In Tagungsband
GI/ITG-Arbeitstreffen "Verteilte Multimedia Systeme", Stuttgart, 18.-19. Februar 1993, S. 33-47
W. Effelsberg, K. Rothermel (Ed.), Saur-Verlag, 1993

CINEMA: Eine konfigurierbare, integrierte Multimedia-Architektur

Ingo Barth, Gabriel Dermler, Tobias Helbig, Kurt Rothermel, Frank Sembach, Thomas Wahl
Universität Stuttgart
Institut für Parallele und Verteilte Höchstleistungsrechner (IPVR)
Breitwiesenstraße 20-22
W-7000 Stuttgart 80
Email: {Barth, Dermler, Helbig, Rothermel, Sembach, Wahl}@informatik.uni-stuttgart.de

Anwendungen benötigen Unterstützung bei der Manipulation multimedialer Daten, um die besonderen Probleme bezüglich zeitlicher Anforderungen und Ressourcenverwaltung zu bewältigen. Diese zu erbringen, ist die Zielsetzung des *CINEMA*-Projekts. Hierbei wird eine Systemplattform entwickelt, durch die die Trennung der Funktionen zur Bearbeitung multimedialer Daten von Problemen der Ressourcenverwaltung, der Transportmechanismen und von zeitlichen Aspekten ermöglicht wird. Zusätzlich bietet die Architektur Abstraktionen zur Modellierung multimedialer Anwendungen bezüglich der Funktionalität und der Dienstgüte.

1 Einleitung

Die seit längerem verfolgte Integration multimedialer Kommunikation in bestehende Rechner- und Kommunikationssysteme ermöglicht einerseits die Verwendung neuartiger verteilter Applikationen, stellt auf der anderen Seite eine Vielzahl besonderer Anforderungen an das Gesamtsystem. Auf der Applikationsseite sind etliche Szenarien beschrieben, die von multimedialer Unterstützung erheblich profitieren können. Hierzu gehören vor allem Szenarien aus dem Bereich rechnergestützter Kooperation sowie der multimedialen Netzwerk-Dienste.

Auf der Systemseite sind in der Zwischenzeit die technologischen Voraussetzungen für multimediale Kommunikation gegeben, sowohl in lokalen als auch in verteilten Umgebungen. Die Charakteristika multimedialer Daten erfordern jedoch darüber hinaus eine geeignete Organisation des Datenaustausches, die sich von der herkömmlichen wesentlich unterscheidet. So macht die potentiell hohe Verarbeitungslast eine Reservierung und Einplanung von Ressourcen notwendig, um eine gewünschte Kommunikationsqualität konstant einzuhalten. Ferner sind Synchronisationsmechanismen erforderlich, die die zeitlichen Eigenschaften multimedialer

Datenströme sowohl isoliert als auch in Bezug zueinander, falls nötig, wiederherstellen. Nicht zuletzt sind Mittel erforderlich, die es einer Multimedia-Anwendung erlauben, die gewünschte Funktionalität auf verschiedenen Abstraktionsstufen zu spezifizieren.

Cinema (Configurable INtEgrated Multimedia Architecture) ist eine Architektur, die eine integrierte Lösung für diese Aspekte anstrebt. Sie unterscheidet zwischen Anwendungen und der sie unterstützenden Systemplattform. Anwendungen spezifizieren gewünschte die Funktionalität und Kommunikationstopologie, indem sie Funktionalitätskomponenten zu "End-to-End"-Kommunikationspfaden verknüpfen. Diese Komponenten werden dabei entweder von einer Anwendung selber definiert oder aus einem Pool bereits existierender Komponenten entnommen. Durch Schachtelung können aus einer oder mehreren bestehenden Komponenten neue Komponenten definiert werden, die komplexere Funktionalität erbringen und eventuell höheren Abstraktionsstufen entsprechen. Neue Anwendungen können somit auf verfügbare, zunehmend komplexe Bausteine zurückgreifen. Die Beschreibung der Applikationssicht auf *Cinema* bildet den Schwerpunkt dieses Artikels.

Der Artikel ist wie folgt gegliedert. Zuerst wird die Architektur im Überblick eingeführt. Danach werden zwei zentrale Ebenen von *Cinema* besprochen und zueinander in Relation gesetzt. Besonderes Gewicht wird dabei auf die Ebene der Komponenten und der darauf definierten Abstraktionen und Strukturierungsmechanismen gelegt. Im Anschluß wird kurz die nötige Managementumgebung vorgestellt. Der Artikel schließt mit einer Diskussion von *Cinema* und einem Ausblick auf zukünftige Forschungsaktivitäten.

2 *Cinema*-Architektur

Cinema unterscheidet zwei Ebenen. Die eine Ebene enthält die vom Rechnersystem bereitgestellten Einheiten, die Ressourcen, während die andere Ebene die zur Strombearbeitung notwendigen Software-Teile, die Komponenten, beinhaltet. Die Ebenen unterscheiden sich durch ihre Funktionalität. Die Ressourcenebene stellt die Basisfunktionalität eines verteilten Rechnersystems zur Verfügung und hat dabei keinen speziellen Bezug zu multimedialen Daten. Die Komponentenebene dagegen stellt die konkreten Funktionen zur Bearbeitung von Multimediaströmen bereit (siehe Abb. 1).

