und das Ergebnis an den Link zurückgibt. Der Link bildet die T-QoS auf die A-QoS einer AFS ab. Die Links an Ausgangsports mit mehrfachen Verbindungen (Multicast), bilden aus den AFS, die sie vom Transportsystem zurückerhalten, die Schnittmenge.

- *Filter*: Aus der in der ersten Phase Inform gespeicherten AFS entfernt der variable Filter die A-QoS, die kleiner als die kleinste A-QoS der erhaltenen AFS ist, führt eine Reservierung durch, entfernt die A-QoS für die keine ausreichenden Ressourcen reserviert werden konnten und übergibt die AFS wieder dem Protokollagenten. Zusätzlich sendet er an den Gleichungslöser die Relation zwischen seinem Eingangs- und Ausgangsport zusammen mit der ID und dem Proportionalitätsfaktor, die in der gespeicherten AFS (aus der ersten Phase) enthalten sind und seine eigene ID mit der A-QoS für seinen Ausgangsport. Wenn er der letzten Filter vor einer Senke ist, sendet er eine entsprechende Kennung mit.
- *Mixer*: Ein *Mixer* erzeugt aus der für seinen Ausgangsport erhaltenen AFS je eine AFS für jeden Eingangsport. Dazu benutzt er die entsprechenden Relationen zwischen den Ports. Zusätzlich sendet er die Relationen seiner Ports, zusammen mit der zu den Eingangsport gehörenden ID und dem Proportionalitätsfaktor (die in der ersten Phase Inform gespeichert wurden) und seine eigene ID, an den Gleichungslöser.
- *fester Filter*: Ein fester Filter verhält sich wie ein Mixer mit einem Eingangsport.
- *Quelle*: Eine *Quelle* führt entsprechend der erhaltenen AFS eine Reservierung durch und entfernt A-QoS, für die keine ausreichenden Ressourcen reserviert werden konnten. Diese A-QoS und ihre ID sendet sie an den Gleichungslöser. Dem Protokoll signalisiert sie das Ende der zweiten Phase Reserve.

Wenn alle Quellen das Ende der zweiten Phase Reserve signalisiert haben, startet das Protokoll den Gleichungslöser.

### 3.2.3 Gleichungslöser

Die dem Gleichungslöser zugesendete Daten enthalten:

- QoS-Wertebereiche von Filtern und Quellen.
- Eine Kennung, die für Filter und Quellen angibt, ob sie direkt vor einer Senke liegen. Direkt bedeutet hier, daß zwischen dem Filter und einer Senke bzw. der Quelle und einer Senke kein weiterer (variabler) Filter vorkommt.
- Eine Relation des Ausgangstroms eines Filters zu dem Ausgangstrom des direkt davor liegenden Filters bzw. der direkt davor liegenden Quelle.
- Die Relationen zwischen den Strömen die auf den gleichen Mixer führen und die ID der Filter bzw. der Quellen die diese Ströme bearbeiten bzw. erzeugen.

Aus diesen Daten kann der Gleichungslöser die beste Lösung für das Geflecht ermitteln, indem er die maximal mögliche QoS für die Ausgangsports der Quellen und Filter, die direkt vor einer Senke liegen, ermittelt und die entsprechende minimale QoS für die anderen Quellen und Filter. Dabei werden QoS-Werte nur für Quellen und variable Filter errechnet und dementsprechend auch nur an diese versendet, weil diese die QoS flexibel ändern können, während alle anderen Komponenten eine feste Relation zwischen der QoS des Eingangs- und des Ausgangsport besitzen.

Die ermittelten QoS-Werte werden an die Komponenten versendet. Danach signalisiert der Gleichungslöser den Start der dritten Phase Relax. Wenn keine Lösung existiert, wird das Scheitern signalisiert.

### 3.2.4 Dritte Phase, Relax

In dieser Phase wird der ausgehandelte QoS-Wert stromabwärts propagiert. Komponenten und Links führen ihre Initialisierungsmethoden aus und reduzieren ihre Reservierungen auf das benötigte Maß. Dabei ist zu beachten, daß die Quellen und Filter bereits ihre endgültigen QoS-Werte erhalten haben.

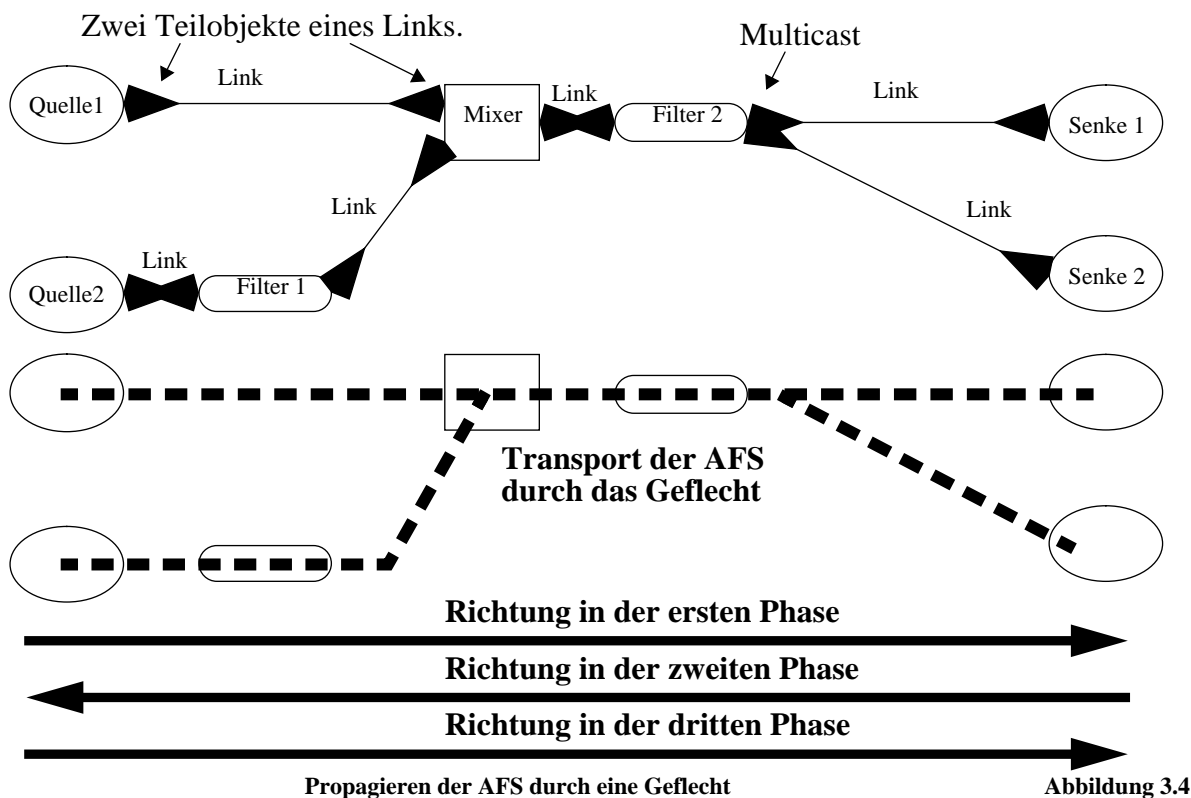
- *Quelle*: Eine Quelle erhält vom Gleichungslöser eine A-QoS, die sie zur Initialisierung verwendet und in einer AFS weitergibt. Nicht benötigte Ressourcen werden von ihr freigegeben.
- *fester Filter*: Ein fester Filter gibt nicht benötigte Ressourcen frei und skaliert die A-QoS der AFS entsprechend seinem Skalierungsfaktor.
- *Mixer*: Ein Mixer verhält sich entsprechend einem festen Filter. In dieser Phase erhält er für seine Eingangsports nur konsistente A-QoS, d.h. die Werte entsprechen den Relationen zwischen seinen Eingangsports.
- *Filter*: Ein variabler Filter erhält vom Gleichungslöser eine A-QoS, die er in einer AFS weitergibt. Seinen internen Skalierungsfaktor setzt er entsprechend der Relation zwischen der weitergegebenen und der A-QoS, die er für seinen Eingangsport erhalten hat. Nicht benötigte Ressourcen werden freigegeben.
- *Link*: Ein Link verhält sich weitestgehend wie in der zweiten Phase Reserve, jedoch wird vom Transportsystem eine Anpassung der Ressourcen durchgeführt.
- *Senke*: Die Senken geben nicht benötigte Ressourcen frei und signalisieren dem Protokoll das Ende der Aushandlung.

Haben alle Senken das Ende der Aushandlung signalisiert, so ist die dritte Phase Relax und damit das Protokoll erfolgreich beendet. In dieser Phase kann die Aushandlung nicht mehr scheitern, da der Gleichungslöser bereits eine realisierbare QoS ermittelt hat.

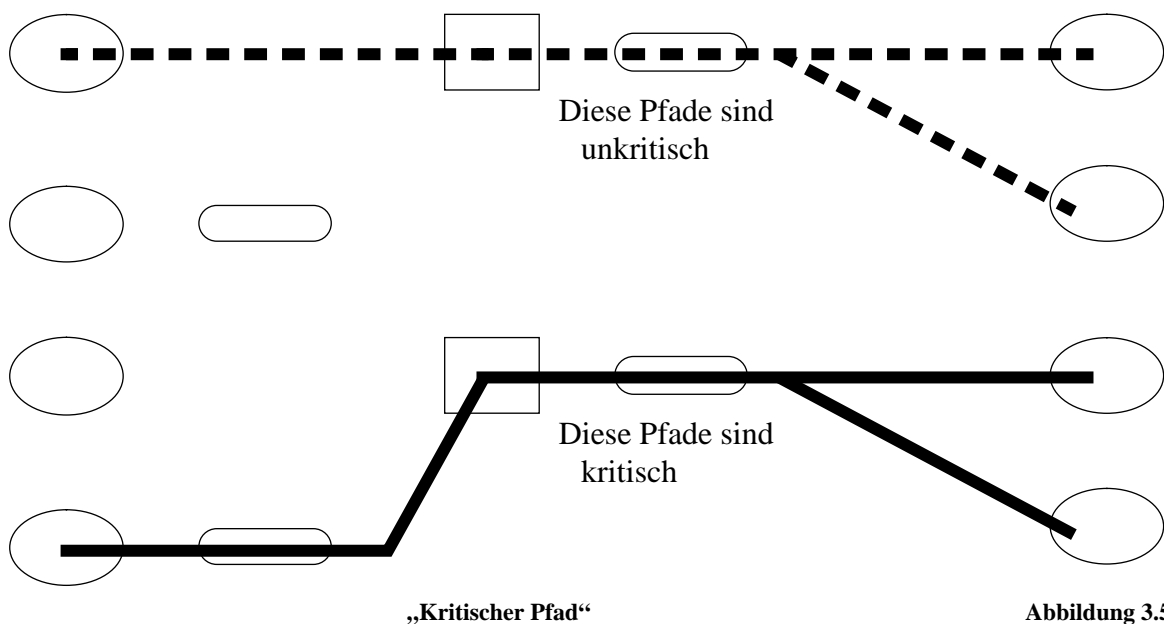
## 3.3 Zusammenfassung und Analyse der Protokolle

Beide Protokolle laufen in drei Phasen ab, in denen jeweils AFS stromabwärts oder stromaufwärts propagiert werden (Kapitel 3). Der Transport der AFS und die Aushandlungen an den Komponenten und Links kann dabei parallel durchgeführt werden. Ausnahmen sind Mixer und Multicast Verbindungen, bei denen mehrere AFS für die Aushandlungen benötigt werden.

Bei dem stromabwärts Propagieren der AFS, muß mit der Fortsetzung des Protokolls gewartet werden, bis für alle Eingangsports eines Mixers die AFS angekommen ist. Entsprechend muß, bei dem stromaufwärts Propagieren der AFS, an Multicast Verbindungen gewartet werden.



Die benötigte Zeit für eine Phase wird von der längsten Zeit bestimmt, die von einer AFS für den Weg von einer Quelle zu einer Senke bzw. von einer Senke zu einer Quelle benötigt wird. Wenn man davon ausgeht, daß eine AFS länger unterwegs ist wenn sie mehr Komponenten passieren muß, kann man einen kritischen Pfad zwischen Quellen und Senken anhand der Komponentenanzahl abschätzen.



Die Anzahl der Komponenten ist für genaue Angaben nicht immer ausreichend. Wenn der kritische Pfad mehr lokale Verbindungen als andere Pfade enthält, kann sich die benötigte Zeit verkürzen, da diese einen schnelleren Transport der AFS bewirken.

Die Gesamtzeit, für die Ausführung des NRP-Protokolls ist demnach unabhängig von der Komplexität des Geflechts und der Gesamtzahl von Komponenten und Links. Sie ergibt sich aus der von der AFS benötigten Zeit für das dreimalige Durchlaufen des kritischen Pfads.

XNRP entspricht weitgehend NRP (siehe oben). Es muß jedoch noch der Zeitbedarf für den Gleichungslöser berücksichtigt werden. Dieser Zeitbedarf setzt sich aus dem Zeitbedarf für die Kommunikation und dem Zeitbedarf für die Berechnung der Lösung zusammen. Dabei ist zu beachten, daß der Transport der Nachrichten asynchron erfolgt. Die Nachrichten von den Komponenten werden parallel zum Ablauf der zweiten Phase übertragen und der Start der dritten Phase kann unmittelbar nach dem Absenden der Lösungen erfolgen. Durch die Konzentration der Nachrichten am Gleichungslöser kann es aber trotzdem zu Verzögerungen kommen.

## 4. Architektur des Cinema-Systems

In diesem Kapitel wird die Architektur und die Implementierung des Cinema-Systems beschrieben. Zuerst wird die Aufteilung der Funktionen auf einzelne Prozesse beschrieben und dann die Organisation der einzelnen Prozesse in einer Threadstruktur.

### 4.1 Organisation der Funktionen in Cinema

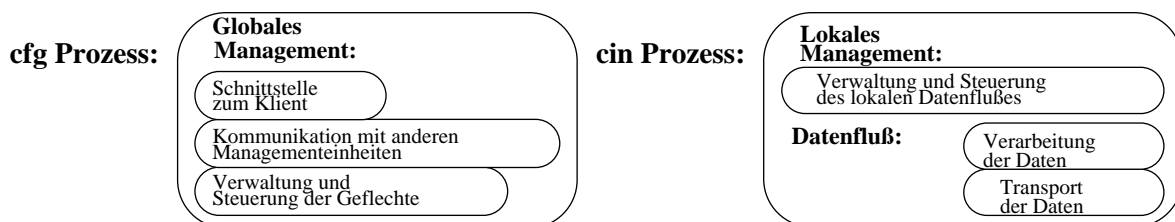
Die für ein verteiltes Multimedia-Systeme benötigten Funktionen lassen sich in die Bereiche Datenfluß und Management aufteilen. Das Management enthält das globale und das lokale Management.

Der Bereich Datenfluß umfaßt die Funktionen für den Transport und die Verarbeitung der Daten. Hier werden Datenströme empfangen, verarbeitet und versendet.

Das lokale Management umfaßt die Verwaltung und Steuerung des lokalen Datenflusses. Dazu gehört das Erzeugen und Zerstören von Datenströmen und Verarbeitungseinheiten, sowie das Starten und Stoppen des Transports und der Verarbeitung der Daten.

Das globale Management umfaßt die Verwaltung und Steuerung der Geflechte, die Schnittstelle zum Klient und die Kommunikation mit anderen Managementeinheiten. Dazu gehört das Empfangen der Konfiguration eines Geflechts vom Klient, der Aufbau des Geflechts durch entsprechende Anweisungen an das lokale Management und durch Nachrichten an andere Managementeinheiten im Netzwerk, das Weiterleiten von Befehlen des Klient zum Starten, Stoppen oder Steuern des Datenflusses und der Abbau des Geflechts.

Die Funktionen der Bereiche Datenfluß und lokales Management sind in Cinema in dem Prozeß „cin“ (cinema) zusammengefaßt, die Funktionen des globalen Management in dem Prozeß „cfg“ (configuration-manager).



