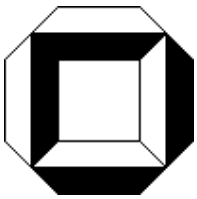
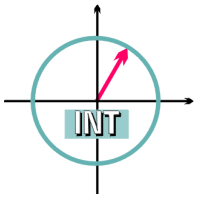


Universität Karlsruhe
Institut Für Nachrichtentechnik
Prof. Dr.rer.nat. Friedrich Jondral




# Projektbereich III.5.1

## Hochleistungsnetzwerk in und zwischen Studios:

## *Client-Server-Simulationsmodelle und Netzwerkoptimierung*

Verfasser:

**Dipl.-Ing. Ulrich Kaage**


29. Dezember 2000

# I Inhaltsverzeichnis

**1 EINFÜHRUNG ..... 6**

**2 KONZEPTION ..... 8**

2.1 Problembeschreibung ..... 9
2.2 Mediendaten ..... 11
*2.2.1 MPEG-Videostandard ..... 12*
*2.2.2 Datenblöcke ..... 15*
2.3 Beschreibung der Verfahren ..... 16
*2.3.1 Festplatten-Scheduling ..... 19*
*2.3.2 Datenstrom ..... 23*
2.4 Verfügbarkeitsprüfung ..... 27
*2.4.1 Deterministischer Ansatz ..... 29*
*2.4.2 Statistischer Ansatz ..... 30*
2.5 Strategie ..... 34

**3 MODELLIERUNG DES MEDIENSERVERS ..... 36**

3.1 Ansätze ..... 36
3.2 Beschreibung des Clients ..... 37
*3.2.1 Videoeinheit ..... 38*
*3.2.2 Übertragungseinheit ..... 39*
3.3 Beschreibung des Medienservers ..... 39
*3.3.1 Network-Manager (Nwk-Mgr) ..... 40*
*3.3.2 Client-Manager(Clt-Mgr) ..... 41*
*3.3.3 Directory-Manager (Dir-Mgr) ..... 42*
*3.3.4 Device-Manager (Dev-Mgr) ..... 43*
*3.3.5 Memory Manager (Mem-Mgr) ..... 43*
*3.3.6 Admission-Controller (AC) ..... 43*
*3.3.7 Operating-System (OS) ..... 45*

**4 IMPLEMENTIERUNG DES MODELLS MIT OMNET++ ..... 46**

4.1 OMNeT++ ..... 46
4.2 Annahmen und Einschränkungen ..... 47
4.3 Simulation der Mediendaten ..... 50
4.4 Datenstrom ..... 53
4.5 Simulation des Clients ..... 54
*4.5.1 Steuerdatei ..... 54*
*4.5.2 Client-Modul ..... 58*
4.6 Die Simulation des Medienservers ..... 59
*4.6.1 Prozesse und Unterprozesse ..... 59*
*4.6.2 Simulation der Peripherie ..... 60*
*4.6.3 Name der Ereignisse ..... 61*
*4.6.4 Abspielen ..... 63*
*4.6.5 Aufnahme ..... 65*
4.7 Parameter ..... 67
4.8 Statistik ..... 69
*4.8.1 Szenario ..... 69*
*4.8.2 Statistikdaten ..... 72*
4.9 Zusammenfassung und Ausblick ..... 76

**5 ANHANG ..... 77**

5.1 Schnittstellen ..... 77
*5.1.1 Schnittstellen zwischen Client und Medienserver ..... 77*
*5.1.2 Message zwischen Modulen des Medienservers ..... 79*
*5.1.3 Schnittstelle zum Operating-System (OS) ..... 86*
5.2 Literaturverzeichnis ..... 89

# II Abbildungsverzeichnis

Abbildung 2-1 Die MPEG-Bildtypen ..... 14
Abbildung 2-2 SCAN-Festplatten-Scheduling ..... 20
Abbildung 2-3 SCAN-EDF-Festplatten-Scheduling ..... 22
Abbildung 2-4 Kumulative Datenlieferung und Datennutzung ..... 24
Abbildung 2-5 Datenmengen des Datenstromes mit CTL-Verfahren ..... 25
Abbildung 2-6 Datenmengen des Datenstromes mit GCTL-Verfahren ..... 27
Abbildung 2-7 Beispiel für ein Histogramm ..... 31
Abbildung 2-8 Grafische Darstellung der Faltung zweier Histogramme ..... 32
Abbildung 2-9 Histogramm des Grenzbereichs ..... 33
Abbildung 3-1 Komponenten des Medienservers und seine Schnittstellen ..... 40
Abbildung 4-1 Die Abläufe für das Abspielen eines Videofilmes ..... 64
Abbildung 4-2 Die Abläufe bei der Aufnahme eines Videofilmes ..... 66
Abbildung 4-3 Das Szenario der Simulation ..... 70
Abbildung 4-4 Die Größe der Datenpakete ..... 72
Abbildung 4-5 Die Ankunftzeit der Datenpakete ..... 73
Abbildung 4-6 Die Zugriffe auf das Bandarchiv ..... 74
Abbildung 4-7 Die Zugriffe auf die Festplatte ..... 75

# 1 Einführung

In den letzten Jahren ist die Entwicklung von Multimediasystemen so weit fortgeschritten, dass bereits dem Normalverbraucher Zugang zu umfangreichen Multimedia-Anwendungen ermöglicht wird. Der Bedarf an Multimedia-Anwendungen wächst ständig und die Systeme, die täglich neu auf den Markt kommen, versuchen, diesem Bedarf gerecht zu werden. Im professionellen Bereich, wie zum Beispiel in Fernsehstudios, sind die Ansprüche besonders hoch. Der Firmenbereich ‚Digitale Medien' der Firma TECMATH entwickelt für die Fernsehstudios spezielle Medienserver-Systeme, mit denen die speziellen Bedürfnisse der Fernsehstudios befriedigt werden sollen. In Fernsehstudios werden Medienserver für die Verarbeitung großer Mengen von Filmmaterial eingesetzt. Das digitalisierte Filmmaterial wird in Archivsystemen und Massenspeichern gelagert und mit einem Datenbanksystem katalogisiert. Mit Hilfe dieser Systeme können Filmmaterialien nach einzelnen Szenenausschnitten, Themen und Inhalt gesucht werden. Die gefundenen Materialien können in digitaler Form abgespielt bzw. verarbeitet und abgespeichert werden.

Die Ergebnisse mehrerer vorangegangener Arbeiten wurden als Grundlage für diese Arbeit genutzt. In diesem Zusammenhang sind folgende Arbeiten zu erwähnen:

- Client/Server-Modelle: Diplomarbeit von Herrn Faller [Fal99]
- Clients: Studienarbeit von Herrn Kaltenbach [Kal99]
- SCSI-Bus: Diplomarbeit von Herrn Barquero [Bar99]
- Tape Drive: Diplomarbeit von Herrn Ferreira [Fer99]

- TCP/IP LAN-Adapter: Diplomarbeit von Herrn Böhm [Böh99]

Erfahrungsgemäß stellt der Serverzugriff auf die Peripheriekomponenten, wie Plattenarrays und Bandarchive, einen zeitkritischen Systemablauf dar. Diese Arbeit soll geeignete Scheduling-Algorithmen eines Medienservers untersuchen und erarbeiten. Das Ergebnis der theoretische Arbeit soll in einem Simulationsmodell bewertet werden. Die Modellierung und Simulation erfolgt mit dem objektorientierten diskreten ereignisbasierten Simulationswerkzeug OMNeT++.

Der für diesen Meilenstein geplante Abschnitt zur Netzwerkoptimierung befindet sich derzeit noch in Bearbeitung und wird in den nächsten Meilenstein „Erarbeitung von Optimierungsstrategien“ einfließen.

# 2 Konzeption

Die Architektur des Medienservers ähnelt einem normalen Dateiserver und besteht meist aus einer Anzahl von Clients, die sporadisch Anfragen an den Server schicken und daraufhin Daten vom Server erwarten. Die Clients und der Medienserver sind über ein Datennetzwerk miteinander Verbunden und alle Kommunikation wird über das Netzwerk abgewickelt. Der Unterschied zwischen den beiden Servertypen sind Datenart, Datenmengen und die Art und Weise, wie die Daten vom Server zum Client verschickt werden.

Bei Mediendaten sind die einzelnen Datenpakete meist sehr groß. Die Verbindung besteht während eines langen Zeitintervalls und die Daten sollen als kontinuierlicher Datenstrom übertragen werden. Diese Struktur kann mit Hilfe von Datenflüssen beschrieben werden.

Für den Client sollen Funktionalitäten wie Abspielen, Aufnahme, vorwärts und rückwärts Spulen, Pause, Stop, zur Verfügung gestellt werden, was dem Arbeitsablauf redaktioneller Client-Systeme nahe kommt. Diese Funktionalitäten beeinflussen die Flussrichtung und die Flussgeschwindigkeit der Daten zwischen Client und Server.

Die Aufgabe des Medienservers ist es, die Speicherung der Mediendaten und die Organisation optimal zu gestalten, so dass, je nach Wunsch des Clients, flexibel reagiert und die Datenflüsse optimal geregelt werden.

Um den hohen Anforderungen gerecht zu werden, müssen bei der Konzeption folgende Punkte betrachtet werden:

- Speicherung der Daten

Dies umfasst die Codierung der Daten, die Speicher- und Zugriffseinheiten, sowie ihre Optimierung.

- Organisation und Strategie

Dies umfasst die interne Organisation sowie die Strategie des Medienservers. Der Medienserver ist der Vermittler zwischen den Speichermedien und den Clients. Die Organisation des Medienservers ist stark von der Organisation der Speichermedien und der Clients abhängig. Die Strategie des Scheduling-Algorithmusses ist von der Speicherung der Mediendaten auf den Speichermedien und von der Form des Datenempfangs des Clients abhängig.

- Serviceumfang

Der einfachste Service beschränkt sich auf das reine Abspielen der Mediendaten. Zusätzlich können Services wie vor- und rückwärts Spulen und die Aufnahme von Mediendaten vorgesehen werden. Abhängig vom Serviceumfang, den der Medienserver dem Client anbieten will, muss der Medienserver entsprechend konzipiert sein.

- Servicequalität

Für den angebotenen Serviceumfang soll der Medienserver die Servicequalität garantieren. Zur Servicequalität gehören die Reaktionszeit des Medienservers auf die Anfrage des Clients und die Qualität der Abspiel- bzw. Aufnahmedaten Diese lässt sich in Form von Datenfehlern, Bildfehlern oder Jittern darstellen und durch Fehlerwahrscheinlichkeiten berechnen. Als weitere Servicequalität können der erforderlichen Speicher- und Ressourcenverbrauch des Client genannt werden.

## 2.1 Problembeschreibung

Der Medienserver verwaltet die digitalisierten, komprimierten Daten, die auf sekundäre Speicher, wie Festplatten, oder tertiäre Speicher, wie Bandarchive, gespeichert sind. Im Vergleich zum tertiären Speicher erlaubt der sekundäre Speicher schnelleren Zugriffe durch kürzere Suchzeit. Auf der anderen Seite bietet der

tertiäre Speicher große Speicherkapazität mit niedrigen Kosten. Die Konzeption das Medienservers unterscheidet sich stark von der Konzeption eines konventionellen Dateiservers. Wegen der Echtzeitanforderung der Mediendaten muss die Konzeption des Medienservers für den Zugriff auf zeitkritische Daten gerecht werden.

Die Mediendaten können wegen ihre Größe zunächst nicht vollständig zum Client übertragen und dann anschließend abgespielt werden. Dieser Ansatz würde sehr große Zwischenspeicher auf dem Client-Rechner und lange Totzeiten, bis das komplette Datenmaterial übertragen ist, erfordern. Die Verzögerungszeit für den Start ist dementsprechend recht lang und ist für den professionelle Gebrauch nicht akzeptabel. Da der Medienserver viele Clients gleichzeitig bedienen soll, würde die Datenübertragung eines gesamten Filmes an einen Client die Ressourcen des Medienservers extrem stark belasten, obwohl nicht die alle Daten sofort gebraucht werden. Gleichzeitig müssten die anderen Clients auf ihre Daten warten.

Die angestrebte Lösung wäre, die Daten in Form kontinuierlicher Datenströme zum Client zu übertragen, d.h. die Datenmengen werden in Paketen zerteilt und diese je nach Bedarf zu liefern. Dies hat den Vorteil, dass der Client-Rechner keinen großen Zwischenspeicher braucht, um den gesamten Film vorübergehend zu speichern, gleichzeitig kann der Server mit noch freien Ressourcen andere Clients bedienen. Der Speicherbedarf auf dem Server wie auf den Clients reduziert sich somit auf die Speicherung der Datenpakete für das nächste Zeitintervall. Zusätzlich würde dieser Ansatz die Verzögerungszeit der Starts auf ein Minimum reduziert.

Um diese Ansatz zu realisieren, muss die Tatsache berücksichtigt werden, dass die Mediendaten in der Regel komprimiert sind. Je nach Komprimierungsverfahren schwankt der Komprimierungsfaktor einzelner Datenpakete stark. Dies führt dazu, dass die Datenmengen im Datenstrom starken Schwankungen unterliegen, was wiederum zu einem schwankendem Ressourcenverbrauch führt. Solche Da-

tenströme werden in der englischen Fachliteratur als „variable bit rate stream (VBR stream) “ bezeichnet.

Im folgenden Abschnitt werden die Mediendaten in Form von Datenströmen genauer betrachtet und verschiedene Ansätze, sowie Steuerungs- und Optimierungsverfahren auf Anwendbarkeit hin analysiert.

## 2.2 Mediendaten

Heutige Multimediadaten können aus verschieden Informationsarten wie Ton, Grafik, Bild, Animation, Audio und Video oder Kombinationen daraus bestehen. Die Datenmengen dieser Mediendaten sind immens und benötigen zur Speicherung, Weiterverarbeitung, Übertragung und zum Abspielen sehr große Systemressourcen. Damit sich die Systemressourcen der digitalen Mediensysteme in Grenze halten, müssen die Datenmengen für die digitale Übertragung und die Speicherung entsprechend codiert und komprimiert werden. D. h., die Mediendaten werden durch einen Codierer geschickt und mit einem geeigneten Kompressionsverfahren komprimiert. Diese komprimierten Daten können gespeichert und übertragen werden. Für das Abspielen und Weiterverarbeitung sie jedoch dekomprimiert werden, um die ursprüngliche Information wieder herzustellen.

Für die Codierung und Decodierung werden die Daten mit geeigneten Verfahren codiert beziehungsweise decodiert. Dafür steht meist spezielle Hardware zur Verfügung. Die Daten des Videosignals werden nach der Codierung zu einem konstanten Datenstrom mit hoher Datenrate umgewandelt. Der Datenstrom wird daraufhin komprimiert. Bei der Codierung und Decodierung entstehen meist keine Verluste. Anders sieht es bei der Komprimierung aus. Es gibt verlustfreie und verlustbehaftete Kompressionsverfahren. Leider können verlustfreie Kompressionsverfahren nicht die gewünschte Reduzierung bringen, wie es für viele Übertragungsmedien mit geringer Bandbreite gefordert wird. Deshalb werden in der Regel verlustbehaftete Kompressionsverfahren eingesetzt.

Die Kompressionsverfahren versuchen die Seh- bzw. Höreigenschaften der menschlich Augen bzw. Ohren auszunutzen, in dem sie die weniger empfindlichen Bereichen stärker Komprimieren und damit mehr Verluste in Kauf nehmen. Zum Beispiel sind leise Töne für das Ohr unter bestimmten Randbedingungen gar nicht hörbar, wenn laute Töne gleichzeitig abgespielt werden. Und das menschliche Auge ist für Farbänderung beispielsweise weniger empfindlich als für Helligkeitsveränderung. Bei Bildfolgen ist es häufig der Fall, dass sich bei aufeinanderfolgenden Bilden nicht der komplette Bildinhalt ändert, hier müssen nur die Differenzen zwischen den Bilden für die Codierung visualisiert werden[Bar98]. Die Kompressionsverfahren versuchen die Reduktionsrate zu erhöhen, indem neben den redundanten auch die irrelevanten Information entfernt werden. Die Datenmenge wird zwar durch das Kompressionsverfahren auf das Nötigste reduziert, trotzdem bedeutet das Abspielen eines Videofilms mit 25 Bildern pro Sekunde bei eine Auflösung von nur $352 \times 288$ Pixel pro Bild und einer Farbtiefe von 8 Bit pro Pixel ein Datenaufkommen von über zwei Megabyte pro Sekunde [Mil95].

### 2.2.1 MPEG-Videostandard

Bei den meisten Multimedia-Anwendungen wird der MPEG-Videostandard benutzt. Deshalb wird in diesem Abschnitt der MPEG-Videostandard genauer betrachtet und als Beispiel für die Mediendaten verwendet. Allerdings wird bei dieser Betrachtung nur Augenmerk auf Punkte wie, Bandbreite, Speichereinheit, Eigenschaften bei der Codierung und der Decodierung sowie das Abspielen, die für die Aufstellung der Scheduling-Strategie von Bedeutung ist, gelegt. Weitere ausführliche Untersuchungen werden auf die von H. Barquero [Bar99] erstellte Studienarbeit und andere Literature verwiesen.

Der MPEG-Videostandard (Motion Picture Experts Group) wurde 1993 von der International Standards Organization (ISO) konzipiert und entwickelt. Dieser

Standard bietet sich für ein breites Spektrum von Anwendungen an. Die Datenraten können von 1 bis 1,5 Mbits/s bei MPEG-I und von 2 bis 80 Mbits/s bei MPEG-II schwanken. Die Kompressionsrate liegt bei ca. 200:1 [Fur95].

Die MPEG-Kompression versucht neben den Daten innerhalb eines Bildes auch die zeitliche Redundanz aufeinanderfolgender Bilder der Bildsequenz für die Komprimierung zu nutzen. Dieses Verfahren wird Prädiktion (preditive coding) genannt. Untersuchungen zeigen, dass nur Teile der Daten der hintereinander folgenden Bilder verändert werden. Aus diesem Grund müssen nur die Differenzen zum neuen Bild gespeichert werden.

Videodaten bestehen aus einer Folge von Bilden, um gleichzeitig die hohe Kompressionsrate zu erreichen und den wahlfreien Zugriff auf die einzelnen Videosequenzen zu ermöglichen, werden die Bilder in Bildgruppen (engl. ‚Group of Pictures' , GoP) in eine feste Anzahl von Bilder unterteilt. Diese Bilder bestehen aus einer festen Anzahl von noch kleineren Einheiten wie Slices, Makroblöcke, usw. Die Betrachtung der Datenstruktur kleinerer Einheiten als ein Bild sind für die Codierung und Decodierung bzw. für die Komprimierung wichtig, für die hier untersuchten Scheduling-Verfahren reicht eine Betrachtung der einzelnen Bilder und der Datenstrukturen in Form von GoP.

Das erste Bild einer GoP ist ein I-Bild (Intra Coded Pictures) und es enthält alle Daten, die für die Darstellung des Bildes nötig sind. Die Kompression dieses Bildes beschränkt sich auf die Bildkompression, die Kompressionsrate ist entsprechend gering. Es bietet einen wahlfreien Zugriff und dient als Referenz zur Codierung und Decodierung anderer Bildtypen. Basierend auf diesem I-Bild werden bei den darauf folgenden Bilder von P- und B- Bildtyp nur die Änderung gespeichert. Zur Codierung und Decodierung eines P-Bildes (Predictive Coded Pictures) werden die Daten des vorherigen I-Bildes benötigt. Die Kompressionsrate ist bei P-Bildern dadurch höher als bei I-Bilder. B-Bilder (Bidirectionally Predictive Coded Pictures) brauchen für die Codierung und Decodierung die Informationen des vor-

angegangenen und darauffolgenden I- oder P-Bildes. Dafür ist die Kompressionsrate hier auch am höchsten.

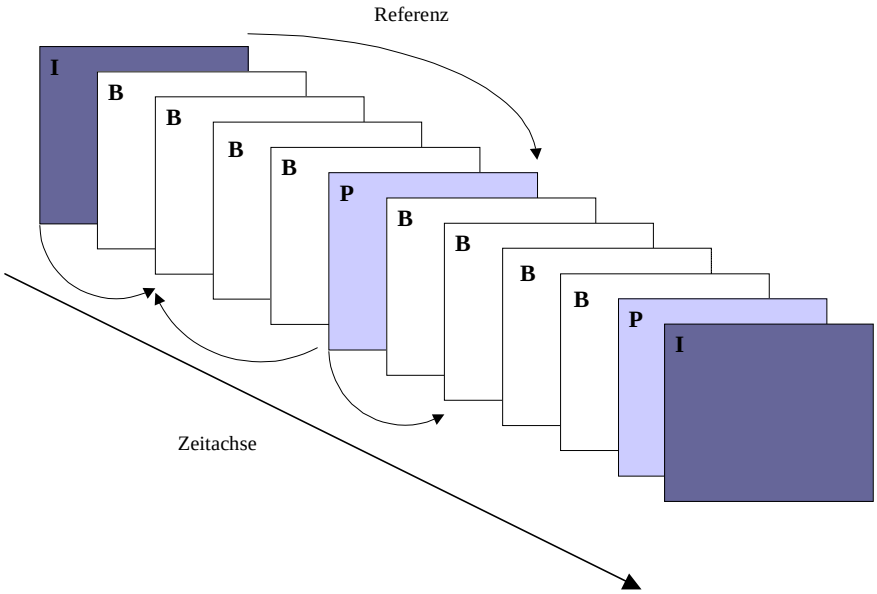



**Abbildung 2-1 Die MPEG-Bildtypen**

Der MPEG-Videostandard benutzt für die Kompression die beschriebenen Vorwärts- und Rückwärts-Prädiktionsverfahren. Für die Dekompression der den B-Bildern vorausgehenden Bilder sind die Daten der später folgenden Daten erforderlich, allerdings werden nur Daten innerhalb einer GoP genutzt. Die GoP bildet also eine abgeschlossene Einheit in der Videosequenz, welche den wahlfreien Zugriff ermöglicht. Diese GoP enthält alle Daten, die für das Abspielen der Bilder innerhalb der GoP benötigt werden. Für das fehlerfreie Abspielen der Bilder der GoP braucht man immer die Daten der kompletten GoP. Würde der hintere Datenteil einer GoP fehlen, könnten größere Teil der Bilder nicht dekomprimiert werden.