Die *Cinema* System-Plattform stellt Mechanismen auf beiden Ebenen zur Verfügung. Das Ressourcen-Management sichert u.a. ab, daß Ressourcen den Komponenten so zugeteilt werden, daß diese ihre Aufgaben wie gewünscht erledigen können. Das Konfigurations- und das

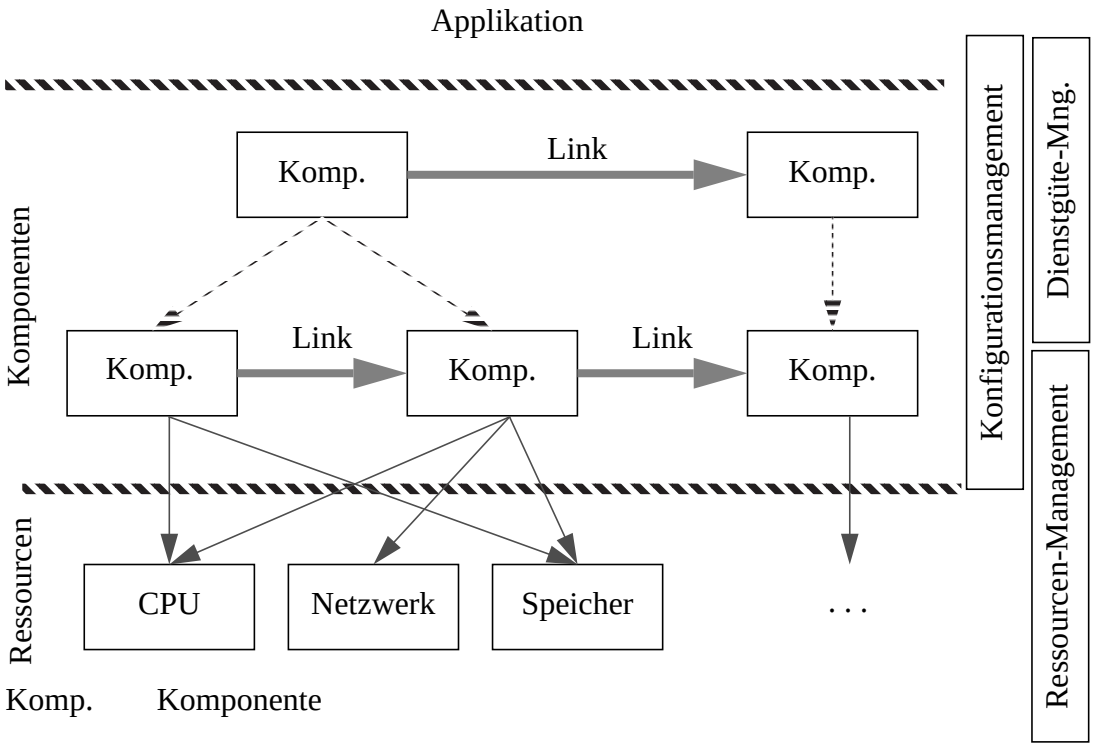


Abbildung 1: Architektur von CINEMA

Dienstgüte-Management sind dagegen auf der Komponentenebene aktiv. Hier führen sie die von einer Anwendung verknüpften Komponenten auf eine Verknüpfung atomarer Basis-Komponenten zurück, die dem Ressourcen-Management bekannt sind. Bei diesem Vorgang wird auch sichergestellt, daß die von der Anwendung spezifizierte Dienstgüte auf Parameter abgebildet wird, die den Basis-Komponenten verständlich sind.

Eine Anwendung sieht nur Komponenten, deren Verknüpfung sie spezifiziert und mit einer Wunsch-Dienstgüte versieht. Alle existierenden Komponenten können im Prinzip von einer Anwendung verwendet werden. Jedoch haben Anwendungen die Möglichkeit, zusammengesetzte Komponenten zu verwenden, die die bereits erfolgte Verknüpfung mehrerer Komponenten in sich verbergen. Auf diese Weise können Anwendungen z.B. Funktionalitäten unterschiedlicher Abstraktionsstufen als Komponenten ansprechen.

2.1 Ressourcenebene

Typisch für Multimedia-Anwendung ist die Anforderung an das System, daß ein gewisser Teil der Gesamtsystemleistung für die Bearbeitung der Multimedia-Datenströme zu Verfügung

gestellt werden muß. Dieser Anteil muß garantiert werden, damit die Multimedia-Anwendung sinnvoll arbeiten kann. Diese Garantie kann nur eingehalten werden, wenn

- sichergestellt ist, daß immer genügend Systemleistung vorhanden ist, oder wenn
- die Systemleistung auf die einzelnen Nutzer aufgeteilt wird, und diese Aufteilung von einer Kontrollinstanz überwacht wird.

Heutzutage ist speziell im Multimedia-Bereich die Systemleistung noch lange nicht als ausreichend zu bezeichnen (vgl. die Datenmengen bei unkomprimiertem Video in VGA-Qualität von 22MByte/s [13]). Da bisher die Leistungsanforderungen der Applikationen an ein System mit den Systemleistungen gestiegen sind, gehen wir davon aus, daß nur durch explizite Aufteilung und Überwachung der Gesamtleistung eine Garantie möglich ist. Die Ressourcen werden von allen strombearbeitenden Einheiten in gleicher, einheitlicher Weise benutzt und durch das Ressourcen-Management verwaltet.

Eine Ressource ist eine Hardware- und/oder Softwareeinheit des Rechnersystems, die eine Grundfunktionalität des Gesamtsystems erbringt. Beispiele für Ressourcen sind die CPU, der Speicher, das Netzwerk und die Peripheriegeräte (Kamera, Mikrophon, Lautsprecher, Bildschirm, Hardware-Komprimierer). Jede Ressource wird durch einen Ressourcen-Manager, der speziell auf sie abgestimmt ist, verwaltet.

