GI/ITG-Fachtagung "Kommunikation in Verteilten Systemen",
Chemnitz, Germany, February 22-24, 1995

Implementierung multimedialer Systemdienste in *CINEMA*

Ingo Barth, Tobias Helbig, Kurt Rothermel
Universität Stuttgart
Institut für Parallele und Verteilte Höchstleistungsrechner
Breitwiesenstr. 20-22
D-70565 Stuttgart

Email: {barth,helbig,rothermel}@informatik.uni-stuttgart.de

Zusammenfassung. Die Erstellung verteilter Multimedia-Anwendungen setzt leistungsfähige Entwicklungsplattformen voraus, innerhalb deren Probleme wie die Übertragung multimedialer Daten, die Gewährleistung der Dienstgüte durch Reservierung von Ressourcen und die Synchronisation von Datenströmen gelöst werden. *CINEMA* (Configurable INtEgrated Multimedia Architecture) bietet als eine solche Entwicklungsplattform Abstraktionen zur Nutzung multimedialer Systemdienste an, womit funktionale Bearbeitungseinheiten erstellt, komplexe Verarbeitungstopologien in verteilten Systemen aufgebaut und unter Einhaltung von Synchronisations- und Dienstgüte-Spezifikationen gesteuert werden können. In diesem Papier werden nach einem kurzen Überblick über die Abstraktionen der *CINEMA*-Dienstschnittstelle die internen Strukturen und Abläufe vorgestellt, mit denen die Multimedia-Dienste in verteilten Systemen erbracht werden.

1 Einleitung

Voraussetzung für die effiziente Erstellung verteilter Multimedia-Anwendungen sind geeignete und leistungsfähige Entwicklungsplattformen. Solche Plattformen bieten vielfältige Unterstützungsfunktionen, die sowohl die Bereiche der Multimedia-Kommunikation, der Synchronisation von Datenströmen als auch der Ressourcen-Verwaltung abdecken. Dabei wird die Funktionalität von Betriebs- und Transportsystemen benutzt und angereichert durch spezielle, auf Multimedia-Verarbeitung zugeschnittene Abstraktionen. Eine solche Plattform stellt *CINEMA* (Configurable INtEgrated Multimedia Architecture) dar. Mit *CINEMA* werden multimediale Datenströme beliebiger Komplexität (mehrere Quellen und Senken, beliebige Verbindungstopologie) übertragen, verarbeitet und präsentiert.

In *CINEMA* werden als zentrale Abstraktion funktionale Bausteine (sogenannte Komponenten) definiert, in welchen die Verarbeitung der multimedialen Datenströme stattfindet. Sie lassen sich zu komplexen Topologien verknüpfen und schachteln. Um die Kommunikation und Verarbeitung multimedialer Daten zu ermöglichen, werden Sessions definiert, an die eine bestimmte Dienstgüte gebunden ist, welche durch *CINEMA* unter Reservierung der notwendigen Ressourcen erbracht wird. Der Fluß von Dateneinheiten in beliebig strukturierten Fluß-Graphen wird über sogenannte Medien-Uhren gesteuert. Mit diesen werden die zeitlichen Eigenschaften der Datenströme sowie Synchronisationsbeziehungen zwischen Datenströmen spezifiziert und der Datenfluß durch Aufruf von Steueroperationen beeinflußt. Die sich aus der Verwaltung und Reservierung von Ressourcen, der Verwaltung der Konfiguration komplexer Szenarien und der Synchronisation von Datenströmen ergebenden Schwierigkeiten werden vor dem Dienstnutzer von *CINEMA*, dem Klienten, verborgen.

Mit den von *CINEMA* bereitgestellten Abstraktionen erhält ein Klient Zugriff auf multimediale Dienste in verteilten Systemen, deren Implementierung in diesem Papier näher vorgestellt wird. Zuerst wird in Abschnitt 2 ein Blick auf verwandte Arbeiten geworfen und in Abschnitt 3 ein

Überblick über die Abstraktionen der *CINEMA*-Dienstschnittstelle gegeben. Im Abschnitt 4 wird die Umsetzung der Systemdienste anhand der drei wichtigsten Problembereiche dargestellt. Es wird auf das *Konfigurationsmanagement* eingegangen, mit dem ganze Anwendungstopologien über Rechner und Prozesse hinweg verteilt, instanziiert und durch Kommunikationskanäle verbunden werden sowie auf das *Ressourcenmanagement*, das die Bereitstellung der Betriebsmittel zur Erbringung der Multimedia-Dienste erbringt. Weiterhin werden die Strukturen des *Synchronisationsmanagements* zur Steuerung und Synchronisation von Datenströmen dargestellt. Das Papier schließt mit einer Zusammenfassung.

2 Verwandte Arbeiten

Die vielfältigen Probleme, die durch die Integration von multimedialer Informationsverarbeitung in konventionelle Rechnersysteme entstehen, haben auf verschiedenen Gebieten Aktivitäten angeregt. Aus dem Wissen über die Unzulänglichkeiten konventioneller Transport- und Betriebssysteme (vgl. [BuLi91]) für die Handhabung multimedialer Daten, wurden Erweiterungen von Kommunikationsdiensten angestrebt (z.B. Integration von Synchronisationsmechanismen [EDP92], [BCG+92] und Einbeziehung von Dienstgüte-Parametern [CCGH92]). Mit HeiTS [HHS+91] wird ein Transportsystem für Multimedia-Systeme bereitgestellt, welches Garantien für die Übertragung von Datenströmen auf ausgewählten Netzwerken geben kann. Im SUMO-Projekt [CBRS93] wird das Chorus-Betriebssystem [RAA+90] für die Verarbeitung von kontinuierlichen Medien erweitert. Dazu wird ein strombasierter Kommunikationsdienst realisiert und die Dienstgüteverwaltung in das Betriebssystem integriert. Der Schwerpunkt solcher Arbeiten liegt auf der Schaffung der jeweils speziellen Funktionalität, auf die dann mit geeigneten Schnittstellen zugegriffen werden kann. Das Ziel von *CINEMA*, eine universelle Plattform zum Entwurf verteilter Multimedia-Systeme bereitzustellen und verschiedenste Systemdienste zu integrieren, steht hierbei nicht im Vordergrund.

Ein vielverwendetes Konzept, verteilte Anwendungen aus einfachen Bausteinen aufzubauen, die nur über Ports miteinander verknüpft werden müssen, wurde mit Conic [KrMa85] und dem Nachfolgeprojekt REX [MKSD90] aufgegriffen und findet sich in einer Reihe anderer Projekte in abgewandelter Form wieder (IMA [Hewl93], QuickTime [Appl91], SUMO [CBRS93], *CINEMA*). In Conic wird dabei eine Sprache für die Programmierung von Komponenten und Applikationen angeboten, die allerdings keine Aspekte der multimedialen Datenverarbeitung berücksichtigt. Die Konfiguration wird durch einen zentralisierten Konfigurationsmanager aufgebaut und Änderungen werden durch diesen umgesetzt. Die gemeinsame Benutzung von Komponenten in mehreren Konfigurationsbeschreibungen ist nicht vorgesehen.