Wie bereits erwähnt ist das angestrebte Ziel des Medienservers, die Mediendaten in Form von kleinen Paketen zwischen Client und Server zu übertragen . Das bedeutet, dass bei der Teilung der Daten in Paketen die Dateneinheit in Form der GoP berücksichtigt werden muss, damit ein fehlerfreies Abspielen der gelieferten Datenpakete auch garantiert werden kann. Das heißt, eine GoP soll die kleinste Dateneinheit für die Speicherung und Weiterleitung der Mediendaten sein. Speicherung und Weiterleitung der Mediendaten in kleinere Dateneinheiten wäre nicht sinnvoll und führt beim Abspielen zu Bildfehlern.

### 2.2.2 Datenblöcke

Die Datenmengen der Mediendaten sind sehr groß. Für die Clients sollen die Daten in Form eines kontinuierlichen Datenstroms, welcher aus kleineren Dateneinheiten besteht, geliefert werden. Die Mediendaten können in Einheiten aufgeteilt und gespeichert werden. Anschließend kann auf diese Daten wieder zugegriffen werden.

Es gibt zwei Möglichkeiten, um die Dateneinheit eines Filmes festzulegen:

- Dateneinheit mit konstanter Datenmenge
- Dateneinheit mit konstanter Abspieldauer

Bei Teilung der Daten in Einheiten mit konstanter Datenmenge hat man eine sehr einfache Datenverwaltung. Andererseits ist die Abspieldauer nicht konstant. Wegen der statistischen Natur der Mediendaten kann man aus der Größe der gelieferten Datenmengen nicht erkennen, ob der Client genügend oder zu viele Daten zur Verfügung hat. Dies erfordert eine Rückmeldung vom Client, einen Regelungsmechanismus im Medienserver und eine schnelle Reaktion der Scheduling-Algorithmen sowie schnellen Plattenzugriff, um die fehlenden Daten rechtzeitig aus der Festplatte auszulesen und zum Client weiterzuleiten.

Bei Teilung der Daten in Einheiten mit konstanter Abspieldauer sind die Datenmengen der einzelnen Dateneinheiten nicht konstant. Die Speicherpuffer im Zwischenspeicher sowie auf der Festplatte müssen dynamisch an die Datenmenge angepasst werden. Die meisten Speichermedien sind für die Speicherung von konstanten Datenmengen konzipiert. Die Speicherung der Daten in Einheiten mit variable Länge nutzt die Speichermedien nicht optimal aus. Es erfordert spezielle Datenverwaltung, Datenspeicherung und Datenzugriff. Auf der anderen Seite kann man mit große Sicherheit annehmen, dass während eines Lesezugriffs die gesamten Daten für eine konstante Zeit von der Festplatte gelesen werden können, und dass die an den Client gelieferten Daten auch eine bestimmte Abspielzeit besitzen, entsprechend kann man die Verwaltung der Datenströme über die Anzahl der Dateneinheiten steuern.

Im letzten Unterkapitel über den MPEG-Videostandard ist auch ersichtlich, dass die Natur der Mediendaten bzw. die der kontinuierlichen Datenströme eine zeitliche Abhängigkeit besitzen. Die oben beschriebenen Gründe sprechen mehr für eine Teilung der Mediendaten in Einheiten mit konstante Abspieldauer. Deshalb wird in dieser Arbeit die Dateneinheit mit konstanter Abspieldauer gewählt. Es wird vorausgesetzt, dass die Peripherie die Mediendaten in Form von Dateneinheiten mit konstanter Abspieldauer speichern und auslesen kann. Alle weitere Untersuchungen werden nun im Zusammenhang mit diesem Ansatz betrachtet.

Nach der Entscheidung für die Dateneinheit mit konstanter Abspieldauer soll nun auch untersucht werden, wie lang die optimale Dauer der Zyklen ist. Im nächsten Abschnitt werden zwei von E. Biersack [Bie1] vorgeschlagene Ansätze diskutiert.

## 2.3 Beschreibung der Verfahren

Die Anforderungen an die Verfahren besteht darin, dass die Ressourcen im Medienserver derart zu verwalten sind, dass der Durchsatz und die Effizienz des Me-

dienservers maximiert wird, und dass zu keiner Zeit Ressourcenengpässe entstehen können. Dabei müssen die Verfahren nicht nur die Ressourcen des Medienservers, sondern auch die des Clients berücksichtigen. Dazu soll das Verhalten von Client und Server und ihr Zusammenspiel genauer untersucht werden. Das Hauptziel aller Verfahren ist die geforderte Servicequalität zu gewährleisten. Zur Servicequalität gehört die schnelle Reaktion auf Anfragen der Clients, die Fehlerminimierung, die Optimierung des Ressourcenverbrauch im Medienserver und den Clients.

Die Mediendaten werden hauptsächlich auf sekundäre und tertiäre Speichermedien wie Festplatte und Bandarchive gespeichert. Physikalisch erlaubt die Festplatte einen schnellen und direkten Datenzugriff, was den gleichzeitigen Zugriff auf Daten verschiedener Videosequenzen ermöglicht. Der ideale Fall wäre, wenn alle benötigten Mediendaten auf der Festplatte vorhanden wären. Da die erforderlichen Kapazitäten auf der Festplatte – nach heutigem technischen Stand – mit enormen Kosten verbunden wäre, müssen als Speichermedien Bandarchive genutzt werden.

Bei Bandarchiven dauert der Datenzugriff deutlich länger, ein direkter Zugriff ist recht schwierig und ist mit extrem langen Wartezeiten verbunden. Anderseits ist es physikalisch aufwendig und ineffizient, dass mehrere Dateien parallel aus dem Bandarchive gelesen werden. Jedoch bietet das Bandarchiv eine viel höhere Speicherkapazität bei geringeren Kosten. Aus diesen Gründen werden die Mediendaten nicht direkt von den Bandarchiven abgespielt. Ist der vom Client angeforderte Film aber nicht auf der Festplatte, sondern nur auf Bandarchiven vorhanden, so muss der Film vom Bandarchiv auf die Festplatte kopiert werden. Im Kopiervorgang können die Mediendaten zwischen den beiden Komponenten mit eine höhere Datenrate als beim Abspielen übertragen werden. Dies bietet die Möglichkeit, das Kopieren auf kurze Zeit zu beschränken, mit dem Vorteil, dass die Bandarchive nur für kurze Zeit belegt sind und nach dem Kopieren für andere Zugriffe zur Verfügung stehen. Das Kopieren heißt gleichzeitiges Lesen aus dem Bandarchiv und Schreiben auf die Festplatte. In der Geschwindigkeit wie die Daten vom Bandar-

chive gelesen werden, müssen mit sie auch auf die Festplatte geschrieben werden können. Dies führt aber dazu, dass die Bandbreite der Festplatte voll in Anspruch genommen werden könnte. Eine Möglichkeit dieses Problem zu vermeiden ist die Kopiergeschwindigkeit an die Systemsressourcen anzupassen. Die Datenrate für den Kopiervorgang soll um ein Mehrfaches höher sein als die Datenrate für das Abspielen.

Für die Aufnahme werden die vom Client geschickten Daten zuerst auf die Festplatte gespeichert und gegebenen falls auf die Bandarchive ausgelagert.

Die Festplatte ist wegen ihrer Kapazität und Zugriffsgeschwindigkeit das Speichermedium, das die meiste Flexibilität bietet. Für das Abspielen und für das Aufnehmen ist die Festplatte das optimale kurzfristige Speichermedium. Entsprechend ist es sinnvoll, die Verfahren und die Strategie für die Scheduling-Algorithmen des Medienservers auf die Eigenschaften der Mediendaten und den Zugriffseigenschaften der Festplatte auszurichten und zu optimieren. Die Verwaltung der Bandarchive beschränkt sich auf die Verwaltung des Kopiervorganges der Videosequenzen von und zur Festplatte.

Die Eigenschaften der Mediendaten wurden im Abschnitt 2.2 „Mediendaten“ erläutert. Nun werden die Scheduling-Algorithmen des Medienservers bzw. dessen Implementierung, dem Scheduler, erläutert. Die Optimierung der Scheduling-Algorithmen enthält zwei wichtig Komponenten, erstens muss der Festplattenzugriffe auf die Bedürfnisse der kontinuierlichen Datenströme ausgelegt werden und zweitens muss der Ressourcenverbrauch der Datenströme mimimiert werden.

Im nächsten Abschnitt wird das Festplatten-Scheduling und seiner Einfluss auf die Datenströme sowie die in [Bie1] vorgeschlagenen Optimierungsverfahren für den Client-Service genauer betrachtet.

### 2.3.1 Festplatten-Scheduling

Die Eigenschaften der Peripheriegeräte, wie Festplatte und Bandarchive, wurden in den Arbeiten der Herren Faller [Fal99], Barquero [Bar99] und Ferreira [Fer99] ausführlich behandelt. Für eine detaillierte Betrachtung soll hier auf diese Arbeiten verwiesen werden. Da die Festplatte bzw. das Mehrplattenssystem die entscheidenden Komponenten im Medienserver darstellt, werden hier noch einmal kurz die Punkte angesprochen, um den Zusammenhang mit dem Festplatten-Scheduling zu erläutern.

Für den Zugriff auf die Festplatte stellt das Betriebssystem Schnittstellen zur Verfügung. Die Anfrage (engl.: Request) an die Festplatte wird vom Betriebsystem weiter an den Treiber und den entsprechenden Controller weitergeleitet. Hier entstehen Verarbeitungsverzögerungen, so genannte *Betriebsystem-Overheads*. Der so erzeugte Request veranlasst das Laufwerk, die Schreib-/Leseköpfe auf einen bestimmte Punkt der Plattenoberfläche zu positionieren. Diese Zeit heißt Positionierzeit. Angegeben wird hier immer die mittlere Positionierzeit $t_{seek}$ (engl. :Seek Time). Sind die Schreib-/Leseköpfe über der entsprechenden Spur positioniert, muss gewartet werden, bis der gewünschten Block unter ihnen vorbeikommt. Diese Wartezeit wird auch Umdrehungswartezeit $t_{rot}$ (eng.: rotational time) genannt. Angegeben wird immer die Zeit, welche die Festplatte für eine halbe Umdrehung benötigt. Dann erst werden die Daten übertragen und zwar zuerst diejenigen Daten, die von der Festplatte zum Festplattencontroller gelangen müssen. An dieser Stelle kommt nun die interne Transferrate der Festplatte ins Spiel. Sie gibt die Anzahl der übertragenen Bytes in einer Sekunde an [Fal99].

#### 2.3.1.1 SCAN

Statistisch betrachtet sind die Datenblöcke auf der Festplatte gleich verteilt , das bedeutet für jeder Anfrage eine Neupositionierung des Schreib-/Lesekopfes. Um die Positionierungszeit zu verkürzen, wird ein Platten-Scheduling verwendet. Im Serverbereich ist das traditionelle Verfahren SCAN und dessen Erweiterungen, wie N-SCAN, C-SCAN, T-SCAN weit verbreitet. SCAN sortiert die eingehenden Anforderungen zur Minimierung der Positionierungszeit. Bei der Sortierung bezieht SCAN die momentane Bewegungsrichtung des Kopfes in diese Entscheidung mit ein. Zunächst werden alle Anforderungen bearbeitet, die in der momentanen Bewegungsrichtung des Kopfes liegen, bis keine solche mehr vorhanden sind. Anschließend wird die Richtung umgekehrt und der Vorgang wiederholt. Da die am Rand liegenden Anforderungen besser bedient werden, erreicht SCAN sehr gute Zugriffszeiten. Wesentliches Ziel dieser Verfahren ist die Kostenreduktion der Suchoperationen, um einen höheren Durchsatz zu erreichen und für jeden Prozess einen fairen Plattenzugriff anzubieten [Ste99].

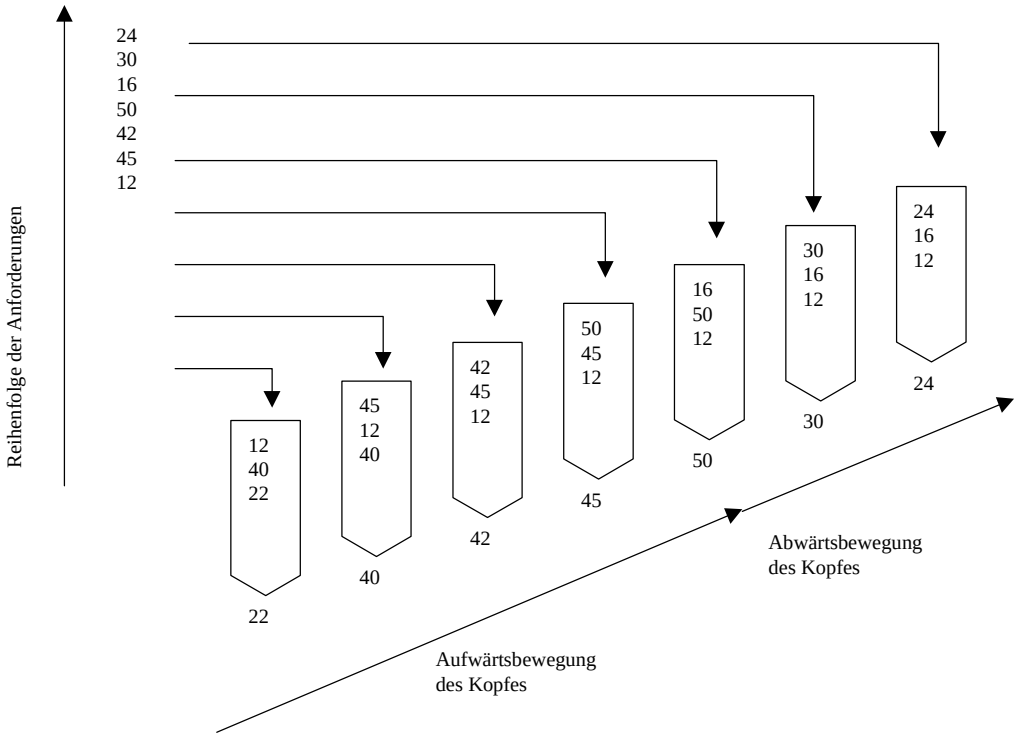



**Abbildung 2-2 SCAN-Festplatten-Scheduling**

Beim Platten-Scheduling für Mediendaten sind die Echtzeitanforderungen an die Bearbeitung sehr hoch, hier hat man es mit vielen kontinuierlichen und konkurrierenden Datenströmen zu tun. Bei kontinuierlichen Datenströmen bedeutet dies, dass die erforderlichen Mediendaten zu einem gegebenen Zeitpunkt zur Verfügung stehen müssen. Die wichtigsten Ziele des Festplatten-Schedulings für den Medienserver sind die Erfüllung der gegebene Zeitschranken und die Maximierung der Zugriffseffizienz. Leider stehen diese beiden Ziele im Widerspruch zu einander. Das bedeutet, dass das Festplatten-Scheduling einen Kompromiss eingehen muss, um beide Ziele im Rahmen zu erreichen. Das bereits erwähnte SCAN-Verfahren optimiert die Suchoperation und maximiert den Durchsatz. Leider kann SCAN die zeitliche Anforderung nicht erfüllen. Ein anderes Verfahren, das EDF-Verfahren, schenkt der Erfüllung der zeitliche Anforderung die höhere Priorität. Das SCAN-EDF-Verfahren ist eine Kombination zwischen dem SCAN- und EDF-Verfahren, berücksichtigt beide Anforderungen und ermöglicht die Balance zwischen den beiden Effekten. Um das SCAN-EDF-Verfahren zu verstehen wird nun zuerst die Grundidee des EDF-Verfahren erläutert, danach werden die Modifikationen zum SCAN-EDF-Verfahren besprochen.

#### 2.3.1.2 Earliest-Deadline-Frist (EDF)

Das Earliest-Deadline-First-Verfahren wurden für das CPU-Scheduling entwikkelt. Bei CPU-Scheduling wird derjenige Prozeß als erster bedient, dessen Zeitschranke am nächsten liegt. Dieser Prozeß erhält die CPU-Ressourcen sofort zur Verfügung, und die Verarbeitung kann unmittelbar daran angeschlossen werden.

Beim EDF-Verfahren für das Festplatten-Scheduling werden für die Anfragen nicht nur die Blocknummer des Datenblockes sondern auch eine Zeitschranke mitgegeben, EDF sortiert die Anfragen nach diese Zeitschranke und liest den Datenblock mit der am nächsten liegenden Zeitschranke zuerst. Dieses Verfahren gibt der Zeitanforderung die höchste Priorität und ignoriert die Zugriffseffizienz.

Beim Festplatten-Scheduling führt dieses Vorgehen aber zu langen Suchzeiten und schlechtem Durchsatz. Was beim CPU-Scheduling sehr gut funktioniert, da die CPU für den Prozeß sofort zur Verfügung steht, kann nicht ohne weitere Anpassungen für Festplatten-Scheduling genutzt werden.. Das Festplatten-Scheduling muss die Suchoperation berücksichtigen. Die Suchoperation der Festplatte beansprucht recht viel Zeit.

### 2.3.1.3 SCAN- Earliest-Deadline-Frist (SCAN-EDF)

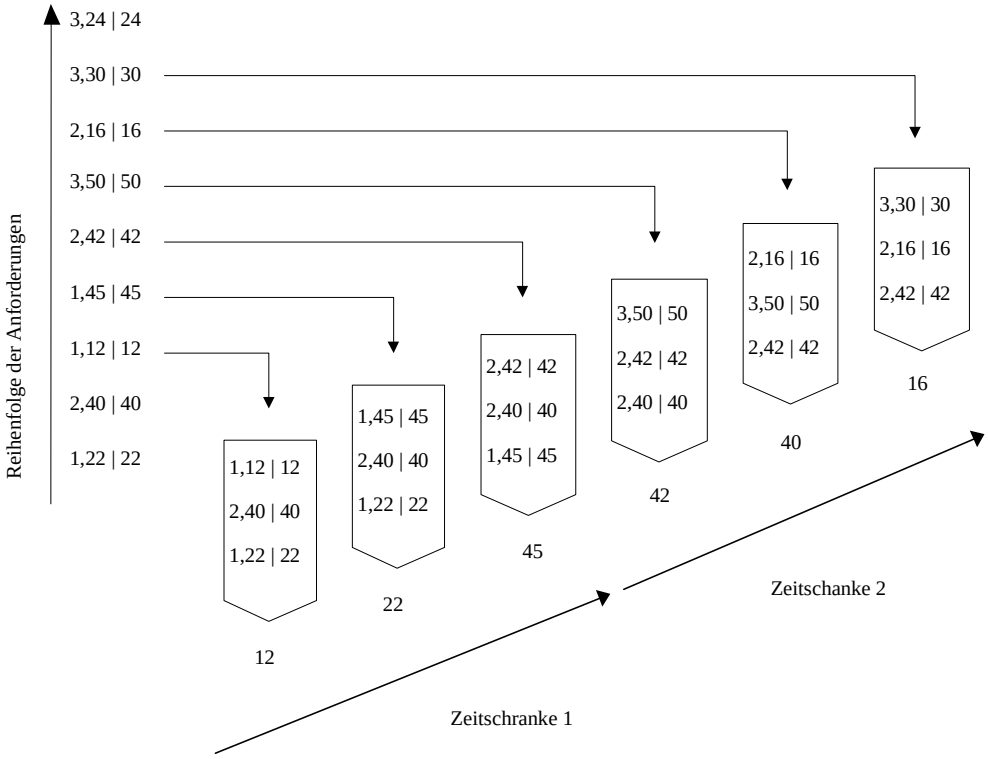



**Abbildung 2-3 SCAN-EDF-Festplatten-Scheduling**

Genauso wie beim EDF-Verfahren müssen bei einer Anfrage neben der Angabe der Blocknummern auch die Zeitschranken für die Anfrage mit angegeben werden. Unter alle Anfragen werden die Anfragen mit der frühesten Zeitschranke vom SCAN-EDF-Verfahren zuerst bedient. Anfragen mit den selben Zeitschran-

ken werden Anfragen, die in der Bewegungsrichtung des Lesekopfes liegen, vorgezogen ausgewählt und zuerst bedient. Diese Prinzip wird solang wiederholt, bis keine Anfrage zu dieser Zeitschranke mehr übrig sind. Da diese Optimierung nur in Kraft tritt, wenn vielen Anfragen mit der selben Zeitschränke vorliegen, hängt die Effizienz des Algorithmus davon ab, wie oft er angewendet werden kann [Ste99].