Das Ressourcen-Management faßt alle Manager der einzelnen Ressourcen zusammen. Die ressourcenspezifischen Manager sind auf die unterschiedlichen Eigenschaften der Ressourcen eingestellt, teilen die Ressourcen den Benutzern zu und überwachen den Zugriff und die Einhaltung der Garantie. Das Ressourcen-Management sichert dem Benutzer die ausgehandelten Anteile zu und kontrolliert diese auch. Der Benutzer verwendet die Ressourcen als eine Dienstleistung des *Cinema*-Systems.

2.2 Komponentenebene

Aufsetzend auf den Ressourcen der existierenden Rechnersysteme soll mittels des *Cinema*-Projekts die Möglichkeit gegeben werden, multimediale Datenströme in einfacher Weise zu manipulieren. Um nun eine auf die speziellen Anforderungen der Erzeugung, Übertragung, Bearbeitung und Wiedergabe multimedialer Datenströme zugeschnittene Umgebung zu schaffen, wird die Abstraktionsebene der Komponenten eingeführt. Die anfallenden Aufgabenstellungen lassen sich hierbei grob in zwei Gruppen unterteilen:

- Strombearbeitung
  Beispiele: (De-)Komprimierung, Verschachtelung mehrerer Datenströme zu einem einzigen, Mischen von Audioströmen, Erzeugen und Wiedergeben multimedialer Daten, anwendungsspezifische Manipulation von Strömen
- Übertragung von Strömen zwischen Bearbeitungsorten

Angestrebt wird innerhalb des Projektes, daß die Erzeugung von Bearbeitungsinstanzen für Datenströme so einfach wie möglich gemacht wird. Das heißt insbesondere, daß immer wiederkehrende Problemstellungen (Ressourcenverwaltung, Datentransfers, Überwachung von zeitlichen Anforderungen, etc.) so weit wie möglich abgetrennt von der eigentlichen Bearbeitungsfunktionalität und letztendlich durch das zugrundeliegende *CINEMA*-System automatisch entsprechend vorgegebener Spezifikationen eingesetzt und unterhalten werden. Unter die vorzugebenden Spezifikationen fallen die Anforderungen der Anwendung bezüglich der gewünschten

- Konfiguration von Bearbeitungsinstanzen
- Güte des zu erbringenden Dienstes
- Synchronisationsanforderungen

Um diese Zielsetzungen zu erreichen, werden innerhalb der Komponentenebene die folgenden Abstraktionen eingeführt.

Link
: Links dienen zur Verbindung von Komponenten untereinander. Sie stellen die Transportfunktionalität lokal oder über Rechnergrenzen hinweg zur Verfügung und sind somit die Voraussetzung für den Austausch von Datenströmen.

Komponente
: Eine Komponente ist eine Softwareeinheit, in welcher die Manipulation von Strömen stattfindet. Sie verfügt über generische Schnittstellen, um im Gesamtsystem verwaltbar zu sein. Komponenten sind intern normalerweise zeitunabhängig und befassen sich nicht mit Synchronisation. Die Schnittstelle für die Ein- und Ausgabe des durch die Komponente zu bearbeitenden Datenstroms ist einheitlich für den Fall der entfernten und der lokalen Kommunikation. Komponenten können elementar (Basis-Komponenten) oder aus mehreren Komponenten zusammengesetzt sein.

Port
: Ein Port stellt den Dienstzugangspunkt für eine Komponente dar. Die Komponente sieht

nur ihre lokalen Ports, über die sie ihre Daten verschickt und empfängt. Links zwischen Komponenten werden bezeichnet durch die Ports, an denen sie enden.

Im folgenden soll nun erläutert werden, wie die einzelnen Bausteine der Architektur miteinander in Verbindung stehen, auf welche Weise sie die erforderliche Funktionalität erbringen und wie sie sich dabei die anfallenden Aufgaben aufteilen. Dabei wird im ersten Schritt nur die Interaktion zwischen den einzelnen Bausteinen erläutert, ohne dabei darauf einzugehen, wie der Aufbau von Verbindungen vor sich geht oder wie die Abbildungen auf Ressourcen stattfinden.

Komponenten sollen so einfach wie möglich und universell einsetzbar sein. Sie besitzen deshalb nur einheitliche Schnittstellen, die sogenannten Ports als Ein- und Ausgänge für Datentransporte. Über die Ports erhalten sie die für den nächsten Bearbeitungsschritt notwendigen Daten beziehungsweise geben das Ergebnis ihrer Bearbeitung über die Ports nach außen ab (an den Link - siehe unten). Wünschenswert wäre es gewesen, daß eine Komponente nur genau einen Strom verarbeitet, um somit ein Höchstmaß an Einfachheit zu erreichen. Dies ist jedoch nicht haltbar, da es Funktionalitäten gibt, die erzwingen, daß mehrere Datenströme mit sehr unterschiedlichen Charakteristika innerhalb einer Komponente verarbeitet werden. Ein typisches Beispiel hierfür ist ein “Interleaver”, der aus einem Audio- und einem Video-Strom einen einzigen, verschachtelten Strom erzeugt. Folglich gibt es Komponenten, die mehrere Ports besitzen müssen. Dabei kann es sich sowohl um Ausgabe-Ports (bei Datenquellen), um Eingabe-Ports (bei Datensenken) oder um Ein- und Ausgabe-Ports (bei zwischengelagerten Bearbeitungskomponenten wie einem Interleaver) handeln.