Die weitergehende Frage des Entwurfes von geeigneten Abstraktionen zum Entwurf von Multimedia-Systemen wurde ebenfalls in verschiedenen Arbeiten in Angriff genommen. ACME (Abstractions for Continuous Media [HGA90]) stellt Abstraktionen für die Aufnahme und Wiedergabe von Audio- und Videodaten zur Verfügung. Sie dienen der Erweiterung konventioneller Fenster-Oberflächen um kontinuierliche Daten. Mit ihnen können einfach Client-Server-Strukturen zur Kommunikation und Präsentation von Audio- und Videodaten entwickelt werden. Eine ähnliche Zielsetzung verfolgt Tactus [DNNR92], wohingegen mit QuickTime [Appl91] und IBM's Multimedia Presentation Manager [IBM92] Toolkits zur lokalen Präsentation von Audio und Video auf Rechnern kommerziell zur Verfügung stehen. In [BCA+92] wird eine integrierte Plattform und ein Verarbeitungsmodell für offene verteilte Multimedia-Anwendungen auf der Basis von ANSA vorgestellt. Dabei werden die Kommunikation und Synchronisation durch Dienste modelliert, die zu einer verteilten Multimedia-Anwendung zusammengesetzt werden können. Der *Request for Technology* [IMA92] der Interactive Multimedia Association

(IMA) zeigt die grundsätzlichen Anforderungen an eine Architektur für verteilte Multimedia-Anwendungen. In dem von mehreren Firmen erstellten Vorschlag [Hewl93] zu diesem RFT werden Abstraktionen für die Struktur und den Aufbau einer verteilten Multimedia-Umgebung unter dem Aspekt der herstellerübergreifenden Verarbeitung vorgestellt. Die Schachtelung von Grundbausteinen zur Erzeugung höherer Abstraktionsebenen ist jedoch mit den bisher vorgeschlagenen Abstraktionen und Konzepten ebensowenig möglich wie die automatische Umsetzung von Synchronisationsbeziehungen.

3 Anwendungskonzepte und Abstraktionen in CINEMA

In diesem Abschnitt werden kurz die Abstraktionen vorgestellt, die einem Nutzer von *Cinema* zur Erstellung und Steuerung verteilter Multimedia-Anwendungen zur Verfügung stehen. Für eine detaillierte Diskussion der Konzepte Komponente, Port, Link, Session und Medien-Uhr sei auf [RBH94] und [RoHe94a] verwiesen.

3.1 Konfiguration von Verarbeitungstopologien

Kontinuierliche Datenströme bestehen aus einer Sequenz von Dateneinheiten, die mit Zeitmarken assoziiert sind (vgl. z.B. [Herr91]). Die Verarbeitung der Dateneinheiten kontinuierlicher Ströme findet in *Cinema* in **Komponenten** statt, welche die aktiven Einheiten bilden. Sie besitzen neben den funktionsspezifischen Schnittstellen, wie beispielsweise Lautstärke- oder Kontrast-Regler, auch generische Schnittstellen, über welche sie vom *Cinema*-System unabhängig von ihrer internen Funktionalität und Implementierung verwaltet werden. Wir unterscheiden zwischen Quellen, die Datenströme erzeugen, Senken, die nur konsumieren und zwischengelagerten Komponenten, die Daten lesen, transformieren und verändert wieder abgeben. Komponenten konsumieren und produzieren Daten durch Zugriffe auf typisierte Ports. **Ports** bilden konfigurationsunabhängige Datenzugriffspunkte, d.h. einer Komponente bleibt verborgen, ob vom Port gelesene Daten von einer anderen Komponente erzeugt wurden, die auf dem gleichen oder einem anderen Rechner lokalisiert ist. Die Implementierung von Komponenten und das Testen ihrer Funktionsfähigkeit ist somit vollständig von dem späteren Anwendungsszenario entkoppelt, was deutlich die Modularisierung und Strukturierung von Anwendungen fördert.

Komponenten sind entweder elementare Bausteine oder werden durch Schachtelung aus anderen Komponenten erzeugt. Dadurch lassen sich höhere funktionale Abstraktionsebenen bilden. Das Konzept der Schachtelung erlaubt es, die von Programmiersprachen bekannten Eigenschaften der Kapselung von Funktionalität auf die Erzeugung multimedialer Verarbeitungsbausteine zu übertragen, was die Strukturierung von Anwendungen und die Wiederverwertbarkeit von Code erleichtert. Bei Komponenten wird unterschieden zwischen solchen, die nur für eine Anwendung bekannt und zugreifbar sind und den über global eindeutige Bezeichner adressierten, von mehreren Anwendungen gemeinsam benutzten Komponenten, den sogenannten "**globalen Komponenten**", welche in Mehrbenutzeranwendungen die Bindeglieder zwischen den beteiligten Anwendungen bilden.

Multimedia-Anwendungen werden konfiguriert, indem **Links** zwischen den Ausgabe- und Eingabeports von Komponenten definiert werden. Mit diesem Konfigurationsmechanismus lassen sich beliebige, auf die Problemstellung zugeschnittene Anwendungstopologien, sogenannte Fluß-Graphen, aufbauen. Während mit Links die Topologie von Anwendungen beschrieben wird, dient die Abstraktion **Session** zur Reservierung von Ressourcen. Datenströme können erst verarbeitet und übertragen werden, wenn die zugehörige Session eingerichtet wurde. Dies

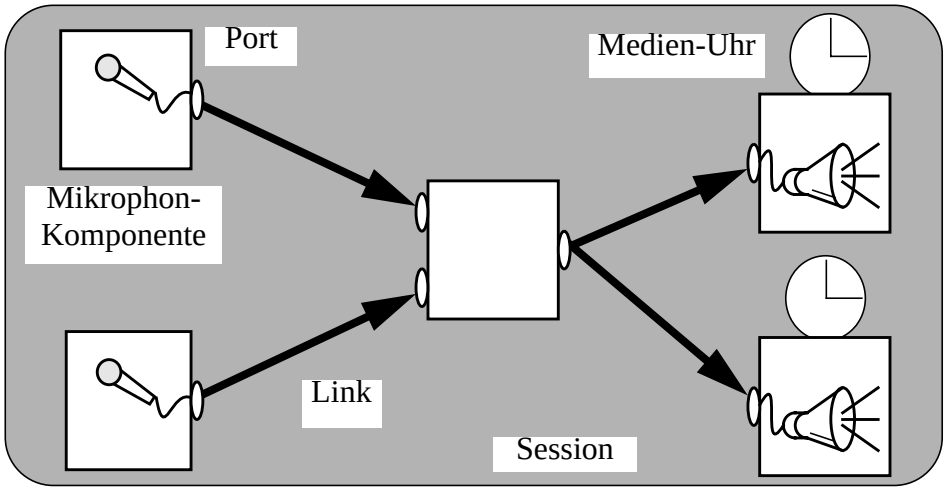


Abb.: 1 Beispielszenario: Audio-Konferenz

geschieht durch Angabe einer Menge von Quellen und Senken, die über ein beliebiges Netzwerk von zwischengelagerten Komponenten verbunden sind. Sessions sind assoziiert mit Dienstgüte-Parametern, aus denen der Bedarf an Ressourcen abgeleitet wird. Eine Session besteht somit aus einem Teil der Anwendungstopologie, deren Instanziierung ganz oder gar nicht erfolgt. Nach der Erstellung einer Session kann der Fluß von Dateneinheiten gestartet werden.

3.2 Steuerung und Synchronisation von Datenströmen

Zur Steuerung des Flusses von Dateneinheiten wird die Abstraktion der Medien-Uhr und das Konzept der Uhren-Hierarchie eingeführt. **Medien-Uhren** dienen der Spezifikation temporaler Eigenschaften von Datenströmen. Sie spannen dazu das Medienzeit-System auf, d.h. sie verwalten eine Menge von zeitbezogenen Parametern wie das Raten-Verhältnis R zwischen Medien- und Echtzeit, den Startwert der Medienzeit M, die Zeit T des Echtzeit-Beginns der Verarbeitung sowie einen relativen Geschwindigkeitsfaktor S zur Skalierung der Verarbeitung. Medien-Uhren sind mit Quellen- oder Senken-Komponenten assoziiert. Mit dem Starten einer Uhr wird die Verarbeitung von Daten entsprechend der zeitlichen Spezifikationen für den zugehörigen Datenstrom eingeleitet. Die Skalierung der Datenübertragung (d.h. Änderung von Richtung oder Geschwindigkeit) oder das Anhalten von Strömen erfolgt durch Anpassung des Voranschreitens der Zeit von Uhren.