Aufteilung der Funktionen auf die Prozesse des Cinema-Systems

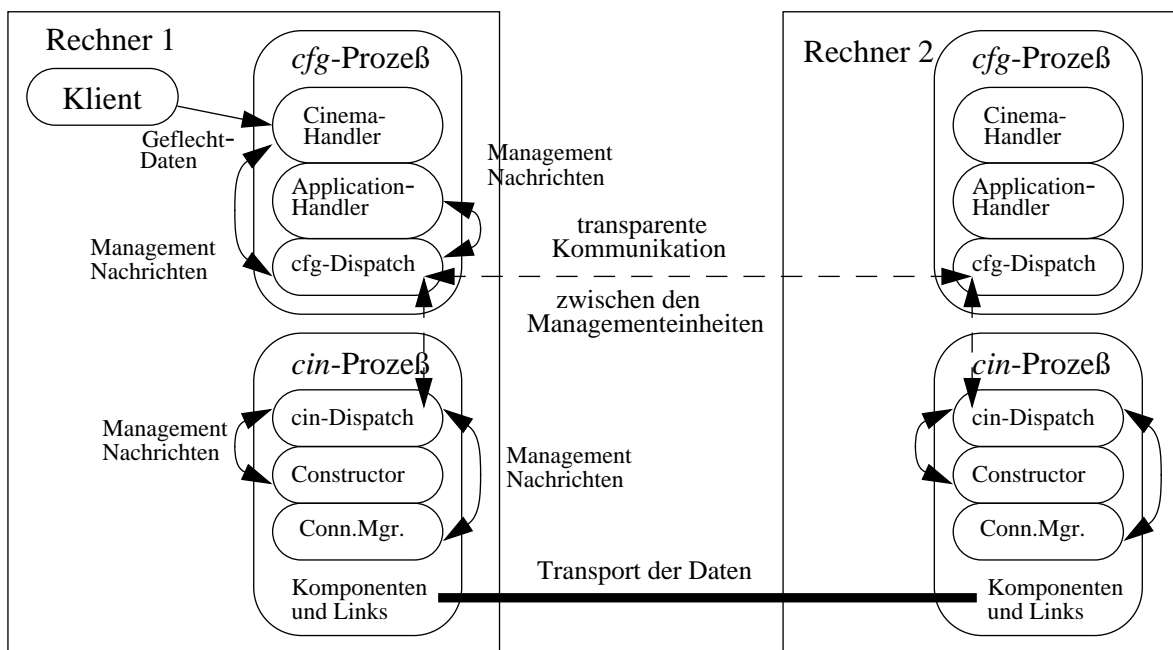
Abbildung 4.1

## 4.2 Thread Struktur

Die Prozesse des Cinema-Systems enthalten eine Threadstruktur, in der die Funktionen organisiert sind.

Der *cfg*-Prozeß enthält den Thread Cinema-Handler, der die Schnittstelle für den Klient zur Verfügung stellt, den Application-Handler für die Verwaltung und Steuerung der Geflechte, und er enthält den Thread *cfg*-Dispatch, der Funktionen für die Kommunikation mit anderen Managementeinheit zur Verfügung stellt und eine transparente Kommunikation mit den Managementeinheiten anderer *cfg*-Prozesse, sowie den Managementeinheiten des lokalen *cin*-Prozesses realisiert.

Der *cin*-Prozeß enthält die Threads für das lokale Management. Der Constructor Thread verwaltet und steuert die Komponenten. Der Connection-Manager Thread verwaltet und steuert die Links. Zusätzlich enthält der *cin*-Prozeß den Thread *cin*-Dispatch, der den anderen Threads des *cin*-Prozesses eine transparente Schnittstelle zu dem *cfg*-Dispatch zur Verfügung stellt. Die Threads für den Datenfluß, das sind die Komponenten für die Verarbeitung der Daten und die Links für den Transport der Daten, werden dynamisch zur Laufzeit erzeugt und auch wieder zerstört.



Threadstruktur und Kommunikation des Cinema-Systems

Abbildung 4.2

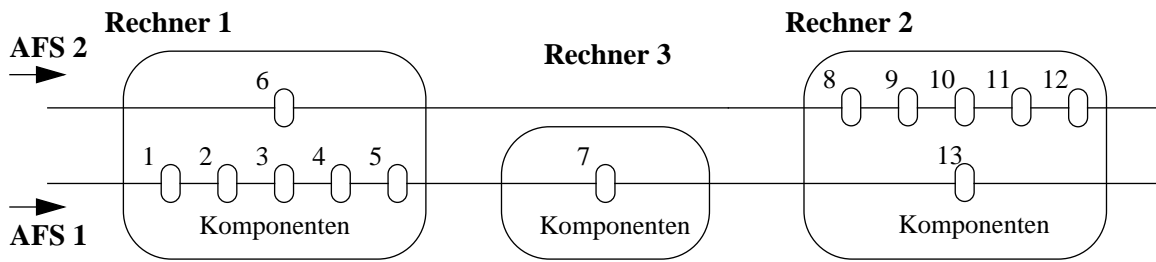
## 5. Implementierung der Protokolle

### 5.1 Protokollagenten

In den Protokolldefinitionen ist jeweils ein Protokollagent für jede Komponente und jeden Link vorgesehen, der dynamisch während des Protokollablaufs erzeugt und bei Beendigung des Protokolls wieder zerstört wird. Der Vorteil dieser Lösung ist eine hohe Parallelisierung des Protokollablaufs. Für die Integration in Cinema zeigte sich dieses Konzept jedoch als nachteilig. Die Architektur des Cinema-Systems sieht das dynamische Erzeugen und Zerstören von Threads nur für Komponenten und Links vor, aber nicht für Threads mit Verwaltung- oder Steueraufgaben. Threads für diesen Aufgabenbereich existieren nur einmal innerhalb des Prozesses und werden beim Start des Prozesses erzeugt und bei dessen Beendigung zerstört. Auch für die Kommunikation über den cin-Dispatch ist ein dynamischer Auf- und Abbau der Verbindungen nicht vorgesehen. Die Verwaltungs- und Steuermethoden der Komponenten und Links werden innerhalb des Constructor und des Connection-Manager Threads ausgeführt und besitzen keine eigene Kommunikationsmöglichkeit. Eine Implementierung der Protokolle mit dynamischen Protokollagenten würde nicht der Architektur des Cinema-Systems entsprechen und es müßten eine Threadverwaltung und eine Kommunikationsschnittstelle für die Protokollagenten integriert werden.

Eine Lösung mit einem einzelnen statischen Protokollthread, der die Aufgaben der Protokollagenten für alle Komponenten und Links auf einem Rechner bzw. in einem Cinema-System übernimmt, hat den Vorteil, der Architektur und Implementierung des Cinema-Systems zu entsprechen. Dadurch könnten die Schnittstellen zum System einfach gehalten werden und es ist auch mit einer höheren Geschwindigkeit bei den einzelnen Abläufen zu rechnen. Der Nachteil dieser Lösung wäre jedoch die Sequentialisierung der Aushandlung.

Bei einem Vergleich mit einer parallelen Bearbeitung zeigen sich jedoch nur in speziellen Konfigurationen reale Nachteile. Bei der Bearbeitung einer einzelnen AFS sind sogar Vorteile durch den geringeren Verwaltungsaufwand zu erwarten. Wenn jedoch mehrer AFS gleichzeitig ausgehandelt werden sollen treten Verzögerungen auf. Eine AFS muß warten, bis alle vor ihr kommenden AFS fertig bearbeitet sind. Wird die AFS nur von einer einzelnen Komponente bearbeitet, so ist keine große Verzögerung zu erwarten, da die AFS im weiteren Ablauf verteilt werden und die Wahrscheinlichkeit sinkt, daß zwei gleichzeitig einen Rechner erreichen. Wenn sich jedoch längere Ketten von Komponenten auf den Rechnern befinden, kann es leichter zu Situationen kommen, bei denen sich die AFS „gegenseitig“ aufhalten. In Abbildung 5.1 ist ein Szenario gezeigt bei dem es zu erheblichen Verzögerung kommen kann.



Verzögerungen bei sequenziellem Ablauf des Protokolls

Abbildung 5.1

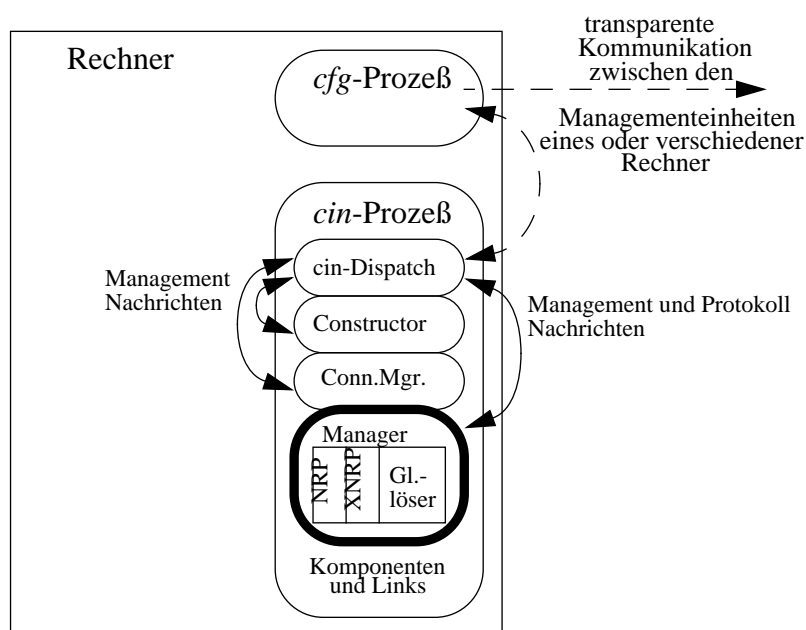
Wird auf dem Rechner 1 die AFS 1 zuerst abgearbeitet, so muß die AFS 2 warten, bis die AFS 1 die Komponenten 1 bis 5 durchlaufen hat. Erst wenn die AFS 1 fertig ist, kann die AFS 2 an der Komponente 6 bearbeitet werden. Durch die Aushandlung auf Rechner 3 wird die AFS 1 verzögert und die AFS 2 kann früher an Rechner 2 ankommen, damit muß die AFS 1 warten. Auch wenn die AFS 1 früher an Rechner 2 ankommt, gibt es einen Zeitverlust, da die AFS 2 bereits an Rechner 2 fertig sein könnte. Im besten Fall entspricht der Durchlauf der beiden AFS die Aushandlung an 6 bzw. 7 Komponenten, im schlechtesten Fall entspricht sie der Aushandlung an 11 bzw. 12 Komponenten. Das Ergebnis läßt sich jedoch verbessern, wenn nicht eine AFS vollständig abarbeitet werden muß, sondern die Aushandlung verschränkt durchgeführt werden kann. Wenn man mit jeder AFS jeweils nur einen Teil der Aushandlung durchführt und dann zu einer anderen AFS wechselt, verbessert sich der schlechteste Fall für das angegebene Beispiel auf 7 bzw. 8 Aushandlungen. Beide AFS werden maximal für den Zeitraum einer Aushandlung aufgehalten.

Bei einer parallelen Lösung kann man bei einer solchen Konfiguration sogar mit einem schlechteren Gesamtergebnis der Aushandlung rechnen, da auch hier die Ketten sequenziell abgearbeitet werden und durch den größeren Verwaltungsaufwand zusätzliche Verzögerungen entstehen. Die Implementierung der Protokolle wurde deshalb mit einem statischen Thread innerhalb des cin-Prozesse realisiert, wobei nach jedem Bearbeitungsschritt zu einer anderen AFS gewechselt wird.

Ein Bearbeitungsschritt besteht hierbei aus der Aushandlung einer Komponente und der Links, die mit dieser Komponente verbunden sind. Die Definition eines Bearbeitungsschritts wurde so gewählt, weil bei der Aushandlung einer Komponente, für jeden ihrer Ports auch eine Aushandlung an einem Link auf demselben Rechner erfolgt. Ein Link zwischen zwei Komponenten auf verschiedenen Rechnern, besteht immer aus zwei Teilen (Kapitel 3). Von diesen befindet sich jeweils ein Teil auf demselben Rechner mit einer der beiden Komponenten. Beim Ablauf der Protokolle müssen stets bei beiden Teilen die Aushandlungsmethoden aufgerufen werden. Hinzu kommt, daß die Verwaltung von Komponenten und Links innerhalb von Cinema unterschiedlich realisiert ist und die in den Protokolldefinitionen vorgesehene Gleichbehandlung für die Aufrufe der Aushandlungsmethoden nicht umsetzbar war. Deshalb wurde die Aushandlung als eine Sequenz von drei Schritten realisiert, die einen Bearbeitungsschritt bilden. Zuerst werden die Aushandlungsmethoden der Links, für die eine AFS empfangen wurde, aufgerufen. Dann erfolgt die Aushandlung an der Komponente und danach die Aushandlung an den Links, für die eine AFS von der Komponente zurückgegeben wurde. Unterbrechungen können bei der stromabwärts Aushandlung eines Mixers und bei der stromaufwärts Aushandlung eines Aus-

gangsports mit mehreren Verbindungen auftreten. Wenn bei der stromabwärts Aushandlung eines Mixers nicht alle AFS für die Eingangsports verfügbar sind. Dann wird für die vorhandenen AFS nur die Aushandlung der Links durchgeführt, das Zwischenergebnis gespeichert und die Bearbeitung abgebrochen. Sobald die fehlenden AFS eintreffen, wird die Aushandlung entsprechend fortgesetzt. (In der Zwischenzeit können andere AFS bearbeitet werden.) Entsprechend wird bei der stromaufwärts Aushandlung eines Ausgangsports mit mehreren Verbindungen, die Aushandlung der vorhandenen AFS für die zugehörige Verbindung durchgeführt und das Ergebnis gespeichert, bis alle benötigten AFS verfügbar sind.

Der Protokollthread wurde innerhalb des cin-Prozesses eingerichtet, um den direkten Zugriff auf die Methoden der Komponenten und Links zu ermöglichen (siehe Abschnitt 4.2). Er enthält vier Objekte die in C++ realisiert sind. Jeweils ein Objekt für NRP und XNRP, einen Gleichungslöser und ein Manager Objekt.



Erweiterung des cin Prozesses um einen Thread

Abbildung 5.2

Das Manager Objekt enthält die Cinema-spezifischen Teile. Es initialisiert den Thread und baut die Kommunikationsverbindung mit dem cin-Dispatch auf. Während des Protokollablaufs nimmt es Nachrichten vom cin-Dispatch entgegen und ruft die Methoden der anderen Objekte, entsprechend der erhaltenen Nachricht auf.

Das NRP Objekt enthält NRP spezifische Methoden. Mit NRPStart wird die Aushandlung gestartet, als Parameter wird eine „leere“ AFS (d.h. ohne QoS-Angaben) für eine Quellenkomponente übergeben. Die Methoden ReserveDown, InformUp, RelaxDown werden während des weiteren Ablaufs aufgerufen und erhalten als Parameter jeweils eine AFS. Alle Methoden führen einen Bearbeitungsschritt durch und senden die erzeugten AFS weiter. Weitere Methoden sind Answer und PortParameters. Die Methode Answer speichert die Adresse des Application-Handler, der das NRP Protokoll gestartet hat. Diese Adresse wird von den Senken verwendet, um den (Miß-)Erfolg des Protokolls zu melden. Die Methode PortParameter wird verwendet, um die vom Klient gewünschte Beschränkungen der QoS an den Eingangsports festzulegen.

Das XNRP Objekt enthält die Methoden XNRPSstart, InformDown, ReserveUp, RelaxDown, Answer und PortParameters, mit einer den NRP Methoden entsprechenden Funktionalität. Zusätzlich sind die Methoden Result und XNRPContinue enthalten. Die Methode Result übergibt Ergebnisse des Gleichungslöser an die Komponenten. Die Methode XNRPContinue startet die dritte Phase an einer Quelle.