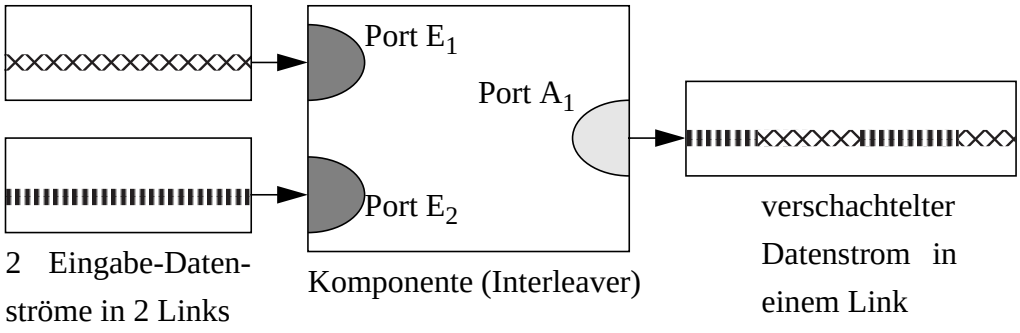


Abbildung 2: Verwendung von Ports an einem Interleaver

Die von einer Komponente via ihrer Ports konsumierten bzw. produzierten Dateneinheiten werden von Links zur nächsten Komponente weitergereicht. Es gibt mehrere Gründe, weshalb Links nicht direkt an Komponenten gekoppelt werden und stattdessen noch Ports zwischengelagert sind. Als Beispiel sei hier die Notwendigkeit genannt, eine 1:*n*-Verbindung aufzubauen, d.h. die von einer Komponente A ausgegebenen Dateneinheiten sollen von *n* verschiedenen Komponenten weiterbearbeitet werden. Die Komponente A muß für ihre Arbeit darüber jedoch nichts wissen. Der Port erbringt hierbei transparent für die Komponente die 1:*n*-Abbildung der Daten auf die verschiedenen Links.

Eine Verbindung zur Übertragung multimedialer Daten ist also auf der Komponentenebene die Kettung der für die Verarbeitung des Datenstroms zuständigen Komponenten mit ihren Ports und den jeweils zwischengelagerten Links. Die Links werden beim Aufbau von Verbindungen durch das *CINEMA*-System transparent eingefügt und automatisch verwaltet. Dies kann geschehen, indem die Lokationsinformationen der zu verbindenden Komponenten ausgewertet werden und der bestmögliche Transportmechanismus entsprechend gewählt wird. Innerhalb von Links verbergen sich alle Aspekte der weiteren Übertragung der Daten, beispielsweise lokal innerhalb eines Rechners über gemeinsamen Speicher, oder aber über das Netz hinweg mittels verschiedener Transportmechanismen. Transportmechanismen und -systeme sind außerhalb von Links nicht sichtbar. Die vollständige Transportfunktionalität wird entsprechend der verfügbaren Möglichkeiten und der Erfordernisse der Konfiguration (lokal oder verteilt) erbracht. Dies hat insbesondere den Vorteil, daß selbst bei einer Umkonfiguration des Systems eine einheitliche Sicht für die Komponenten erhalten bleibt.

Nachdem nun ein Überblick über die Interaktion zwischen den Komponenten und den anderen Bausteinen der Architektur innerhalb der Komponentenebene gegeben wurde, sollen als nächstes die Schnittstellen einer Komponente näher betrachtet werden. Prinzipiell kann hierbei zwischen zwei verschiedenen Arten von Schnittstellen unterschieden werden. Eine Komponente besitzt einerseits generische Schnittstellen, die im Rahmen der *CINEMA*-Umgebung vorgeschrieben sind und die durch das System-Management von *CINEMA* verwendet werden. Teilweise bleiben diese Schnittstellen vor der Anwendung völlig verborgen (beispielsweise jene, über die in Zusammenarbeit mit dem Ressourcen-Management die Belegung der Ressourcen ausgehandelt wird), teilweise werden sie auch durch die Anwendung benutzt (beispielsweise für die Veränderung der Dienstgüte). Andererseits verfügen Komponenten über anwendungsspezifische Schnittstellen, welche auf die jeweilige Funktionalität der Komponente zugeschnitten sind und die dem System-Management verborgen bleiben. Ein Beispiel hierfür stellt ein Lautstärkeregler an einer Audiokomponente dar.

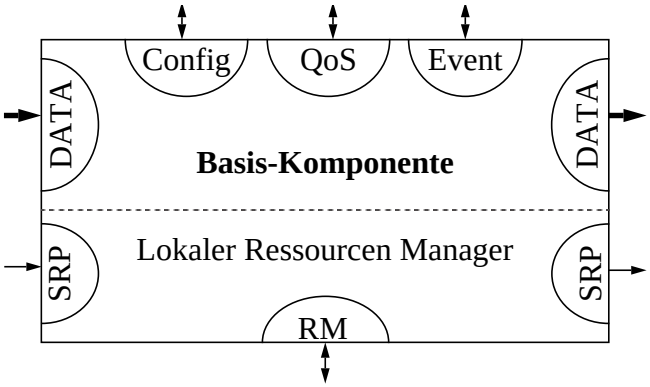


Abbildung 3: Schnittstellen einer Basis-Komponente

Im einzelnen existieren die folgenden Schnittstellen:

DATA: Diese entspricht der oben beschriebenen Schnittstelle zu den Ports, über die die zu bearbeitenden Stromdaten gelesen bzw. geschrieben werden.

SRP: Über diese Schnittstelle läuft das Ressourcen-Reservierungsprotokoll. Bei diesem wird ermittelt, welche Ressourcen reserviert werden müssen, um die Funktionalität einer Komponente mit garantierter Qualität erbringen zu können. Dies kann z.B. mitteln SRP [2] geschehen.