Uhren-Hierarchien beschreiben Beziehungen zwischen Medien-Uhren und damit zwischen den zugehörigen Datenströmen. Sie werden benutzt, um Gruppen von Datenströmen zu bilden, diese als Einheit zu steuern sowie um Synchronisationsbeziehungen zu spezifizieren. Die Gruppierung von Datenströmen erfolgt durch Zuordnen der sie steuernden Medien-Uhren zu einer gemeinsamen, übergeordneten Uhr. Operationen, deren Aufruf an übergeordneten Medien-Uhren erfolgt, werden in Richtung der Blätter von Uhren-Hierarchien propagiert. Durch die Gestaltung der Struktur der Uhren-Hierarchie und die Auswahl der Hierarchie-Stufe, auf der Operationen ausgelöst werden, ist eine flexible Gruppierung von Strömen gegeben. Dies wird ergänzt durch die Möglichkeit zum Sperren der Propagierung von Operationen, was es erlaubt, ganze Teil-Hierarchien auszublenden und darüber beispielsweise die Zuständigkeitsbereiche verschiedener Benutzer gegeneinander abzugrenzen.

Die Propagierung von Operationen und die Spezifikation von Synchronisationsbeziehungen erfolgt durch die Kopplung der von Medien-Uhren aufgespannten Medienzeitsysteme. Das geschieht auf der Basis von Referenz-Punkten. Ein **Referenz-Punkt** *RP [$C_1 : P_1$, $C_2 : P_2$]* definiert, daß die Medienzeit P_1 im Zeitsystem der Uhr C_1 zur Medienzeit P_2 im Zeitsystem von C_2 korrespondiert. Das erlaubt die Transformation der Medienzeiten ineinander: $m_2 = f(m_1)$ mit

$$m_2 = (m_1 - P_1) \cdot \frac{S_2 \cdot R_2}{S_1 \cdot R_1} + P_2$$

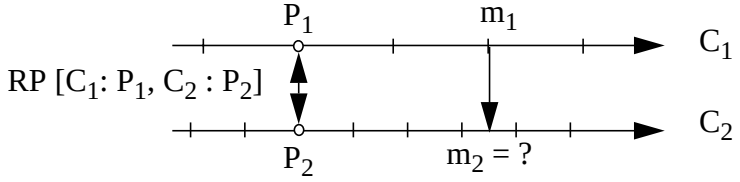


Abb.: 2 Transformation von Medienzeit

Uhren-Hierarchien bilden dynamische Strukturen, die auch zur Laufzeit einer Anwendung an Änderungen in der Systemstruktur angepaßt werden können. So lassen sie sich erweitern bzw. verkleinern, wenn bei Mehrbenutzer-Anwendungen neue Benutzer hinzukommen oder andere ausscheiden. Veränderte Anforderungen an die Gruppierung von Datenströmen (z.B. das Erzeugen oder Auflösen von Gruppen) lassen sich durch Umgestaltung der Struktur der Hierarchie leicht realisieren. Dies unterstützt die Handhabung der in interaktiven und kooperativen Multimedia-Anwendungen auftretenden Dynamik.

Zusätzliche Flexibilität in der Steuerung von gruppierten Datenströmen wird durch Attribute der Kanten von Uhren-Hierarchien erreicht. So ist es möglich, die Beziehung zwischen Uhren auf eine Steuerbeziehung zu beschränken. Hierdurch wird eine lose Kopplung der Zeitsysteme ohne feste Synchronisation festgelegt. Trotz zusammengefaßter Ausführung von Operationen der Gruppe ist ein Drift von Strömen oder eine getrennte Skalierung möglich. Anders bei der Synchronisationsbeziehung. Mit ihr wird der Gleichlauf der Medienzeiten und darüber die zeitliche Bindung von Datenströmen abgesichert. Dies geschieht sowohl während der Verarbeitung von Daten wie auch beim späteren Hinzuschalten weiterer Ströme.

4 Realisierung der Systemdienste von CINEMA

Mit den im vorherigen Abschnitt beschriebenen Abstraktionen und Konzepten steht eine Dienstschnittstelle zur Verfügung, die es erlaubt, komplexe und dynamische Multimedia-Anwendungen aufzubauen und zu steuern. Um die damit ausdrückbare Vielfalt von Funktionen tatsächlich zu erbringen, wird *CINEMA* auf Betriebs- und Transportsysteme aufgesetzt, deren Funktionalität für die Verarbeitung multimedialer Daten angereichert wird. In diesem Abschnitt erfolgt dazu die Beschreibung ausgewählter Systemstrukturen und Abläufe von *CINEMA*. Nach der Vorstellung der Programmierschnittstelle für Zugriffe auf Systemdienste werden die Architekturen zur Umsetzung der wesentlichen Systemdienste, d.h. der Verwaltung und Konfiguration komplexer Topologien, der Steuerung und Synchronisation von Datenströmen sowie der Ressourcen-Reservierung (vgl. Abb. 3) vorgestellt. Auf die Realisierung der Ereignisverwaltung wird nicht im Detail eingegangen.

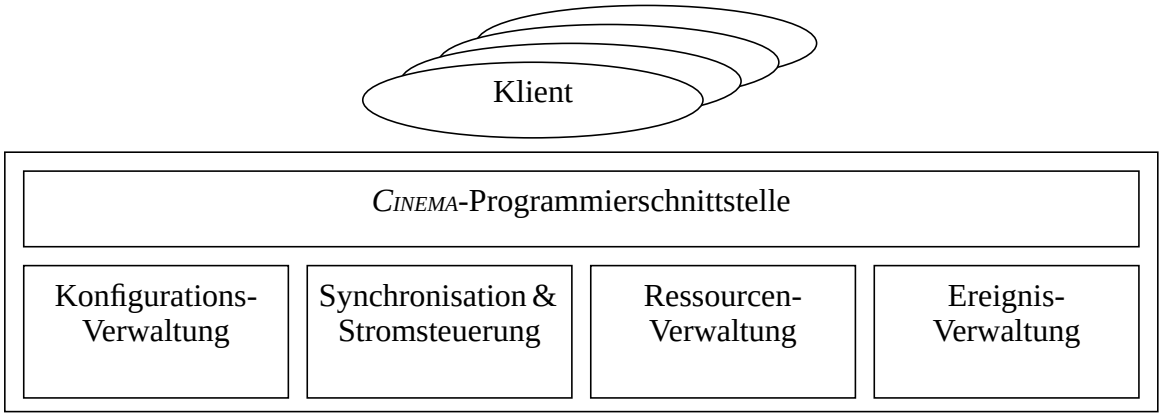


Abb.: 3 Architektur des CINEMA-Systems

4.1 CINEMA-Programmierschnittstelle

Die von *CINEMA* bereitgestellten Abstraktionen zum Aufbau verteilter Multimedia-Anwendungen werden den Klienten über die *CINEMA*-Programmierschnittstelle zugänglich gemacht. Mit dieser Programmierschnittstelle kann der Klient mittels einer objektorientierten Programmiersprache multimediale Verarbeitungstopologien erzeugen und steuern. Der Aufbau von Topologien ist dynamisch zu einem beliebigen Zeitpunkt zur Laufzeit möglich.

Dazu erzeugt der Klient zuerst Komponentenobjekte (im Beispiel Zeile 1 `camera` und Zeile 2 `display`), für die er als Parameter die Komponentenklasse und die gewünschte Lokation spezifiziert. Die Ports (im Beispiel `video`) der Komponentenobjekte werden dann durch Links (Zeile 3) miteinander verbunden. Über die Quellen- und Senkenports der Topologie erfolgt die Definition einer Session unter Angabe von Dienstgüte-Parametern (Zeilen 5-10). Neben den Komponentenobjekten kann der Klient Uhrenobjekte erzeugen, die er mit Quellen und Senken assoziiert (Zeile 4) und ggf. in einer Uhren-Hierarchie miteinander in Beziehung setzt. Die Steuerung des Flusses von Dateneinheiten wird durch Aufruf der Methoden von Uhrenobjekten vorgenommen (Zeile 11).