Beide Objekte enthalten noch private Methoden für die einzelnen Aushandlungen an Komponenten und Links, für das Zwischenspeichern der AFS bei stromabwärts Aushandlungen an Mixern (siehe oben), für das stromaufwärts und stromabwärts Senden der AFS und für das Senden der (Miß-)Erfolgs Meldung bei Beendigung der Protokolle. Das XNRP Objekt enthält noch eine Methode, um die Beendigung der zweiten Phase zu melden.

Das Gleichungslöser Objekt enthält die Methode CNPSave, um zugesendete QoS und Relationen zu speichern, sowie die Methode CNPStart, mit der die Berechnung gestartet wird. Durch die Integration des Gleichungslöser in den cin-Prozeß ist auf jedem Rechner, innerhalb des Geflechts, ein Gleichungslöser vorhanden. Vom XNRP Protokoll wird jeweils der Gleichungslöser verwendet, der sich auf dem gleichen Rechner wie der Klient befindet. Durch diese Lösung entsteht kein zusätzlicher „Single Point of Failure“, wie es bei anderen Realisierungen, z.B. als Prozeß auf einem entfernten Rechner, der Fall wäre, da bei einem Ausfall des Cinema-Systems auf diesem Rechner, der Aufbau des Geflechts auf jeden Fall scheitert.

## 5.2 Schnittstelle zum Klient

Die Implementierung innerhalb des cin-Prozesses ist weitgehend transparent für Benutzer und Applikationen. Die Programmierschnittstelle des Cinema-Systems wurde lediglich um Funktionen für die Auswahl eines Protokolls und eine Eingabemöglichkeit für die gewünschte QoS erweitert.

Die Auswahl des Protokolltyps erfolgt durch die Methode TransmitNRP des Objekts „CIN\_PortList\_t“ (in dem auch die Methoden CreateSession und DeleteSession enthalten sind). Als Parameter wird ein Integer übergeben, der folgendermaßen interpretiert wird: Bei dem Wert 1 wird NRP verwendet, bei dem Wert 2 wird XNRP verwendet, bei einem anderen Wert wird kein Protokoll verwendet.

Die Eingabe der gewünschten QoS ist über die Methode SetQoSParams des Objekts CIN\_Port\_t möglich. Als Parameter wird als erstes ein Array aus Stringadressen (\*(char[])) und als zweites eine Integer Zahl erwartet. Die Integer Zahl muß die Anzahl der Adressen in dem Array angeben. Das Array selbst enthält jeweils drei zusammengehörende Adressen, die erste von einem String mit der Bezeichnung des Parameters, die zweite auf einen String mit der maximal gewünschten QoS und die dritte auf einen String mit der minimal gewünschten QoS. Die Parameter müssen dem Stromtyp entsprechen. Es müssen nicht alle Strings einen Inhalt besitzen, in diesem Fall wird der maximal bzw. minimal mögliche Wert angenommen. Wenn für einen Port keine QoS angegeben wurde, wird die maximal mögliche QoS verwendet. In der Protokolldefinition ist die Angabe der QoS an den Inports der Senken vorgesehen, in der Realisierung sind Angaben für alle Komponenten eines Geflechts möglich.

## 5.3 Kommunikation der Protokollthreads

Die Kommunikation zwischen den Protokollthreads, dem Application-Handler und dem Gleichungslöser erfolgt über die „send“ Methode und Nachrichtenpuffer, die das Cinema-System zur Verfügung stellt. Diese realisieren eine transparente und asynchrone Nachrichtenübermittlung zwischen den Threads eines oder verschiedener Cinema-Systeme [4]. Sie werden innerhalb des Cinema-Systems für den Aufbau und die Steuerung der Geflechte benutzt. Der Vorteil der Nutzung dieser Methode für die Kommunikation der Protokollthreads, gegenüber dem Aufbau eigener Verbindungen, ist das Vermeiden des zeitintensiven Aufbaus von Verbindungen innerhalb des Protokollablaufs. Alle für die Protokolle notwendigen Verbindungen, zwischen den verschiedenen Rechnern bzw. Cinema-Systemen, wurden bereits für den Aufbau der Komponenten sowie Links erstellt und können für den Protokollablauf verwendet werden.

Als Parameter erwartet die „send“ Methode ein Adressobjekt, das den Prozeß und optional den Thread des Empfängers beschreibt, eine Opcode und die zu transportierenden Daten. Der Opcode muß innerhalb eines im cfg- und cin-Dispatch vorgegebenen Zahlenbereich für den Zielthread liegen. Für NRP und XNRP wurde der Bereich auf 2500 bis 2599 gelegt. Der Transport erfolgt für den Aufrufer transparent, innerhalb des Rechners oder über das Netzwerk zu dem angegebenen Prozeß. Ist der Thread des Empfängers nicht definiert, so wird dieser durch den Opcode vom cfg-Dispatch ermittelt. Die Nachricht wird in dem Empfangspuffer des Threads abgelegt. Der Transport einer Nachricht von einem Protokollthread zu einem anderen Protokollthread über den cin-Dispatch. Dieser sendet die Nachricht zum cfg-Dispatch, von wo aus sie über das Netzwerk an den cfg-Dispatch auf dem Zielrechner gesendet wird. Dieser leitet sie zum cin-Dispatch des Zielrechners, der die Nachricht dem Empfängerthread übergibt.

Durch die Verwendung der send Methode ergab sich eine einfache Möglichkeit, den Wechsel der AFS zwischen den Bearbeitungsschritten zu realisieren. Nach der Durchführung eines Bearbeitungsschritts, wird die AFS (eventuell auch mehrerer AFS) mit der „send“ Methode weitergegeben, unabhängig davon ob die nächste Komponente auf dem gleichen oder einem anderen Rechner liegt. Liegt das Ziel auf einem anderen Rechner, wird die AFS vom cin-Dispatch weitergeleitet und die nächste AFS kann bearbeitet werden. Liegt das Ziel auf dem gleichen Rechner, so wird die AFS als letztes Element in den Empfangspuffer gelegt und der Protokollthread fährt mit der Bearbeitung der ersten AFS im Puffer fort.

Aufgrund der Ähnlichkeit der Protokolle wurde eine gemeinsame Struktur für die AFS gewählt. Die Grundstruktur der AFS enthält alle Elemente, die nicht von den Stromtypen abhängig sind.

Für beide Protokolle gemeinsam sind das:

- ein Feld, das den Typ des verwendeten Protokolls angibt. Dieses Feld wird von den Komponenten benötigt, da hier die Methoden für die verschiedenen Protokolle nicht explizit getrennt sind.
- ein Feld, das den Stromtyp angibt. Dieses Feld wird für die Interpretation der QoS-Daten benötigt.
- ein Feld, das die Größe der QoS-Daten in Bytes angibt. Dieses Feld wird für das Versenden benötigt, da die Größe nicht konstant ist.
- einen Zeiger auf die QoS-Daten. In diesem Speicherbereich sind die QoS und alle stromspezifischen Informationen enthalten.

Für NRP sind keine speziellen Informationen enthalten, der DRV (Abschnitt 3.1) ist vom Stromtyp abhängig und in den QoS-Daten enthalten.

:Für XNRP sind zwei spezielle Elemente enthalten:

- eine Feld, das von den Senken gesetzt wird und während dem stromaufwärts propagieren von den variablen Filtern gelöscht wird. Filter und Quellen können so erkennen, ob noch ein Filter zwischen ihnen und den Senken liegt.
- eine ID der nächsten, stromaufwärts liegenden Komponente vom Typ Quelle oder Filter (variabel).

Der Proportionalitätsfaktor (Abschnitt 3.2) ist vom Stromtyp abhängig und in den QoS-Daten enthalten.

Die Komponenten müssen die QoS-Daten einer AFS, die an die Protokollthreads übergeben wird, in einem einzelnen Speicherbereich ablegen, dessen Größe und Anfangsadresse in der AFS eingetragen werden. Der Transport einer AFS, zwischen zwei Protokollthreads, erfolgt entlang eines Links zwischen zwei Komponentenports. Die AFS wird für den Transport in eine Nachrichtenstruktur verpackt, in der auch eine Identifikation des Ports und der Komponente, zu denen der Link führt, enthalten ist. Die Identifikation erfolgt durch eine eindeutige Kennung innerhalb des Cinema-Systems, in Form einer Thread-, einer Komponenten- und einer Portnummer, mit denen der Protokollthread des Zielrechners die entsprechenden Elemente ermitteln und den Protokollablauf weiterführen kann.

## 5.4 Aushandlungsmethoden der Komponenten und Links

Komponenten enthalten Verwaltungs-, Steuer- und eine „Echtzeit“ Methode. Die Aufgabe der „Echtzeit“ Methode ist die kontinuierliche Verarbeitung der Daten. Sie wird daher in einem eigenen Thread ausgeführt. Die Verwaltungs- und Steuermethoden sind nicht in einem eigenen Thread enthalten, sondern werden bei Bedarf von anderen Threads des cin-Prozesses ausgeführt. Zu diesen Verwaltungs- und Steuermethoden wurden die Methoden look\_up, SendCNP, GetResult, SetQoS, Inform, Reserve, Relax und Free hinzugefügt, die für die Aushandlung der QoS und die Reservierung von Ressourcen notwendig sind. Die Methode look\_up liest die Formatbeschränkungen und Relationen aus der Komponentenbeschreibungsdatei ein und speichert sie innerhalb der Komponente. Die Formatbeschränkungen jedes Ports und die Relationen zwischen den Ports werden in einem speziellen Format im Quellcode der Komponente angegeben und beim Übersetzen vom Compiler in die Datei mit der Komponentenbeschreibung eingetragen. SendCNP wird in der zweiten Phase des XNRP Protokolls benötigt, um Daten an den Gleichungslöser zu senden. GetResult empfängt die Antwort des Gleichungslöser und setzt die QoS der Komponente auf den empfangenen Wert. Inform, Reserve, Relax, und Free werden von den Protokollthreads aufgerufen (Kapitel 3). Diese Methoden sind vom Stromtyp und der Komponentenklasse abhängig und müssen für jede Komponenten speziell definiert werden. Bei einem Scheitern geben die Methoden eine leere AFS zurück, d.h. eine AFS ohne QoS-Daten.

Für die Durchführung der Messungen und zur Demonstration der Protokolle wurden vier Komponentenklassen mit entsprechenden Protokollmethoden, Formatbeschränkungen und Relationen realisiert. Alle Komponenten sind für die Bearbeitung unkomprimierter Videoströme mit 8 Bit Farbtiefe.

- *VideoFileSource*: Typ Quelle, liest eine Sequenz von Videobildern aus einer Datei und gibt diese, als Bilder mit einer festen Bildgröße, weiter. Besitzt keine Formatbeschränkungen.
- *VideoFilter*: Typ (variabler) Filter, skaliert Höhe und Breite eines Bildes. Besitzt eine kontinuierliche Formatbeschränkung am Eingangsport und eine diskrete Formatbeschränkungen am Ausgangsport.
- *VideoMixer*: Typ Mixer, erzeugt aus zwei Bildern gleicher Größe, durch übereinanderstellen, ein Bild mit doppelter Höhe. Besitzt eine kontinuierliche Formatbeschränkung für die Eingangsports und den Ausgangsport.
- *VideoWindow*: Typ Senke, gibt Bilder in einem Grafikfenster aus. Besitzt eine kontinuierliche Formatbeschränkung für den Eingangsport.

Die Protokollmethoden Inform, Reserve und Relax sind bisher nur für die Bildgröße implementiert und erwarten diskrete QoS-Mengen in der AFS, eine Erweiterung, z.B. für die Aushandlung der Bildrate und für die Angabe kontinuierlicher QoS-Bereiche in der AFS, ist vorbereitet. Die Ressourcenmanager für Speicher und CPU sind bisher nicht in Cinema integriert. Eine reale Reservierung dieser Ressourcen findet daher nicht statt. Für den Test der Protokolle wurde jedoch die Möglichkeit realisiert, Ressourcenbeschränkungen zu simulieren.

Die Protokollmethoden sind nach einem einheitlichen Schema aufgebaut: Der Protokollthread übergibt die AFS für einen Port. Aus den Bildgrößen in der übergebenen AFS und den Bildgrößen der gespeicherten Formatbeschränkungen des Ports wird die Schnittmenge gebildet. Bei den Eingangsports des VideoMixer geschieht dies für jeden Eingangsport und aus den Resultaten wird eine einzelne AFS berechnet. Danach wird die entsprechende Funktion der Methode durchgeführt, z.B. wird eine Reservierung simuliert und Bildgrößen für die keine ausreichenden Ressourcen verfügbar sind werden entfernt. Aus dem Resultat wird wiederum das Ergebnis für den Protokollthread berechnet, z.B. wenn ein VideoMixer als Ergebnis eine AFS für seinen Ausgangsport berechnet, muß er die Angaben der Bildhöhe verdoppeln. Wenn die Formatbeschränkungen und die Berechnungsschritte innerhalb der Komponente konsistent sind, so entspricht das Ergebnis den Formatbeschränkungen des Ports für das es berechnet wurde und muß nicht nochmals überprüft werden. Zur Sicherheit wurde dennoch eine entsprechende Überprüfung in den Komponenten eingefügt.

Alle oben genannten Komponente können für die Bearbeitung entsprechender Videoströme in realen Applikationen eingesetzt werden. Nur für die Messungen wurde die Komponente XMixer realisiert. Sie hat keine reale Funktion, die Bilder an den Eingangsports müssen die gleiche Größe besitzen und das erzeugte Bild besitzt die Größe eines Bildes an den Eingangsports. Sie besitzt kontinuierliche Formatbeschränkungen für die Eingangsports und den Ausgangsport. Die Anzahl von Eingangsports kann im Code einfach variiert werden, um Messungen mit einer unterschiedlichen Anzahl von Eingangsports durchführen zu können.

Die Links wurden um die Methoden Inform, Reserve, Relax, Free und MixAFS für den Protokollablauf erweitert. Die Methoden Inform, Reserve, Relax und Free wurden jedoch nicht aus-

programmiert, sondern verweisen auf entsprechende Methoden die bereits im Rahmen einer Studienarbeit [13] im Ressourcenmanager für das Transportsystem implementiert wurden. Die Methode MixAFS wird in der zweiten Phase der Protokolle benötigt. Sie berechnet die AFS für den Ausgangsport einer Komponente aus den verschiedenen AFS einer Multicast-Verbindung (Kapitel 3). Dabei erwartet sie jeweils zwei AFS als Parameter und gibt eine AFS als Ergebnis zurück. Bei mehr als zwei Verbindungen muß der Aufruf entsprechend oft wiederholt werden.

## 5.5 Gesamtablauf

In diesem Abschnitt werden die Vorgänge in den Protokollthreads während eines Protokollablaufs beschrieben. Die Methoden der Komponenten entsprechen den Protokolldefinitionen in Kapitel 3.

Nach der Übertragung, der Konfiguration des Geflechts, der Angabe des gewünschten Protokolls und der Angabe der gewünschten QoS, z.B. Bildgröße und Bildrate, an den Senken, kann der Klient das Kommando zum Starten der Übertragung und der Verarbeitung der Daten geben. Der Applikation-Handler steuert dann den Aufbau, den Protokollstart und den eigentlichen Start des Geflechts durch Nachrichten an die Constructor, Connection-Manager und Protokoll Threads der beteiligten cin Prozesse. Wenn die maximal mögliche QoS erwünscht wird, müssen keine Angaben an den Senken gemacht werden.