RM: Schnittstelle zum Ressourcen-Manager zur Reservierung von Ressourcen, die durch die Komponente benutzt werden müssen.

QoS: Spezifikation von Dienstgüte-Anforderungen (QoS) durch die Anwendung bzw. Verhandlung über die erbringbare Dienstgüte.

Event: Weitergabe von Ereignissen (Event), beispielsweise über die Verletzung vorgegebener Dienstgüte-Parameter.

Config: Interaktion mit dem Konfigurationsmanagement.

Der Aufbau von Verbindungen zwischen Basis-Komponenten stellt einen zweistufigen Prozeß dar. Im ersten Schritt werden durch das Konfigurationsmanagement die Basis-Komponenten durch Links miteinander verbunden (siehe Abb. 4). Nach diesem Konfigurationsschritt muß zwischen den Basis-Komponenten die QoS der Datenübertragung ausgehandelt werden. Die QoS-Parameter, die zwischen den Basis-Komponenten verhandelt werden, sind Parameter für einen reinen Bitstrom (bzw. Bytestrom). Diese Art von Strömen kann durch LBAP-Parameter (LBAP=Linear Bounded Arrival Process [1]) beschrieben werden. Dabei werden die folgenden QoS-Parameter unterschieden: *Durchsatz* (throughput), *Verzögerung* (delay), *Varianz der Verzögerung* (jitter), *Bündelungscharakteristik* (burstiness) und *Paket-Verlust-Rate* (packet-

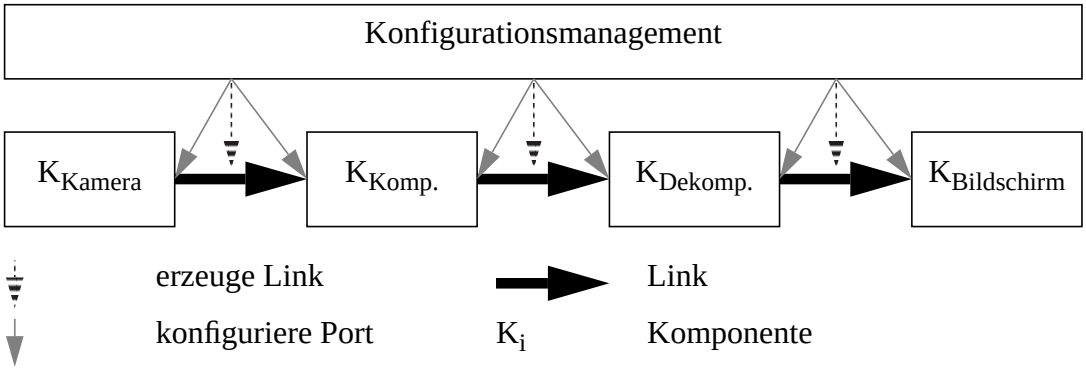


Abbildung 4: Konfiguration von Komponenten und Links

loss-rate). Die Aushandlung der gewünschten Dienstgüte findet durch ein Ressourcen-Reservierungsprotokoll nach der Konfiguration statt. Eine solche Aushandlung kann auch während der Laufzeit zur Anpassung von Parametern verwendet werden. Mit dem Aushandeln der Dienstgüte geht die Reservierung von Ressourcen einher.

Damit eine Komponente weiß, welche Dienstgüte sie für die Übertragung der Daten fordern muß, erhält sie über die QoS-Schnittstelle eine Beschreibung der für sie geltenden Dienstgüte. Diese Beschreibung ist abhängig von der Funktion der Komponente. Eine Videoquelle (Kamera-Komponente) kann die QoS-Parameter Bildgröße und Bildrate haben, während eine Komprimiererkomponente die QoS-Parameter Bildgröße und Kompressionsfaktor besitzen kann. Jede Komponente berechnet aus den eingehenden LBAP-Parametern der SRP-Schnittstelle und der QoS-Parameter der QoS-Schnittstelle die benötigten Ressourcen und die angepaßten LBAP-Parameter für den Ausgangsstrom (siehe Abb. 5). Eine Quelle berechnet ihre ausgehenden LBAP-Parameter natürlich nur aus den QoS-Parametern der QoS-Schnittstelle. Senken hingegen benutzen die ausgehenden LBAP-Werte ($LBAP_a$) zum Einleiten der Freigabe der überschüssig reservierten Ressourcen (Freigabephase im SRP).

Zusammengesetzte Komponenten

Die vielfältige Funktionalität innerhalb einer Multimediaanwendung wird nicht von einer einzigen Komponente zur Verfügung gestellt. Durch das Verbinden von Komponenten entstehen Komponentengeflechte, die in ihrer Summe eine höhere Funktionalität erbringen. Ein einfaches Beispiel hierfür ist eine Videoquelle (Kamera), der ein Komprimierer, ein Dekomprimierer und eine Videosenke (Bildschirm) folgen. Vor allem bei komplexen Anordnungen erscheint es nun sinnvoll, zusammengehörige Komponenten zu einer einzigen Einheit zusammenzufassen, die wiederum wie eine Komponente angesprochen werden kann. Dadurch hat der Nutzer