Für diese Optimierung muß die Frage geklärt werden, wie die richtige Balance bzw. das Verhältnis zwischen den Effekten der beiden SCAN- und EDF-Komponenten ist. Hier gibt es verschiedene Ansätze für diese Optimierung. Die erste Möglichkeit ist, die Zeitschranke zu quantisieren, d.h. jede Zeitschränke ist das Vielfache einer Zeiteinheit *p*. Damit wird die Wahrscheinlichkeit, daß mehreren Anfragen dieselben Zeitschranken haben, erhöht. Mit der Entscheidung über die Größe der Zeiteinheit *p* wird auch das Verhältnis zwischen SCAN- und EDF-Effekten festgelegt. Eine andere Möglichkeit ist die Verzerrung der Zeitschranke. Ist $D_i$ die Zeitschränke der Anfrage *i*, so soll diese zur Zeitschranke $Di + f(N_i)$ verzerrt werden. Die Funktion $f(N_i)$ ist eine Verzerrungsfunktion, welche die Bewegungsrichtung und die Position des Schreib-/Lese-Kopfes berücksichtigt. Es gibt verschieden Ansätze, wie die Verzerrungsfunktion $f(N_i)$ erstellt werden soll, da diese Ansätze mit komplexen Berechnungen verbunden sind, werden sie hier nicht weiter betrachtet. Für weitere Details wird auf [Ste99] verwiesen.

### 2.3.2 Datenstrom

Die Versorgung des Clients mit Daten erfolgt in Form eines kontinuierlichen Datenstroms. Diese Versorgung kann am besten durch Zyklen verwaltet werden. In jedem Zyklus soll der Medienserver für jeden Datenstrom in bestimmten Zyklen eine bestimme Mengen von Daten von der Festplatte holen und dem Client zur Verfügung stellen. Die Menge der Daten hängt von der Datenrate der Videosequenz ab. Allerdings schwanken die Datenmengen wegen der statistischen Eigen-

schaften der komprimierten Daten sehr stark. Wie in Abschnitt 2.2.2 „Datenblöcke“ beschrieben, sollen die Daten in Einheiten mit konstanter Abspieldauer gespeichert werden. Das bedeutet, die Zyklen des jeweiligen Datenstromes kann ein Mehrfaches der Abspieldauer der Dateneinheit sein. Diese Zyklen werden auch *Service-Round* genannt. Generell versucht der Medienserver, die Daten für die nächsten Zyklen im Voraus von der Festplatte zu lesen, diese Daten zwischenzuspeichern und dann rechtzeitig an den Client zu liefern.

Auf der Client-Seite werden die vom Medienserver kommende Datenpakete zwischengespeichert und kontinuierlich verwendet. Für eine gegebene Speichergröße dürfen nicht mehr Daten ankommen als der Client aufnehmen kann, da sonst die Daten verloren gehen. Kommen anderseits die Daten nicht rechtzeitig an, entstehen Bildfehler. Wegen der Echtzeitanforderung des kontinuierlichen Datenstroms muss der Medienserver eine optimale Versorgung gewährleisten. Die benötigten Daten müssen zur Abspielzeit vorhanden sein. Die Scheduling-Algorithmen des Medienservers müssen dafür sorgen, dass es für den Client zu keiner Zeit zu einer Speicherüberfüllung (engl.: buffer overflow) bzw. Speicherunterfüllung (engl.: buffer starvation) kommen darf.

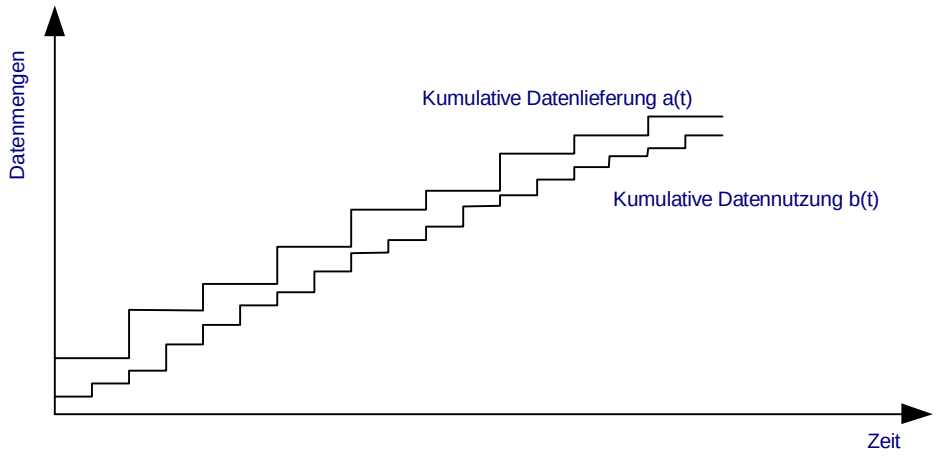



**Abbildung 2-4 Kumulative Datenlieferung und Datennutzung**

Den Speicherstatus des Clients zum Zeitpunkt *t* kann man aus der Differenz zwischen den gelieferten und den benutzten Datenmengen bestimmen. Würde man die gelieferten Datenmengen in einer kumulativen Funktion *a(t)* und die benutzten Datenmengen in einer kumulativen Funktion *b(t)* darstellen, so tritt für eine Speichergröße $b_{total}$ eine Speicherüberfüllung auf, wenn $a(t) - b(t) > b_{total}$, und eine Speicherunterfüllung auf, wenn $a(t) - b(t) < 0$.

### 2.3.2.1 Constant Time Length (CTL)

Bei CTL-Verfahren greift der Scheduler auf Datenmengen mit konstanter Abspieldauer für die Zyklusdauer $\tau$ auf der Festplatte zu. Bei Daten mit Speichereinheiten mit konstanter Abspieldauer entspricht diese Zyklusdauer $\tau$ der Anzahl der Datenblöcke multipliziert mit der Abspieldauer des Datenblocks. Die gelesenen Datenblöcke werden dann direkt zum Client übertragen. Danach macht der Scheduler eine Pause und wartet bis zum nächsten Zyklus, um dann die Prozedur zu wiederholen. Die Datenmengen des Datenstroms entsprechen den Datenmengen für den Zyklus.

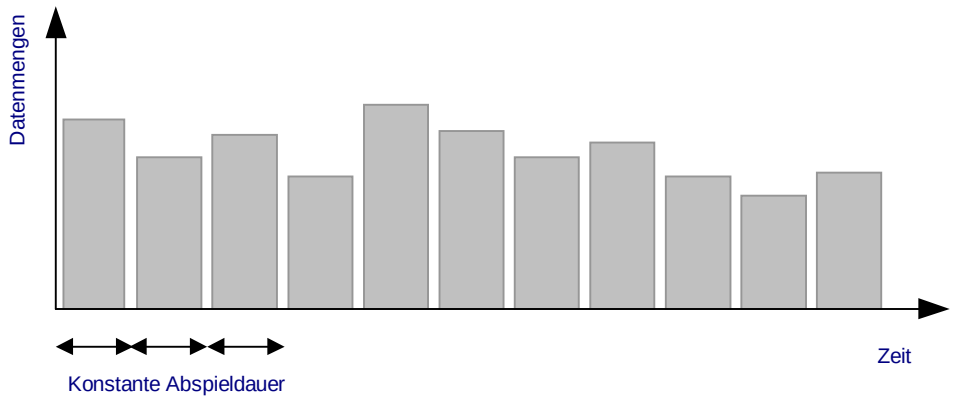



**Abbildung 2-5 Datenmengen des Datenstromes mit CTL-Verfahren**

Dieses Verfahren garantiert, dass der Client immer die benötigten Datenmengen erhält, die er für die Zyklusdauer $\tau$ braucht. Die Zyklusdauer $\tau$ kann von einer bis zu mehreren Sekunden betragen. Bei längere Zyklusdauer hat man den Vorteil, dass die Festplatte viele Zugriffsanfragen bekommt und dementsprechend das Platten-Scheduling besser optimieren kann. Außerdem schwankt die Datenmenge im Datenstrom weniger stark . Man hat aber auch den Nachteil, dass der Speicherbedarf für die Zwischenspeicherung und die Anfangsverzögerung proportional mit der Zyklusdauer $\tau$ wächst, dies gilt sowohl für den Medienserver als auch für den Client. Damit dieser Nachteil nicht überwiegt, muss die Zyklusdauer klein gehalten werden.

#### 2.3.2.2 General Constant Time Length (GCTL)

Beim GCTL-Verfahren wird eine Trennung zwischen Festplatten-Zugriffzyklen (Disc Service Round) und Client-Servicezyklen (Client Service Round) durchgeführt.

Während der Festplatten-Zugriffzyklen $\tau$ werden die Daten für einen Datenstrom genauso wie beim CTL-Verfahren von der Festplatte gelesen. Auf alle benötigten Daten wird auf einmal zugegriffen. Allerdings wird die Dauer $\tau$ klein gehalten. Für die Client-Servicezyklen oder GCTL-Zyklen wird der Verbrauch für den Zyklus $\tau_i$ errechnet, wobei die Client-Servicezyklen ein Vielfaches des Festplatten-Zugriffzyklus $\tau$ sein sollten. Ist A[t, t + $\tau_i$] die Datenmengen, die in einem Datenstrom für die Client-Servicezyklen $\tau_i$ gebraucht werden, so ist die Datenmenge A[t, t + $\tau_i$] / $m_i$, wobei $m_i = \tau_i / \tau$ der Verhältnisfaktor zwischen den beiden Zyklen ist, auf die in einem Festplatten-Zugriffzyklen zugegriffen werden soll.

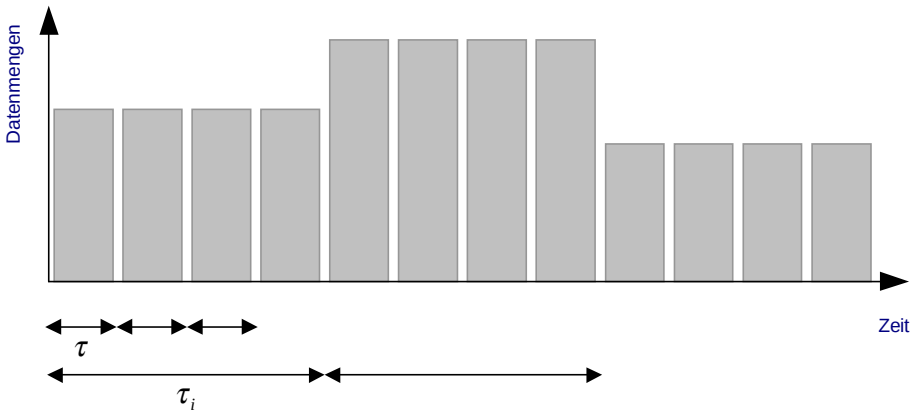



**Abbildung 2-6 Datenmengen des Datenstromes mit GCTL-Verfahren**

Bei diesem Verfahren ist die Anforderung bzw. Belastung auf den Festplattenzugriff gleich wie beim CTL-Verfahren. Da der Client-Servicezyklus länger ist und die Festplatten-Zugriffzyklen aber trotzdem kurz gehalten werden, hilft dies, die Schwankungen im Datenstrom abzuflachen. Der Speicherbedarf für die Zwischenspeicherung auf den Medienserver sowie auf den Client und die Anfangsverzögerung wächst nicht proportional mit der Zyklusdauer $\tau_i$.

Dieses Verfahren hat allerdings den Nachteil, dass man die Datenmengen für den Client-Servicezyklus im Voraus kennen muss. Um die Schwankungen im Datenstrom abzuflachen, werden die Datenmengen pro Festplattenzugriff auf einen Mittelwert begrenzt. Das bedeutet für die Speicherung der Daten in Einheiten mit konstante Abspieldauer eine Teilung der Dateneinheit. Zusätzlich ist nicht gesichert, dass die Begrenzung der Datenmengen in den Festplatten-Zyklen $\tau$ den Datenbedarf des Clients garantiert werden können.

## 2.4 Verfügbarkeitsprüfung

Um den Datenfluss für die gesamte Abspieldauer nicht zu gefährden, müssen Systemressourcen reserviert werden. Das heißt, für die gegebenen Systemressourcen können nur eine bestimmte Anzahl von Datenflüssen gleichzeitig aktiv sein, bzw. nur eine bestimmte Anzahl von Clients gleichzeitig bedient werden. Dies bedeutet für den Medienserver, dass für jede neu ankommende Anfrage eines Clients eine Verfügbarkeitsprüfung (englisch: Admission-Control) durchgeführt wird, um zu Überprüfen, ob noch genügend Ressourcen zur Verfügung stehen und damit einen zusätzlichen Datenstrom mit den geforderten Servicegarantien auch gewährleisten zu können. Falls nicht, muss die Anfrage abgelehnt werden. Für dieses Problem soll untersucht werden, welches Verfahren den Ressourcenverbrauch minimieren und die Leistung des Medienservers maximieren kann. Die Leistung des Medienservers wird durch die Anzahl der Datenströme gemessen.

Aktivitäten wie das Abspielen, die Aufnahme und das Kopieren von Mediendaten zwischen den Peripherien verbrauchen viele Ressourcen wie Systemspeicher, Festplattenvolumen, Festplattentransferrate und Netzbandbreite. Eine Verfügbarkeitsprüfung setzt voraus, dass der Medienserver seine Systemressourcen und alle Aktivitäten mit entsprechendem Ressourcengebrauch kennt. Der Medienserver benötigt einen Mechanismus, womit er den aktuellen Bedarf feststellen kann, um dementsprechend die Neuzugänge kontrollieren zu können.

Die Ressourcenverwaltung kann am besten durch eine zentrale Instanz durchgeführt werden. Jeder Datenstrom muss für jeden Ressourcenbedarf eine Anfrage an diese Ressourcenverwaltung machen, um die benötigte Ressourcen zu reservieren. Damit die Ressourcenverwaltung auch richtig arbeiten kann, müssen die nicht mehr benötigten Ressourcen an die Ressourcenverwaltung zurückgegeben werden. Diese freigewordene Ressourcen können dann für andere Aktionen zur Verfügung gestellt werden. Die Reservierung und Freigabe müssen vor und nach jeder ressourcen-intensiven Aktion gemacht werden. Dies gilt für das Abspielen, die Aufnahme und das Kopieren von Mediendaten.

Die kontinuierlichen Datenströme benötigen in allen Instanzen, von der Datenquelle bis zum Datensenke die gleichen Datenmengen bzw. Bandbreiten. Dies gilt für Speichervolumen auf der Festplatte, der Festplattenbandbreite, dem Zwischenspeicher auf dem Medienserver, der Netzbandbreite und dem Zwischenspeicher auf dem Client. Es besteht ein linearer Zusammenhang zwischen den erforderlichen Festplatten-, Netzwerkbandbreite und Speichervolumen. Der kritische Punkt dieser Untersuchung ist in der Festplattenbandbreite zu sehen. Deshalb wird in den weiteren Betrachtungen vorwiegend hierauf Augenmerk gelegt. Das Ergebnis der Untersuchung soll aber auch für die anderen Komponenten gelten. Der Ressourcenbedarf kann dementsprechend umgerechnet werden.

Im folgenden werden die deterministischen und statistischen Ansätze erläutert.

### 2.4.1 Deterministischer Ansatz

Um die Ressourcenverfügbarkeit für die Echtzeitanforderung der kontinuierlichen Datenströme garantieren zu können, muss bei der deterministischen Verfügbarkeitsprüfung die ‚Worst-Case'-Annahme gemacht werden. Dies ist nötig, um die Ressourcen für den maximalen Bedarf reservieren zu können.
In [Fal99] wurde der Vorschlag gemacht, daß die Entscheidung mit dem Vergleich, ob die erforderliche Zugriffszeit aller Datenströme kleiner ist als die Zeit eines Servicezyklen, getroffen werden soll. Dabei werden die Anzahl der Datenblöcke, Blockgröße, die Transferrate sowie die Zeit für die Suchoperation aller Datenströme berücksichtigt.

Bei genauer Untersuchung der Mediendaten findet man heraus, dass in der meisten Zeit die für den ‚Worst-Case' reservierten Ressourcen nicht genutzt werden. Die Zeit für diesen maximalen Bedarf könnte nur ein Bruchteil der Abspieldauer der Videosequenz sein. D.h., die reservierten Ressourcen werden nur zu einem Teil der Zeit voll ausgenutzt, die meiste Zeit werden die Ressourcen jedoch nicht

gebraucht. Das bedeutet eine Verschwendung der Systemressourcen, die eigentlich für anderen Aktionen genutzt werden könnten.

### 2.4.2 Statistischer Ansatz

Beim statistischen Ansatz arbeitet man mit statistischen Garantien. Es wird garantiert, dass Fehler für den Service unterhalb des Grenzwerts der vorgegebenen Fehlerwahrscheinlichkeit liegen werden.

Dies ermöglicht auch die Zugangskontrolle über das statistische Verhalten des Systems, das alle aktiven Datenströme und den noch hinzukommenden Datenstrom berücksichtigt. Die möglichen Fehlerwahrscheinlichkeiten werden aus den statistische Daten errechnet. Liegt die Fehlerwahrscheinlichkeit unterhalb des Grenzwerts der vorgegebenen Fehlerwahrscheinlichkeit, wird die Anfrage angenommen, ansonsten wird sie abgelehnt.

Mit diesem Ansatz können die Ressourcen des Medienserver voll genutzt werden. Allerdings benötigt man für die Berechnung die Wahrscheinlichkeitsdichte der Paketgössen des Datenstroms. Eine Möglichkeit die statistische Verteilungsdichte der Datenmengen zu beschreiben ist die Darstellung als Gauß'sche Verteilungsdichte mit Mittelwert $\mu$ und Standardabweichung $\sigma$.

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2}\frac{(x-\mu)}{\sigma^2}}$$

Die Benutzung der Gauß'schen Normalverteilung ist einerseits für die Veranschaulichung des Problems sehr gut verwendbar, leider für die rechnerische Berechnung recht kompliziert und aufwendig zu implementieren. Andererseits ist die Darstellung als Gauß'sche Verteilungsdichte mit Mittelwert und Standardabweichung nur als eine Näherung an die Wirklichkeit zu betrachten.

In [Bie1] wurde vorgeschlagen, dass die Statistikdaten in Form von Histogrammen dargestellt werden könnten. Diese Histogramme können für die Darstellung der statistische Datenrate des Datenstromes sowie für die Darstellung des statistische Systemlast bzw. des Ressourcenbedarfs genutzt werden. Für die Datenströme können aus der Analyse der Mediendaten recht genaue Statistikdaten gewonnen werden. Mit dieser Darstellung in Form von Histogrammen hat man auch eine recht einfache Möglichkeit, die Statistikdaten der Systemlast aus den Statistikdaten des einzelnen Datenstroms zu errechnen. Die Fehlerwahrscheinlichkeit des Gesamtsystem kann wiederum aus dem Histogramm der Systemlast berechnet werden.

Ein Histogramm kann als Polynom dargestellt werden. Der große Vorteil beim Polynom ist, dass für die Berechnung seiner Elemente einfache Algorithmen implementiert werden können.

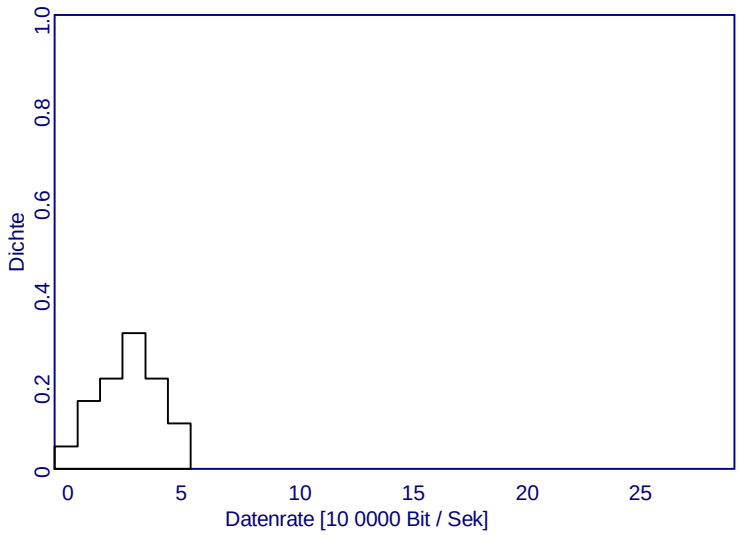



**Abbildung 2-7 Beispiel für ein Histogramm**

Ist nur ein Datenstrom im Medienserver aktiv, so entspricht das Histogramm der Gesamtsystemlast dem des Datenstromes. Kommt ein neuer Datenstrom hinzu, kann das Histogramm der neuen Gesamtlast durch die Faltung des Datenratenhistogramms mit der aktuellen Systemlast errechnet werden. Die Faltung kann als

Multiplikation zweier Polynome und deren Elemente – die Werte $H_n$ und $h_{n+1}$ – betrachtet werden. Sind $l_H$ und $l_k$ die Länge des Polynoms, so hat das Ergebnis ein Histogramm der Länge $l_H + l_h + 1$ und lässt sich wie folgt darstellen:

$$H_{n+1}(k) = \sum_{i=1}^{k-1} H_n(i) \cdot h_{n+1}(k-i)$$

Das Histogramm der Gesamtlast ist höher als die Datenrate des einzelnen Datenstroms, entsprechend ist das Histogramm nach recht verschoben. Die Faltung kann für beliebig viele Datenströme durchgeführt werden. Im folgenden Bild wird die Faltung zweier Datenströmen graphisch dargestellt.