Der erste Schritt ist der Aufbau des Geflechts bei dem die Komponenten und Links auf den Rechnern eingerichtet werden. Für das Einrichten der Komponenten sendet der Applikation-Handler, für jede Komponente jeweils eine Nachricht mit der Angabe der Komponentenklasse, an den Constructor des Cinema-Systems des Rechners auf dem die Komponenten plaziert werden soll. Der Constructor lädt den Code der Klasse in einen Speicherbereich und erzeugt einen Thread für die Echtzeit Methode, der jedoch noch nicht gestartet wird. Für das Einrichten der Links sendet der Applikation-Handler, für jeden Link jeweils eine Nachricht mit der Angabe der zu verbindenden Komponenten und des jeweiligen Ein- und Ausgangsports, an den Connection-Manager auf dem Rechner der Komponente die Daten an dem Eingangsport empfangen soll. Der Connection-Manager richtet einen Link für den Eingangsport der Komponente ein und sendet eine Nachricht zu dem Connection-Manager auf dem Rechner der Komponente die Daten an ihrem Ausgangsport senden soll. Dieser Connection-Manger richtet einen Link für den Ausgangsport der Komponente ein, der ein entsprechendes Gegenstück des Links auf der Empfängerseite darstellt. Die eigentliche Verbindung für den Datentransport wird noch nicht eingerichtet.

Wenn alle Komponenten und Links des Geflechts eingerichtet sind, ist der Aufbau beendet.

Für den Start des ausgewählten Protokolls, sendet der Applikation-Handler, für jede Quellkomponente, eine Nachricht an den Protokollthread auf dem Rechner, auf dem sich die Komponente befindet. Die Nachricht enthält die Identifikation der Komponente und den Opcode für den Start des entsprechenden Protokolls. Das Manager Objekt nimmt die Nachricht entgegen und Ruft die zum Opcode gehörende Methode auf. In diesem Fall NRPStart des NRP Objekts oder XNRPStart des XNRP Objekts.

### 5.5.1 NRP

Nach dem Start des Protokolls sendet der Application-Handler für jede Senke eine Anfrage nach dem Ergebnis an die Protokollthreads und wartet auf eine Antwort, während das Protokoll abläuft.

Die NRPSstart Methode ruft zuerst die Reserve Methode der Quelle auf. Als Parameter übergibt sie eine leere AFS (d.h. ohne QoS-Angaben) und erhält als Ergebnis eine AFS mit den QoS-Angaben der Quelle zurück. Mit dieser AFS ruft sie dann die Reserve Methode des Links für den Ausgangsport der Quelle auf. Der Link gibt als Ergebnis, für jede Verbindung des Ausgangsports, eine AFS zurück. Diese AFS werden von NRPSstart zusammen mit dem Opcode der ReserveDown Methode an die Protokollthreads der stromabwärts liegenden Komponenten versendet. Danach kann das Manager Objekt die nächste Nachricht bearbeiten. Diese Nachricht kann eine Start Nachricht für eine weitere Quelle auf diesem Rechner oder eine Nachricht von einem Protokollthread sein.

Erhält der Manager eine Nachricht mit dem Opcode der ReserveDown Methode, so wird diese von ihm aufgerufen. Die ReserveDown Methode ruft zuerst die Reserve Methode des Links auf, für den die AFS der Nachricht bestimmt ist. Besitzt die zugehörige Komponente mehrere Eingangsports und sind nicht alle benötigten AFS vorhanden, so wird das Ergebnis gespeichert und die Bearbeitung abgebrochen. Sind alle benötigten AFS vorhanden, so wird die Reserve Methode der Komponente aufgerufen und mit der resultierenden AFS wiederum, die Reserve Methode des Links des Ausgangsports. Der Link gibt als Ergebnis, für jede Verbindung des Ausgangsports, eine AFS zurück. Diese AFS werden von der ReserveDown Methode zusammen mit dem Opcode der ReserveDown Methode an die Protokollthreads der stromabwärts liegenden Komponenten versendet. Besitzt die Komponente jedoch keinen Ausgangsport, so handelt es sich um eine Senke. Dann ruft ReserveDown Methode, mit der resultierenden AFS der Reserve Methode der Komponente, die Inform Methode des Links jedes Eingangsports auf. Die resultierenden AFS sendet sie, zusammen mit dem Opcode der InformUp Methode, an die Protokollthreads der stromaufwärts liegenden Komponenten, die mit dem jeweiligen Eingangsport verbunden sind. Danach kann das Manager Objekt die nächste Nachricht bearbeiten.

Erhält der Manager eine Nachricht mit dem Opcode der InformUp Methode, so wird diese von ihm aufgerufen. Die InformUp Methode ruft zuerst die Inform Methode des Links auf, für den die AFS der Nachricht bestimmt ist. Besitzt die zugehörige Komponente mehrere Verbindungen am Ausgangsport und sind nicht alle benötigten AFS vorhanden, so wird das Ergebnis gespeichert und die Bearbeitung abgebrochen. Sind alle benötigten AFS vorhanden, so wird die Inform Methode der Komponente aufgerufen und mit der resultierenden AFS wiederum, die Inform Methoden der Links der Eingangsports. Jeder Link gibt als Ergebnis, für die Verbindung des Eingangsports, eine AFS zurück. Diese AFS werden von der InformUp Methode zusammen mit dem Opcode der InformUp Methode an die Protokollthreads der stromaufwärts liegenden Komponenten versendet. Besitzt die Komponente jedoch keinen Eingangsport, so handelt es sich um eine Quelle. Dann ruft die InformUp Methode, mit der resultierenden AFS der Inform Methode der Komponente, die Relax Methode des Links des Ausgangsports auf. Der Link gibt als Ergebnis, für jede Verbindung des Ausgangsports, eine AFS zurück. Diese AFS werden von InformUp zusammen mit dem Opcode der RelaxDown Methode an die Protokollthreads der stromabwärts liegenden Komponenten versendet. Danach kann das Manager Objekt die nächste Nachricht bearbeiten.

Erhält der Manager eine Nachricht mit dem Opcode der RelaxDown Methode, so wird diese von ihm aufgerufen. Die RelaxDown Methode ruft zuerst die Relax Methode des Links auf, für den die AFS der Nachricht bestimmt ist. Besitzt die zugehörige Komponente mehrere Eingangsports und sind nicht alle benötigten AFS vorhanden, so wird das Ergebnis gespeichert und die Bearbeitung abgebrochen. Sind alle benötigten AFS vorhanden, so wird die Relax Methode der Komponente aufgerufen und mit der resultierenden AFS wiederum, die Relax Methode des Links des Ausgangsports. Der Link gibt als Ergebnis, für jede Verbindung des Ausgangsports, eine AFS zurück. Diese AFS werden von der RelaxDown Methode zusammen mit dem Opcode der RelaxDown Methode an die Protokollthreads der stromabwärts liegenden Komponenten versendet. Besitzt die Komponente jedoch keinen Ausgangsport, so handelt es sich um eine Senke und die RelaxDown Methode sendet für die Senke eine (Miß-) Erfolgsmeldung an den Application-Handler. Danach kann das Manager Objekt die nächste Nachricht bearbeiten.

Der Mißerfolg kann nicht anhand medienspezifischer Parameter der QoS festgestellt werden, da die Protokollthreads Stromtyp unabhängig sind. Es ist jedoch auch nicht notwendig die QoS auszuwerten, da von den Komponenten nur QoS-Werte eingetragen werden die realisiert werden können. Wenn das Protokoll scheitert, so sind keine QoS-Daten mehr in der AFS enthalten und dies kann von den Protokollthreads jederzeit nachgeprüft werden. So muß der Mißerfolg nicht an den Senken erfolgen, sondern kann bereits an anderen Stellen des Geflechts eintreten und gemeldet werden.

Nachdem der Application-Handler von allen Senken die Erfolgsmeldung erhalten hat, fährt er mit dem Start des Geflechts fort. Wird ein Mißerfolg gemeldet, so sendet er für alle Komponenten und Links eine entsprechende Nachricht an die Protokollthreads, bereits reservierte Ressourcen freizugeben. Die Protokollthreads rufen dazu die Free Methoden der Komponenten und Links auf.

## **5.5.2 XNRP**

Nach dem Start des Protokolls sendet der Application-Handler für jede Quelle eine Anfrage nach dem Ergebnis an die Protokollthreads und wartet auf eine Antwort, während das Protokoll abläuft.

Die XNRPStart Methode ruft zuerst die Inform Methode der Quelle auf. Als Parameter übergibt sie eine leere AFS (d.h. ohne QoS-Angaben) und erhält als Ergebnis eine AFS mit den QoS-Angaben der Quelle zurück. Mit dieser AFS ruft sie dann die Inform Methode des Links für den Ausgangsport der Quelle auf. Der Link gibt als Ergebnis, für jede Verbindung des Ausgangsports, eine AFS zurück. Diese AFS werden von XNRPStart zusammen mit dem Opcode der InformDown Methode an die Protokollthreads der stromabwärts liegenden Komponenten versendet. Danach kann das Manager Objekt die nächste Nachricht bearbeiten. Diese Nachricht kann eine Start Nachricht für eine weitere Quelle auf diesem Rechner oder eine Nachricht von einem Protokollthread sein.

Erhält der Manager eine Nachricht mit dem Opcode der InformDown Methode, so wird diese von ihm aufgerufen. Die InformDown Methode ruft zuerst die Inform Methode des Links auf, für den die AFS der Nachricht bestimmt ist. Besitzt die zugehörige Komponente mehrere Eingangsports und sind nicht alle benötigten AFS vorhanden, so wird das Ergebnis gespeichert

und die Bearbeitung abgebrochen. Sind alle benötigten AFS vorhanden, so wird die Inform Methode der Komponente aufgerufen und mit der resultierenden AFS wiederum, die Inform Methode des Links des Ausgangsports. Der Link gibt als Ergebnis, für jede Verbindung des Ausgangsports, eine AFS zurück. Diese AFS werden von der InformDown Methode zusammen mit dem Opcode der InformDown Methode an die Protokollthreads der stromabwärts liegenden Komponenten versendet. Besitzt die Komponente jedoch keinen Ausgangsport, so handelt es sich um eine Senke. Dann ruft die InformDown Methode, mit der resultierenden AFS der Inform Methode der Komponente, die Reserve Methode des Links jedes Eingangsports auf. Die resultierenden AFS sendet sie, zusammen mit dem Opcode der ReserveUp Methode, an die Protokollthreads der stromaufwärts liegenden Komponenten, die mit dem jeweiligen Eingangsport verbunden sind. Danach kann das Manager Objekt die nächste Nachricht bearbeiten.

Erhält der Manager eine Nachricht mit dem Opcode der ReserveUp Methode, so wird diese von ihm aufgerufen. Die ReserveUp Methode ruft zuerst die Reserve Methode des Links auf, für den die AFS der Nachricht bestimmt ist. Besitzt die zugehörige Komponente mehrere Verbindungen am Ausgangsport und sind nicht alle benötigten AFS vorhanden, so wird das Ergebnis gespeichert und die Bearbeitung abgebrochen. Sind alle benötigten AFS vorhanden, so wird die Reserve Methode der Komponente aufgerufen. Handelt es sich um einen variablen Filter, Mixer oder eine Quelle, so sendet die Komponente beim Aufruf ihrer Reserve Methode die entsprechenden Daten selbstständig an den Gleichungslöser. Mit der, aus dem Aufruf der Reserve Methode der Komponenten resultierenden AFS, übergibt die ReserveUp Methode den Reserve Methoden der Links der Eingangsports. Jeder Link gibt als Ergebnis, für die Verbindung des Eingangsports, eine AFS zurück. Diese AFS werden von der ReserveUp Methode zusammen mit dem Opcode der ReserveUp Methode an die Protokollthreads der stromaufwärts liegenden Komponenten versendet. Besitzt die Komponente jedoch keinen Eingangsport, so handelt es sich um eine Quelle und die ReserveUp Methode sendet für die Quelle eine (Miß-) Erfolgsmeldung an den Application-Handler. Danach kann das Manager Objekt die nächste Nachricht bearbeiten.

Nachdem der Application-Handler von allen Quellen die Erfolgsmeldung erhalten hat, sendet er die Nachricht an den Gleichungslöser mit den Berechnungen zu beginnen und wartet auf eine Antwort. Der Gleichungslöser beginnt darauf hin mit der Berechnung einer Lösung für die Quellen und variablen Filter. Hat er eine Lösung ermittelt, sendet er die jeweils errechnete QoS an die entsprechende Komponente und sendet danach die Erfolgsmeldung an den Applikation-Handler. Wenn der Gleichungslöser denn Erfolg der Ermittlung der Lösung gemeldet hat, sendet der Applikation-Handler für jede Quellenkomponente, eine Nachricht an den Protokollthread auf dem Rechner, auf dem sich die Komponente befindet mit der Identifikation der Komponente und den Opcode für die Fortführung des XNRP Protokolls. Das Manager Objekt nimmt die Nachricht entgegen und ruft die zum Opcode gehörende XNRPCContinue Methode auf. Danach sendet der Application-Handler für jede Senke eine Anfrage nach dem Ergebnis an die Protokollthreads und wartet auf eine Antwort, während das XNRP Protokoll weiter abläuft.

Die XNRPCContinue Methode ruft zuerst die Relax Methode der Quelle auf. Als Parameter übergibt sie eine leere AFS (d.h. ohne QoS-Angaben) und erhält als Ergebnis eine AFS mit der QoS-Angabe der Quelle zurück, die diese vom Gleichungslöser erhalten hat. Mit dieser AFS ruft sie dann die Relax Methode des Links für den Ausgangsport der Quelle auf. Der Link gibt als Ergebnis, für jede Verbindung des Ausgangsports, eine AFS zurück. Diese AFS werden von XNRPCContinue zusammen mit dem Opcode der RelaxDown Methode an die Protokollthreads der

stromabwärts liegenden Komponenten versendet. Danach kann das Manager Objekt die nächste Nachricht bearbeiten. Diese Nachricht kann eine Continue Nachricht für eine weitere Quelle auf diesem Rechner oder eine Nachricht von einem Protokollthread sein.

Erhält der Manager eine Nachricht mit dem Opcode der RelaxDown Methode, so wird diese von ihm aufgerufen. Die RelaxDown Methode ruft zuerst die Relax Methode des Links auf, für den die AFS der Nachricht bestimmt ist. Besitzt die zugehörige Komponente mehrere Eingangsports und sind nicht alle benötigten AFS vorhanden, so wird das Ergebniss gespeichert und die Bearbeitung abgebrochen. Sind alle benötigten AFS vorhanden, so wird die Relax Methode der Komponente aufgerufen und mit der resultierenden AFS wiederum, die Relax Methode des Links des Ausgangsports. Der Link gibt als Ergebniss, für jede Verbindung des Ausgangsports, eine AFS zurück. Diese AFS werden von der RelaxDown Methode zusammen mit dem Opcode der RelaxDown Methode an die Protokollthreads der stromabwärts liegenden Komponenten versendet. Besitzt die Komponente jedoch keinen Ausgangsport, so handelt es sich um eine Senke und die RelaxDown Methode sendet für die Senke eine (Miß-) Erfolgsmeldung an den Application-Handler. Danach kann das Manager Objekt die nächste Nachricht bearbeiten.

Der Mißerfolg wird, entsprechend dem NRP Protokoll, nicht anhand medienspezifischer Parameter der QoS festgestellt, da die Protokollthreads Stromtyp unabhängig sind, sondern das Scheitern des Protokolls wird durch das Fehlen der QoS-Daten in der AFS von den Protokollthreads festgestellt. Ein Mißerfolg kann an jeder Stelle des Geflechts eintreten und gemeldet werden. Bei XNRP kann ein Mißerfolg zusätzlich durch den Gleichungslöser gemeldet werden. Meldet der Gleichungslöser jedoch einen Erfolg der Berechnungen, so kann ein Scheitern der Aushandlung in der dritten Phase unter normalen Umständen nicht mehr eintreten.

Nachdem der Application-Handler von allen Senken die Erfolgsmeldung erhalten hat, fährt er mit dem Start des Geflechts fort. Wird ein Mißerfolg gemeldet, so sendet er für alle Komponenten und Links eine entsprechende Nachricht an die Protokollthreads, bereits reservierte Ressourcen freizugeben. Die Protokollthreads rufen dazu die Free Methoden der Komponenten und Links auf.

## 6. Messungen

In diesem Kapitel wird zunächst das Meßschema nach dem die Messungen ausgewählt wurden und die Systemumgebung beschrieben. Danach werden die jeweiligen Szenarien und die Ergebnisse der Messungen dargestellt. Am Ende erfolgt eine Zusammenfassung und Bewertung der Messungen.