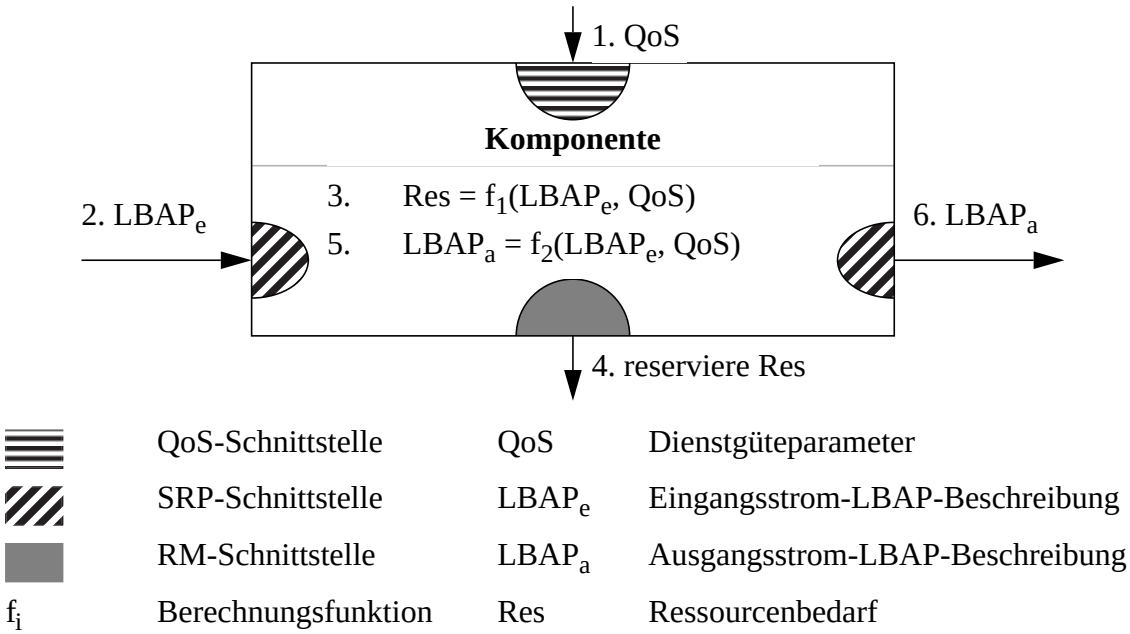


Abbildung 5: Aushandlung der Dienstgüte

der *CINEMA*-Architektur die Möglichkeit, von der Komplexität einzelner, untereinander verbundener Basis-Komponenten zu abstrahieren, unnötige Details zu kapseln und höhere Funktionalität in einfach handhabbarer Form zur Verfügung zu stellen. Das eröffnet die folgenden Möglichkeiten:

Adressierung zusammengefaßter Einzelkomponenten

Abstraktion der Funktionalität

Durch Zusammensetzen einfacher Komponenten zu komplexen erhält man Funktionseinheiten mit höherer Funktionalität. Beispielsweise wird aus einer Kamera-Komponente und einer Komprimierer-Komponente eine zusammengesetzte Komponente, die einen komprimierten Videostrom liefert.

Abstraktion der Dienstgüte

Auf unterster Ebene verbergen sich hinter der Dienstgüte Beschreibungen der Eigenschaften von Bit- oder Byteströmen. Zur Unterstützung der Sicht der Anwendung ist es notwendig, andere, höhere Dienstgüte-Parameter verarbeiten und abbilden zu können. Beispielsweise kann von einem Videostrom gesprochen werden, der eine Bildrate und Bildgröße besitzt. Dies wird dann auf die entsprechende Anzahl von Bytes pro Zeiteinheit abgebildet.

Kapselung von Fehlerbehandlungsmechanismen

Innerhalb zusammengefaßter Komponenten kann abgestuft auf das Eintreten von Fehlersi-

tuationen reagiert werden, wodurch das Auftreten von Fehlern u.U. nicht bis zur Anwendung weitergereicht werden muß.

Bei der Instanziierung einer vordefinierten zusammengesetzten Komponente wird das in ihr verborgene Geflecht von Komponenten aufgebaut. Dabei wird auch entschieden, wo diejenigen Komponenten plaziert werden, für die keine Lokation vorgeschrieben wurde. Kriterien hierbei können kurze Übertragungswege, aber auch der Grad der Auslastung der in Frage kommenden Rechner sein. Im Gegensatz zu Basis-Komponenten können zusammengesetzte Komponenten verteilt sein. Dabei wird die Verteilung nach außen jedoch verborgen, da eine Komponente eine adressierbare Einheit bildet und somit alle Bestandteile als Ganzes ansprechbar sind. Die Adressierung erfolgt über einen eindeutigen Namen. Relativ zu diesem Namen können nun die Ein- und Ausgänge (Ports) bezeichnet werden. Die Konfigurationsinformation, die den Aufbau von zusammengesetzten Komponenten beschreibt, wird durch den Konfigurationsmanager verwaltet.