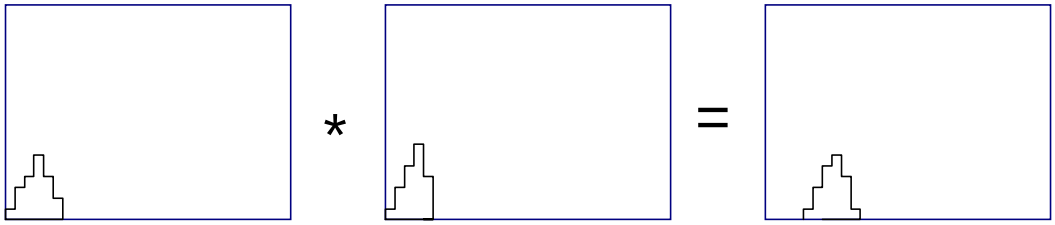


**Abbildung 2-8 Grafische Darstellung der Faltung zweier Histogramme**

Sollen $n$ Datenströme aktiv sein und sind alle charakteristischen Datenratenhistogramme $h_i$ für den Datenstrom $s_i$ vorhanden, so lässt sich die Gesamtlast $H_n$ durch n-malige Faltung aus den einzelnen Datenratenhistogrammen errechnen .

Die Grenzen der Festplatte werden durch ihre Transferrate und ihre Positionierzeit $t_{seek}$,, Umdrehungswartezeit $t_{rot}$ und die Anzahl der Datenströme, die gleichzeitig von der Festplatte gelesen werden sollen, vorgegeben. Ist $D_{n+1}^{\max}$ die theoretische Grenze, bei der auf die Datenmenge während eines Festplattenzugriffzyklus $\tau$ zugegriffen wird, so kann dies wie folgt dargestellt werden :

$$D_{n+1}^{\max} = (\tau - t_{seek} - (n+1) \cdot t_{rot}) \cdot r_{disk}$$

Daraus erhält man die Fehlerwahrscheinlichkeit des Medienservers als die Summe aller Histogrammelemente, die höher sind als die Grenze $D_{n+1}^{\max}$ .

$$p(D > D_{n+1}^{\max}) = \sum_{k=D_{n+1}^{\max}}^{\infty} H_{n+1}(k)$$

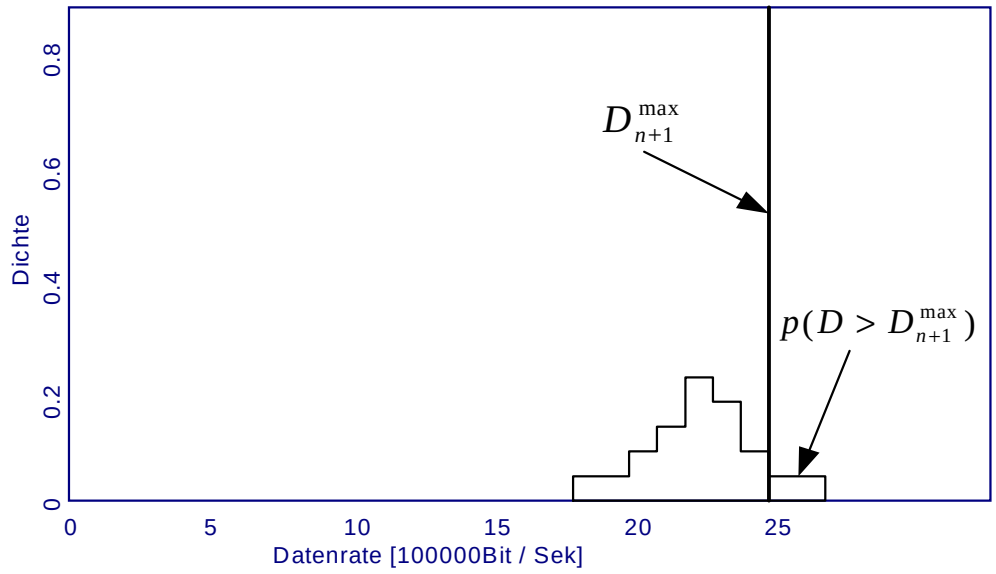



**Abbildung 2-9 Histogramm des Grenzbereichs**

In [Bie1] wurden Messungen für eine gegebene Fehlerwahrscheinlichkeit von $10^{-4}$ gemacht. Im Vergleich zum deterministischen Verfahren können 30 – 40 % mehr Datenströme gleichzeitig vom Medienserver bedient werden. Bei eine Fehlerwahrscheinlichkeit von $10^{-4}$ bedeutet dies, dass es durchschnittlich alle $10^{4}$ Zugriffe zu einem Fehler kommen kann. Bei einem Festplattenzugriffszyklus $\tau$ von einer Sekunde würde es alle 2,8 Stunden zu einem Datenfehler kommen können. Wie bei jeder statistischen Aussage, bedeutete die Angabe der Fehlerwahrscheinlichkeit nur die Aussage, dass ein Fehler auftreten kann. Was für Auswirkungen dieser Fehler aber im Datenstrom verursacht, kann nicht vorhergesagt werden.

Mit Hilfe des statistischen Verfahrens kann die Verfügbarkeitsprüfung mehr Datenströme zulassen als mit dem deterministischen Verfahren. Das statistische Verfahren erlaubt die Leistung des Medienservers bis zur Systemgrenze auszunutzen.

An der Systemgrenze nimmt man absichtlich das Risiko, Fehler zu verursachen, in Kauf. Dieses Risiko ist jedoch im voraus abschätzbar.

## 2.5 Strategie

Nach den Erkenntnissen der oben beschriebenen Analyse werden folgende Entscheidungen für die weitere Modellierung des Medienservers getroffen:

- Die Mediendaten sollen in Einheiten mit konstanter Abspieldauer gespeichert und gelesen werden. Diese Abspieldauer der Speichereinheit ist an der Abspieldauer der ‚Group of Pictures' orientiert. Das bedeutet, dass Videofilme in Speichereinheiten unterschiedlicher Größe gespeichert werden. Innerhalb eines Videofilmes soll die Abspieldauer konstant sein.

- Als Festplatten-Scheduling soll das SCAN-EDF-Verfahren benutzt werden, da dieses Verfahren die Flexibilität besitzt, das Verhältnis der Effekte des SCAN- und EDF-Verfahrens durch die Quantisierung bzw. die Verzerrung der Zeitschranke je nach Anforderung einzustellen.

- Für die Optimierung der Datenströme zum Client soll das CTL-Verfahren benutzt werden. Allerdings wird die Idee der Trennung der Servicezyklen in Client-Servicezyklen und Festplatten-Zugriffzyklen vom GCTL-Verfahren übernommen. Diese Trennung ist erforderlich, da sich die Speichereinheiten der Mediendaten an der GoP orientieren. Feste Client-Servicezyklen für alle Datenströme führen zu nicht optimalen Servicezyklen für die Datenströme. Die Client-Servicezyklen für die Datenströme sollen sich nach einem vorgegebene Wert richten. Unter Berücksichtigung der Abspieldauer der Speichereinheit sollen jedoch Abweichungen erlaubt sein.

- Die Verfügbarkeitsprüfung soll mit einem statistischen Ansatz mit Histogrammen realisiert werden, da dieses Verfahren die Maximierung der Nutzung vorhandener Systemressourcen verspricht und die Möglichkeit einer Risikoabschätzung anbietet.

# 3 Modellierung des Medienservers

In Kapitel 2 wurden die Grundlagen und die Strategien für die Scheduling-Algorithmen des Medienservers besprochen. Basierend auf diesen Grundlagen und Strategien wird nun ein Modell erstellt, in das die Strategien integriert werden und gleichzeitig die geforderten Funktionalitäten des Medienservers erfüllt. Dieses Modell dient wiederum als Basis für die Implementierung eines Simulationsmodells mit dem Simulationstool OMNeT++.

## 3.1 Ansätze

Der Medienserver besteht intern aus mehreren Komponenten, deren Zusammenwirken dem Client einen bestimmten Service anbietet. Dabei übernimmt jede Komponente dedizierte Aufgaben. Die Leistung des Medienservers wird durch die Anzahl der kontinuierlichen Datenströme gemessen. Diese Datenströme sollen so geliefert werden, dass der Client Filme störungsfrei abspielen kann und gleichzeitig einen möglichst geringen Ressourcenbedarf im Server aufweist.

Die Modellbeschreibung soll als Grundlage für den Medienserver dienen. Dabei werden folgende Ansätze benutzt:

- Der Medienserver analysiert die Dateninhalte nicht. Die Daten müssen zuvor in geeigneter Form gespeichert werden. Täte der Fall auf, dass der Medienserver die Dateninhalte analysieren muss, würde er zu stark in Anspruch genommen; vor allem könnten die Ressourcen für die Analyse nicht genau genug berechnet werden. Bei diesem Ansatz wird deshalb verlangt, dass beim Speichern der Daten auf der Festplatte, z. B. bei der Aufnahme, die Mediendaten

werden und die Dateninformationen in Form von Meta-Daten mit abgespeichert werden.

- Der Medienserver speichert und verwaltet die Mediendaten in Form von ‚Group of Pictures' als kleinste Speichereinheit. Die Framerate und die Anzahl der Frames pro GoP sollen zur Abspielzeit bekannt sein. Mit Hilfe der Meta-Daten wird versucht, den Datenbedarf bzw. Framebedarf eines Clients für einen bestimmten Client-Servicezyklen zu berechnen und damit dem Client zu garantieren.

- Abhängig von der Bildauflösung und der Framerate des Filmes kann es für die Datenströme zu einem unterschiedlichen Ressourcenbedarf kommen. Der Ressourcenbedarf wird vom Medienserver berechnet und verwaltet. Hier wird vorausgesetzt, dass das Datenratenhistogramm für die einzelnen Mediendaten im Meta-Datensatz vorhanden ist.

- Wie das Unterkapitel 2.3.2 „Datenstrom" zeigt, gibt es Verfahren, die eine Trennung zwischen den Festplatten-Zugriffzyklen und den Client-Servicezyklen machen. Diese Trennung ermöglicht, dass das Modell offen für andere Strategien bleibt. Der Festplattenzugriff und der Client-Service sollen von unterschiedlichen Managern (Prozessen) verwaltet werden. Der Datenaustausch zwischen diesen Managern soll über den Zwischenspeicher erfolgen.

- Die Datenströme sollen unabhängig voneinander verwaltet werden.

## 3.2 Beschreibung des Clients

Der Client ist der Vermittler zwischen dem Benutzer und dem Medienserver. Auf der einen Seite bietet er dem Benutzer die Funktionalität des Abspielens, Auf-

nehmens, Stoppens, Vorwärts- und Rückwärtsspulens des Videofilms, also die Funktionalitäten eines Videorecorders. Auf der anderen Seite ist die Verfügbarkeit eines Videofilms nicht garantiert, und der Client ist auf den Dienst des Medienservers angewiesen. So dient der Client an zwei Schnittstellen und teilt sich dementsprechend in zwei Komponenten auf: Die Video- und die Übertragungseinheit.

### 3.2.1 Videoeinheit

Die Videoeinheit ist die Schnittstelle zum Benutzer. Hier werden die Mediendaten in Video-, bzw. Audiodaten umgewandelt und abgespielt. Parallel dazu empfängt sie auch alle Kommandos vom Benutzer und leitet sie an den Medienserver weiter. Je nach Benutzerkommando werden die Mediendaten vom bzw. zum Medienserver weitergeleitet. Die Videoeinheit komprimiert Mediendaten zu GoP-Datenblöcke bzw. dekomprimiert GoP-Datenblöcke zu Mediendaten und steuert die Übertragung zum Medienserver.

Beim Abspielen liest sie die Daten in Form von GoP aus dem Client-FIFO, dekodiert die Daten und bildet für den Benutzer die erstellten Bilder auf dem Bildschirm ab. Bei der Aufnahme analysiert die Videoeinheit die Aufnahmedaten, kodiert die Daten in das MPEG-Format und bildet die GoPs auf Datenblöcke ab. Die Datenblöcke werden im Client-FIFO gespeichert und stehen nun der Übertragung zur Verfügung.

Im Normalfall werden die meisten Benutzerkommandos von der Videoeinheit zur Übertragungseinheit bzw. zum Medienserver weitergeleitet, da die Anzahl der Bilder direkt auf die Anzahl der Datenblöcke abgebildet werden kann. Nur im Fall des Vorwärts- bzw. Rückwärtsspulens ist die direkte Abbildung nicht mehr möglich. Hier muß die Videoeinheit aus der Spulzeit die entsprechende Anzahl der Bilder und die Datenblöcke bestimmen. Danach wird die Übertragungseinheit beauftragt, die entsprechenden Datenblöcken vorwärts bzw. rückwärts zu spulen.

Ein Teil der Videoeinheit die (Schnittstelle zum Benutzer, die Dekomprimierung und die Komprimierung der Videoeinheit) entspricht der im Institut für Nachrichtentechnik erstellten Studienarbeit „Client-Modellierung in einem Medienserver-Simulationssystem“ von H. Kaltenbach [Kal99].

### 3.2.2 Übertragungseinheit

Diese Einheit verwaltet die Weiterleitung der Kommandos und der Daten zwischen Client und Medienserver. Abhängig vom Kommando der Videoeinheit bzw. des Benutzers werden die Datenblöcke zwischen dem Client-Zwischenspeicher und dem Medienserver übertragen.

## 3.3 Beschreibung des Medienservers

Für seine interne Verwaltung hat der Medienserver folgende Aufgaben zu bewältigen:

- Die Verwaltung der Datenübertragung
- Die Verwaltung von Client-Anfragen sowie die Steuerung des Datenstromes
- Die Verwaltung der Verzeichnisstruktur
- Die Verwaltung der Zugriffe auf die Peripherie bzw. auf die Mediendaten
- Die Verwaltung des Zwischenspeichers
- Die Verwaltung der Systemressourcen
- Die Verwaltung der Datenspeicherung

Diese Aufgaben werden jeweils von einem separaten Modul übernommen. Entsprechend ihren Aufgaben werden diese Module auch benannt. Der Medienserver

besteht intern also aus den Modulen „Network-Manager“, „Client-Manager“, „Directory-Manager“, „Device-Manager“, „Memory-Manager“, „Admission-Controller“, sowie dem „Operating-System“ als Schnittstelle zum SCSI-Subsystem und der Peripherie. Abbildung 3-1 zeigt die Komponenten des Medienservers und seine Schnittstellen zueinander. In den nachfolgenden Unterkapiteln werden die Aufgaben der Komponenten und ihr Zusammenwirken beschrieben.

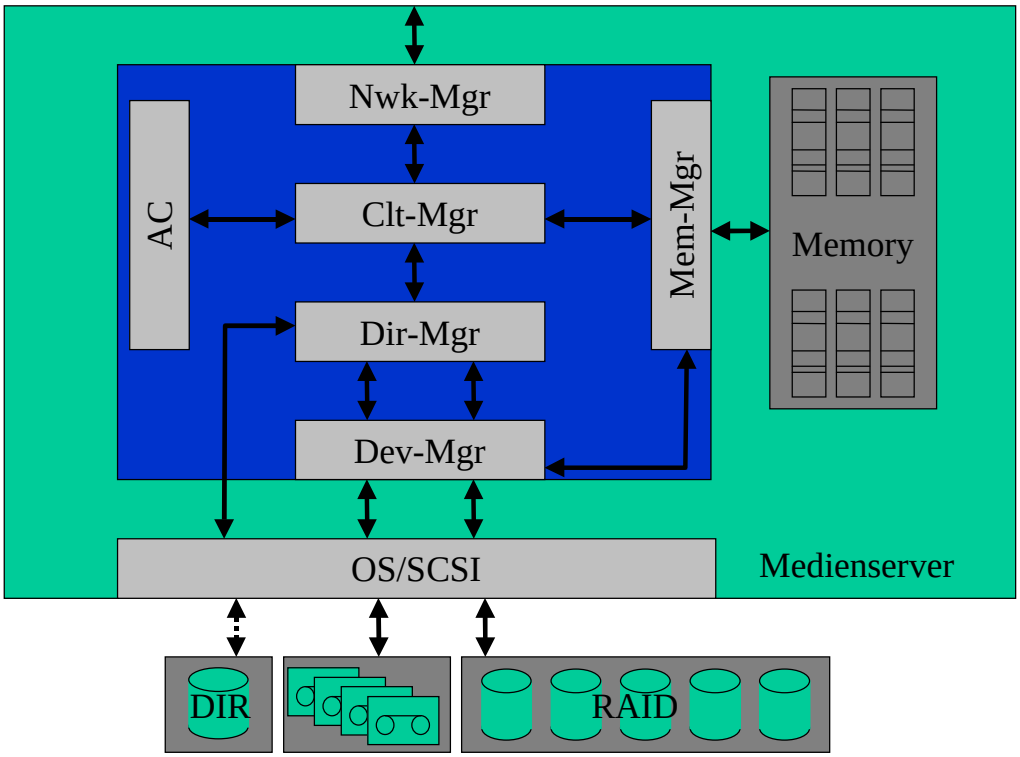



**Abbildung 3-1 Komponenten des Medienservers und seine Schnittstellen**

## 3.3.1 Network-Manager (Nwk-Mgr)

Der Network-Manager übernimmt die Weiterleitung aller Daten zwischen dem Client und dem Medienserver. Er verwaltet die Netzwerkadressen und die Daten-

stromnummer. Des weiteren legt er die Zuordnung zwischen den Netzwerkadressen und der internen Datenstromnummer fest.

### 3.3.2 Client-Manager(Clt-Mgr)

Der Client-Manager empfängt alle Anforderungen vom Client. Er hat Zugriff auf die im Zwischenspeicher gespeicherten Daten und kontrolliert die Datenströme zum Client. Über den Directory-Manager hat er Zugriff auf alle Dateninformationen, Datenstatistiken bzw. Mediendaten. Er ist damit die zentrale Komponente für die Interaktion zwischen Medienserver und Client. Der Client-Manager soll die Aufgabe der Koordination aller Aktionen im Medienserver übernehmen; er ist die Steuereinheit des Medienservers. Somit muss er dafür sorgen, dass die Datenströme in periodischen Intervallen mit Daten versorgt werden.

In diesem Modul wird die Scheduling-Strategie des Medienservers implementiert. Dieser Ansatz, die Strategie des Medienservers auf ein Modul zu beschränken, hat den Vorteil, dass verschiedene Strategien sowie deren Parametrisierung leicht angepasst bzw. ausgetauscht werden können, ohne das Gesamtmodell zu modifizieren. Somit hat man ein Modell, mit dem verschiedene Strategien simuliert werden können. In dieser Arbeit wird das CTL-Verfahren implementiert.

Für jedes Kommando vom Client wird die entsprechende Aktion eingeleitet. Beim Kommando für das Abspielen beauftragt der Client-Manager den Directory-Manager, die Information und die Statistik der Mediendaten von der Festplatte zu lesen. Existiert die Information bzw. die Statistik, bedeutet dies, dass auch die Mediendaten vorhanden sind. Sind die Informationen nicht vorhanden, wird die Anfrage abgelehnt, andernfalls benutzt der Client-Manager die Information und die Statistik der Mediendaten und reserviert beim Admission-Controller die erforderlichen Systemressourcen. Wird die Reservierung mit einer positiven Rückmeldung bestätigt, werden weitere Aktionen veranlasst. Anhand der Datenstatistik

wird die Anzahl der Datenblöcke berechnet, die von der Festplatte geholt und in den Zwischenspeicher übertragen werden sollen. Die Anzahl der benötigten Datenblöcke ist abhängig von der Framerate, der Datenrate, sowie von der Dauer der Client-Servicerunde. Die errechnete Anzahl der Datenblöcke wird an den Directory-Manager weitergegeben. Damit beauftragt der Client-Manager den Directory-Manager, die genaue Anzahl der Datenblöcken zu holen, die im nächsten Schritt für den jeweiligen Datenstrom zur Verfügung stehen müssen, wobei die Beauftragung zum Directory-Manager nur auf die Anzahl der Datenblöcke beschränkt ist. Um welche Datenblöcke es sich handelt, bestimmt der Directory-Manager.

### 3.3.3 Directory-Manager (Dir-Mgr)

Dieses Modul verwaltet Dateninhalte der Peripherie. Es regelt die gesamten Mediendaten und ihre Information. Diese Peripherie können Festplatten bzw. Festplattensysteme und Bandarchive sein. Alle Aktionen des Directory-Managers werden vom Client-Manager angestoßen. Über die Verzeichnisstruktur kennt der Directory-Manager die gesamte Information der Mediendaten der Peripherie, sowie die einzelnen Datenblöcke. Für jeden aktiven Datenstrom wird die Gesamtanzahl der Datenblöcke sowie der aktuelle Datenblock intern aufgeführt, und entsprechend den Kommandos des Client-Managers werden die Aktionen schließlich ausgeführt. Für das Abspielen werden zuerst die Datenblöcke und ihr Speicherort bestimmt und dann der Device-Manager beauftragt, diese von der Festplatte abzurufen. Befinden sich die Mediendaten nicht auf der Festplatte, setzt der Kopiervorgang ein.