Beispiel-Code:

```
 1  comp_camera  = COMPONENT(“camera”,  Camera1);
 2  comp_display = COMPONENT(“display”, Display1);
 3  link(comp_camera->port(“video”),comp_display->port(“video”));
 4  comp_display->associate(clock);
 5  create_session(comp_camera ->port(“video”),
 6                 comp_display->port(“video”),
 7                 QoS(Rate          (min =  15, max =  25),
 8                     Picturewidth  (min = 200, max = 400),
 9                     Pictureheight (min = 150, max = 300),
10                     Delay         (min = 150, max = 250));
11  clock->start();
```

Die Objekte und Funktionen der Programmierschnittstelle werden dem Klienten in einer Bibliothek zur Verfügung gestellt. Die Objekte (wie die Komponenten im obigen Beispiel) sind Proxyobjekte [Shap86]. Als lokale Platzhalter der tatsächlichen, i.a. entfernt instanziierten Objekte nehmen sie Nachrichten des Klienten entgegen und leiten sie transparent an das *CINEMA*-System (z.B. das Konfigurationsmanagement) oder die tatsächlich Objektinstanzen weiter. Die Proxyobjekte sind stets lokal zum Klienten, verhalten sich wie die tatsächliche Realisierung des

entfernten Objekts und erlauben es deshalb, die Dienste von *CINEMA* leicht in objektorientierte Programmiersprachen (wie z.B. C++) zu integrieren.

4.2 Konfigurationsmanagement

Verteilte Multimedia-Anwendungen in *CINEMA* bestehen aus einem Geflecht von Komponenten, die über Kommunikationskanäle miteinander verbunden sind. Die Aufgabe des Konfigurationsmanagements ist es, aus der vom Klienten vorgegebenen Spezifikation des Geflechts eine ablauffähige Instanziierung im verteilten System aufzubauen. Dafür müssen geschachtelte Komponenten aufgelöst und das daraus entstandene Komponentengeflecht auf die Rechnerknoten verteilt und instanziiert werden. Außerdem müssen von mehreren Klienten gemeinsam genutzte Komponenten verwaltet und Information über Topologien und Komponenten für andere Managementbereiche des *CINEMA*-Systems zur Verfügung gestellt werden. Zusätzlich zur Instanziierung von Komponenten werden durch das Konfigurationsmanagement die Kommunikationskanäle zwischen den Ports der Komponenten durch Instanziierung von Linkobjekten aufgebaut.

Im folgenden wollen wir die Instanziierung der Komponenten auf den Knoten näher betrachten. Hierfür gehen wir von bereits lokalisierten Komponenten aus, d.h. für jede Komponente ist vom Klienten vorgegeben, auf welchem Knoten ihre Realisierung zu instanziieren ist.[1] Die Komponenten werden auf den Knoten in speziellen Prozessen, den sogenannten *CINEMA*-Prozessen, ausgeführt, in denen Threads für die Aktivierung der Komponentenmethoden zur Verfügung stehen. In *CINEMA* werden Threads verwendet, da sie sowohl Nebenläufigkeit mit geringem Prozeßwechsel-Aufwand wie auch effiziente Kommunikation durch gemeinsame Adressbereiche unterstützen. Für die Zuordnung von Komponenten zu Threads stehen verschiedene Alternativen zur Auswahl. Sie werden im folgenden unter dem Aspekt des Einflusses auf die Ende-zu-Ende-Verzögerung eines Datenstroms diskutiert.

Die einfachste Form der Verteilung von Komponenten auf Threads ist die Zuordnung jeder Komponente zu einem eigenen Thread. Aufgrund der Möglichkeit, jede Komponente individuell aktivieren zu können, läßt sich die maximale Nebenläufigkeit erreichen. Die hohe Zahl von Threads entlang eines Datenpfades bewirkt aber eine hohe Ende-zu-Ende-Verzögerung. Dies resultiert aus zusätzlichen Verzögerungen, die durch das Scheduling von Threads eingeführt werden und zur reinen Bearbeitungs- und Kommunikationsverzögerung hinzukommen. Die in Multimedia-Systemen üblichen Schedulingverfahren (wie *Rate-Monotonic* und *Earliest-Deadline-First* [6]) gehen von einer Verzögerung der Aktivierung aus, die aus der Differenz zwischen dem ersten möglichen Aktivierungszeitpunkt und der Deadline resultiert und größer als die reine Bearbeitungszeit ist. Da die Verfahren stauvermeidend sind, beträgt diese zusätzliche Verzögerung pro Thread eine Periode[2] und ist unabhängig von der Bearbeitungszeit. Das bedeutet beispielsweise bei einem Videostrom mit einer Rate von 25 Bildern pro Sekunde, daß pro Komponente eine Verzögerung von 40 msec eingeführt wird. Bei fünf aufeinanderfolgenden Komponenten summiert sich dies bereits zu einer Verzögerung von 200 ms, zu der die Verzögerungen aufgrund der Übertragung von Daten noch hinzukommen.

Abhilfe kann durch zwei Maßnahmen geschaffen werden: Einerseits können Schedulingverfahren dahingehend verbessert werden, daß kürzere Schedulingverzögerungen erreicht werden

---

1. Die Möglichkeit einer automatischen Verteilung von zwischengelagerten Komponenten unter Betrachtung der Systemlast und Ressourcenverfügbarkeit wird derzeit untersucht.
2. Periode = Verzögerung durch das Scheduling + Bearbeitungszeit

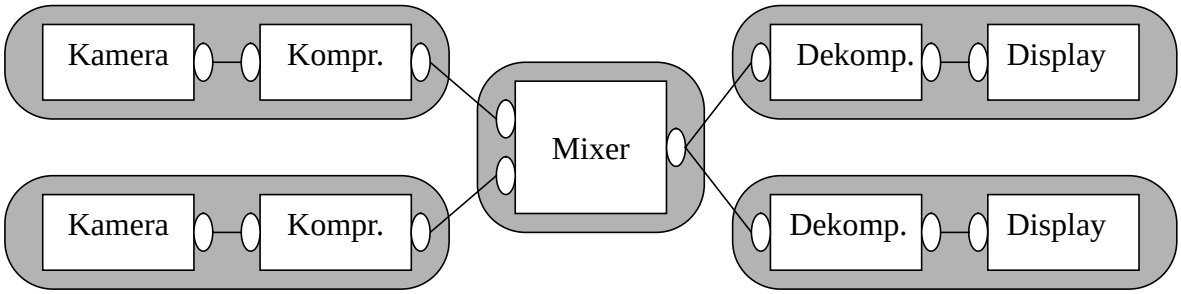


Abb.: 4 Verteilung einer Topologie auf Threads

(vgl. hierzu Abschnitt 4.4), andererseits kann durch die Zusammenfassung von mehreren Komponenten in einen Thread die Gesamtzahl der Threads und somit die durch sie eingeführte Verzögerung verringert werden. Die Zusammenfassung von Komponenten in einen Thread bewirkt, daß nur noch eine Wartesituation auftreten kann. Dies erfordert, daß die Dateneinheiten, die von einer Komponente erzeugt werden, vollständig von der nächsten Komponente konsumiert werden. Es entsteht so eine Bearbeitungspipeline, bei der nur die erste Komponente Daten konsumieren kann, die von Komponenten stammen, die in anderen Threads oder auf anderen Knoten liegen. Abb. 4 zeigt eine Topologie eines Konferenzszenarios mit der Threadzuordnung.