### 6.1 Meßschema

Die Aufgabe der Messungen ist es, die Eigenschaften der Protokolle und der Implementierung aufzuzeigen. Dazu sind die Protokoll- und Implementierungsbedingten Engpässe und die Skalierbarkeit der Protokolle und der Implementierung festzustellen, d.h. die Verwendbarkeit für Geflechte verschiedener Größenordnungen. Diese Angaben sind über Messungen des Zeitbedarfs einzelner Schritte und des gesamten Protokollablaufs möglich. Die Reihenfolge der Messungen erfolgt weitgehend „Bottom up“ von den Messungen einzelner Teil bis zu den Messungen der Geflechte. Zunächst werden Meßwerte für spezielle Teile der Implementierung ermittelt, die unabhängig von der Struktur des Geflechts sind. Diese Teile sind:

- Der Zeitbedarf für den Transport einer Nachricht zwischen zwei Threads des Cinema-Systems.
- Die Größe einer Nachricht, mit einer minimalen und mit einer maximalen Menge an QoS Daten.
- Der Zeitbedarf des Ressourcenmanger des Transportsystems. (andere Ressourcenmanager sind noch nicht integriert)
- Der Zeitbedarf für die Initialisierung der Komponenten. (Die Initialisierung wird von der Relax Methode durchgeführt.)
- Der Zeitbedarf für ein minimales Geflecht, Quelle-Senke.

Danach wird der Einfluß typischer Merkmale des Geflechts und der Komponenten auf den Zeitbedarf des Protokollablaufs gemessen. Diese Merkmale sind:

- Die Anzahl der Links bei einer Multicastverbindung.
- Die Anzahl der Eingangsports eines Mixers.
- Der Umfang der QoS Daten in der AFS.

- Die Anzahl der Nachrichten an den Gleichungslöser.

Für die Feststellung der Skalierbarkeit werden Messungen über lineare Ketten von Filtern und Mixern und Geflechte mit parallelen Strukturen durchgeführt.

## 6.2 Die Systemumgebung

Für die Messungen stehen zwei Workstation zu Verfügung:

- eine Sun SparcStation 20
- eine Sun SparcStation 10

Die Rechner sind mit dem Betriebssystem Sun Solaris 2.4 ausgestattet und über Ethernet verbunden.

Reale Reservierungen der Rechnerressourcen sind nicht möglich. Jedoch ist eine simulierte Reservierung in den Komponenten realisiert.

Reservierungen des Transportsystems erfolgen über RSVP [13].

## 6.3 Durchführung der Messungen

Die Messungen erfolgen mit den oben genannten Komponenten. Die Komponente VideoFileSource wird als Quelle verwendet, VideoFilter als variabler Filter, VideoMixer als Mixer und VideoWindow als Senke. Bei Mixern mit mehr als zwei Eingängen wird eine entsprechende Version des XMixer verwendet. Die Platzierung der Komponenten auf den Rechnern wird so gewählt, daß zwei Komponenten die durch einen Link verbunden sind auf verschiedenen Rechnern liegen. Abweichende Konfigurationen werden angegeben.

Da Rechner und Netzwerk auch von anderen Prozessen benutzt werden, wurden die Werte jeder Messung in einem möglichst kurzen Zeitraum ermittelt. Schwankungen zwischen den einzelnen Messungen konnten jedoch nicht in allen Fällen vermieden werden.

Die Messungen werden in der Form angegeben:

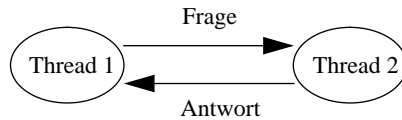
1. Das Thema der Messung.
2. Das Szenario für die Messung, wenn ein spezielles Szenario verwendet wurde.
3. Welche Werte ermittelt wurden und wie.
4. Die Meßwerte.

Eine Einschränkung der Szenarien stellt die Kommunikation in Cinema dar. Der Nachrichtentransport in Cinema ist durch UDP realisiert und daher nicht zuverlässig. (Eine Umstellung auf TCP/IP ist geplant.) Messungen mit mehr als ca. 20 Komponenten waren nicht durchführbar, da durch die Anzahl der Fehlübertragungen keine zuverlässigen Aussagen möglich waren.

### 6.3.1 Messungen einzelner Teile der Implementierung

Bei diesen Messungen werden einzelne Teile der Implementierung der Protokolle und des Cinema-Systems erfaßt, von denen die Dauer des Protokollablaufs abhängt, die jedoch von der Struktur der Geflechte unabhängig sind.

#### Messung 1: Der Zeitbedarf für den Transport einer AFS zwischen zwei Threads, mit den Kommunikations Methoden des Cinema-Systems.



Szenario für die Messung des Zeitbedarfs für die Kommunikation

Abbildung 6.1

Gemessen wird der Zeitraum vom Senden einer Frage bis zum Empfang der Antwort (hin und zurück). Die Zeitangaben in der Tabelle sind durch zwei dividiert. Für die lokale Messung befinden sich die Threads auf dem selben Rechner.

#### Messung 1: Zeitbedarf für den Transport einer Nachricht zwischen zwei Threads

	Minimalwert in msec.	Maximalwert in msec.	arithmetisches Mittel in msec.
lokal	9	10	10
entfernt	2	100	10

#### Messung 1: Zeitbedarf für die Kommunikation zwischen Applikation-Handler und Gleichungslöser

Minimalwert in msec.	Maximalwert in msec.
<1	<1

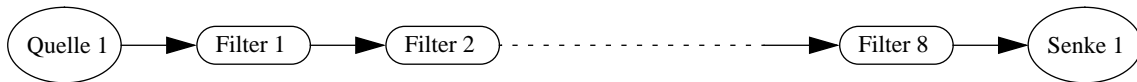
#### Messung 2: Die Größe der Nachrichten zwischen den Protokollthreads, gemessen an den Sendemethoden.

Gemessen wird die Größe der abgesendeten Nachricht.

#### Messung 2: Die Größe der Nachrichten

minimale Größe in Byte	maximale Größe in Byte
84	124

**Messung 3: Der Zeitbedarf für den Ressourcenmanger des Transportsystems (andere Ressourcenmanager sind noch nicht integriert)**



Szenario für die Messung des Zeitbedarfs für den Ressourcenmanger des Transportsystems

Abbildung 6.2

Gemessen wird die Gesamtzeit des Protokolls für eine Sequenz aus 8 Filtern, einer Quelle und einer Senke, mit und ohne Aufrufe des Ressourcenmanager. Aus der Differenz der Messungen und der Anzahl der Links wird die Zeit für einen Link berechnet.

**Messung 3: Der Zeitbedarf des Ressourcenmanger des Transportsystems**

Ressourcen- manager	NRP, Zeit in msec.	XNRP, Zeit in msec.
mit	546	778
ohne	307	468
pro Link	34	44

**Messung 4: Der Zeitbedarf der Initialisierungsmethoden der Komponenten. Wird direkt an den Aufrufen der Initialisierungsmethoden in den Relax Methoden der Komponenten gemessen.**

Gemessen wird die Zeit für die Ausführung der Initialisierung.

**Messung 4: Der Zeitbedarf für die Initialisierung der Komponenten**

Komponente	Initialisierung, Zeit in msec.
Quelle	11
Filter	2
Mixer	< 1
Senke	140

**Messung 5: Die Dauer des Protokollablaufs für ein minimales Geflecht Quelle-Senke.**



Szenario für die Messung des Zeitbedarfs für ein minimales Geflecht, Quelle-Senke

Abbildung 6.3

Gemessen wird die Gesamtzeit des Protokollablaufs. Für die lokale Messung befinden sich die Komponenten auf dem selben Rechner.

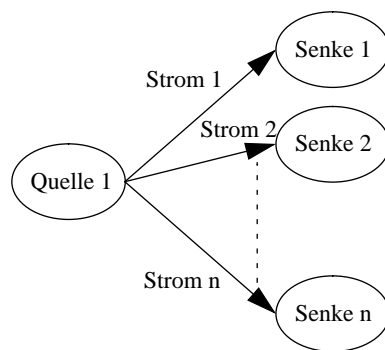
**Messung 5: Ein minimales Geflecht, Quelle-Senke**

	NRP, Zeit in msec.	XNRP, Zeit in msec.
lokal	187	212
entfernt	196	224

**6.3.2 Die Abhängigkeit des Protokollablaufs von typischen Merkmalen eines Geflechts**

Bei diesen Messungen wird die Einflüsse typischer Merkmale eines Geflechts auf die Dauer des Protokollablaufs erfaßt.

**Messung 6: Der Zeitbedarf des Links bei einer Multicastverbindung.**



Szenario für die Messung des Zeibedarfs des Links einer Multicastverbindung

Abbildung 6.4

Gemessen wird der Zeitbedarf der Linkmethode für die Berechnung der AFS für den Ausgangsport der Quelle, aus den AFS der Senken.

**Messung 6: Der Zeitbedarf des Links bei Multicastverbindungen**

Anzahl der Verbindungen	Link, Zeit in msec.
1	<1
2	<1
4	<1
8	0.3
16	0.5

**Messung 7: Der Zeitbedarf für Mixer mit einer unterschiedlichen Anzahl von Eingangsports.**



Szenario für die Messung des Zeitbedarfs für Mixer mit einer unterschiedlichen Anzahl von Eingangsports

Abbildung 6.5

Gemessen wird die Gesamtzeit des Protokollablaufs, abzüglich des Zeitaufwands für die Multicastverbindung und die Quelle-Senke. Aus der Entwicklung der Messung wird der Anstieg der Zeit je Port berechnet.

**Messung 7: Der Zeitbedarf für Mixer mit einer unterschiedlichen Anzahl von Eingangsports**

Ressourcenmanager	Anzahl der Eingangsports	NRP, Zeit in msec.	XNRP, Zeit in msec.	NRP, Anstieg je Port in msec.	XNRP, Anstieg je Port in msec.
mit	2	278	341	-	-
mit	4	350	424	36	41
mit	8	535	679	42	56
mit	16	-	-	-	-
ohne	2	221	277	-	-
ohne	4	294	350	36	36
ohne	8	418	470	32	32
ohne	16	661	733	31	32

Bei der Messung des Mixers mit 16 Eingangsports und der Verwendung des Ressourcenmanagers kam es zu einem Mißerfolg der Reservierung, da durch die 16 Links die Kapazität des Netzwerks überschritten wurde.

**Messung 8: Der Zeitbedarf für den Protokollablauf bei verschiedenen Größen der AFS für eine Sequenz aus Filter.**



Szenario für die Messung des Zeitbedarfs für den Ressourcenmanager des Transportsystems

Abbildung 6.6

Gemessen wird die Gesamtzeit für den Protokollablauf.

**Messung 8: Der Zeitbedarf bei verschiedener Größe der AFS**

Größe der Nachrichten in Byte	NRP, Zeit in msec.	XNRP, Zeit in msec.
84	495	670
124	504	716

**Messung 9: Die Änderung des Zeitbedarfs bei einer unterschiedlichen Anzahl von Nachrichten an/von dem Gleichungslöser.**

Die Messungen erfolgen im Rahmen anderer Messungen mit verschiedenen Szenarien.

Gemessen wird die Dauer der Berechnungen.

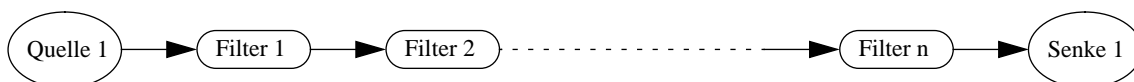
**Messung 9: Der Zeitbedarf der Berechnungen des Gleichungslöser**

Anzahl der Mixer	Anzahl der Filter und Quellen	Dauer der Berechnung in msec.
0	1	<1
8	1	<1
0	8	1.8
3	11	2

**6.3.3 Skalierbarkeit der Protokolle**

In diesen Messungen wird die Änderung des Zeitbedarfs für den Protokollablauf bei einer steigenden Anzahl von Komponenten in einer Sequentiellen Anordnung erfaßt.

**Messung 10: Der Zeitbedarf für eine Sequenz von Filtern.**



Szenario für die Messung des Zeitbedarfs einer Sequenz von Filtern

Abbildung 6.7

Gemessen wird die Gesamtzeit des Protokollablaufs. Aus der Entwicklung der Messung wird der Anstieg der Zeit je Filter berechnet.

**Messung 10: Der Zeitbedarf für eine Sequenz von Filtern**

Anzahl der Filter	NRP, Zeit in msec.	XNRP, Zeit in msec.	NRP, Anstieg je Filter in msec.	XNRP, Anstieg je Filter in msec.
1	225	270	-	-
2	260	317	35	47
4	322	418	32	49
8	504	716	39	63
16	841	1255	41	65

**Messung 11: Der Zeitbedarf für eine Sequenz von Mixern.**



Szenario für die Messung des Zeitbedarfs einer Sequenz von Mixern

Abbildung 6.8

Gemessen wird die Gesamtzeit des Protokollablaufs. Aus der Entwicklung der Messung wird die Zeit für einen Mixer berechnet.

**Messung 11: Der Zeitbedarf für eine Sequenz von Mixern**

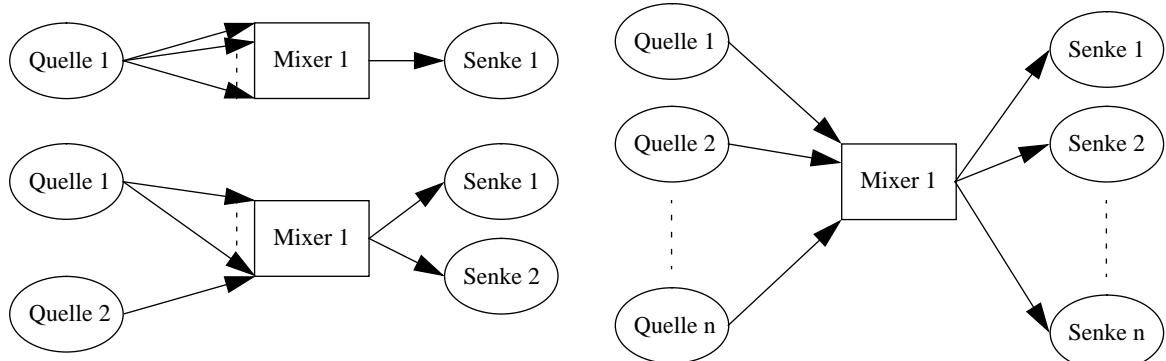
Anzahl der Mixer	NRP, Zeit in msec.	XNRP, Zeit in msec.	XNRP, Anstieg je Mixer in msec.	XNRP, Anstieg je Mixer in msec.
1	277	288	-	-
2	363	391	86	103
4	479	594	67	102
8	-	1052	-	109
16	-	-	-	-

Die Messung mit 8 Mixern und NRP und die Messung mit 16 Mixern war, wegen dem massiven Auftreten von Fehlern bei der Nachrichtenübertragung, nicht möglich.

### 6.3.4 Änderung des Zeitbedarfs bei parallelen Komponenten

In diesen Messungen wird die Änderung des Zeitbedarfs für den Protokollablauf in Abhängigkeit der Anzahl an parallelen Komponenten erfaßt.