Bei der Aufnahme werden freie Speicherblöcke bestimmt, und der Device-Manager veranlasst, die empfangenen Daten in die entsprechenden Speicherplätze zu schreiben. Sind keine freien Speichervolumen vorhanden, werden die nicht mehr benötigten Daten entweder zum Bandarchiv kopiert oder von diesem gelöscht.

### 3.3.4 Device-Manager (Dev-Mgr)

Der Device-Manager übernimmt die Zugriffe auf die Peripherie. Dazu gehören die Lese- und Schreibzugriffe sowie das Kopieren der Mediendaten zwischen den Peripherien. Der Zugriff auf die Peripherien erfolgt blockweise. Hier können die Peripherie-Zugriffsstrategien optimiert werden, bzw. werden hier die Zugriffsstrategien auf die Peripherien implementiert. Die Zugriffe werden über das Betriebssystem an die SCSI-Schnittstelle bzw. an die Peripherie weitergeleitet. Wie diese schließlich auf den Speichermedien gespeichert werden, regelt das darunter liegende Archiv- bzw. RAID-System.

### 3.3.5 Memory Manager (Mem-Mgr)

Der Memory Manager verwaltet den Speicherpuffer, der als Zwischenspeicher für die Mediendaten zwischen Festplatte und Client dient. Der Speicherpuffer soll in Form von FIFOs organisiert werden. Für jeden aktiven Datenstrom wird ein FIFO angelegt bzw. abgebaut, wenn der Datenstrom nicht mehr aktiv ist. Der Zugriff auf den FIFO erfolgt immer blockweise, wobei die Länge der Datenblöcke variabel sein kann.

### 3.3.6 Admission-Controller (AC)

In diesem Modul wird die mögliche Belastung der Systemressourcen errechnet. Für jeden neu aufzunehmenden Client muss der Client-Manager beim Admission-Controller eine Anfrage stellen. Dabei werden alle benötigten Dateiinformation

mitgeliefert. Der Admission-Controller errechnet daraufhin die Fehlerwahrscheinlichkeit, die zu erwarten ist, wenn der die Client-Anfrage akzeptiert wird. Liegt die Fehlerwahrscheinlichkeit unter der vorgegebenen Grenze, so wird der Service für den Client angenommen. Kriterium für die Annahme der Anfrage ist also vorgeschriebene Garantie, die der Medienserver einhalten muss. Umgekehrt müssen die nicht mehr benötigte Ressourcen vom Client-Manager dem Admission-Controller mitgeteilt werden, sobald ein Service beendet ist. In diesem Fall wird die Systembelastung erneut berechnet.

Bei der Berechnung der Systembelastung spielen für den Admission-Controller die folgenden Punkte eine wichtige Rolle :

- Datenrate der Mediendaten

Die Datenrate legt fest, welche Datenmenge pro Zeiteinheit zwischen Client und Medienserver übertragen werden soll. Je höher die Datenrate ist, desto stärker belastet ein Datenzugriff die Systemressourcen, wie Zwischenspeicher, Plattenbandbreite sowie Netzwerkbandbreite.

- Lesen, Schreiben oder Kopieren

Physikalisch bedeutet das Lesen aus der Festplatte ein Messen des Magnetfeldes und das Schreiben eine Veränderung des Magnetfeldes. Das Verändern erfordert mehr Zeit. Daher können die Daten schneller aus der Festplatte gelesen als geschrieben werden. Das Abspielen des Filmes bzw. das Auslesen der Festplatte verursacht weniger Verzögerungen als die Aufnahme bzw. das Schreiben des Films auf die Festplatte. Beim Kopiervorgang werden auch Systemressourcen (Zwischenspeicher, Plattenplatz, Plattenzugriff) benötigt. Dieser Ressourcenverbrauch ist gleich zu setzen mit dem Datentransfer von und zum Client. Das bedeutet auch, dass vor dem Kopiervorgang auch die Admission-Control für das Kopieren zwischen Festplatten und Band durchgeführt werden muss. Folglich müssen vor der Annahme eines Clients die Ressourcen für das Kopieren genauso reserviert werden.

### 3.3.7 Operating-System (OS)

Normalerweise ist das Betriebsystem die Schnittstelle der Scheduling-Algorithmen zu den Systemressourcen, der Peripherie, dem Netzwerk bzw. dem Client. Über diese Schnittstelle bekommt der Scheduler alle Daten, die er für die Verarbeitung braucht, und über das Betriebsystem werden die Ergebnisse weitergeleitet. In diesem Modell wird die Einschränkung gemacht, dass das Operating-System (OS) die Schnittstelle zum SCSI-Bus bzw. zur Peripherie ist, über die der Scheduler auf die Information und die Daten der Medien schreibend und lesend zugreifen kann. Schnittstellen zu anderen Systemressourcen werden in diesem Modell nicht betrachtet.

Im Operating-System (OS) werden alle Informationen über die Mediendaten gespeichert. Für den Zugriff auf die Daten muß der Scheduler Anfragen an Operating-System stellen. Das Operating-System prüft die Anfragen und sendet die entsprechenden Antworten zurück. Hier wird auch das Verhalten der Peripherie simuliert.

# 4 Implementierung des Modells mit OMNeT++

Die Aufgabe dieser Arbeit ist es, ein Modell für einen Medienserver aufzubauen und mit Hilfe dieses Modells Scheduling-Algorithmen zu simulieren. Dabei muss das Verhalten der Software sowie der Hardware berücksichtigt werden. Um den Aufwand für die Hardware auf das Nötigste zu reduzieren, werden die Schnittstellen sowie das Verhalten aller Hardwarekomponenten auf ein einfaches Modell reduziert. Dieses Modell berücksichtigt nur die Antwortzeit und den Ressourcenbedarf. Hier wird versucht, den Konventionen der bisher im Institut für Nachrichtentechnik erstellten Arbeiten so eng wie möglich zu genügen. Allerdings gibt es einen großer Unterschied: Die bisherigen Modelle beschäftigen sich hauptsächlich mit den Hardwarekomponenten. In dieser Arbeit soll jedoch ein Modell für die Scheduling-Algorithmen aufgebaut werden. Die Konzeption des Modells wurde deshalb derart festgelegt, dass man sie leicht als Modell für die Software verwendet werden kann. Folglich weichen die Ansätze von den bisherigen Strukturen ab.

## 4.1 OMNeT++

OMNeT++ ist ein objektorientiertes Simulationssystem für diskrete ereignisgesteuerte Simulationen und wurde an der Universität von Budapest entwickelt. Das Tool wird im Internet veröffentlicht und kann für Forschungszwecken kostenfrei genutzt werden. Der Name OMNeT++ ist eine Abkürzung für „**O**bjective **M**odular **Ne**twork **T**estbed in C++**“**.

Die Struktur eines OMNeT++-Simulationmodells besteht aus hierarchisch geordneten Modulen. Module können als einfache Module, den Simple-Modules, oder aus zusammengesetzten und verschachtelten Modulen, den Compound-Modules, bestehen, wobei die Tiefe und die Anzahl der Module eines Modells nicht beschränkt ist. Die Module eines Modells können mittels Nachrichtenaustaush (Message-Passing) miteinander kommunizieren. Diese Nachrichten können beliebig komplexe Datenstrukturen enthalten. Das Versenden der Nachrichten kann direkt von Modul zu Modul bzw. an sich selbst oder über vordefinierte Pfadstrukturen mit Hilfe von Toren (Gates) und Verbindungen (Connections) erfolgen. Das Versenden kann sofort oder zeitversetzt durchgeführt werden. Darüber hinaus können für die Verbindungen Signallaufzeit, Datenübertragungrate und Bitfehlerrate definiert werden. Jede ankommende Nachricht stellt für den Empfänger ein Ereignis dar, und dieser kann daraufhin entsprechend des Ereignistyps die erforderlichen Aktionen durchführen.

Das Verhalten des Moduls wird mit Hilfe von C++ Funktionen definiert. OMNeT++ bietet auch die Möglichkeit, die Module zu parametrisieren. Das Verhalten der Module in der Simulation kann nun durch diese Parameter eingestellt werden. Zusätzlich bietet OMNeT++ dem Benutzer der Simulation eine graphische Oberfläche, in der er die Simulation in verschiedenen Durchführungsgeschwindigkeiten starten und stoppen, Interaktionen zwischen den Modulen beobachten, die Simulation protokollieren sowie die Ergebnisse grafische auswerten kann.

Für eine ausführliche Beschreibung und für das Beziehen des Sourcecodes des Simulationstools wird hier auf die OMNeT++-Home-Page http://www.hit.bme.hu/phd/vargaa/omnetpp.html verwiesen.

## 4.2 Annahmen und Einschränkungen

Diese Arbeit beschränkt sich auf Scheduling-Algorithmen und auf die Übertragung zwischen Client und Medienserver. D. h., es werden die Datenströme zwischen Client und Medienserver gesteuert. Mögliche Effekte, die bei der Kodierung/Dekodierung oder bei Zugriffen auf die Peripherie auf physikalischer Ebene auftreten könnten, werden nicht genauer betrachtet. Es wird angenommen, dass sich physikalische Effekte nur in Form zeitlicher Verzögerungen auf die Simulation auswirken.

- Der Schwerpunkt der Simulation ist es, eine Aussage über die aufgestellte Strategie des Medienservers und das Verhalten im Grenzbereich zu machen. Augenmerk soll auch auf die Optimierung der Strategien durch die Einstellung der Hauptparameter gelegt werden.

- Da das Grenzverhalten des Medienservers untersucht werden soll, wird das Verhalten des Netzwerks in der Simulation nicht berücksichtigt. Es wird angenommen, dass das Netzwerk genügende Bandbreite für den Datentransfer zwischen Server und allen Clients zur Verfügung stellt und es zu keiner Zeit zu Engpässen kommt. Allerdings wird das Netzverhalten in Form von Übertragungsverzögerungen berücksichtigt.

- In der Simulation werden keine echten Mediendaten von Modul zu Modul übertragen, vielmehr wird die Information über die Datengröße weitergeleitet. Dabei werden die für die Weiterleitung der Datenmengen benötigte Zeit und die Systemressourcen in der Berechnung berücksichtigt.

- Die Mediendaten sind derart gespeichert, dass die Mediendaten in Form von einzelnen Speichereinheiten, wie z. B. in ‚Group of Pictures' (GoP, eine Gruppe aus I-, P-, B- Bildern) von der Platte gelesen bzw. auf die Platte geschrieben werden können.

- Die Verwaltung der physikalischen Speicherung der Mediendaten auf der Platte wird von der Festplatte oder vom RAID-System erledigt und ist nicht

Bestandteil diese Arbeit. Konventionell werden die Daten auf sekundäre Medien (Festplatten) blockweise in 1024kByte, 2048kByte usw. gespeichert. Das RAID-System und die Verzeichnisstruktur sollen in der Lage sein, die vom Medienserver geforderten Zugriffseinheiten in Speichereinheiten zu konvertieren und zu verwalten.

- Die Zwischenspeicher des Medienservers werden in Form von FIFOs verwaltet. Jeder Datenstrom hat sein eigenes FIFO. Die Größe des FIFO ist einstellbar. Die FIFOs werden derart verwaltet, dass die Mediendaten blockweise gelesen und geschreiben werden können. Die Gesamtgröße und der Füllstand des FIFOs können jeder Zeit abgelesen werden. Für das Lesen und das Schreiben der Daten aus dem bzw. in den FIFO, das Anlegen, das Löschen und das Lesen des Füllstands werden in Form Zugriffsfunktionen implementiert.

- Im Falle der Aufnahme kann der Client jeden Videofilm aufnehmen, dessen Name noch nicht benutzt ist. Existiert der Filmname bereits, wird die Aufnahme verweigert.

- Für das Abspielen wählt der Client aus einer Liste einen Videofilm aus, die auf dem Medienserver zur Verfügung steht, und nur dieser Videofilm kann sofort abgespielt werden.

- Beim Vorwärts- bzw. Rückwärts-Spulen wird lediglich der Zeiger auf die aktuelle Bildsequenz des Medienservers vor- bzw. rückwärts bewegt. Das Spulen kann nicht zu einer beliebigen Stelle, sondern nur zum Anfang einer Speichereinheit gesetzt werden, z. B. an den Anfang eines GoPs. Ein gleichzeitiges Spulen und Abspielen ist nicht möglich.

## 4.3 Simulation der Mediendaten

- **Datendatei der Filme (`*.mpg` Datei)**

Die MPEG-I-Filme werden meist in einer Datendatei mit der Dateierweiterung „`.mpg`“ gespeichert. Die Datendatei wird für die Simulation selbst nicht benutzt, aber aus dieser Datendatei können mit Hilfe verschiedener Tools die Statistikdaten, sowie alle benötigten Informationen des Filmes für die Simulation gewonnen werden.

Wie bereits erwähnt, werden bei der Simulation keine echten Mediendaten zwischen Client und Medienserver übertragen. Vielmehr wird die Größe der Dateninformation weitergeleitet, und diese bei der Weiterleitungen in Form von Verzögerungen berücksichtigt. Für die Realisierung der zu simulierenden Mediendaten sind diese Dateninformationen nötig. Damit sich die Simulation so nah wie möglich an der Wirklichkeit anlehnt, wird die Dateninformation für jeden einzelnen Videofilm erstellt und simuliert.

Mit Hilfe des Programms „mpeg_stat“ wird die Datendatei analysiert. Bei der Analyse schreibt dann ‚mpeg_stat‘ die gefundenen Daten in verschiedene Dateien. Für die Auswertung sind die Information über die Datengröße (Size) mit der Dateierweiterung „.sz“ und die Information über die Datenrate (Rate) mit der Dateierweiterung „.rt“ von Bedeutung. In der „Size“-Datei werde die Daten über Anzahl, Größe sowie Typ der einzelnen Bilder gespeichert. In der „Rate“ Datei wird die Datenrate gespeichert.

Im Rahmen diese Arbeit wurde auch das Programm ‚fileinfo‘ programmiert, mit dem aus den von ‚mpeg_stat‘ erzeugten Dateien die benötigte Information gelesen und dann die Dateiinformation für die Mediendaten erzeugt wird.

Leider wurden die von „mpeg_stat“ erzeugten Dateien nicht für eine Weiterverarbeitung konzipiert. Entsprechend wurden nicht alle Daten in den geeigneten Formaten gespeichert. Die Angabe über die Framerate ist in keiner Datei abgelegt und nur in Form von Bildschirmausgaben bekannt. Deswegen steht in der von „fileinfo“ erstellten Informationsdatei keine Angabe über die Framerate. Die Framerate muss von Hand ergänzt werden.

- **Informationsdaten des Filmes (*.inf Datei)**

Für die Steuerung benötigt der Medienserver die Information über die Mediendaten. Für die Verfügbarkeitsprüfung sind darüber hinaus die Statistikdaten der einzelnen Videofilme vonnöten. Die Statistikdaten können durch die Analyse der Mediendaten ermittelt werden. In dieser Informationsdatei stehen alle Informationen des jeweiligen Films wie die Bitrate, Gesamtgröße in Bits, Frame-Rate, Gesamtanzahl der Frames, Gesamtanzahl der Datenblöcke, Anzahl der Frames im Datenblock sowie die Datenratehistogramme.

Diese Datei kann manuell erstellt werden oder automatisch mit Hilfe eines Analysetools wie z. B. dem oben beschriebenen Programm ‚fileinfo‘.

Hier ein Beispiel für eine Informationsdatei:

```
#
# (cMsFileInfo)
#
500000      #= bitrate
50777600    #= bitsize
20          #= framerate
558         #= framesize
50          #= blocksize
20          #= frame per data block

#
# (cMsStreamHist)
#
100         #= num_cells
100000      #= bit rate per cell
```

```
1          #= cellv[] exists
 0.000000
 0.001792
 0.000000
 0.030466
 0.216846
 0.360215
 0.200717
 0.107527
 0.071685
 0.008961
 0.000000
 0.000000
 0.000000
 0.000000
...
```

- **Datenratenhistogramm**

Das Datenratenhistogramm besteht aus einer festen Anzahl von Zellen. Eine Zelle stellt einen Datenratenbereich dar. Die Anzahl der Zellen wird willkürlich auf 100 Zellen und der Datenratenbereich auf 100000 Bit/Sek. Gesetzt. Diese Größen können je nach Bedarf entsprechend geändert werden. Die Zellen haben einen Wertebereich von null bis eins, der die Wahrscheinlichkeitsdichte der Zellen darstellen soll. Damit kann zum Beispiel eine Festplatte bis zu einer Festplatten-Transferrate vom 10.000.000 Bit/Sek. Simuliert werden.

Das Histogramm wird auch für die statistische Darstellung der Systemlast im Medienserver benutzt (siehe Kapitel 2.4.2 ).

- **Mediendaten**

Wie erwähnt, werden für die Simulation keine echten Daten gebraucht. Es wird nur die Länge der Datenblöcke, deren erforderlichen Systembelastungen und für deren Bearbeitung die verursachten Verzögerungen benötigt. Diese Daten können durch einen Zufallsprozess erzeugt werden. Für die Erzeugung der Daten des Datenstroms werden die charakteristischen Daten in der Informationsdatei genutzt.

## 4.4 Datenstrom

In diesem Modell ist der Datenstrom der logische Übertragungskanal für Kommandos und Daten zwischen dem Client und dem Medienserver, sowie zwischen den Modulen im Medienserver. In einem Datenstrom können aber nur Kommandos und Daten für ein Video übertragen werden. Die gleichzeitige Übertragung von Kommandos und Daten verschiedener Videos ist nicht möglich. Allerdings können über den selben Datenstrom nacheinander verschieden Video übertragen werden. Natürlich müssen bestimmte Regel eingehalten werden, um eine gesicherte Übertragung zu garantieren. Diese Regel wird später genauer erläutert.

Mit einem Verbindungsaufbau wird ein Datenstrom aktiviert, der daraufhin dem Client fest zugeordnet ist. Der Datenstrom wird wieder abgebaut, sobald der Client einen Verbindungsabbruch durchführt. Mit der Quittung für den Verbindungsaufbau sendet der Medienserver eine Nachricht mit der Kennung `STREAM_ID` an den Client zurück. Diese `STREAM_ID` ist die eindeutige Identifizierung des Datenstroms und muss mit jedem weiteren Kommando mitgeliefert werden. Die `STREAM_ID` wird für jeden Verbindungsaufbau nur einmal vergeben und wird nach dem Verbindungsabbruch auch nicht wieder benutzt. Damit wird gewährleistet, dass die Identifizierung immer eindeutig ist.

Ist eine Datenverbindung zwischen Client und Medienserver vorhanden, kann der Client Anfragen an den Medienserver schicken und ein Video entweder abspielen oder aufnehmen lassen. Der Zugang hängt natürlich von den verfügbaren Ressourcen des Medienservers ab.Wird der Zugang erlaubt und die entsprechenden Ressourcen für das Video reserviert, kann das Video abgespielt oder aufgenommen, gestoppt, vorwärts und rückwärts gespult (gilt nur für das Abspielen) und wieder gestartet werden. Diese Aktionen kann der Client für das selbe Video beliebig oft wiederholen.

Wenn der Client ein anderes Video haben oder zwischen Abspielen und Aufnahme wechseln will, muss er bestimmte Regeln einhalten. Diese Regeln sind darin begründet, dass der Medienserver für das Abspielen und die Aufnahme jedes Videos die Verfügbarkeitsprüfung durchführen muss, denn ein anderes Video kann ganz anderen Ressourcen beanspruchen. Diese Regeln schreiben vor, dass der Client die reservierten Ressourcen freigeben muss und für das neue Video oder den neuen Service erneut anfragen muss. Die Freigabe kann mit dem Befehl `MS_END` signalisiert werden. Der Medienserver setzt mit diesem Befehl den Datenstrom auf den Anfangszustand zurück und gibt die reservierten Ressourcen frei. Dieser Anfangszustand entspricht dem Zustand nach dem Verbindungsaufbau. In diesem Fall wird die Verbindung nicht abgebrochen und die `STREAM_ID` bleibt erhalten.

Bei einem Verbindungsabbau wird der Medienserver alle Ressourcen freigeben. Der Datenstrom mit der Identifizierung `STREAM_ID,` und somit die Verbindung, wird geschlossen. Nach dem Empfang der Quittung werden alle Kommandos mit der `STREAM_ID` ignoriert.

## 4.5 Simulation des Clients

Die Simulation des Clients lehnt sich an die Arbeit „Client-Modellierung in einem Medienserver-Simulationssystem“ vom H. Kaltenbach [Kal99] an. Allerdings wurde das Modell leicht modifiziert, sowie die Formate der Kommandos entsprechend den Anforderungen geändert.

### 4.5.1 Steuerdatei

Die Simulation erlaubt eine Festlegung der Anzahl der Clients vor dem Simulationstart. Jeder Client hat eine eigene Steuerdatei, die die Kommandos der Benutzer simulieren soll. Deshalb muss vor Simulationsstart gesichert sein, dass alle Steuerdateien vorhanden sind. Ansonsten entsteht ein Fehler und die Simulation wird abgebrochen. Die Steuerdatei hat den Name „`clientxx.cmd`“, wobei die beiden „`x`“ den Index des Clients bezeichen. Alle Kommandos eines Benutzers stehen in dieser Steuerdatei. Jede Zeile der Datei stellt ein Kommando für die nächste Aktion dar.