4.3 Stromsteuerung und Synchronisationsmechanismen

In *Cinema* wird mit Medien-Uhren und Uhren-Hierarchien eine einheitliche, universelle Dienst-Schnittstelle zur Steuerung und Synchronisation von Datenströmen angeboten. Die für die Erbringung der dadurch zur Verfügung gestellten Funktionalität gewählte Architektur wird im weiteren vorgestellt.

Ein Synchronisationsservice wird allgemein durch geeignete Synchronisationsprotokolle erbracht. Viele der in der Literatur vorgeschlagenen Protokolle (vergleiche z.B. [EDP92], [RRVK92], [AnHo91]) sind auf spezielle Systemumgebungen, Datentypen oder Topologien zugeschnitten. Dem rechnungtragend, ermöglicht die *Cinema*-Architektur die Integration unterschiedlicher Synchronisationsprotokolle, welche dann alternativ in Abhängigkeit vom Synchronisationsszenario verwendet werden können. Hierdurch wird sowohl die Auswahl des bestgeeigneten Protokolls für eine bestimmte Konfiguration wie auch die Erweiterbarkeit des Systems unterstützt.

Das Synchronisationssubsystem von *Cinema* besteht aus drei Schichten (Abb. 5). Die Medien-Uhren-Verwaltungsschicht (Media Clock Management Layer - MCML) verwaltet Uhrenobjekte sowie die Struktur von Uhren-Hierarchien und trägt die Verantwortung für die Propagierung von Uhrenoperationen. Der verteilte Synchronisationsdienst wird durch die Stromsynchronisationsschicht (Stream Synchronization Layer - SSL) erbracht. Sie besteht aus einzelnen Flußsteuerungs- und Synchronisationsmodulen, von denen jedes ein spezielles Steuerungs- und Synchronisationsprotokoll realisiert. Diese Synchronisationsmodule benutzen die Basisdienste, wie beispielsweise Pufferverwaltung, Scheduling oder den Kommunikationsdienst für Multimediadaten, die durch die *Cinema*-Basisdienstschicht (*Cinema* Base Services Layer - CBSL) verfügbar gemacht werden. Aus der Sicht des Synchronisationssubsystems subsumiert die CBSL

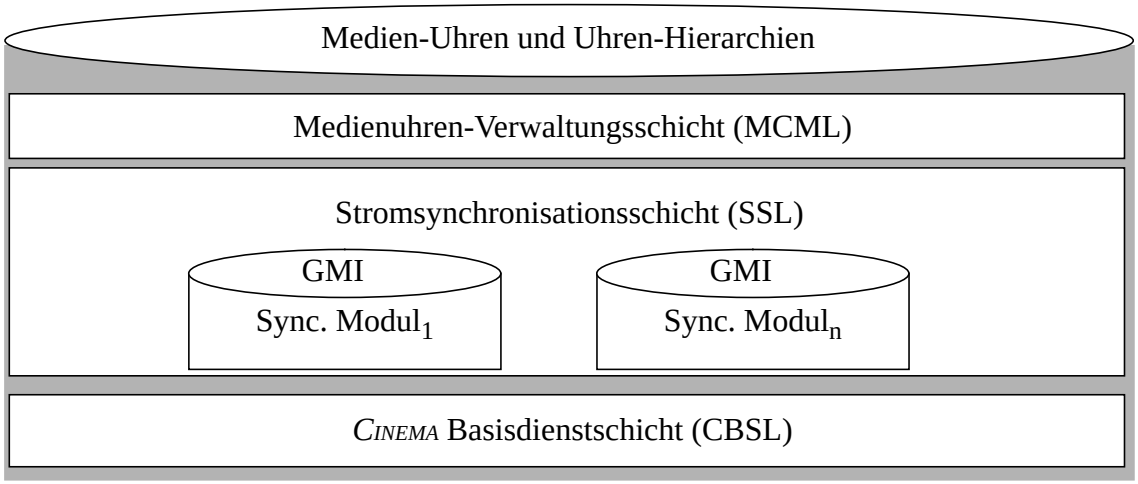


Abb.: 5 Schichten der Architektur des Synchronisationsdienstes

diejenigen Basisfunktionen, die von anderen Managementteilen des *CINEMA*-Systems erbracht werden.

Medien-Uhren-Verwaltungsschicht (MCML)

Die MCML verwaltet die Struktur und den Status von Medien-Uhren und Uhren-Hierarchien. Sie bildet die durch Uhren-Hierarchien gebotene, anwendungsorientierte Schnittstelle auf die stromorientierte Schnittstelle der Synchronisationsmodule ab.

Zur Verwaltung von Medienuhren werden Informationen über die von einer Anwendung angelegten Uhren, ihre Verknüpfung zu Uhren-Hierarchien und ihren aktuellen Status, wie z.B. zu welcher Komponente sie zugeordnet sind, ob sie ticken oder ob Sperren gesetzt sind, unterhalten. Um die strukturelle Konsistenz von Uhren-Hierarchien zu gewährleisten, werden Konsistenzüberprüfungen durchgeführt. Außerdem sind Medien-Uhren einem der Synchronisationsmodule der SSL zugeordnet. Derzeit erfordert die Auswahl eines geeigneten Synchronisationsmoduls für ein Anwendungsszenario entsprechende Hinweise durch den Klient. Es werden jedoch Kriterien zur automatischen Modulauswahl untersucht.

Die MCML ist weiterhin zuständig für die Propagierung von Uhrenoperationen in Hierarchien. Dabei werden die jeweiligen Parameter einer Operation in das Zeitsystem der betroffenen Uhr transformiert. Wird eine Operation zu einer Uhr propagiert, die mit einer Komponente verbunden ist, erfolgt eine Notifikation des zuständigen Synchronisationsmoduls, welches für die Umsetzung im verteilten System sorgt.

Stromsynchronisationsschicht (SSL)

Synchronisationsmodule realisieren die vorhandenen Synchronisationsprotokolle unter Verwendung der von der *CINEMA* Basisschicht bereitgestellten Funktionen.

Auf ein Synchronisationsmodul wird über das stromorientierte, generische Modul-Interface (GMI) zugegriffen, welches eine einheitliche Schnittstelle zu den verschiedenen Synchronisationsmodulen darstellt. Das GMI ermöglicht die Spezifikation von zeitlichen Eigenschaften einzelner Ströme und von Synchronisationsbeziehungen zwischen ihnen. Über das GMI wird dem Synchronisationsmodul die zu steuernde Konfiguration verbundener Komponenten in Form

eines Flußgraphen bekannt gemacht. Weiterhin wird das Starten, Stoppen und Skalieren von Flußgraphen als Ganzes oder in Teilen unterstützt. Der Ansatz profitiert vom objektorientierten Programmierparadigma, da die Schnittstellen zu speziellen Synchronisationsmodulen aus der generellen Schnittstelle durch Überdefinieren der jeweiligen Funktionen erzeugt werden.

Intern besteht ein Synchronisationsmodul aus einem GMI-Agenten und einer Reihe von sogenannten Synchronisationmodul-Agenten (SM-Agenten). Der GMI-Agent implementiert die Schnittstelle zum Modul und instanziiert SM-Agenten auf den Knoten, die am jeweiligen Kommunikationsszenario beteiligt sind. Welche Knoten dies sind, wird aus der Flußgraph-Spezifikation ermittelt, die durch die MCML bereitgestellt wurde. Nach ihrer Instanziierung steuern die SM-Agenten kooperativ den Fluß von Dateneinheiten entsprechend des gewählten Synchronisationsprotokolls.

Derzeit sind in unseren Prototyp zwei Synchronisationsprotokolle integriert. Eines erlaubt die synchrone Wiedergabe gespeicherter Daten [Sche94], das andere die Synchronisation von "Live"-Datenströmen durch Schätzung und Angleichung der Übertragungszeiten unterschiedlicher Übertragungskanäle. Letzteres ist eine modifizierte Version des Flow Synchronization Protocols [EDP92]. Ein weiteres Protokoll [RoHe94b], welches eine weitgehende Anpassung der Dienstgüte unter unterschiedlichen Strategien erlaubt, wird gegenwärtig implementiert.