#### Messung 12: Die Änderung des Zeitbedarfs bei parallelen Komponenten.



Messung des Zeitbedarfs für komplexe Geflechte

Abbildung 6.9

Gemessen Gesamtzeit bei einer verschiedenen Anzahl von Quellen, abzüglich Multicast und Mixer. Aus der Entwicklung der Messung wird der Anstieg der Zeit je Quelle-Senke Erweiterung berechnet

#### Messung 12: Der Zeitbedarf bei parallelen Komponenten

Anzahl der Quellen und Senken	NRP, Zeit in msec.	XNRP, Zeit in msec.	NRP, Anstieg je Quelle-Senke Erweiterung in msec.	XNRP, Anstieg je Quelle-Senke Erweiterung in msec.
1	534	671	-	-
2	688	844	154	173
4	949	1180	138	169
8	1614	1915	154	177

## 6.4 Zusammenfassung und Analyse der Meßergebnisse

In Messung 5 wurden für einen Protokollablauf, bei einem minimalen Geflecht aus einer Quelle und einer Senke, für NRP **196** msec. und für XNRP **224** msec. gemessen. Diese Zeit setzt sich aus mehreren Teilen zusammen:

- Die Initialisierung der beiden Komponenten benötigt **150** msec. (Messung 4). Bei dem VideoWindow dauert das Öffnen und Initialisieren eines Fensters 140 msec. Bei der VideoFileSource benötigen die Vorbereitungen für das Einlesen der Bilder 10 msec.
- Das Transportsystem verzögert den Protokollablauf bei NRP um **35** msec. und bei XNRP um **45** msec. (Messung 3). Diese Verzögerung ist die Summe der Aufrufe der

Reserve und Relax Methoden an den Links (Kapitel 3). Der Unterschied zwischen den Protokollen liegt dabei in der Reihenfolge der Aufrufe an den beiden Links einer Verbindung. Bei NRP wird Reserve zuerst auf der Seite des Outports aufgerufen, bei XNRP auf der Seite des Eingangsports. Die interne Verwaltung einer Reservierung durch RSVP führt bei XNRP zu einer Verzögerung gegenüber NRP.

- Für den Transport der AFS zwischen den Komponenten werden nochmals **30 msec.** benötigt (Messung 1). Dabei wird die AFS dreimal zwischen den Komponenten transportiert. Die Übertragung einer Nachricht benötigt ca. 10 msec. Unabhängig davon, ob der Transport nur innerhalb des Rechners oder über das Netzwerk erfolgt. (Messung 1) Daß diese Verzögerung durch die Verwaltung der Nachrichten in Cinema (Kapitel 4) entsteht, läßt sich aus der Tatsache schließen, daß die Verzögerungen bei lokaler und entfernter Kommunikation fast identisch sind. Eine Übertragung vergleichbarer Daten, mit einer TCP/IP Verbindung eines Testprogramms, benötigte weniger als 1 msec.
- Bei XNRP kommen ca. **20 msec.** durch die Unterbrechung für den Gleichungslöser hinzu. Die Berechnungsdauer (Messung 9) und der Nachrichtenaustausch zwischen Gleichungslöser und Applikation-Handler (Messung 1) spielen keine Rolle. Daß die Kommunikation zwischen dem Applikation-Handler und dem Gleichungslöser ungleich schneller ist als die Kommunikation zwischen den Protokollthreads, läßt sich mit dem verwendeten Adressobjekt erklären. Da XNRP immer den lokalen Gleichungslöser benutzt, erfolgt die Kommunikation über Adressobjekte des Systems für die lokalen Prozesse. Für diese Adressobjekte erfolgt der Transport einer Nachricht schneller.

Bei der Addition der Einzelwerte erhält man eine Gesamtdauer von **210 msec.** für NRP und **245 msec.** für XNRP. Der Unterschied ergibt sich aus den prinzipiellen Schwankungen der verwendeten Multiuser/Multitasking Systemumgebung und der Komplexität des zu messenden Systems. Besonders starke Schwankungen traten bei der Nachrichtenübertragung und bei der Initialisierung der Komponenten auf. Hier wurden Verzögerungen bis zu einer Sekunde gemessen.

In Messung 7 wurde für einen Eingangsport an einem Mixer eine Verzögerung von ca. 35 msec. gemessen. Bei einem Mixer mit zwei Eingangsports wird diese Verzögerung durch die Reservierung an den Links verursacht. Bei einer wachsenden Zahl von Eingangsports kann eine Verringerung dieser Verzögerung beobachtet werden, jedoch steigt gleichzeitig der Zeitbedarf für die Verwaltung der Ports.

In Messung 10 wurden, als Zeitbedarf der Protokolle für einen VideoFilter und einen Link, 35 msec. bei NRP und 45 msec. bei XNRP gemessen. Die Werte steigen bei 16 Filter auf 40 msec. für NRP und 65 msec. für XNRP an. Durch die Begrenzung auf ca. 20 Komponenten konnten keine Messungen mit größeren Geflechten durchgeführt werden. Jedoch zeigt sich bereits bei 16 Filtern ein Anstieg des Zeitbedarfs. Der Grund hierfür liegt in der stärkeren Auslastung der beiden Rechner durch die steigende Zahl der Komponenten und die resultierenden Schwankungen in den Ausführungszeiten der Prozesse. Die Ursache für den stärkeren Anstieg bei XNRP ist die steigende Anzahl an Nachrichten an den Gleichungslöser.

In Messung 11 wurde ein Zeitbedarf für einen VideoMixer und zwei Links von 80 msec. bei NRP und 100 msec. bei XNRP gemessen. Die steigende Zahl von Nachrichtenverlusten machte die Messung einer großen Anzahl von Mixern unmöglich.

In Messung 12 wurden ein ansteigen des Zeitbedarfs beim Hinzufügen einer Quelle und einer

Senke um ca. 150 msec. bei NRP und um ca. 170 msec. bei XNRP gemessen. Dieser Wert wird durch den zusätzlichen Link und die Initialisierung der Komponenten verursacht, da sich alle Quellen und Senken auf einem Rechner befinden und sequentiell bearbeitet werden.

Die Größe der Nachricht, die Anzahl der Verbindungen bei einem Multicast und die Berechnungen der Gleichungslöser haben keinen nennenswerten Einfluß auf den Protokollablauf.

Die Verwendung der Cinema internen Kommunikationsmethoden ist für die Integration der Protokolle ein Vorteil. Die Unzuverlässigkeit dieser Methoden verursacht jedoch eine starke Einschränkung der Geflechte. Größere Geflechte könnten durch eine Verteilung der Komponenten auf mehr Rechner zwar realisiert werden. Diese Lösung war für die Messungen jedoch nicht realisierbar. Die geringe Geschwindigkeit der Nachrichtenübertragung ist ebenfalls ein Problem, wegen der großen Anzahl von Nachrichten während des Protokollablaufs. Die Messungen haben gezeigt, daß einzelne Verzögerungen im Millisekunden Bereich, sich während einer Aushandlung schnell summieren können. Auch der Zeitbedarf des RSVP ist nicht unerheblich. Während das Versenden der Nachrichten asynchron verläuft und andere Protokollabläufe nicht blockiert, werden die Protokollthreads durch die Links für eine nicht unerhebliche Zeit blockiert. Die Methoden der Komponenten benötigen dagegen wenig Zeit. Lediglich ein Mixer mit vielen Eingängen erzeugt eine spürbare Verzögerung. Die Initialisierung der VideoWindow Komponente benötigt zwar eine lange Zeit, jedoch kann man diese als konstant ansehen. Sie ist nicht von der Größe des Geflechts abhängig. Wenn nicht mehrere VideoWindow Komponenten auf einem System gleichzeitig initialisiert werden sollen, ist der Zeitbedarf der Protokolle sogar unabhängig von der Anzahl der VideoWindow Komponenten im Geflecht.

Ein weiteres Problem ist das Fehlen der Ressourcenmanager für den Speicher und die CPU. Bei ihrer Integration ist eine weitere Verlangsamung der Protokolle zu erwarten.

## **7. Bewertung der Protokolle und der Implementierung**

Die Arbeiten zu diesem Thema stehen noch am Anfang und es für diese Arbeit war keine mit NRP oder XNRP vergleichbare Lösung verfügbar. Aus diesem Grund werden die theoretischen Grundlagen hier genauer betrachtet und verwandte Arbeiten vorgestellt um eine Basis für die Bewertung der Protokolle und der Implementierung zu schaffen.

### **7.1 Definition der Anforderungen an die Protokolle**

Die Aufgabe der Protokolle ist die Aushandlung der QoS und die Reservierung von Ressourcen in einem verteilten Multimedia-System. Dazu lassen sich verschiedene Anforderungen formulieren:

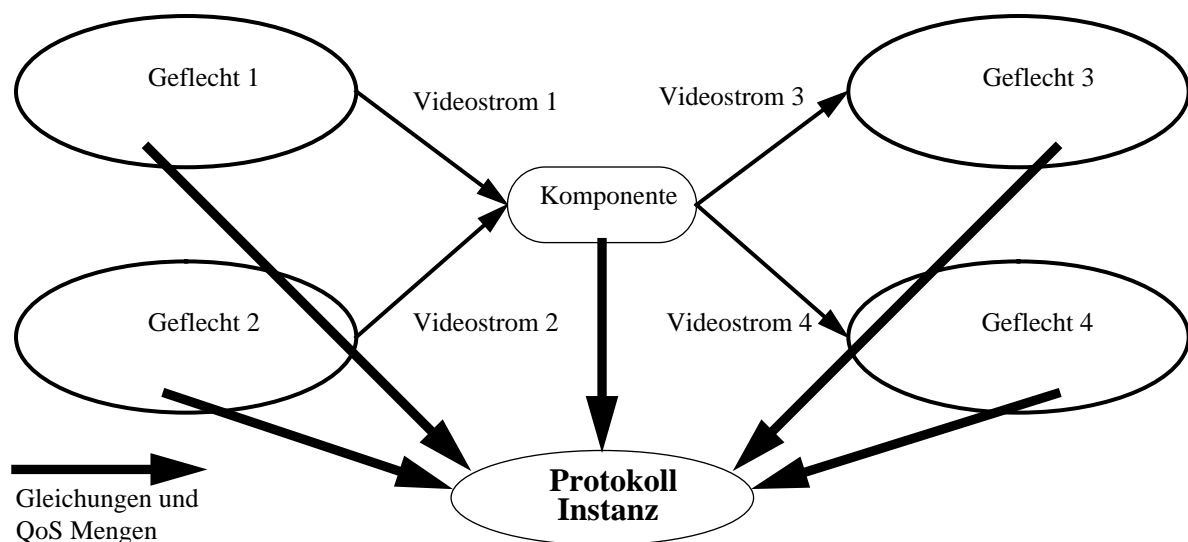
- Die maximale Dauer der Aushandlung soll beschränkt sein.  
Dies ist wünschenswert um auch bei einem Scheitern der Aushandlung in jedem Fall eine Fehlermeldung zu erhalten. Auf diese Fehlermeldung kann dann mit einer Korrektur der gewünschten QoS oder z.B. mit einer Erhöhung der Priorität reagiert werden.
- Der Zeitaufwand für das Protokoll soll gering sein.  
Der Start eines verteilten Multimedia-Systems muß nicht in jedem Fall interaktiv mit dem Benutzer erfolgen (Ein Videokonferenzsystem kann bereits längere Zeit vor Beginn der eigentlichen Konferenz gestartet werden). Bei einem interaktiver Einsatz ist jedoch eine kurze Startzeit erforderlich. Zwischen der Eingabe eines Benutzers und der sichtbaren Reaktion einer Applikation sollten nicht mehr als 2 bis 3 Sekunden liegen.
- Die maximal mögliche QoS soll ermittelt werden.  
Wenn die gewünschte QoS nicht möglich ist, dann soll die maximale QoS des Systems verwendet werden, wenn diese in dem vorgegebenen Rahmen liegt.
- Die Reservierung von Ressourcen soll minimal sein.  
Es sollen nur Ressourcen reserviert werden, die auch wirklich benötigt werden.
- Die Struktur des Protokolls soll verteilt sein.  
Als Bestandteil eines verteilten Systems sollte das Protokoll dem Modell verteilter Systeme entsprechen. Diesen Punkt kann man einschränken, da das Protokoll nicht direkt zur Funktionalität des Systems gehört. (siehe XNRP Kapitel 3.2)

- Eine hohe Flexibilität bezüglich der unterstützten Geflechte.  
Eine „sinnvolle“ Beschränkung ist jedoch möglich. (siehe NRP Kapitel 3.1)
- Eine hohe Flexibilität bezüglich der unterstützten Stromtypen. (Audio, Video, Text, usw.)  
Die Unterstützung neuer Stromtypen soll möglich sein. (z.B. Komprimierungsverfahren)
- Eine hohe Flexibilität bezüglich der unterstützten Komponententypen  
Bei neuen Stromtypen sind auch neue Komponenten erforderlich.

## 7.2 Lösungskonzepte der Protokolle

### 7.2.1 Zentrale Instanz

Eine Lösung ist der Transport aller Parameter an ein zentrale Instanz. Diese Instanz kann eine QoS und die benötigten Ressourcen für jedes Element des Systems, aus der Relationen, Format- und Ressourcenbeschränkungen ermitteln.



Konzentration aller Informationen an einer zentralen Instanz.

Abbildung 7.1

Ein Nachteil dieses Konzeptes ist die große Anzahl an Nachrichten und Daten die sich an einen Punkt konzentrieren. Es wird eine Relation für je zwei Links benötigt, um die Zusammenhänge im gesamten System zu beschreiben. Für jede Komponente wird die Angabe der Formatbeschränkungen und der Ressourcen des Rechners benötigt. Für die Links werden Angaben der Ressourcen des Netzwerks benötigt.

Ein weiterer Kritikpunkt sind die prinzipiellen Probleme einer zentralen Lösung [11]. Die zentrale Instanz stellt einen „Engpaß“ und einen „Single Point of Failure“ dar. Alle Komponenten des Systems müssen Nachrichten an sie versenden und eine Antwort erhalten. Die Anzahl der Komponenten und die Geschwindigkeit des Protokolls kann durch die Anzahl der Nachrichten begrenzt werden, die das Netzwerk an dieser Stelle erlaubt. Bei einem Ausfall des Rechners, der Netzwerkverbindung oder des Programms selbst, ist das gesamte System inoperabel.

Zusätzlich widerspricht die dezentrale Organisation eines verteilten Multimedia-Systems. Bei einer Integration in ein verteiltes Multimedia-System sind Probleme mit der Schnittstelle zwischen System und Protokoll zu erwarten.

Der Vorteil dieser Lösung ist, daß alle Informationen über das System zur Verfügung stehen. Es können problemlos Randbedingungen integriert werden, wie z.B. die Aufteilung der Ressourcen zwischen Komponenten auf einem Rechner oder Links die einen Teil des Netzwerks gemeinsam nutzen.

### **7.2.2 Verteilter Ansatz**

Ein verteilter Ansatz beruht auf einer möglichst autonomen (und parallelen) Ermittlung der QoS, für jedes Element des Systems. Ein einzelnes Element (Komponente oder Link) soll keine Informationen über das gesamte System besitzen bzw. benötigen. Im Idealfall werden nur Informationen über die direkten Nachbarn verwendet. Dieser Idealfall wird hier als Protokollansatz vorgestellt.

Aufgrund der Beziehungen in einem verteilten Multimedia-System kann die QoS eines Elements des Systems von der QoS eines entfernten Elements beeinflusst werden. So muß sich eine Komponente die einen Datenstrom liefert auf die Möglichkeiten einer Komponente einstellen, die eventuell am anderen Ende des Systems diesen Strom verarbeitet. Die Komponenten müssen keine unmittelbaren Informationen übereinander besitzen, es genügt wenn die Informationen schrittweise, zwischen benachbarten Elementen durch das System propagiert werden. Diese Methode entspricht einem Broadcast [11], bei dem ein Element dem gesamten System seine Beschränkungen der QoS mitteilt. Da ein Element jedoch nur seine Nachbarn (mit denen es durch Links verbunden ist) kennt, erfolgt der Broadcast entlang der Links.

Der Zeitraum, um eine Lösung zu ermitteln, wird daher durch die kürzeste Verbindung zwischen den beiden am weitesten voneinander entfernten Komponenten bestimmt. Wobei sich die Entfernung auf die Anzahl der Links zwischen den Komponenten bezieht und Verbindungen zwischen Komponenten sind die Pfade entlang der Links. Die kürzeste Verbindung ist dann der Pfad mit den wenigsten Links.