Die einzelnen Zeilen in der Steuerdatei haben das folgende Format:

```
[command]   [time]      [ ... ]
```

In `[command]` stehen alle Benutzer-Kommandos für den Medienserver. Bei `[time]` ist die Zeit in Sekunden zu finden, in der das Kommando gelten soll, bevor die nächste Zeile gelesen wird. Das Kommando gilt unabhängig davon, ob in dieser Zeit irgendeine Aktion aktiv ist oder nicht. Für `[ ... ]` können weitere Parameter des Kommandos hinzukommen.

In der Steuerdatei können folgende Kommandos mit den entsprechenden Formaten stehen:

- **`CONN_REQ  time`**

Mit diesem Kommando wird eine Anfrage mit einem Verbindungsaufbau und Aktivierung eines Datenstroms veranlasst, danach wird die Quittung `CONN_ACK` mit der Identifizierung des Datenstroms `STREAM_ID` erwartet.

- **`MS_PLAY  time  name`**

Dieses Kommando veranlasst die Anfrage für das Abspielen eines Videofilmes. Unabhängig von der Abspieldauer des Videofilmes wird dieses Kommando für die Dauer von `[time]` gelten. Dieses Kommando soll von dem Kommando

MS_STOP gefolgt werden. Ist die Abspieldauer länger als die Zeit time soll das folgende Stop-Kommando die Aktion beenden. Ist dagegen die Abspieldauer kürzer, erhält der Client eine Nachricht vom Medienserver, dass der Videofilm endet und veranlasst, dass das nächste Kommando aus der Datei zu lesen ist.

- **MS_STOP time**

Mit diesem Kommando wird die gerade aktive Aktion gestoppt.

- **MS_WAIT time**

Mit diesem Kommando wird keine Aktion gestartet, sondern nur abgewartet. In diesem Zeitraum gibt es auch keinerlei Aktivitäten zwischen Client und Medienserver.

- **MS_FORWARD time forward_time**

Dieses Kommando darf nur im Zusammenhang mit einem vorausgegangenen MS_STOP und während des Abspielens benutzt werden. Das Vorwärts-Spulen für die Abspielzeit von forward_time wird veranlasst.

- **MS_REWIND time rewind_time**

Mit der selben Regel wie bei MS_FORWARD wird bei diesem Kommando das Rückwärts-Spulen veranlasst.

- **MS_REC time name**

Dieses Kommando veranlasst eine Anfrage an den Medienserver für die Aufnahme eines Videofilmes. Unabhängig von der Aufnahmedauer des Videofilmes gilt dieses Kommando für die Dauer von time. Dieses Kommando soll von dem Kommando MS_STOP gefolgt werden. Ist die Aufnahmedauer länger als die Zeit time, soll das folgende Stop-Kommando die Aktion beenden. Ist dagegen die

Aufnahmedauer kürzer, so veranlasst der Client, das nächste Kommando aus der Datei zu lesen.

- **MS_END time**

Dieses Kommando gibt die reservierten Ressourcen wieder frei und versetzt den Datenstrom wieder in den Anfangszustand zurück. Dieses Kommando ist erforderlich, wenn der Client einen neuen Service (neues Abspielen oder neue Aufnahme) vom Medienserver anfragen will oder er sich auf Verbindungsabbruch vorbereitet.

- **DISC_REQ time**

Dieses Kommando veranlasst den Verbindungsabbruch. Ein Verbindungsabbruch bedeutet nicht unbedingt das Ende der Steuerdatei, sondern es können nach einer Wartezeit `time` weitere Verbindung aufgebaut werden und weitere Aktionen folgen.

Ein Beispiel:

```
MS_WAIT     2
CONN_REQ    5
MS_PLAY     30    xmen
MS_STOP     10
MS_REWIND   2.5
MS_PLAY     30    xmen
MS_STOP     10
MS_END      1
MS_REC      30    stargate
MS_STOP     10
MS_END      1
DISC_REQ    30
CONN_REQ    5
MS_PLAY     30    stargate
MS_STOP     10
MS_END      1
DISC_REQ    30
```

### 4.5.2 Client-Modul

Die Aufgabe des Clients wurde im Unterkapitel 3.2 beschrieben. Das Client-Modul beschränkt sich auf die Simulation der Benutzerschnittstelle und die Kommunikation mit dem Medienserver. Das Client-Modul liest die Benutzerkommandos aus der Steuerdatei, wandelt die Kommandos in die geeigneten Nachrichten um und schickt diese an den Medienserver. Abhängig vom Benutzerkommando und von der Quittung des Medienservers wird entsprechend reagiert. Für die Gesamtsimulation ist das Abspielen und die Aufnahme eines Videofilms von besonderer Bedeutung, deswegen wird der Ablauf hier kurz erläutert.

Für das Abspielen sendet das Client-Modul eine Anfrage-Nachricht mit dem Namen des gewünschten Films an den Medienserver. Erhält es eine positive Quittung, so erwartet es die Mediendaten im regelmäßigen Zeitintervallen. Die Datenmengen und Ankunftszeiten werden protokolliert. Erhält es vom Benutzer in dieser Phase ein Kommando, z. B. ein Stop, so wird sofort eine Stop-Nachricht an den Medienserver geschickt. Beim Abspielen ist der Client in einer passiven Rolle, deshalb protokolliert das Client-Modul hauptsächlich die ankommenden Daten.

Bei der Aufnahme ist das Client-Modul der aktivere Teil. Es muss alle Informationen des Videofilms aus der Dateiinformation (siehe 4.3 „Simulation der Mediendaten“) lesen und in eine Datenstruktur ‚cMsFileInfo‘ packen. Diese Datenstruktur wird mit der Anfrage für die Aufnahme an den Medienserver geschickt. Die Daten in der Datenstruktur benötigt der Medienserver für seine Verfügbarkeitsprüfung. Erhält es eine positive Quittung, so muss es ab diesem Zeitpunkt die simulierten Mediendaten in regelmäßigen Zeitintervallen schicken, d. h., es muss die Daten aus der Dateiinformation nutzen und die Datenmenge des Videofilmes mit einem Zufallsprozeß erzeugen, diese dann in eine Nachricht verpacken und an den Medienserver senden. Die Anzahl der Datenblöcke muss bei der Datenrate,

der Framerate, usw. berücksichtigt werden. Ist das Filmende erreicht, so muss das Ende in einer Nachricht dem Medienserver mitgeteilt werden.

## 4.6 Die Simulation des Medienservers

### 4.6.1 Prozesse und Unterprozesse

Wie bereits im Kapitel 3.3 „Beschreibung des Medienservers“ erläutert wurde, besteht der Medienserver aus mehreren voneinander unabhängigen Prozessen, dem „Network-Manager“, dem „Client-Manager“, dem „Directory-Manager“, dem „Device-Manager“, dem „Memory-Manager“, dem „Admission-Controller“ und dem „Operating-System“. Diese Prozesse haben ihre eigenen Aufgaben und sind verantwortlich für die Verarbeitung aller Datenströme.

Um die Datenströme voneinander zu unterscheiden, erhält jeder Datenstrom ein eigener Nummer für die Identifizierung, die `STREAM_ID`. Die `STREAM_ID` wird für jeden Verbindungsaufbau vom Client neu vergeben und beim Verbindungsabbau wieder gelöscht. Damit die Datenströme getrennt und unabhängig voneinander verwaltet werden können, wird ein ‚Eltern-Kind‘-Prozessstruktur verwendet. D. h., für jeden aktiven Datenstrom wird von allen Manager-Prozessen (Elternprozess) ein Unterprozess (Kindprozess) dynamisch erzeugt, jedem Kindprozess wird ein Datenstrom zugewiesen. Der Kindprozess wird gelöscht, wenn der entsprechende Datenstrom nicht mehr aktiv ist. Alle Ereignisse bzw. Nachrichten werden zwar zuerst zwischen den Elternprozess ausgetauscht, die Elternprozess führen aber nur die Vorverarbeitung durch. Danach werden die Ereignisse für die Verarbeitung an den Kindprozess weitergeleitet. Diese Prozessstruktur hat den Vorteil, dass alle Daten und deren Speicherplätze, die für die Verarbeitung der Ereignisse eines Datenstroms erforderlich sind, unabhängig voneinander in einem Kindpro-

zess erzeugt, verwaltet, verarbeitet und abgebaut werden können. Um das Weiterleiten der Ereignisse an den richtigen Kindprozess zu ermöglichen, müssen sowohl die Kindprozesse und als auch die Ereignisse mit der jeweiligen `STREAM_ID` verknüpft sein.

Die gerade beschriebene Prozessstruktur gilt für alle Prozesse mit Ausnahme des „Admission-Controller“ Prozesses. Dies ist dadurch begründet, dass der „Admission-Controller“ eine zentrale Instanz des Medienservers ist und keine spezielle Verarbeitung für den Datenstrom durchführt. Seine Aufgabe ist die Verwaltung der Systemressourcen im Medienserver.

Alle Prozesse, Eltern- und Kindprozesse, werden jeweils in einem OMNeT++ Modul realisiert.

### 4.6.2 Simulation der Peripherie

Für die Simulation der Peripherie wird eine virtuelle Verzeichnisstruktur in der Verzeichnisstruktur des Simulationsrechners angelegt.

Das ‚Root‘-Verzeichnis des Medienservers startet bei `RootDir`. Es enthält die Datenstrukturen `DiscDir` und `TapeDir`, welche die Festplatte und das Bandarchive darstellen. Die in der Dateistruktur `DiscDir` und `TapeDir` stehenden Daten- und Statistikdateien stellen die auf der Festplatte oder des Bandarchives gespeicherntene Mediendaten dar. Um die Simulation nicht unnötig zu verkomplizieren, wird die Annahme gemacht, dass sich alle Mediendaten der Festplatte und des Bandarchivs in einem Verzeichnis befinden. Unterverzeichnisse sollen nicht verwendet werden.

In der jeweiligen Datenstruktur werden alle Dateiinformationen des jeweiligen Films gespeichert. Die Existenz einer Dateiinformation bedeutet die Existenz ei-

nes Filmes. Die Gesamtgröße aller Daten in der Datenstruktur bedeutet die Gesamtbelegung der Festplatte bzw. des Bandarchives. Die Gesamtkapazität kann beliebig gewählt und in der Simulation berücksichtigt werden.

Für das Kopieren der Mediendaten vom Bandarchiv zur Festplatte und umgekehrt werden die Datendateien von einer Datenstruktur zur anderen kopiert. Die Verzögerung und der Ressourcenbedarf des Kopiervorgangs werden in der Simulation durch das verzögerte Lesen und Schreiben berücksichtigt.

Für die Festplatte und das Bandarchive existiert eine Verzeichnisdatei. In diese Datei werden die Namen aller Filme der Peripherie gespeichert. In jeder Zeile steht der Name eines Films. In der Initialisierungsphase der Simulation liest das Modul „Operation-System OS“ die beiden Dateien `disc.dir` und `tape.dir`. Für jeden Namen in der Datei wird die entsprechende Informationsdatei gesucht. Der Name der Informationsdatei entspricht dem Namen des Films mit der Dateierweiterung `.inf`. Die Inhalte der Informationsdatei werden in einer speziellen Datenstruktur gespeichert. Diese Datenstruktur wird dann in das Verzeichnis der Peripherie geschrieben. Das Verzeichnis der Peripherie wird mit Hilfe der objektorientierten Datenstruktur `map der C++-Standard-Template-Library` (STL) implementiert. Mit dieser Datenstruktur läßt sich mit Hilfe des Filmnamens die entsprechende Informationsdateien des Films finden.

### 4.6.3 Name der Ereignisse

Wegen der großen Anzahl der Prozesse im Medienserver und der Anzahl der Ereignisse zwischen dem Client und dem Medienserver sowie zwischen den Modulen im Medienserver wird eine Namens-Konvention für die Ereignisse festgelegt, damit aus dem Namen eines Ereignisses auch sofort erkannt werden kann, von welchem Modul das Ereignis initialisiert, für welches Modul das Ereignis gedacht und welche Aktion damit verbunden ist. Es wird folgende Regel festgelegt:

Alle Module enthalten eine Abkürzung mit der die Module eindeutig identifiziert werden können. Hiermit werden die Unterschiede zwischen Client-Server-Kommunikation und Kommunikation zwischen den Modulen im Medienserver deutlich.

- Client-Server-Kommunikation:

| **Modul** | **Abkürzung** |
|---|---|
| Client | `CLT` |
| Medienserver | `MS` |

Der Name aller Ereignisse vom Client zum Medienserver soll mit dem Präfix ‚`CLT_`' beginnen. Der Name aller Ereignisse von Medienserver soll mit dem Präfix ‚`MS_`' beginnen.

- Kommunikation zwischen den Modulen im Medienserver:

| **Modul** | **Abkürzung** |
|---|---|
| Network-Manager | `NWK` |
| Client-Manager | `CLT` |
| Directory-Manager | `DIR` |
| Memory-Manager | `MEM` |
| Device-Manager | `DEV` |
| Admission-Controller | `AC` |
| Operating-System | `OS` |

Der Name aller Ereignisse zwischen den Modulen im Medienserver soll mit dem Präfix aus der Abkürzung des Senders und des Empfängers beginnen. Ein Ereignis das vom Client-Manager stammt und zum Directory-Manager gesendet wird, soll z. B. mit dem Präfix ‚`CLTDIR_`' beginnen, umgekehrt soll der Antwortname mit dem Präfix ‚`DIRCLT_`' beginnen.

Die Namen der Ereignisse für das Modul „Operating-System“ unterliegen nicht dieser Regel. Normalerweise wird auf die Ressourcen des Betriebsystems über die Systemfunktionen zugegriffen. Die Namen wurden so gewählt, dass man die verwendeten Systemfunktion erkennen kann . Der Name aller Ereignisse des Moduls „Operating-System“ beginnt mit dem Präfix `OS_` und seine Quittung fängt mit `OSRET_` an.

Allen Ereignisnamen sowie deren Parameter werden im Anhang 5.1 „Schnittstellen“ beschrieben. Die Namen der Ereignisse können in dem folgenden Ablaufdiagramm für das Abspielen und die Aufnahme eines Videofilmes demonstriert werden.

### 4.6.4 Abspielen

In Abbildung 4-1 werden die Abläufe für das Abspielen eines Videofilms von der Festplatte dargestellt. Es wurde vorausgesetzt, dass genügend Systemressourcen für das Abspielen vorhanden sind.

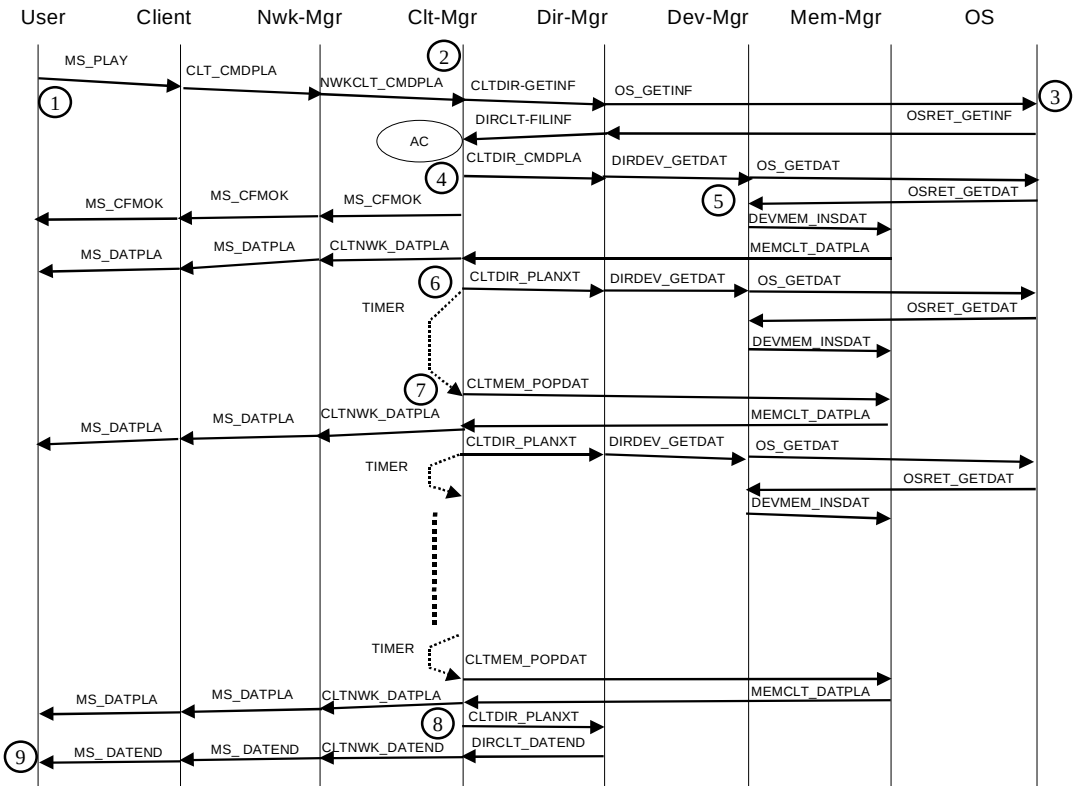



**Abbildung 4-1 Die Abläufe für das Abspielen eines Videofilmes**

Das Abspielen wird (1) vom Benutzer initiiert. Beim Erhalt dieses Kommandos sendet der Client die Anfrage mit dem gewünschten Filmnamen zum Medienserver. Im Medienserver übernimmt der Client-Manager die Anfrage. Der Client-Manager überprüft die Existenz des Videofilms durch eine Anfrage an den Directory-Manager (2). Der Directory-Manager liest darauf die Dateiinformation aus dem Verzeichnis (3) und gibt diese Information an den Client-Manager zurück. Existiert die Dateiinformation so reserviert der Client-Manager die erforderlichen Ressourcen beim Admission-Controller. Bei der Reservierung wird die Dateiinformation mitgeliefert. Mit Hilfe der Dateiinformation berechnet der Admission-Controller die Gesamtlast einschließlich dem zusätzlichen Datenstrom und überprüft, ob die mögliche Fehlerwahrscheinlichkeit unter dem gegebenen Wert liegt. Ist dies der Fall, so wird die Anfrage für die Reservierung positive quittiert, ansonsten wird die Anfrage abgelehnt. Sind die nötigten Ressourcen vorhanden (4), sendet der Client-Manager dem Client eine positive Quittung und bereitet sich auf

das Abspielen vor. Dabei wird die Dauer der Client-Servicezyklen sowie die Anzahl der Datenblöcke aus den Daten der Informationsdatei errechnet und der Directory-Manager beauftragt, die Datenblöcke zu holen (5). Parallel dazu teilt er dem Memory-Manager mit, dass er die Datenblöcke erwartet. Danach wartet der Client-Manager auf die ersten Datenblöcke. Der Directory-Manager bestimmt die Speicherorte der Datenblöcke und beauftragt den Device-Manager, die Datenblöcke von der Festplatte zu lesen. Sind die Daten gelesen, werden sie zuerst an den Memory-Manager weitergeleitet und in den Zwischenspeicher geschrieben. Der Memory-Manager erkennt die Daten und leitet sie an den Client-Manager weiter. Dieser kann daraufhin die Datenblöcke sofort an den Client weiterleiten (6). Sind die ersten Datenblöcke geliefert, beauftragt er den Directory-Manager erneut, die nächsten Datenblöcke zu holen und legt sich für die Zyklusdauer schlafen. Ist die Zeit abgelaufen, wird er von einem Timer geweckt und die Prozedur wird wiederholt, bis die letzten Datenblöcke abgespielt sind (8). Ist das Ende erreicht, erhält der Client eine „Ende“-Mitteilung.

### 4.6.5 Aufnahme

In Abbildung 4-2 werden die Abläufe für die Aufnahme eines Videofilms dargestellt. Die Aufnahmedaten werden auf die Festplatte geschrieben. Es wurde vorausgesetzt, dass genügend Systemressourcen für die Aufnahme vorhanden sind.