CINEMA Basisdienstschicht (CBSL)

Wichtigste Aufgabe der CBSL ist die Erweiterung von Betriebssystemfunktionen, um die Anforderungen verteilter multimedialer Verarbeitung zu erfüllen. So wird beispielsweise das Echtzeit-Scheduling unabhängiger Verarbeitungsprozesse durch einen erweiterten Betriebssystemscheduler verfügbar gemacht. Multimediale Datenkommunikation ist gekapselt in Link-Objekte, welche über einheitliche Schnittstellen die Übertragung von Daten zwischen Komponenten erbringen. Dabei können sich Komponenten entweder auf dem gleichen Knoten oder entfernt voneinander auf verschiedenen Knoten befinden, wodurch die Konfiguration verteilter Anwendungen erleichtert wird. Letztlich bietet die CBSL eine erweiterte Pufferverwaltung, die es erlaubt, Speicher für eine Session zu reservieren und fest im Hauptspeicher zu verankern.

Beispiel

Das Zusammenwirken der unterschiedlichen Schichten der Synchronisationsarchitektur soll mit dem folgenden Beispiel (Abb. 6) verdeutlicht werden. Es wird von einem Synchronisationsprotokoll ausgegangen, das auf dem Ausgleich von Übertragungszeiten verschiedener Kanäle basiert. Vereinfachend wird angenommen, daß alle Kommunikationsverzögerungen zwischen GMI-Agent und SM-Agenten sowie zwischen Quelle und Senke der Multimedia-Datenpfade bekannt seien. Aus Platzgründen wird im folgenden nur die Startphase der synchronen Datenübertragung betrachtet.

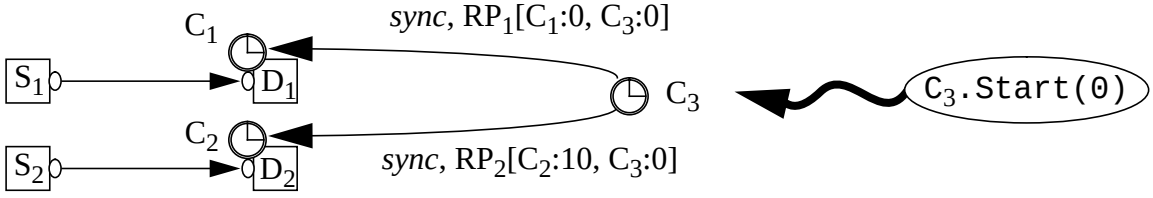


Abb.: 6 Beispielszenario

Sobald der Klient die Startoperation C_3.Start(0) aufruft, werden im MCML die folgenden Aktionen ausgelöst: Zuerst findet die Propagierung der Operation zu den Medienuhren C_1 und C_2 statt, wobei die Parameter entsprechend abgebildet werden. Anschließend wird die ermittelte Menge von Operationen {C_1.Start(0), C_2.Start(10)} auf die Modulschnittstelle des ausgewählten Synchronisationsmoduls abgebildet:

```
ModulGMI->StartFlow(C1, M1=0, C2, M2=10, Zeit=sofort);
```

Der GMI-Agent des Moduls führt daraufhin die Kommunikation zu den SM-Agenten der entfernten Lokationen und hierüber den tatsächlichen Start der Ströme aus. Dazu werden die Startzeiten der periodischen Verarbeitungseinheiten auf den Knoten der Quellen S_1 und S_2 sowie der Senken D_1 und D_2 im voraus berechnet und über Steuerbotschaften an die SM-Agenten der Knoten übermittelt (siehe Abb. 7). Mit dem Eintreffen der Botschaften wird das Scheduling der Verarbeitungseinheiten durch die SM-Agenten initialisiert und zur gegebenen Zeit gestartet.

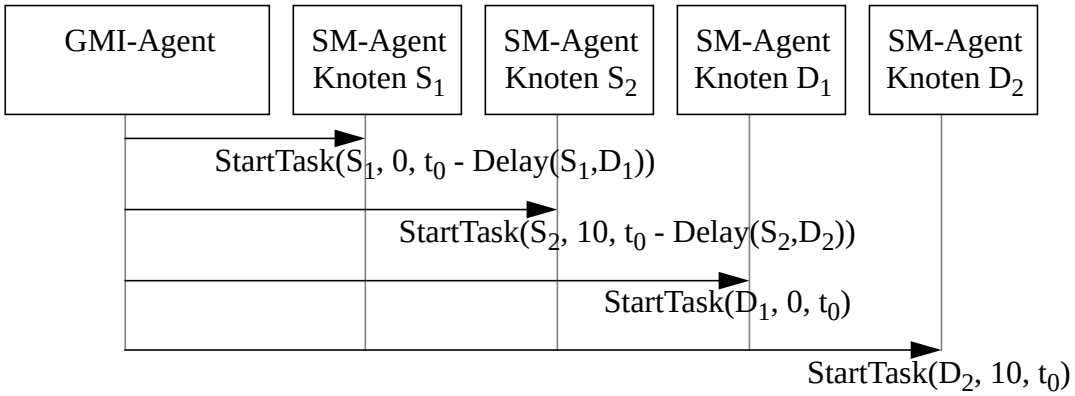


Abb.: 7 Austausch von Steuerbotschaften

4.4 Ressourcenmanagement

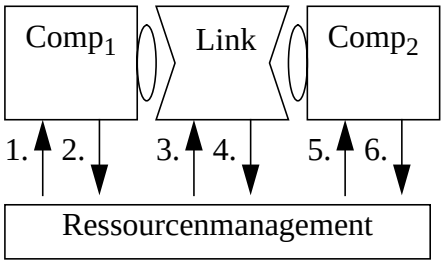
Die Qualität multimedialer Anwendungen wird von der Einhaltung der vom Klienten spezifizierten Randbedingungen, den Dienstgüte-Parametern, bestimmt. Da die Verarbeitung multimedialer Daten hohe Anforderungen an die Leistungsfähigkeit von Ressourcen (z.B. CPU, Bandbreite, Speicher) stellt, ist es notwendig, die Einhaltung der Dienstgüte durch geeignete Verwaltung, Reservierung und Zuteilung der Ressourcen abzusichern. Aufgabe des Ressourcenmanagements ist es deshalb, aus Dienstgüte-Vorgaben des Klienten den Bedarf an Ressourcen auf den jeweiligen Knoten des verteilten Systems zu ermitteln, beim Verwalter der Ressource zu reservieren und zur Laufzeit zuzuteilen. Bei der Reservierung von Ressourcen handelt es sich um einen dynamischen Verhandlungsprozeß, bei dem eine Anpassung zwischen den Wünschen des Klienten und den Möglichkeiten des Systems (maximal verfügbare Kapazität, Lastsituation) erzielt werden muß.

Bisherige Arbeiten zur Reservierung von Ressourcen in Multimedia-Systemen entstammen vor allem dem Netzwerkbereich (SRP [AHS90], RSVP [ZDE+93], ST-II [Topo90]). Sie unterstützen Punkt-zu-Punkt-Verbindungen und Multicast-Topologien. Die betrachteten Dienstgüte-Parameter werden durch die Anforderungen an die Kommunikationssysteme bestimmt (z.B. Übertragungsverzögerung, Jitter, Burst, Paket-Größe, Paket-Verlust-Rate etc.). Da die Schnittstelle zur Dienstgüteaushandlung bei einer Multimedia-Plattform wie *CINEMA* auf einer höheren

Abstraktionsebene liegt, sind diese Parameter für eine direkte Übernahme nicht geeignet. Vielmehr orientieren sich die mit Sessions assoziierten Parameter an den Eigenschaften und Typen der zu übertragenden und verarbeitenden Datenströme. So sind beispielsweise bei einem Videostrom Bildrate, Bildgröße oder Farbtiefe geeignete Parameter. Zusätzlich ist die Aushandlung von Ende-zu-Ende-Eigenschaften (wie z.B. der Gesamtverzögerung) in komplexen Verarbeitungstopologien mit mehreren Quellen und Senken bei beliebiger interner Vernetzung notwendig.