Neben den prinzipiellen Vorteilen verteilter Systeme [11], wird die Integration durch die Übereinstimmung der Konzepte von Protokoll und System vereinfacht. Und besonders bei großen Systemen mit starker Vernetzung ist eine höhere Geschwindigkeit zu erwarten als bei einer zentralen Lösung.

Ein Kritikpunkt ist die Anzahl der Nachrichten, die größer als bei einer zentralen Lösung ist. Jedoch verteilen sich die Nachrichten über das Netzwerk.

Ein weiterer Nachteil ist die mangelnde Information über das Gesamtsystem. z.B. können zwei Komponenten auf dem gleichen Rechner existieren ohne sich zu „kennen“, wenn sie innerhalb des Systems „entfernt“ sind (d.h. sie sind nicht durch einen Link verbunden). Da die Komponenten keine Informationen übereinander besitzen, konkurrieren sie um die Ressourcen des Rechners und der „Verlierer“ bestimmt die QoS des Gesamtsystems.

## 7.3 Verwandte Arbeiten

Der QoS Broker, von K.Nahrstedt und J.Smith [10], ist ein Middleware System das die Ressourcenverwaltung (CPU, Speicher, Netzwerk, usw.) für alle Applikationen auf einem Rechner übernimmt und mit anderen Systemen über QoS verhandelt. Diese Struktur entspricht einem verteilten Ansatz. Die Aushandlung der QoS findet auf einer „Client-Server“ Basis statt, d.h. eine Empfänger-Applikation startet eine Anfrage mit der Übergabe eines QoS Rahmens an den lokalen QoS Broker. Der QoS Broker gibt die Anfrage an den Zielrechner mit der Sender-Applikation weiter. Die QoS Broker der beiden Rechner beginnen nun die QoS zu „verhandeln“. Dabei werden die verfügbare Ressourcen des Netzwerks, der beiden Rechner und auch Abhängigkeiten zwischen Applikationen auf den Rechnern in Betracht gezogen. Einigen sich die Broker auf eine QoS, reservieren sie die benötigten Ressourcen. Sonst wird der Empfänger von dem Mißerfolg informiert.

Die Stärke dieses Systems ist die genaue Kenntnis über den Zustand und die Historie der beteiligten Rechner und Programme. So kann eine Aushandlung auch dynamisch, während der Laufzeit einer Applikation, geändert werden. Dabei kann der QoS Broker Änderungen auf dem eigene Rechner einbeziehen. Wenn z.B. eine Video-Wiedergabe vom Benutzer vergrößert wird, kann der QoS Broker anderen Applikationen Ressourcen entziehen und dieser, im Augenblick für den Benutzer wichtigeren Applikation, zu Verfügung stellen. Zusätzlich kann er, wenn das Video über Netzwerk empfangen wird, mit dem QoS Broker des Senders über eine Erhöhung und mit Broker anderer Applikationen über eine Verminderung der QoS verhandeln.

Diese Lösung bietet eine hohe Flexibilität und Leistungsfähigkeit bezüglich der QoS Aushandlungen. Durch den Client-Server Ansatz ist der QoS Broker jedoch nur begrenzt für komplexe verteilte Systeme einsetzbar. Systeme mit einer Baum-Struktur lassen sich jedoch realisieren.

L.C.Wolf beschreibt in seiner Publikation „Resource Management for Distributed Multimedia Systems“ [9] ein Protokoll zur Aushandlung der QoS und Reservierung von Ressourcen mit einem zentralen Ansatz.

Dabei müssen die Ressourcen drei Methoden für den Protokollablauf zur Verfügung stellen:

- „Throughput test“: Prüft ob eine gewünschte Reservierung durchgeführt werden kann.
- „Resource reservation“: Führt eine Reservierung durch.
- „QoS computation“: Ermittelt die maximal mögliche QoS für einen Datenstrom.

Eine zentrale Instanz gibt QoS vor, die Schrittweise durch das System propagiert wird. Dabei wird an den Ressourcen ein „Throughput test“ durchgeführt.

- Ist eine Reservierung möglich wird „Resource reservation“ durchgeführt.
- Ist sie nicht möglich wird eine neue QoS mit „QoS computation“ ermittelt und der zentralen Instanz mitgeteilt. Alle reservierten Ressourcen werden freigegeben und das Protokoll beginnt mit der neuen QoS von neuem.

Ein Problem des Protokolls ist, daß die Dauer der Aushandlung nicht beschränkt ist und durch das vollständige Zurücksetzen bei einem Scheitern einer Reservierung der Zeitaufwand nicht

unerheblich ist. Auch kann durch die Freigabe der Ressourcen sich die Situation im System sich ständig ändern und eine Reservierung mit einer geringeren QoS an einer Stelle scheitern, an der früher bereits Ressourcen für eine höhere QoS reserviert waren.

## 7.4 Bewertung

### 7.4.1 Implementierung

Die Protokolle wurden unmittelbar in die Struktur des *Cinema*-Systems eingebettet und die Implementierung ist für den Benutzer weitgehend transparent. Existierende Applikationen können nach der Änderung der Komponenten weiter verwendet werden, müssen jedoch um eine Eingabemöglichkeit für die gewünschte QoS erweitert werden.

Subjektiv konnte eine starke Vereinfachung und Beschleunigung der Arbeit mit Cinema bemerkt werden. Für die Initialisierung mußte bisher an verschiedenen Komponenten Parameter vorgegeben werden. z.B. bei einem Videofilter die Bildhöhe und Bildbreite, die er an seinem Eingang zu erwarten hat und die er an seinem Ausgang liefern soll. Dabei mußten die Beziehungen der Links an Mixern und die Beschränkungen verschiedener Komponenten beachtet werden. Nach der Integration der Protokolle verringerte sich die notwendige Zeit und der Aufwand für den Aufbau einer Sitzung.

Ein Grundlegendes Problem der Implementierung ist die intensive Nutzung von Teilen des Systems die nicht speziell für die Verwendung durch die Protokolle implementiert wurden. Diese erzielen für sich betrachtet eine ausreichende Geschwindigkeit, verzögern in der Summe jedoch den Protokollablauf erheblich. Besonders davon betroffen ist die Kommunikation und die Reservierung von Ressourcen. Daraus läßt sich die Notwendigkeit ableiten bei einer Integration der Protokolle in ein existierendes System diese Teile zu überarbeiten oder ebenfalls zu erneuern.

Das zeitliche Verhalten der Protokollmethoden der hier verwendeten Komponenten hat sich als unkritisch erwiesen. Dieses Verhalten ist jedoch von der jeweiligen Realisierung durch den Programmierer der Komponente abhängig. Eine Komponente mit langsamen Methoden kann den gesamten Protokollablauf verzögern. Auch wird die Realisierung von Komponenten aufwendiger. Es sollten daher standardisierte Methoden vom System zur Verfügung gestellt werden.

Die Realisierung mit Protokollthreads und einer sequentiellen Bearbeitung hat sich als ausreichend gezeigt. Die Verzögerungen durch die Kommunikation spielen keine Rolle, da das Versenden der Nachrichten asynchron erfolgt. Die Verzögerungen durch die Links lassen sich durch Parallelisierung nicht verhindern, da RSVP keine parallele Durchführung von Reservierungen vorsieht. Bei einer Realisierung mit mehreren Protokollagenten müßten die Aufrufe der Linkmethoden durch Semaphore gesteuert werden. Ein Problem ist jedoch die Initialisierung der Komponenten in der Relax Phase. Komponenten mit einer zeitaufwendigen Initialisierung, wie die VideoWindow Komponente, können hier eine erhebliche Verzögerung verursachen. Eine einfache Lösung die Delegation dieser Aufgabe an den Constructor des cin-Prozesses. Dieser könnte vom Protokollthread eine entsprechende Nachricht erhalten und die Initialisierung durchführen während das Protokoll weiterläuft.

## 7.4.2 Protokolle

In der von der Implementierung vorgegebenen Grenze von 20 Komponenten zeigten die Protokolle ein unkritisches Verhalten. Zwar zeigte sich stellenweise ein stärkerer Anstieg des Zeitbedarfs bei den Messungen mit den größten möglichen Geflechten. Dieser war jedoch mit der steigenden Anzahl an Komponenten auf den beiden verwendeten Rechnern erklärbar. Bei einer Umstellung des Cinema-Systems auf eine zuverlässige Kommunikation kann aus den gemessenen Werten auf eine Grenze von 50 bis 100 Komponenten für die verwendete Systemumgebung geschlossen werden. Bei dieser Anzahl ist ein akzeptable Aushandlungsdauer erreichbar. Größere Geflechte wären nur mit einer entsprechend größeren Anzahl an Rechnern sinnvoll.

Grundsätzlich zeigte sich ein sehr geringer Ressourcenbedarf der Protokolle. Die Größe der Nachrichten ist selbst bei umfangreichen Parameterangaben auf einige hundert Byte beschränkt und eine Belastung der Rechner durch die Aushandlungen war nicht feststellbar. Bei einem System für die Übertragung und Bearbeitung von multimedialen Daten ist mit keinen Ressourcen-Engpässe durch den Protokollablauf zu rechnen.

Beide Protokolle besitzen eine hohe Flexibilität bezüglich der unterstützten Stromtypen. Die Unterstützung beliebiger Typen wie Audio, Video oder Text ist möglich. Bei der Einführung eines neu Stromtyps müssen lediglich die Linkmethoden ergänzt werden. Da Komponenten für einen neuen Stromtyp in jedem Fall neu entwickelt werden müssen, zählt die Implementierung der Protokollmethoden nicht zu der Erweiterung des Protokolls.

Beide Protokolle besitzen eine Beschränkung der Aushandlungsdauer, entsprechend der Größe des Geflechts. Diese Resultiert aus der Beschränkung auf drei Phasen. In den drei Phasen wird von beiden Protokollen die maximal mögliche QoS ermittelt und eine minimale Reservierung für diese QoS durchgeführt. Beide Protokolle bilden daher eine effektive Lösung.

NRP besitzt eine rein verteilte Ablaufstruktur und läßt sich problemlos in ein verteiltes Multimedia-System integrieren. Der Ablauf des Protokolls kommt der im Abschnitt 7.2.1 beschriebenen Lösung mittels eines Broadcast nahe. Bei Geflecht mit überwiegend paralleler Struktur kann die NRP Lösung sogar schneller sein. Aufgrund der guten Anpassung an die verteilte Struktur der Geflechte und die Beschränkung auf drei Phasen kann NRP als eine prinzipiell sehr schnelle Lösung angesehen werden. Aus dem verteilten Ablauf und der Beschränkung auf drei Phasen resultiert jedoch auch die Einschränkung der Geflechte. Für eine Verwendung von NRP muß jeweils geprüft werden ob die Geflechte sich, dem „zwei Zonen“ Model entsprechend, in einen Mixer- und einen Multicast-Teil aufteilen lassen. Durch die große Ähnlichkeit mit XNRP bietet sich eine parallele Implementierung der beiden Protokolle an, wie sie in dieser Arbeit durchgeführt wurde. So kann man zwischen der schnelleren NRP oder dem flexibleren XNRP wählen.

XNRP besitzt keine Einschränkung der Geflechte, dieser Vorteil wird jedoch durch die Einführung des Gleichungslöser erreicht. Durch das zentrale Konzept erschwert sich die Integration in ein verteiltes Multimedia-System. Jedoch ist XNRP keine rein zentrale Lösung, sondern eine Kombination der beiden Konzepte. Der Vorteil dieser Lösung ist eine Parallelisierung des Protokollablaufs und eine Verminderung der Kommunikationslast am Gleichungslöser. Die Kommunikation zwischen Komponenten und Gleichungslöser erfolgt parallel zum Ablauf der zweiten und dritten Phase. Zusätzlich werden durch die Aushandlungen in den ersten beiden Phasen, die Anzahl und der Umfang der erforderlichen Nachrichten an den Gleichungslöser re-

duziert. Die Messungen für NRP und XNRP zeigen nur einen geringen Unterschied, obwohl die Anzahl der benötigten Nachrichten bei XNRP höher ist.

Ein Kritikpunkt bei beiden Protokollen ist, daß sie keine Angaben über Verzögerungen beim Transport der Daten unterstützen. Dies wäre durchaus ein wichtiger Qualitätsfaktor z.B. bei Videokonferenzen. Ein noch größeres Problem ist jedoch das Fehlen einer Aufteilung von Ressourcen zwischen Komponenten auf demselben Rechner oder Links auf derselben Netzwerkverbindung. Bei NRP ergibt sich durch das verteilte Konzept ein prinzipielles Problem die Randbedingung in die Aushandlungen zu integrieren. XNRP besitzt mit dem Gleichungslöser zwar eine Instanz, die eine Aufteilung der Ressourcen durchführen könnte. Jedoch werden die Reservierungen bereits in der zweiten Phase durchgeführt, bevor der Gleichungslöser gestartet wird.

### **7.4.3 Zusammenfassung**

Subjektiv brachte die Integration der Protokolle eine große Vereinfachung und Erleichterung der Arbeit mit dem Cinema-System. Die Realisierung mit einem einzelnen Protokollthread je Cinema-System bzw. je Rechner ergab keine erheblichen Nachteile für den Protokollablauf. Bei den Messungen konnte jedoch eine starke Abhängigkeit der Protokolle vom Zeitbedarf der Kommunikation und der Ressourcenreservierung festgestellt werden. Besonders wichtig ist daher eine effiziente und zuverlässige Kommunikation, sowie schnell durchführbare Reservierungen. Bei der Integration der Protokolle in existierende Systeme ist eine Überarbeitung oder Erneuerung der entsprechenden Funktionen anzuraten. Auch die Abhängigkeit der Protokolle von der jeweiligen Implementierung der Protokollmethoden der Komponenten ist ein Schwachpunkt. Soweit möglich sollten standardisierte Methoden eingesetzt werden.

NRP besitzt einen rein verteilten Ablauf. Die Integration zeigte daher keine prinzipiellen Probleme. Bei NRP muß der Gleichungslöser alle Komponenten bekannt sein. Dies konnte durch die Platzierung des Gleichungslöser auf dem Rechner der das Protokoll startet erreicht werden. Der Ressourcenbedarf für die Aushandlung ist bei beiden Protokollen gering und von einem System für die Verarbeitung von multimedialen Daten leicht zu bewältigen. Beide Protokolle führen die Aushandlung der QoS und die Reservierung von Ressourcen effektiv aus. Über die Skalierbarkeit kann keine endgültige Aussage gemacht werden. Die Kommunikation in Cinema ließ keine Geflechte mit mehr als 20 Komponenten zu. Aus den Messungen läßt sich jedoch schließen, daß der Zeitbedarf der Reservierungen den minimalen Zeitbedarf der Protokolle am stärksten bestimmt. Die Größe der Geflechte ergibt sich daher aus der maximal tolerierten Zeit für den Protokollablauf. Dabei konnten Unterschiede zwischen den Anwendungen festgestellt werden. Beim starten einer Videoplayer Applikation in Cinema werden vom Benutzer maximal zwei Sekunden Verzögerung toleriert. Bei komplexeren Anwendungen wurden wesentlich längere Wartezeiten akzeptiert. Bei diesen Anwendungen mußten ohne die Protokolle umfangreiche Einstellungen manuell durchgeführt werden. Die Verwendung der Protokolle Vereinfachte und Beschleunigte den Start der Anwendung trotz der Wartezeit. Bei der Frage nach der Skalierbarkeit muß also auch die Art der Anwendung mit einbezogen werden. Mit den gemessenen Geschwindigkeiten ließen sich komplexe Anwendungen mit bis zu einhundert Komponenten realisieren. Bei einer Überarbeitung der Kommunikation und Reservierung ließe sich dieser Wert noch steigern.

Ein Kritikpunkt bei den Protokollen ist das Fehlen einer Lösung für die Aufteilung der Ressourcen zwischen Komponenten bzw. Links, wenn sich diese auf dem gleichen Rechner befinden bzw. die gleiche Netzwerkverbindung verwenden.