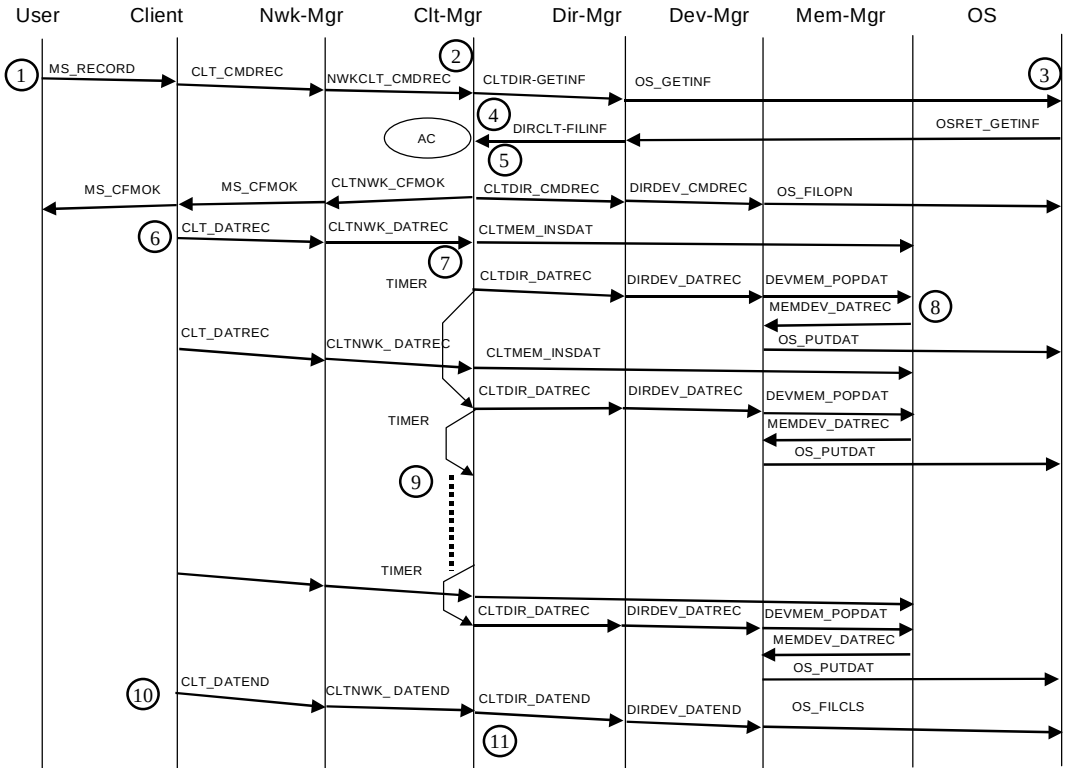



**Abbildung 4-2 Die Abläufe bei der Aufnahme eines Videofilmes**

Die Abläufe für die Aufnahme sind ähnlich denen des Abspielens. Diesmal fließen die Mediendaten vom Client zum Medienserver. In diesem Fall ist der Medienserver passiv, seine Hauptaufgabe ist es, die erhaltenen Daten auf die Festplatte zu schreiben.

Genauso wie das Abspielen wird die Aufnahme vom Benutzer initiiert (1). Wie bereits in Unterkapitel 4.5.2 „Client-Modul“ beschrieben, sammelt der Client für die Anfrage erforderliche Daten und sendet sie an den Medienserver. Hier führt der Client-Manager auch verschieden Prüfungen durch, bevor er die Annahme positive quittiert. Er vergleicht den Filmnamen mit dem Namen in dem Verzeichnis (3) und prüft, ob ein Videofilm mit dem gleichen Namen bereits existiert. Als nächstes werden die erforderlichen Ressourcen beim Admission-Controller reserviert (4). Ist der Film noch nicht auf der Festplatte und die Ressourcen sind vorhanden (5), wird eine Quittung zum Client geschickt. Danach werden die Vorbereitungen für die Aufnahme getroffen und die erwarteten Datenblöcke sowie Zy-

klusdauer aus der Dateiinformation errechnet. Parallel wird der Directory-Manager beauftragt, eine Datei für die Speicherung der Daten zu öffnen. Kommen die Daten vom Client an (6), werden diese sofort zum Memory-Manager weitergeleitet und in den Zwischenspeicher geschrieben (7). Bevor sich der Client-Manager für eine Zyklusdauer in den Schlafmodus versetzt, wird der Directory-Manager noch beauftragt, die Daten aus dem Zwischenspeicher auf die Festplatte zu schreiben. Der Directory-Manager bestimmt die Speicherplätze auf der Festplatte und beauftragt den Device-Manager die Speicherung zu übernehmen(8). Im Schlafmodus kann der Client-Manager auch die vom Client ankommenden Daten an den Memory-Manager weiterleiten und die Datenmengen werden registriert. Weitere Verarbeitungen werden nicht gemacht. Nach Ablauf der Zyklusdauer wird der Client-Manager vom Timer geweckt und die Verarbeitungsprozedur wiederholt (9). Kommt vom Client ein Stop oder ein Ende, wird die Aufnahme sofort gestoppt (10) und alle noch im Zwischenspeicher gespeicherten Daten werden auf die Festplatte geschrieben (11).

## 4.7 Parameter

Für die Simulation sind einige Parameter nötigt. Es werden zwei Arten von Parametern unterschieden:

1. Parameter der Mediendaten

Die Parameter der Mediendaten werden in Form von Informationsdateien zusammengefasst. Für jeden Videofilm wird eine Informationsdatei erstellt. In dieser Informationsdatei sollen alle Informationen des Videofilmes enthalten sein (eine detaillierte Beschreibung hierzu siehe Unterkapitel 4.3 „Simulation der Mediendaten“ )

2. Parameter des Clients und des Medienservers

Die Parameter des Clients und des Medienservers werden in der Modellbeschreibung deklariert. Diese Parameter können dann als feste oder einstellbare Parameter definiert werden. Die einstellbaren Parameter können entweder in der Initialisierungsdatei `omnetpp.ini` der Simulationsumgebung festgelegt oder beim Simulationsstart von OMNeT++ abgefragt werden.

Um eine flexible Konfiguration der Simulation zu ermöglichen, wird eine Reihe von einstellbaren Parametern eingeführt. Diese Parameter sind:

- Server.MsErrorProbability

Dieser Parameter definiert die Fehlerwahrscheinlichkeit, die der Medienserver für seinen Service garantieren soll. Diese Fehlerwahrscheinlichkeit wird für die Verfügbarkeitsprüfung genutzt. Beispielwert: 0,01 = 1%.

- Server.OsServiceRound

Dieser Parameter gibt die Dauer der Festplatten-Zugriffszyklen vor. Beispielwert : 1 Sek.

- ClientServiceRound

Dieser Parameter gibt die Dauer der Client-Servicezyklen vor. Dieser Parameter wird sowohl vom Medienserver als auch vom Client genutzt. Beispielwert : 1 Sek.

- Server.CopySpeed

Dieser Parameter gibt das Verhältnis der Kopiergeschwindigkeit zwischen Festplatte und Bandarchive im zur Abspielgeschwindigkeit vor. Beispielswert : 5.

- Server.ResourceMax

Dieser Parameter legt die Systemgrenze des Medienservers fest. Wie im Unterkapitel 2.4.2 „Statistischer Ansatz“ und 4.3 „Simulation der Mediendaten“ beschrieben, wird die Systembelastung in Form von Histogrammen dargestellt. Das Histogramm hat 100 Zellen. Beispielswert : 90.

- Client[*].home_dir

Dieser Parameter gibt an, in welchem Verzeichnis die Steuerdateien der Clients zu finden sind. Beispielswert: "." (das aktuelle Verzeichnis).

## 4.8 Statistik

Das Ziel dieses Abschnitts ist, das erstellte Szenario zu beschreiben und die in der Simulation gemessenen Statistiken zu interpretieren. Anhand der Statistikdaten kann das Antwortverhalten des Medienservers bzw. seines Scheduling-Algorithmus bewertet werden.

### 4.8.1 Szenario

Es wird ein einfaches Szenario mit einem Medienserver und zwei Client gewählt. Das primäre Ziel dieses Szenarios ist es, die Statistikdaten zu sammeln und weitergehende Untersuchungen zu ermöglichen.

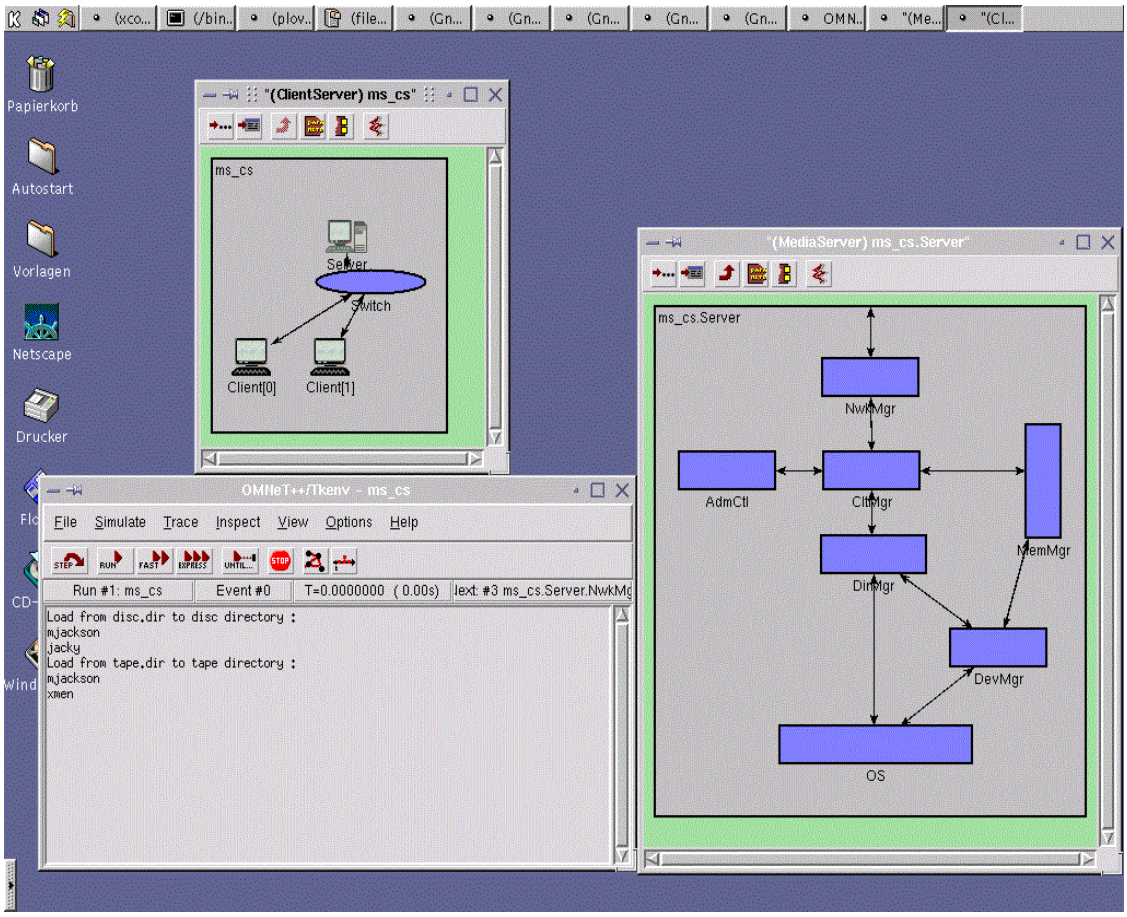



**Abbildung 4-3 Das Szenario der Simulation**

In diesem Szenario sollen für die beiden Clients zwei verschieden Videofilme abgespielt werden, die Filmdaten sollen also vom Medienserver geliefert werden. Für den „Client0“ werden die Videofilme so gewählt, daß sich der erste Videofilm auf der Festplatte und der zweiten Videofilm auf dem Bandarchiv befindet, so daß der Medienserver den zweiten Videofilm für das Abspielen vom Bandarchiv auf die Festplatte kopieren muß. Die beiden Videofilme für den ‚Client1‘ befinden sich bereits auf der Festplatte.

Für die weiteren Beschreibungen werden die Aktionen und Statistikdaten des „Client0“ betrachtet.

- In der Steuerdatei „`client0.cmd`“ des „Client0“ werden folgende Kommandos vorgegeben:

```
MS_WAIT     0.5
CONN_REQ    10
MS_PLAY     30    jacky
MS_STOP     10
MS_END      5
MS_PLAY     30    xmen
MS_STOP     10
MS_END      1
DISC_REQ    10
```

„Client0“ soll also nach dem Verbindungsaufbau den Videofilm „jacky“ für 30 Sekunden und nach einer Pause von 15 Sekunden den Videofilm „xmen“ für 30 weitere Sekunden abspielen.

- Folgenden Parameter werden für die Simulation benutzt:

```
MsErrorProbability = 0.1
BandwithMax = 90
CopySpeed = 4
OsServiceRound = 1.0
ClientServiceRound = 1.0
```

Diese Parameter legen fest, daß die Fehlerwahrscheinlichkeit für den Admission-Controller nicht höher als 10% sein darf, die Bandbreite der Festplatte bei 9.000.000 Bit/Sek liegt, die Kopierdatenrate viermal höher als beim Abspielen ist und die Zeiten sowohl für den Peripherie-Zugriff als für den Client eine Sekunde sind.

- Die Mediendaten haben folgenden Inhalt:

| **Filmname** | **Bitrate** | **Gesamtgröße in Bits** | **Framerate** | **Framezahl** | **Anzahl der Blöcken** | **Bild pro Datenblock** |
|---|---|---|---|---|---|---|
| jacky | 604450 | 28932380 | 25 | 1210 | 82 | 14 |
| xmen | 604450 | 28932380 | 25 | 1210 | 41 | 30 |

Hier werden Filme mit gleicher Daten- und Framerate aber mit verschiedener Anzahl der Bilder pro Datenblock (GoP) benutzt.

## 4.8.2 Statistikdaten

Die folgenden Abbildungen zeigen die Statistikdaten des „Client0“ für die ersten 100 Sekunden. Anhand dieser Abbildungen können die Statistikdaten erklärt und Rückschlüsse auf das Verhalten des Medienservers gemacht werden.

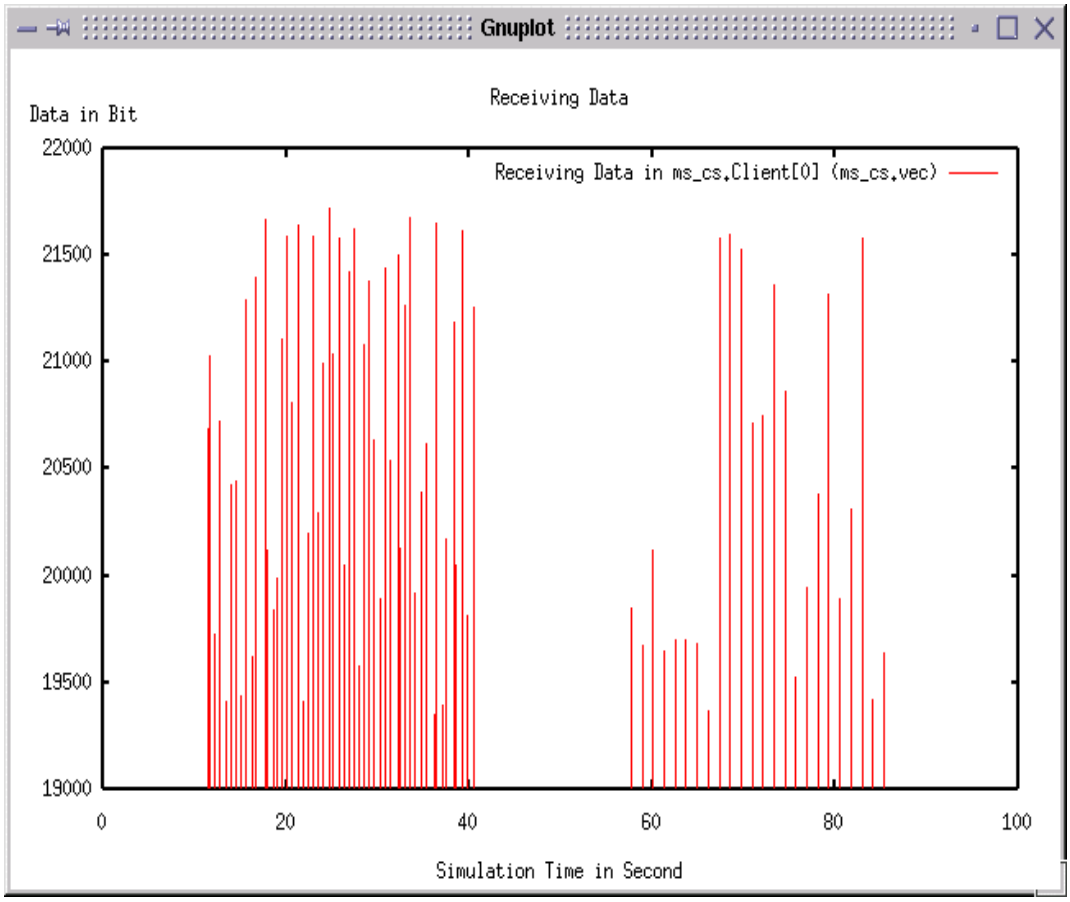



**Abbildung 4-4 Die Größe der Datenpakete**

In der Abbildung Abbildung 4-4 werden die Größe der Datenpakete aufgezeigt, die der „Client0“ bekommt. Die Größe der Datenpakete wird mit einem Zufallprozeß im Modul „Operating-System“ erzeugt und zum „Client0“ weitergeleitet. Die starken Schwankungen der Datengröße soll die statistische Eigenschaft der Mediendaten simulieren.

In der Abbildung ist sichtbar, daß die Anzahl der Datenpakete, die pro Zeiteinheit beim Client ankommen, für den ersten Videofilm höher ist als bei dem zweiten. Dies ist darin begründet, dass die Anzahl der Bilder pro Datenblock (GoP) beim ersten Videofilm kleiner ist.

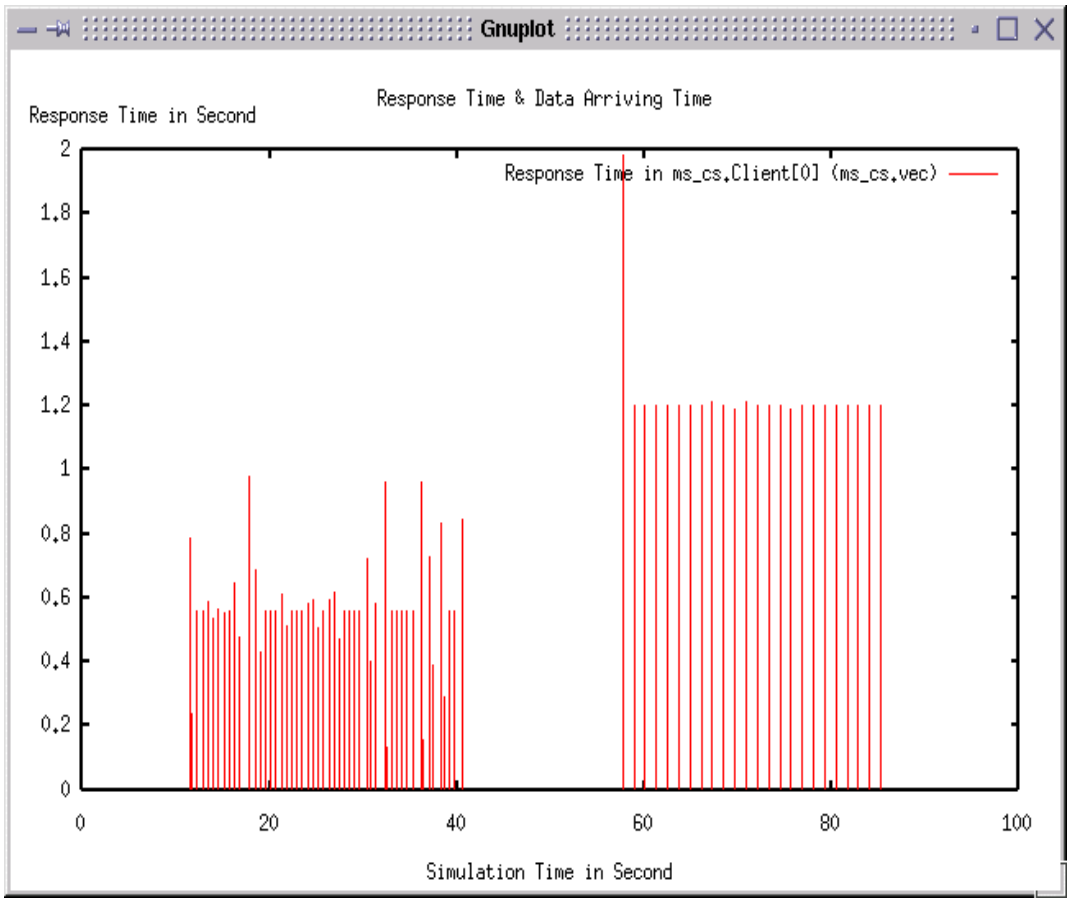



**Abbildung 4-5 Die Ankunftzeit der Datenpakete**

In Abbildung 4-5 sind die Zeitabstände zwischen den Datenpaketen dargestellt. Die Totzeit zu Beginn ist der zeitliche Abstand zwischen dem Client-Kommando zum Empfang des ersten Datenpakets, also die Anfangsverzögerung. Der erste Videofilm wird von der Festplatte abgespielt, entsprechend ist die Anfangsverzögerung kürzer als beim zweiten Videofilm. Der Zugriff auf das Bandarchiv und das Kopieren der Daten zur Festplatte verursacht eine lange Anfangsverzögerung.

Aus diesen Statistikdaten ist auch ersichtlich, daß die Client-Servicezyklen für den Datenstrom an die Abspieldauer der Datenblöcke automatisch angepasst werden. Obwohl in der Konfiguration der Client-Servicezyklus mit einer Senkunde vorgeben ist, wird für die beiden Filme eine Anpassung gemacht, die zu einem Service-Zyklus größer zwei Sekunden führt. Der erste Videofilm braucht zwei Datenblöcke pro Zyklus, die Zeitabstände zwischen den Datenblöcken ist ca. 0,58 Sekunden. Der zweite Videofilm benötigt einen Datenblock pro Zyklus, entsprechend ist der Zeitabstand 1,2 Sekunden.