In *Cinema* ergeben sich somit zwei Ebenen von Dienstgüte-Parametern. Die höhere Abstraktionsebene enthält die vom Typ des Datenstromes (wie Audio, Video) abhängigen Parameter, die zur Gestaltung der Dienstschnittstelle der Multimedia-Plattform benötigt werden. Die tiefere Abstraktionsebene enthält Parameter, die von den Eigenschaften der Ressourcen (wie bsw. dem Netzwerk) bestimmt sind und nicht vom Typ des Stromes abhängen. Mit ihnen arbeiten die individuellen Ressourcenverwalter und nutzen sie u.a. zur Ermittlung der freier Kapazitäten. Zwischen beiden Abstraktionsebenen ist eine Abbildung der Dienstgüte-Parameter notwendig. Diese Abbildung findet neben der Aushandlung von Dienstgüte-Parametern im Ressourcen-Reservierungsprotokoll von *Cinema* (vgl. [BaFi94]) statt. Sie ist von der Verarbeitungsfunktionalität der Komponenten abhängig. Der Abbildungsschritt kann folglich nur von Komponenten selbst durchgeführt werden. Hierzu bieten sie eine Methode an, die aus stromtypabhängigen Dienstgüte-Parametern der Eingabeports stromtypabhängige und -unabhängige Parameter für die jeweiligen Ausgabeports und den Ressourcenbedarf für die Bearbeitung bestimmt.

Dies soll an dem in Abb. 7 dargestellten Ablauf der Ressourcenreservierung näher erläutert werden. Es wird exemplarisch die Abbildung der Bildgröße, als Vertreter eines stromtypabhängigen Parameters, auf die Paketgröße als Vertreter für einen stromtypunabhängigen Parameter betrachtet. Das Delay als Ende-zu-Ende-Parameter wird dabei ebenfalls akkumuliert. Auf den Ressourcenbedarf wird nicht näher eingegangen.



1. QoS-Wunsch: 400-200x300-150
2. QoS-Ausgabe: 300x225, 67500 Bytes, 20 ms
3. QoS-Eingabe: 67500 Bytes, 200 ms
4. QoS-Ausgabe: 67500 Bytes, 100 ms
5. QoS-Eingabe: 300x225
6. QoS-Ausgabe: 20 ms

Als Parameter werden betrachtet: stromorientiert: Bildgröße
stromunabhängig: Paketgröße und Delay

Abb.: 8 Beispiel des Ablaufs einer Ressourcenreservierung

Die Quellenkomponente erhält als Parametervorgabe den Dienstgütewunsch des Klienten (1.). Daraus wählt sie, z.B. auf Grund der von der Komponente tatsächlich unterstützten Bildgrößen, geeignete Parameterwerte aus den vorgegebenen Intervallen aus. Für diese Parameterwerte berechnet sie nun die Paketgröße, z.B. indem sie die Breite mit der Höhe multipliziert (8 Bit Farbtiefe). Zusätzlich berechnet die Komponente die Zeit, die sie für die Bearbeitung benötigt. Die ausgewählten Parameter, die Paketgröße und die Bearbeitungszeit sind das Ergebnis des Aufrufs (2.).

Zur Bearbeitungszeit wird der Scheduling-Overhead (hier: 30 ms) addiert und dann von der max. Verzögerungsvorgabe (z.B. 250 ms) des Klienten abgezogen, was die neue Verzögerungsvorgabe ergibt. Diese wird als Grenzwertvorgabe für die Kommunikationsverbindung im Linkobjekt zusammen mit der Paketgröße an das Linkobjekt übergeben (3.). Das Ergebnis der Verhandlung über die Kommunikationsverbindung wird in Form der max. möglichen Paketgröße und dem für die Übertragung festgestellten Verzögerungswert zurückgegeben (4.). Dieser Verzögerungswert wird wiederum von der Verzögerungsvorgabe abgezogen und somit eine neue Vorgabe ermittelt. Der Senkenkomponente wird die Bildgröße als Eingabeparameter übergeben (5.). Sie berechnet daraus die Zeit, die sie für die Bearbeitung benötigt (6.). Diese Bearbeitungszeit wird zusammen mit dem Scheduling-Overhead von der Verzögerungsvorgabe abgezogen. Bleibt die Verzögerungsvorgabe positiv, so war die Reservierung erfolgreich und es kann in einem zweiten Durchlauf die minimale Verzögerung durch Freigabe von Ressourcen bzw. Einführung zusätzlicher Puffer eingestellt werden.

Die Reservierung und Zuteilung von Ressourcen setzt Verwalter für die einzelnen Betriebsmittel voraus. In der Literatur sind hier beispielsweise das Bandbreitenmanagement in [VHN92], Speicher-Verwalter [McRo93] und Scheduler der CPU [6] beschrieben. In *CINEMA* gibt es derzeit Implementierungen für die Verwaltung der Ressourcen Speicher und CPU. Für letzteres wurde das Rate-Monotonic-Schedulingverfahren erweitert [Bart94]. Es werden Scheduling-Klassen mit unterschiedlichen Garantie-Stufen eingeführt. Die oberste Klasse bekommt eine Verzögerungsgarantie des Schedulings, die kürzer als beim normalen Rate-Monotonic-Verfahren ist. In der anderen gelten die normalen Werte des Rate-Monotonic-Verfahrens, jedoch sind im ungünstigen Fall Fristverletzungen möglich.

5 Zusammenfassung

In diesem Papier wurde die *CINEMA*-Architektur vorgestellt, welche eine Entwicklungsplattform für die Erstellung und Steuerung komplexer, verteilter Multimedia-Anwendungen bereitstellt. *CINEMA* bietet als Dienstschnittstelle eine Menge von Abstraktionen an, über die transparent die darunterliegenden multimedialen Systemdienste angesprochen werden. Mit den Systemdiensten wird die Funktionalität bereitgestellt, um Konfigurationen von Komponenten und Links, die zu Fluß-Graphen verknüpft wurden, in verteilten Systemen zu installieren, um die über Sessions ausgehandelte Dienstgüte durch die Reservierung von Ressourcen zu erbringen und um den Fluß von Dateneinheiten unter Einhaltung von Synchronisationsbeziehungen zu steuern. In der Summe zielen die bereitgestellten Funktionen darauf ab, die Lücke zwischen der von Betriebs- und Transportsystemen bereitgestellten Menge an Diensten und den Anforderungen bei Aufbau und Betrieb verteilter Multimedia-Anwendungen zu verringern. Immer wiederkehrende, automatisierbare Aufgaben, sollen dabei durch die System-Plattform erbracht werden, um einen Großteil der Schwierigkeiten, der Details und der Komplexität beim Entwurf von Multimedia-Anwendungen vor Anwendungsprogrammierern zu verbergen.

Derzeit ist ein erster Prototyp des *CINEMA*-Systems lauffähig. Er erlaubt den Aufbau einfacher, verteilter Anwendungsszenarien unter Verwendung der beschriebenen Abstraktionen innerhalb unseres lokalen Rechnernetzes. Datenströme innerhalb der Anwendungsszenarien können gesteuert und synchronisiert übertragen werden. Die Implementierung zunehmend erweiterter Funktionalität ist derzeit in vollem Gange. Parallel dazu wird weitere Forschungsarbeit geleistet, um beispielsweise den Entwurf von Protokollen zur Ressourcen-Aushandlung und Strom-Synchronisation voranzutreiben, um Multicasting von Daten durch Teilstrombildung zu optimieren oder die Auflösung geschachtelter Komponenten zu ermöglichen.