#### **7.4.4 Ausblick**

Beide Protokolle bieten bereits eine Vereinfachung der Arbeit mit Cinema. Eine Umstellung der Kommunikationsmethoden des Cinema-Systems auf einen Zuverlässigen Nachrichtentransport ist jedoch notwendig um die gesamte Leistungsfähigkeit der Implementierung feststellen und nutzen zu können. Eine Verwendung für Geflechte mit einigen hundert Komponenten wird jedoch nur nach einer Überarbeitung des gesamten Systems möglich sein.

Für die Protokolle NRP und XNRP wird am IPVR der Universität Stuttgart an Ergänzungen gearbeitet. Diese beinhalten Aushandlung des Delay, Aufteilung der Ressourcen und Automatische Platzierung von Komponenten. Bei der Aushandlung des Delay wird die Angabe der gewünschten QoS um die Angabe einer maximalen Verzögerung erweitert. Diese Verzögerung ist die Zeit die vom erzeugen der Daten an der Quelle bis zur Wiedergabe an der Senke vergehen darf. Das Protokoll muß durch entsprechende Aushandlung der QoS dafür sorgen, daß dieses Maximum nicht überschritten wird. Die Aufteilung der Ressourcen betrifft Komponenten bzw. Links eines Geflechts, die sich auf dem gleichen Rechner befinden bzw. die dieselben Netzwerkverbindungen nutzen. Das Protokoll teilt die Ressourcen zwischen den Beteiligten so auf, daß für das Geflecht die maximale QoS möglich ist. Die automatische Platzierung der Komponenten betrifft die Konfiguration des Geflechts. Das Protokoll verschiebt Komponenten zwischen den Rechnern, entsprechend der vorhandenen und der benötigten Ressourcen.

Wenn es im Augenblick auch keine Anwendungen gibt, für die NRP oder XNRP benötigt würden. So ist durch die schnelle Entwicklung im Bereich der Netzwerke und der großen Nachfrage nach multimedialen Anwendungen ein steigender Bedarf zu erwarten.

Eine weitere Verwendung der Implementierung in Cinema ist jedoch fraglich. Das Cinema-System hat bereits seine konzeptionellen Grenzen erreicht und eine Überarbeitung des Systems wäre notwendig.

# Anhang A      Installation

Aufgrund des Umfangs der Änderungen und Ergänzungen im Cinema-System (es handelt sich um mehrere tausend Zeilen Code) werden hier lediglich die notwendigsten Informationen angegeben.

Übersetzen von Cinema mit den Protokollen:

- Das Verzeichnis NRP muß sich im Cinema Verzeichnis befinden
- Die Kode in der Datei „NRP/src/cinema.neu“ muß an den in Abschnitt A.5 und A.6 beschriebenen Stellen eingefügt werden.
- Die entsprechenden Definitionen (A.3) müssen in die CONFIG.make Dateien der Verzeichnisse „CAE/etc/“ „CFG/etc/“ „CIF/etc/“ „CIN/etc/“ „CLIB/etc/“ „COM/etc/“ „COMPONENTS/etc/“ eingetragen werden.

Entfernen des Kodes:

Alle im Abschnitt A.5 beschriebenen Änderungen sind im Kode mit „\_NRP\_“ gekennzeichnet. Die Änderungen in Abschnitt A.6 sind für die Verwendung der Protokolle notwendig betreffen diese jedoch nicht direkt und müssen nicht entfernt werden.

Hinweis für die Verwendung:

In allen Komponenten müssen die Protokollmethoden definiert sein. Die Programmierschnittstelle ist in Kapitel 5 beschrieben. Umgebungsvariablen müssen gesetzt sein.

## A.1 Dateien

NRP\_Opcodes.H + NRP\_Types.H

Definitionen der Opcode für NRP und XNRP, Definitionen der Nachrichten, der AFS und der Darstellung der QoS Daten.

NRP\_Protocol.C + NRP\_Protocol.H

Das NRP Objekt.

XNRP\_Protocol.C + XNRP\_Protocol.H

Das XNRP Objekt.

CNP\_solver\_cfg.C CNP\_solver\_cfg.H

Der Gleichungslöser.

NRP\_Manager.C NRP\_Manager.H

Das Manager Objekt für die Verwaltung von NRP, XNRP und Gleichungslöser

CNP\_chain.C CNP\_chain.H

Die Listenverwaltung des Gleichungslöser, für die QoS Daten und die Relationen.

CNP\_eqneq.C + CNP\_eqneq.H  
 CNP\_eq.C + CNP\_eq.H  
 CNP\_neq.C + CNP\_neq.H  
 Objekte des Gleichungslöser für Relationen  
 CNP\_variable.C + CNP\_variable.H  
 Objekt des Gleichungslöser für QoS Daten.

## A.2 Umgebungsvariablen

„cdpath“ . . . . . zeigt auf das Verzeichnis mit den Deskription-Dateien!

## A.3 Definitionen für die Compilierung von Cinema mit NRP

„\_NRP\_“ . . . . . Um Protokoll Verwaltungsfunktionen einzubinden.  
 „\_INCLUDE\_NRP\_PROTOCOL\_“ und „\_INCLUDE\_XNRP\_PROTOCOL\_“  
 . . . . . Entsprechend dem gewünschten Protokoll definieren, es können  
 auch beide gleichzeitig definiert werden. (\_NRP\_ muß definiert  
 sein!).  
 „\_NetResMgr\_“ . . . . . Muß in NRP\src\Config.mak definiert sein wenn RSVP verwen-  
 det werden soll.

## A.4 Definitionen in den Header-files

„\_NRP\_INCLUDED\_“ . . wird in NRP\_Manager.H definiert.  
 NRP- und XNRP-Opcode . wird in NRP\_Opcodes.H festgelegt.

## A.5 Änderungen an Cinema für die Protokoll Einbindung

### Notation:

(Verzeichnis) Dateiname. . Art der Änderung

(CIN) CIN\_Main.C . . . . NRP\_Manager starten und beenden.  
 (CFG,CIN,COMPONENTS,CLIB) CONFIG.make  
 . . . . . „\_NRP\_“, „\_INCLUDE\_NRP\_PROTOCOL\_“ und  
 „\_INCLUDE\_XNRP\_PROTOCOL\_“ Flags setzen und  
 -I\$(TOPDIR)/NRP/src \$(SMD\_HEADERS) Include-Verzeich-  
 nis angeben.  
 (CFG) CFG\_Dispatch.C . . Nachrichten an NRP\_Manager weiterleiten.  
 (CIN) CIN\_Dispatch.C . . Nachrichten an NRP\_Manager weiterleiten.  
 (CFG) CFG\_Component.\* . Komponenten-Objekt um die Methoden NRP- und XNRPSstart  
 erweitern.

- (CFG) CFG\_AppHandler.C NRP- und XNRPSstart Aufrufe eintragen (in DoCreateSession).
- (CIN + CFG) Makefile . . NRP\_Opcodes.H als Include-Datei eintragen.
- (CIN) CIN\_ConnMgr.C + .H  
 . . . . . NRP\_ConnMgr Objekt einfügen, für den direkten Zugriff des NRP-Protokolls auf die Informationen der Komponenten Verbindungen.
- (CFG) CDL\_ReadDesc.C . In CFG\_ComponentDesc\_t::Read(), switch-Anweisung ändern, damit die Kennzahlen, in den „Deskription“ Dateien, richtig erkannt werden.
- (CLIB) CIN\_Component.C + .H  
 . . . . . virtuelle Methoden GetResult, SetQoS, GetRessources, inform, reserve, relax und Free für Komponenten einfügen. Die Methoden SendCNP, get\_formula, get\_restrict und look\_up für Komponenten einfügen. NRP\_Types.H mit AFS Definitionen included. NRP Methoden einfügen um Constrains und Stream-Relations aus Deskription-file einlesen.
- (CIN) CIN\_Constructor.C . Constrains und Stream-Relations aus Deskription-file einlesen.
- (COMPONENTS) \*.CDL . Methoden GetResult, SetQoS, GetRessources, inform, reserve, relax und free müssen enthalten sein.
- (\*\*\*) CONFIG.mak . . . Include-Pfad nach NRP/src zufügen, NRP-Flags setzen.
- (COM) CFG\_Task\_nr.H . . Opcode CFG\_NRP\_PROTOCOL\_ACK 1220 einfügen
- (CAE) CAE\_Ctl.\* + CAE\_Ctl\_MotifAic.\* + CAE\_Mod.\* + applicationShell1.\*  
 + (CIF) CIN\_PortList.\* . . Menü für NRP und XNRP einfügen.
- (CAE) CAE\_Ctl.\* + CAE\_Ctl\_MotifAic.\* + CAE\_Mod.\* + applicationShell1.\*  
 + (CIF) CIN\_Port.\* . . . QoS Parameter-Übergabe für Session und Ressourcen Manager-Simulation einfügen.
- (CFG) CFG\_AppHandler.C CAI\_QOSPARAM einfügen.
- (CFG) CFG\_Component.\* CFG\_Port.\*  
 . . . . . Methoden zur Eintragung von QoS-Parametern in Port-Objekte.
- (CIN) CIN\_Link.\* . . . Methode MixAFS zum Vereinen der AFS's bei Multicast Verbindungen.
- (CIF) CIN\_CinemaObject.C  
 . . . . . TimeOutIntervall von 15 auf 65 erhöhen. (warten bis (X)NRP fertig ist)
- (CLIB) CIN\_Component.\* . Methode einfügen die Daten an den CNP sendet.

## A.6 Sonstige Änderungen an Cinema

Diese Änderungen sind für die Benutzung von NRP und XNRP notwendig. Sie betreffen den normalen Betrieb jedoch nicht und müssen nicht entfernt werden, wenn der NRP und XNRP Kode entfernt werden soll.

**Notation:**

(Verzeichnis) Dateiname. . Art der Änderung

(CIN + CFG) CFG\_ + CIN\_Dispatch.C

. . . . . Sync- (Opcode:2300-2499) und Event-Manager (Opcode:2600-2699) Aufrufe durch Fehler-Meldung ersetzen, da es sich um NULL-Pointer handelt.

(CIN) CIN\_ConnMgr.C . . Weiterleiten von Nachrichten (mit Opcode außerhalb 2200-2299) an den CIN\_Dispatcher. Da bisher alle Sendungen zwischen Cinema-Prozessen beim ConnMgr landen!

(CLIB) CIN\_Component.C + .H

. . . . . Problem: QoS-Methoden in Komponenten werden noch vor der Initialisierung ausgeführt! Der Fehler tritt zwischen dem Ladevorgang und der Initialisierung der Komponente auf.) Bei Komponenten, bei denen der obige Fehler auftritt, arbeitet die Parameter-Übergabe an die INIT-Routine nicht mehr korrekt. Grund: Das Einfügen der QoS-Methoden (in CIN\_Component.H) wird nicht richtig erkannt, die QoS-Methoden werden anstatt der Umwandlungsmethoden („arganal“-Methoden) aufgerufen. Lösung: Einfügen der QoS-Methoden am ENDE des Objekts (CIN\_Component\_t). Sicherung: Eine zusätzliche Variable „CheckInitialisation“ die von der Initialisierung gesetzt wird, und von den Methoden (bei einem Aufruf) abgeprüft wird.

(CLIB) CIN\_Component.C + .H

. . . . . Problem: Die Methode (look\_up) für das Einlesen der Stream-Relations und Constrains arbeitet nicht richtig. Grund: Sie erwartet nach jedem Port-Bezeichner einen Faktor. Lösung: Alle Portangaben mit „\*1“ ergänzen (Außer es ist bereits ein Faktor vorhanden).

(COM) CIN\_Task\_nr.H + CPC\_Task\_nr.H

. . . . . CPC\_SRP\_THREAD und CPC\_SRP\_THREAD\_t auskommentiert, werden nicht benutzt und liegen im Bereich für NRP-Opcode.

(COM) CMG\_UdpGlobal.H Erhöhen des UDP Buffers von 1k auf 60k,sonst vermehrt Fehler bei der Kommunikation und auch Abstürze von Cinema!

(COM) CMG\_ReadUdp.C + CMG\_WriteUdp.C

. . . . . erhöhen der Anzahl der gepufferten Messages und entfernen der Semaphor-Operationen wegen Abstürzen (Signal fault in critical Section vermutlich durch zu schnelles Wiederholen der Semaphor Operationen.

(COM) CMG\_ReadUdp.C . Etwas optimiert, um Nachrichtenverluste zu verringern.

(COM + CIF) CIF\_Task\_nr.H

. . . . . struct CAI\_QOSPARAM\_t einfügen.

(CAE) CAE\_View\_MotifAic.C

. . . . . QoS Eingaben für Outports entfernen, kein MvCrQosDia für Out-Ports.

## **A.7 Einschränkungen des Cinema-Systems bei Benutzung der Protokolle**

Keine „Compound“ Komponenten,

Keine AIX-Unterstützung beim Einlesen der Deskription-Dateien

Globale Komponenten werden nicht unterstützt!

CAE Interface Änderungen nicht auf aic!

Die AFS muß diskrete Werte enthalten. Die Erweiterung auf kontinuierliche Wertebereiche (d.h. min. max. Angaben) ist vorgesehen.

## **A.8 Wichtige Anmerkung für Wartung oder Änderungen**

Antwort Nachrichten in Cinema müssen einen Opcode = Opcode der Originalnachricht + 1 haben. Es kann nur eine Antwort je Nachricht zurückgesendet werden.

## Anhang B      Literaturverzeichnis

- [1] Steinmetz R., Nahrstedt K., *Multimedia: Computing, Communications and Applications*, ISBN 0-13-324435, Prentice Hall P T R, Upper Saddle River, NJ 07458
- [2] The ATM Forum, Technical Committee: *Native ATM Services: Semantic Description*, Version 1.0. Februar 1996.
- [3] L. Zhang et al. RSVP: *A New Resource ReSerVation Protocol*, IEEE Network, p. 8-18, 9 1993
- [4] K. Rothermel, I. Barth, T. Helbig. *Architecture and Protocol's for High-Speed Networks*, Kapitel Cinema - An Architecture for Distributed Multimedia Applications, p. 253-271. Kluwer Academic Publishers, 1994.
- [5] 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, 1996.
- [6] 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, Ottawa, Canada, 1997
- [7]. G. Dermler, W.Fiederer, K. Rothermel. *QoS Negotiation and Resource Reservation Protocol for Distributed Multimedia Applications*. Fakultätsbericht 1996/09, Fakultät Informatik, Universität Stuttgart, Stuttgart 1996
- [8] H. Kern, R. Johnson, M. Hawkins, H. Lyke. *Networking the New Enterprise*. ISBN 0-13-263427-9, Prentice Hall P T R, Upper Saddle River, NJ 07458
- [9] L.C. Wolf. *Resource Management for Distributed Multimedia Systems*. Kapitel 3.2, p. 27. Kluwer Academic Publishers, Boston/Dordrecht/London 1996.
- [10] K. Nahrstedt, J. Smith. *The QOS Broker*. IEEE Multimedia, Vol.2, No.1, S. 53-67, Spring 1995.
- [11] A. S. Tanenbaum: *Computer Networks. Lehrbuch in der dritten Auflage*. Prentice Hall, S. 60-66, S. 344-353, S. 449-475, S. 545-555, 1996.
- [12] K. Wagner, R. Bodendiek. *Graphentheorie I, II, III*. BI-Wissenschaftsverlag, 1989, 1990, 1993.
- [13] M. Paulus. *Integration von Transportsystemen mit Reservierungsmöglichkeit in Cinema*. Studienarbeit. Universität Stuttgart 1997.
- [14] T. Helbig, K. Rothermel. *An Architecture for a Distributed Stream Synchronization Service*. European Workshop on Interactive Distributed Multimedia Systems and Services, Berlin, Germany, March 4-6, 1996.
- [15] T. Helbig. *Timing Control of Stream Handlers in a Distributed Multi-Threaded Environment*. The 6th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 96), April 23 - 26, 1996.
- [16] I. Barth. *Configuring Distributed Multimedia Applications Using Cinema*. IEEE Workshop on Multimedia Software Development MMSD 96 Berlin, Germany, March 25, 1996