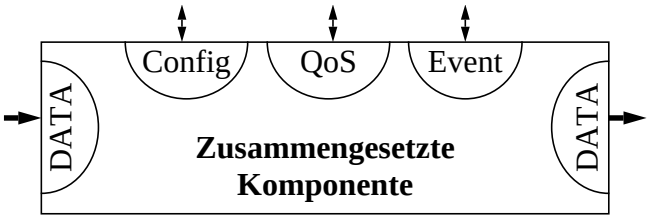
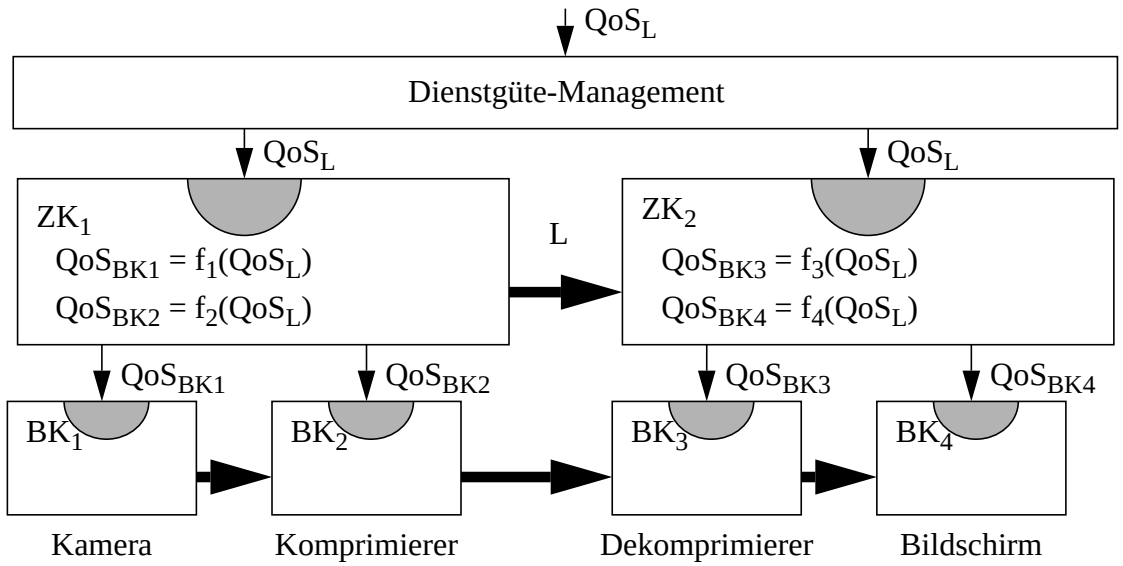


Abbildung 6: Schnittstellen einer zusammengesetzten Komponente

Die Schnittstellen einer zusammengesetzten Komponente entsprechen aus der Sicht der Anwendung prinzipiell denen der Basis-Komponenten. Aus System-Sicht entfallen jedoch (die für die Anwendung unsichtbaren) Schnittstellen zur Durchführung der Ressourcen-Reservierungen. Das ist dadurch zu erklären, daß eine zusammengesetzte Komponente letztendlich in Basis-Komponenten zerlegbar ist, welche dann über diese Schnittstellen und die entsprechende Funktionalität verfügen.

Im folgenden wird die Behandlung von QoS-Parametern zwischen zusammengesetzten Komponenten (Z-Komponenten) beschrieben. Dadurch ergibt sich die Möglichkeit zur Abstraktion über die QoS-Parameter. Am Beispiel einer Videoverbindung mit Komprimierung soll die Umsetzung der QoS-Parameter auf die QoS-Parameter der Basis-Komponenten beschrieben werden (siehe Abb. 7).



	QoS-Schnittstelle	l_i	Link
ZK_i	zusammengesetzte Komponente	QoS_{li}	Dienstgüte von Link l_i
BK_i	Basis-Komponente		

Abbildung 7: Beispiel einer komprimierten Videoverbindung

In diesem Beispiel wird von der Anwendung eine QoS-Beschreibung der Form (Bildrate, Bildgröße, Bildqualität) übergeben. Durch das Dienstgüte-Management werden diese strombezogenen Informationen an die beiden Teilnehmer der Verbindung, die Z-Komponenten ZK_1 und ZK_2 über deren QoS-Schnittstelle gegeben. Diese Z-Komponenten berechnen nun über die Funktionen f_1, f_2, f_3 und f_4 die QoS-Parameter der Komponenten BK_1, BK_2, BK_3 und BK_4. Die Basis-Komponente BK_1 erhält die Informationen Bildrate und Bildgröße (QoS_{BK1}). Die Basis-Komponente BK_2 erhält die Informationen Bildgröße und Bildqualität (QoS_{BK2}). Die Basis-Komponente BK_3 erhält wie die Basis-Komponente BK_4 die Informationen Bildgröße und Bildrate (QoS_{BK3} und QoS_{BK4}).

Dieser Vorgang der QoS-Abbildung ist im beschriebenen Beispiel nur einstufig. Es ist aber durchaus denkbar, daß noch weitere Abstraktionsstufen für die Dienstgüte bestehen, die in mehreren gleichartigen Schritten ausgeführt werden.

2.3 Managementumgebung

Die Managementumgebung von *CINEMA* sieht folgende Funktionsbereiche vor:

Konfigurationsmanagement

Das Konfigurationsmanagement setzt nach den Angaben der Applikation oder der zusammengesetzten Komponenten die Komponenten zusammen und verbindet diese über die Links. Das Konfigurationsmanagement stellt Mechanismen zur Konfiguration und zur Speicherung von Konfigurationsinformationen zur Verfügung. Diese Informationen können von anderen Managementfunktionen abgerufen werden. Es verwirklicht keine Strategien für einzelne Konfigurationsaspekte (z.B. für die Verteilung der Komponenten auf Knoten), sondern stellt nur allgemeine Mechanismen bereit.