6 Literatur

[AHS90] David P. Anderson, Ralf Guido Herrtwich, Carl Schaefer. SRP: A Resource Reservation Protocol for Guaranteed-Performance Communication in the Internet. Technischer Bericht No. UCB/CSD 90/562, Computer Science Division (EECS) University of California, Berkeley, CA, 2 1990.

[AnHo91] David P. Anderson, George Homsy. Synchronization Policies and Mechanisms in a Continuous Media I/O Server. *Report No. UCB/CSD 91/617, Computer Science Division (EECS), University of California, Berkeley, CA*, 2 1991.

[Appl91] Apple Computer Inc., Cupertino, CA, USA. *QuickTime Developer's Guide*, 1991.

[BaFi94] Ingo Barth, Walter Fiederer. Session Reservation in *Cinema. in Vorbereitung*, 1994.

[Bart94] Ingo Barth. Extending the Rate Monotonic Scheduling Algorithm to Get Shorter Delays. In *2nd International Workshop on Advanced Teleservices and High-Speed Communication Architectures, IWACA '94, Heidelberg*, S. 104-114, 9 1994.

[BCA+92] G.Blair, G. Coulson, P. Auzimour, L.Hazard, F. Horn, J.B. Stefani. An Integrated Platform and Computational Model for Open Distributed Multimedia Applications. In *3rd International Workshop on Network and Operating Systems Support for Digital Audio and Video*, S. 209-222, 11, 1992.

[BCG+92] Gordon Blair, Geoff Coulson, Francisco Garcia, David Hutchison, Doug Shepherd. Towards new Transport Services to Support Distributed Multimedia Applications. In *4th IEEE ComSoc International Workshop on Multimedia Communications*, 4 1992.

[BuLi91] Dick C. Bulterman, Robert van Liere. Multimedia Synchronisation and UNIX. In *2nd International Workshop on Network and Operating System Support for Digital Audio and Video*, 11 1991.

[CBRS93] Geoff Coulson, Gordon S. Blair, Philippe Robin, Doug Shepherd. Extending the Chorus Micro-Kernel to Support Continuous Media Applications. In *Proceedings of the 4th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, S. 49–60, 11 1993.

[CCGH92] Andrew Campell, Geoff Coulson, Francisco Garcia, David Hutchison. A Continuous Media Transport and Orchestration Service. In *SIGCOMM'92 Conference Proceedings Communications Architectures and Protocols*, S. 99–110, 8 1992.

[DNNR92] Roger B. Dannenberg, Tom Neuendorffer, Joseph M. Newcomer, Dean Rubine. Tactus: Toolkit-Level Support for Synchronized Interactive Multimedia. In *3nd International Workshop on Network and Operating System Support for Digital Audio and Video*, 11 1992.

[EDP92] Julio Escobar, Debra Deutsch, Craig Partridge. Flow Synchronization Protocol. In *IEEE Global Communications Conference*, 12 1992.

[Herr91] Ralf Guido Herrtwich. Time Capsules: An Abstraction for Access to Continuous-Media Data. *The Journal of Real-Time Systems, Kluwer Academic Publishers*, S. 355–376, 3 1991.

[Hewl93] Hewlett-Packard Company, International Business Machines Corporation, SunSoft Inc. *Multimedia System Services, Version 1.0, verfügbar über ftp: ibminet.awdpa.ibm.com*, 7 1993.

[HGA90] George Homsy, Ramesh Govindan, David P. Anderson. Implementation Issues for a Network Continuous–Media I/O Server. *Report No. UCB/CSD 90/597, Computer Science Division (EECS), University of California, Berkeley, CA*, 9 1990.

[HHS+91] D. Hehmann, R. G. Herrtwich, W. Schulz, T. Schuett, R . Steinmetz. Implementing HeiTS: Architecture and Implementation Strategy of the Heidelberg High-Speed Transport System. In *2nd Intl. Workshop on Network and Operating System Support for Digital Audio and Video*, 11 1991.

[IBM92] IBM Corporation. *Multimedia Presentation Manager Programming Reference and Programming Guide 1.0, IBM Form: S41G-2919-00 and S41G-2920-00*, 3 1992.

[IMA92] Interactive Multimedia Association, Compatibility Project, Annapolis, MD, USA. *Request for Technology: Multimedia System Services, Version 2.0, verfügbar über ftp: ibminet.awdpa.ibm.com*, 11 1992.

[KrMa85] Jeff Kramer, Jeff Magee. Dynamic Configuration for Distributed Systems. *IEEE Transaction on Software Engineering*, SE-11(4):424–436, 4 1985.

[LiLa73] C.L. Liu, James W. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of the ACM*, 20(1):46–61, 1 1973.

[McRo93] Brian Craig McKellar, Jan Roos. Buffer Management in Communication Systems. *Technical Report 43.9312, IBM ENC, Heidelberg, Germany*, 1993.

[Mill90] David L. Mills. On the Accuracy and Stability of Clocks Synchronized by the Network Time Protocol in the Internet System. *Computer Communications Review*, S. 65–75, 1990.

[MKSD90] Jeff Magee, Jeff Kramer, Morris Sloman, Naranker Dulay. An Overview of the REX Software Architecture. In *2nd IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems*, 10 1990.

[NaSm92] Klara Nahrstedt, Jonathan M. Smith. The Integrated Media Approach to Networked Multimedia Systems. 2 1992.

[RAA+90] M. Rozier, V. Abrossimov, F. Armand, I. Boule, M. Gien, M. Guillemont, F. Herrmann, C. Kaiser, S. Langlois, P. Léonard, W. Neuhauser. Overview of the Chorus Distributed Operating System. *Chorus Systémes CS/TR-90-25*, 4 1990.

[RBH94] Kurt Rothermel, Ingo Barth, Tobias Helbig. CINEMA - An Architecture for Configurable Distributed Multimedia Applications. In *Architecture and Protocols for High-Speed Networks*, O. Spaniol, A. Danthine, W. Effelsberg. Kluwer Academic Publishers 1994, S. 253-271.

[RoHe94a] Kurt Rothermel, Tobias Helbig. Clock Hierarchies: An Abstraction for Grouping and Controlling Media Streams. *Technical Report 2/94, University of Stuttgart*, 4 1994, (zur Veröffentlichung eingereicht).

[RoHe94b] Kurt Rothermel, Tobias Helbig. ASP: Adaptive Synchronization Protocol for Continuous Data Streams. *Technical Report 14/94, University of Stuttgart*,12 1994.

[RRVK92] P. Venkat Rangan, Srinivas Ramanathan, Harrick M. Vin, Thomas Kaeppner. Media Synchronization in Distributed Multimedia File Systems. In *Proceedings of the 4th IEEE ComSoc Int. Workshop on Multimedia Communications, Monterey (CA), USA*, S. 315–324, 4 1992.

[Shap86] Marc Shapiro. Structure and Encapsulation in Distributed Systems: The Proxy Principle. *Proceedings of the 6th International Conference on Distributed Computer Systems*, 5 1986.

[Sche94] Volker Scheel. Ausgabesynchronisation und Skalierung gespeicherter multimedialer Datenströme *CINEMA*. Diplomarbeit, University of Stuttgart/IPVR, 3 1994.

[Topo90] C. Topolcic. Experimental Internet Stream Protocol, Version 2 (ST-II). *RFC 1190*, 10 1990.

[VHN92] Carsten Vogt, Ralph G. Herrtwich, Ramesh Nagarajan. HeiRAT: The Heidelberg Resource and Administration Technique - Design Philosophy and Goals. *IBM ENC Technical Report No. 43.9213*, 1992.

[ZDE+93] Lixia Zhang, Steve Deering, Debora Estrin, Scott Shanker, Daniel Zappala. RSVP: A New Resource ReSerVation Protocol. *IEEE Network*, 9 1993.