Ressourcen-Management

Das Ressourcen-Management verhandelt die Ressourcennutzung mit den Komponenten und überwacht die Zuteilung der Ressourcen an die Komponenten. Es setzt sich aus Ressource-Managern zusammen, die die einzelnen Ressourcen entsprechend der spezifischen Anforderungen verwalten. Durch die Ressourcen-Management-Schnittstelle (siehe Abb. 8)

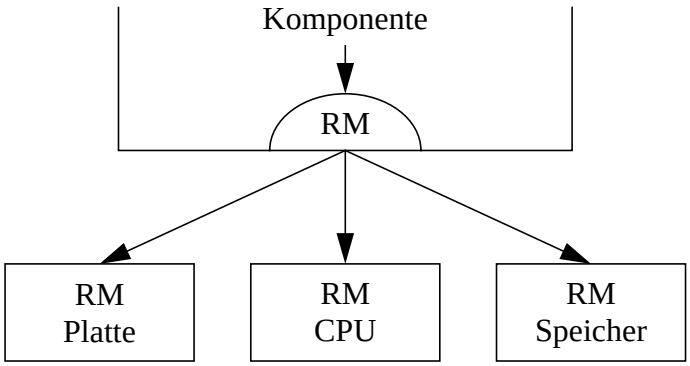


Abbildung 8: Schnittstelle zum Ressourcen-Management

der Komponenten wird der Komponente eine Sicht auf das übergreifende Ressourcen-Management zur Verfügung gestellt. Das Ressourcen-Management stellt das verwaltende Bindeglied zwischen der Komponentenebene und der Ressourcenebene dar.

Dienstgüte-Management

Wie schon in obigem Beispiel beschrieben, stellt das Dienstgüte-Management Mechanismen zur Umwandlung der Dienstgüte-Parameter eines Links in Dienstgüte-Parameter für die Komponenten zur Verfügung.

3 Zusammenfassung und Ausblick

In diesem Papier wurden die Architektur des *Cinema*-Projekts und die darin enthaltenen Abstraktionen vorgestellt. Die Zielsetzung des Projektes ist es, eine Umgebung zur Verfügung zu stellen, mit deren Hilfe multimediale Datenströme unter verschiedensten Anforderungen durch Applikationen gehandhabt werden können. Zu diesem Zweck werden die eigentlichen Bearbeitungsfunktionen der Ströme getrennt von stets wiederkehrenden Anforderungen bezüglich der Ressourcenverwaltung und der Transportfunktionalität. Hierzu wurden zwei Ebenen eingeführt, die Ressourcen- und die Komponentenebene. Die erstere bietet die Reservierung von Grundfunktionalitäten (CPU, Speicher, . . .) von Rechnern an, wodurch eine bestimmte Bearbeitungsqualität garantiert werden kann. Die zweite beinhaltet die eigentliche Funktionalität zur Übertragung und Verarbeitung der Multimedia-Daten. Diese wird erbracht durch Komponenten (Basis-Komponenten und zusammengesetzte Komponenten) als Verarbeitugsinstanzen, Links als Übertragungsinstanzen und Ports als Bindeglieder zwischen beiden.

Neben dem *Cinema*-Projekt gibt es andere Projekte mit ähnlichen Zielsetzungen. Teilweise beschäftigen sich dabei die bestehenden Ansätze mit isolierten Aspekten wie Synchronisation von Strömen ([5], [4], [11]), Konfiguration von Präsentationen ([3], [7]), Garantie von Qualitätsparametern ([10]), Adaption der Dienstqualität ([6]). Bei anderen wird der Schwerpunkt auf die Erbringung von Transport- und Kommunikationsfunktionalität (beispielsweise bei HeiTS [9] oder DASH [1]) oder auf die Unterstützung der Präsentation multimedialer Daten an einer Ausgabelokation (Quicktime [8]) gelegt.

Die Schwerpunkte unserer weiteren Forschung richten sich einerseits auf den Entwurf von Synchronisationsmechanismen für verschiedenste Arten von Strömen, welche über eine einfache, high-level Synchronisationsschnittstelle aktivierbar sein sollen. Als Basis hierfür ist ein Referenzsystem aus logischen Uhren [12] geplant. Andererseits untersuchen wir, welche Auswirkungen die Gruppierung von Strömen (Zusammenfassung verschiedener Ströme zu einem einzigen Strom) auf die Spezifikation der QoS-Parameter und die Synchronisationbeziehungen hat. Außerdem wird die Unterstützung höherer Abstraktionsebenen durch die Bildung zusammengesetzter Komponenten betrachtet, sowie das Management der Ressourcen und solche Aspekte, wie durch graduelle Senkung der Dienstgüte auf veränderte Systembedingungen reagiert werden kann.

Literaturverzeichnis

[1] David P. Anderson, Ralf Guido Herrtwich. Resource Management for Digital Audio and Video, 1 1990.

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

[3] Rusti Baker, Alan Downing, Kate Finn, Earl Renninson. Multimedia Processing Model for a Distributed Multimedia I/O System. In *3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, S. 154–165, 11 1992.

[4] Dick C. A. Bulterman. Synchronization of Multi-Sourced Multimedia Data for Heterogeneous Target Systems. In *3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, S. 110–120, 11 1992.

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

[6] Stephen T.-C. Chou, Hideyuki Tokuda. System Support for Dynamic QOS Control of Continuous Media Communication. In *3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, S. 322–327, 11 1992.

[7] Roger B. Dannenberg, Tom Neuendorffer, Joseph M. Newcomer, Dean Rubine. Tactus: Toolkit-Level Support for Synchronized Interactive Multimedia. In *3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, S. 264–275, 11 1992.

[8] Erfert Fenton. Quick Time Tools. *MACWORLD*, S. 240–244, 2 1992.

[9] 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.

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

[11] T.D.C. Little, F. Kao. An Intermedia Skew Control System for Multimedia Data Presentation. In *3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, 11 1992.

[12] Kurt Rothermel, Gabriel Dermler. Synchronization in Joint-Viewing Environments. In *3nd International Workshop on Network and Operating System Support for Digital Audio and Video*, 11 1992.

[13] Ralf Steinmetz, Ralf Guido Herrtwich. Integrierte verteilte Multimedia-Systeme. *Informatik Spektrum*, 14(5):249–260, 10 1991.