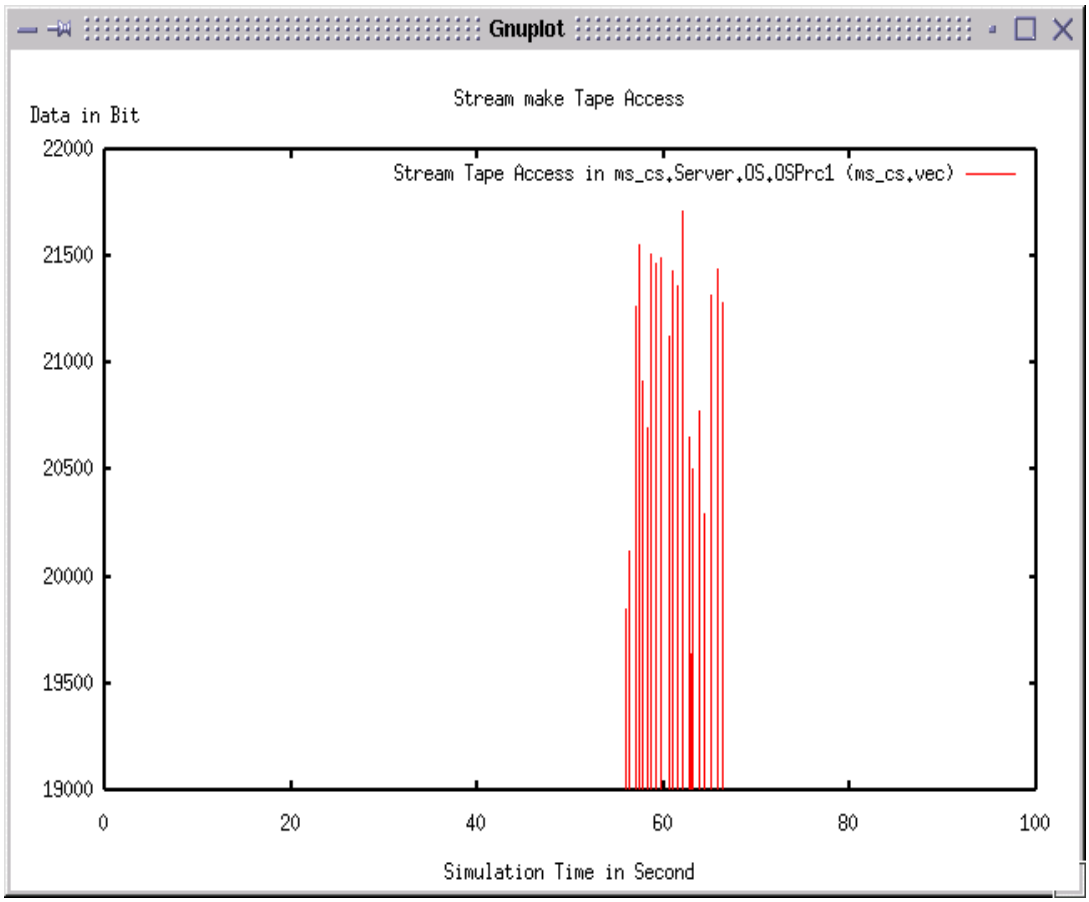



**Abbildung 4-6 Die Zugriffe auf das Bandarchiv**

Abbildung 4-6 zeigt die für das Kopieren des zweiten Videofilms erforderlichen Zugriffe auf das Bandarchiv. Das Kopieren wird mit einer höheren Datenrate durchgeführt. Aus der Abbildung ist zu erkennen, daß die Anzahl der Zugriffe bzw. die Anzahl der zugegriffenen Datenblöcke pro Zeiteinheit höher sind.

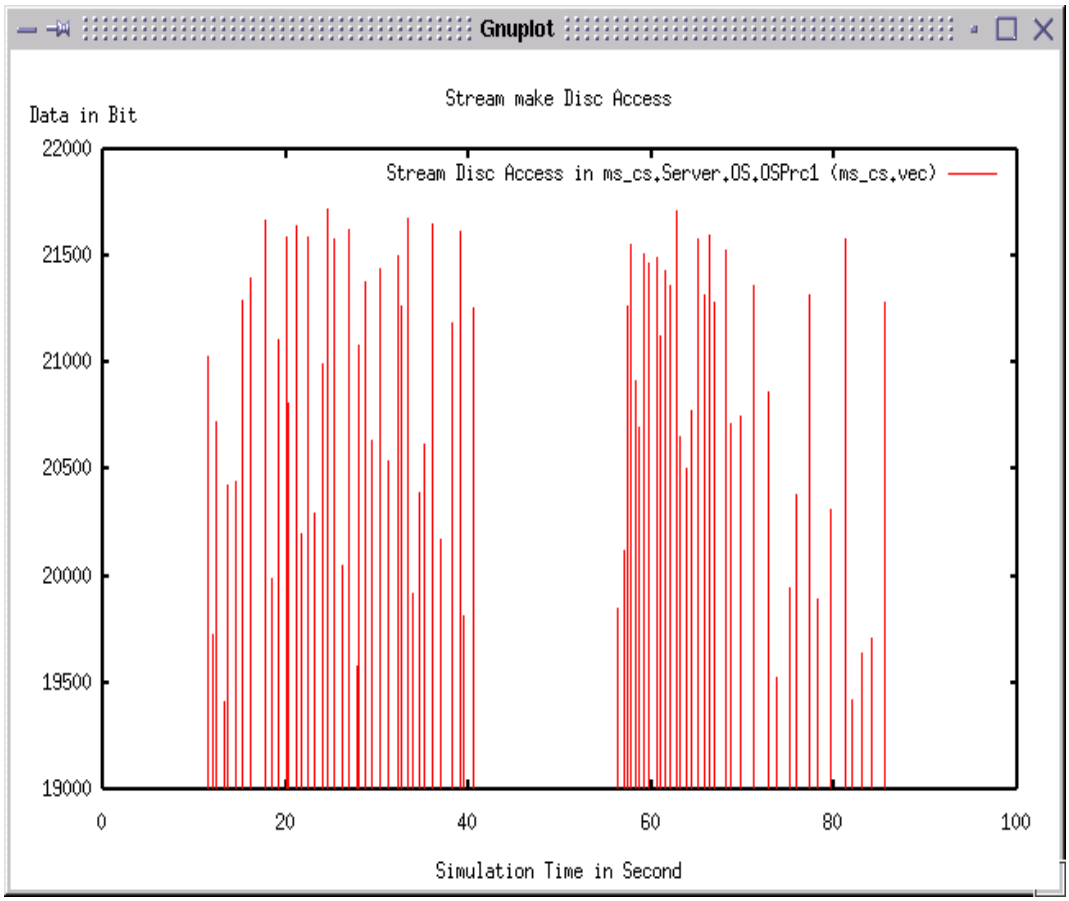



**Abbildung 4-7 Die Zugriffe auf die Festplatte**

Abbildung 4-7 zeigt die Zugriffe auf die Festplatte. Bei einem Vergleich mit Abbildung 4-4 wird ersichtlich, daß für den ersten Videofilm die Statistikdaten identisch sind. Die von der Festplatte gelesene Datenmenge wird auch vom Client empfangen. Darüber hinaus lässt sich erkennen, dass beim zweiten Videofilm für die Dauer des Kopiervorganges neben den Lesezugriffen auch Schreibzugriffe stattfinden. Nach Ende des Kopiervorgangs entspricht das Bild für den Festplattenzugriff dem Datenempfang des Clients (Abbildung 4-4).

## 4.9 Zusammenfassung und Ausblick

In diese Arbeit wurden für die Scheduling-Algorithmen verschiedene Ansätze und Verfahren diskutiert. Die Untersuchung umfasst die Analyse der Mediendaten, das Festplatten-Scheduling als auch die Analyse der Verfahren für Datenweiterleitung an den Client und die statistische Verfügbarkeitsprüfung. Es wurde ein Modell für den Scheduler entwickelt und in dem Simulationstool OMNeT++ implementiert. Die gemessenen Statistiken zeigen, daß dieses Modell das Verhalten des Medienservers gut simuliert und für die weitere Analyse des Medienservers genutzt werden kann.

In dieser Arbeit wurde das Modell für die Zugriffe auf die Peripherie und ihr Verhalten auf einfacheren Zufallprozessen aufgebaut, die dazu dienen, die Scheduling-Algorithmen zu überprüfen. Damit können qualitative Aussagen über das Verhalten gemacht werden. Quantitative Aussage sowie Aussagen über das Verhalten des Medienserver im Grenzbereich kann das Modell nicht liefern. Es ist aber so konzipiert, daß es für weitere Verfeinerungen oder die Intergration mit anderen Modellen offen steht. Damit kann es als Komponent in ein umfangreicheres Modells integriert werden.

# 5 Anhang

## 5.1 Schnittstellen

### 5.1.1 Schnittstellen zwischen Client und Medienserver

#### 5.1.1.1 Message des Client zum Medienserver

- Verbindungsaufbau

| | | |
|---|---|---|
| Message | `CLT_CONREQ` | Client Command Connection Request |
| Parameter | `Client_ID` | Client Nummer |

- Verbindungsabbau

| | | |
|---|---|---|
| Message | `CLT_DSCREQ` | Client Command Disconnection Request |
| Parameter | `Stream_ID` | Stream Nummer |

- Abspielen eines Filmes

| | | |
|---|---|---|
| Message | `CLT_CMDPLA` | Client Command Play |
| Parameter | `Stream_ID` | Stream Nummer |
| | `Fileinfo` | Dateiinformation und Filmname |

- Aufnahme eines Filmes

| | | |
|---|---|---|
| Message | `CLT_CMDREC` | Client Command Record Data |
| Parameter | `Stream_ID` | Stream Nummer |
| | `File_info` | Dateiinformation und Filmname |

- Vorwärts Spulen

| | | |
|---|---|---|
| Message | `CLT_CMDFWD` | Client Command Forward |

| | | |
|---|---|---|
| Parameter | `Stream_ID` | Stream Nummer |
| | `Time` | Vorwärst Spuldauer |

- Rückwärts Spulen

| | | |
|---|---|---|
| Message | `CLT_CMDREW` | Client Command Rewind |
| Parameter | `Stream_ID` | Stream Nummer |
| | `Time` | Rückwärst Spuldauer |

- Zurücksetzen

| | | |
|---|---|---|
| Message | `CLT_CMDEND` | End Current Action |
| Parameter | `Stream_ID` | Stream Nummer |

- Anhalten letzte Aktion

| | | |
|---|---|---|
| Message | `CLT_CMDSTP` | Client Message Stop Current Action |
| Parameter | `Stream_ID` | Stream Nummer |

- Aufnahmedaten

| | | |
|---|---|---|
| Message | `CLT_DATREC` | Client Message Record Data |
| Parameter | `Stream_ID` | Stream Nummer |
| | `Size` | Datengröße |
| | Data | Daten |

#### 5.1.1.2 Messages vom Medienserver zum Client

- Bestätigung

| | | |
|---|---|---|
| Message | `MS_CFMOK` | Server Confirm OK |
| Parameter | `Stream_ID` | Stream Nummer |

- Fehlermeldung

| | | |
|---|---|---|
| Message | `MS_CFMERR` | Server Confirm Error |
| Parameter | `Stream_ID` | Stream Nummer |

- Bestätigung der Verbindungsaufbau

| | | |
|---|---|---|
| Message | `MS_CONACK` | Server Confirm Connection Request |
| Parameter | `Stream_ID` | Stream Nummer |

- Bestätigung der Verbindungsabbau

| Message | MS_DSCACK | Server Confirm Disonnection Request |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |

- Abspieldaten

| Message | MS_DATPLA | Server Send Play Data |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |
| | Data | Daten |

## 5.1.2 Message zwischen Modulen des Medienservers

### 5.1.2.1 Messages vom Network-Manager zum Client-Manager

- Anlegen eines neue Stream

| Message | NWKCLT_CRTSTR | Create Stream |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |

- Stream Löschen

| Message | NWKCLT_DLTSTR | Delete Stream |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |

- Film Abspielen

| Message | NWKCLT_CMDPLA | Play Stream |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |
| | File_info | Dateiinformation und Filmname |

- Film Aufnehmen

| Message | NWKCLT_CMDREC | Record Stream |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |
| | File_info | Dateiinformation und Filmname |

- Vorwärts Spulen

| Message | NWKCLT_CMDFWD | Forward Stream |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |
| | Time | Vorwärst Spuldauer |

- Rückwärts Spulen

| Message | NWKCLT_CMDREW | Rewind Stream |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |
| | Time | Rückwärst Spuldauer |

- Anhalten

| Message | NWKCLT_CMDSTP | Stop Stream |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |

- Zurücksetzen letzte Aktion

| Message | NWKCLT_CMDEND | Reset Stream |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |

- Aufnahmedaten

| Message | NWKCLT_DATREC | Record Data |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |

#### 5.1.2.2 Messages vom Client-Manager zum Network-Manager

- Bestätigung

| Message | CLTNWK_CFMOK | Confirm OK |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |

- Fehlermeldung

| Message | CLTNWK_CFMERR | Confirm Error |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |

- Abspieldaten

| Message | CLTNWK_DATPLA | Play Data |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |
| | Size | Datengröße |
| | Data | Daten |

### 5.1.2.3 Messages vom Client-Manager zum AdmissionControl

- Ressource reservieren

| | | |
|---|---|---|
| Message | `CLTAC_ALLRES` | Allocate Ressource |
| Parameter | `Stream_ID` | Stream Nummer |
| | `File_info` | Dateiinformation |

- Ressourcen freigeben

| | | |
|---|---|---|
| Message | `CLTAC_FRERES` | Free Ressource |
| Parameter | `Stream_ID` | Stream Nummer |

### 5.1.2.4 Messages vom Client-Manager zum Directory-Manager

- Neuen Stream Erzeugen

| | | |
|---|---|---|
| Message | `CLTDIR_CRTSTR` | Create Stream |
| Parameter | `Stream_ID` | Stream Nummer |

- Stream löschen

| | | |
|---|---|---|
| Message | `CLTDIR_DLTSTR` | Delete Stream |
| Parameter | `Stream_ID` | Stream Nummer |

- Stream zurücksetzen

| | | |
|---|---|---|
| Message | `CLTDIR_RESSTR` | Reset Stream |
| Parameter | `Stream_ID` | Stream Nummer |

- Filminformation holen

| | | |
|---|---|---|
| Message | `CLTDIR_GETINF` | File Info und Datenstatistik |
| Parameter | `Stream_ID` | Stream Nummer |
| | `File_info` | Dateiinformation |

- Film Abspielen starten

| Message | CLTDIR_CMDPLA | Play Stream First Data Block |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |
| | Block_amount | Anzahl der Datenblöcke |

- Film Abspielen

| Message | CLTDIR_PLANXT | Play Stream Next Blocks |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |
| | Block_amount | Anzahl der Datenblöcke |

- Filmaufnahme starten

| Message | CLTDIR_CMDREC | Record Stream First Data Block |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |
| | Block_amount | Anzahl der Datenblöcke |

- Film Aufnehmen

| Message | CLTDIR_RECNXT | Record Stream Next Blocks |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |
| | Block_amount | Anzahl der Datenblöcke |

- Vorwärts Spulen

| Message | CLTDIR_CMDFWD | Forward Stream |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |
| | Block_amount | Anzahl der Datenblöcke |

- Rückwärts Spulen

| Message | CLTDIR_CMDREW | Rewind Stream |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |
| | Block_amount | Anzahl der Datenblöcke |

- Ende letzte Aktion

| Message | CLTDIR_CMDEND | Reset Stream |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |

- Aufnahmedaten in die Festplatte schrreiben

| Message | CLTDIR_DATREC | Write Record Data |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |
| | Size | Datengröße |
| | Data | Daten |

### 5.1.2.5 Messages vom Client-Manager zum Memory-Manager

- Neuen Stream Erzeugen

| | | |
|---|---|---|
| Message | `CLTMEM_CRTSTR` | Create a new stream |
| Parameter | `Stream_ID` | Stream Nummer |

- Stream Löschen

| | | |
|---|---|---|
| Message | `CLTMEM_DLTSTR` | Delete a stream |
| Parameter | `Stream_ID` | Stream Nummer |

- Stream zurücksetzen

| | | |
|---|---|---|
| Message | `CLTMEM_RESSTR` | Reset Stream |
| Parameter | `Stream_ID` | Stream Nummer |

- FIFO-Information Abfragen

| | | |
|---|---|---|
| Message | `CLTMEM_GETSTA` | Get FIFO Status |
| Parameter | `Stream_ID` | Stream Nummer |

- Abspieldaten Lesen

| | | |
|---|---|---|
| Message | `CLTMEM_POPDAT` | Pop Stream Play Data From FIFO |
| Parameter | `Stream_ID` | Stream Nummer |

- Aufnahmedaten Schreiben

| | | |
|---|---|---|
| Message | `CLTMEM_INSDAT` | Insert stream record data to FIFO |
| Parameter | `Stream_ID` | Stream Nummer |
| | `Size` | Datengröße |
| | `Data` | Daten |

### 5.1.2.6 Messages vom Directory-Manager zum Client-Manager

- Filminformation

| | | |
|---|---|---|
| Message | `DIRCLT_FILINF` | Read File Info and Media Statistic |
| Parameter | `Stream_ID` | Stream Nummer |

| | | |
|---|---|---|
| | File_info | Dateiinformation |

### 5.1.2.7 Messages vom Directory-Manager zum Device-Manager

- Neuen Stream Anlegen

| | | |
|---|---|---|
| Message | DIRDEV_CRTSTR | Create a new stream |
| Parameter | Stream_ID | Stream Nummer |

- Stream Löschen

| | | |
|---|---|---|
| Message | DIRDEV_DLTSTR | Delete a stream |
| Parameter | Stream_ID | Stream Nummer |

- Stream zurücksetzen

| | | |
|---|---|---|
| Message | DIRDEV_RESSTR | Reset Stream |
| Parameter | Stream_ID | Stream Nummer |

- Abspieldatenblock holen

| | | |
|---|---|---|
| Message | DIRDEV_GETDAT | Read Stream play data |
| Parameter | Stream_ID | Stream Nummer |
| | Start_block | Startblock |
| | Number_block | Anzahl der Blöcke |

- Aufnahmedatenblock schreiben

| | | |
|---|---|---|
| Message | DIRDEV_PUTDAT | Write Stream record data |
| Parameter | Stream_ID | Stream Nummer |
| | Start_block | Startblock |
| | Number_block | Anzahl der Blöcke |

- Film von der Festplatte zum Archive kopieren

| | | |
|---|---|---|
| Message | DIRDEV_DSCARC | Copy media from disc to archive |
| Parameter | Stream_ID | Stream Nummer |
| | File_info | Dateiinformation |

- Film vom Archive zur Festplatte kopieren

| | | |
|---|---|---|
| Message | DIRDEV_ARCDSC | Copy media from archive to disc |
| Parameter | Stream_ID | Stream Nummer |

| | | |
|---|---|---|
| | File_info | Dateiinformation |

### 5.1.2.8 Messages vom Device-Manager zum Memory-Manager

- Abspieldaten

| | | |
|---|---|---|
| Message | DEVMEM_INSDAT | Insert stream play data to FIFO |
| Parameter | Stream_ID | Stream Nummer |
| | Size | Datengröße |
| | Data | Daten |

- Aufnahmedaten holen

| | | |
|---|---|---|
| Message | DEVMEM_POPDAT | Pop stream record data from FIFO |
| Parameter | Stream_ID | Stream Nummer |

### 5.1.2.9 Messages vom Memory-Manager zum Client-Manager

- Abspieldaten

| | | |
|---|---|---|
| | MEMCLT_DATPLA | Stream play data |
| Parameter | Stream_ID | Stream Nummer |
| | Size | Datenmengen |
| | Data | Daten |

- FIFO-Status

| | | |
|---|---|---|
| Message | MEMCLT_MEMSTA | FIFO status |
| Parameter | Stream_ID | Stream Nummer |
| | Fifo_size | FIFO’s Speichergröße |
| | Fill_size | Füllmenge |

### 5.1.2.10 Messages vom Memory-Manager zum Device-Manager

- Abspieldaten

| | | |
|---|---|---|
| Message | `MEMDEV_DATREC` | Stream record data |
| Parameter | `Stream_ID` | Stream Nummer |
| | `Size` | Datenmengen |
| | `Data` | Daten |

#### 5.1.2.11 Messages vom Admission-Controller zum Client-Manager

- Ressourcen Reservierung

| | | |
|---|---|---|
| Message | `ACCLT_ALLRES` | Ressource allocated |
| Parameter | `Stream_ID` | Stream Nummer |
| | `Fifo_size` | FIFO's Speichergröße |
| | `Allocation_mode` | Reserverungsmodus |

### 5.1.3 Schnittstelle zum Operating-System (OS)

#### 5.1.3.1 Messages vom MS-Manager zum Operating-System (OS)

- Mediendatei öffnen

| | | |
|---|---|---|
| Message | `OS_FILOPN` | Open a Media File |
| Parameter | `Stream_ID` | Stream Nummer |
| | `File_name` | Dateiname |

- Mediendatei Schließen

| | | |
|---|---|---|
| Message | `OS_FILCLS` | Close a Media File |
| Parameter | `Stream_ID` | Stream Nummer |
| | `File_name` | Dateiname |

- Datenblock lesen

| Message | OS_GETDAT | Read Data Block |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |
| | Start_block | Startblock |
| | Number_block | Anzahl der Blöcke |

- Datenblock schreiben

| Message | OS_PUTDAT | Write data block |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |
| | Start_block | Startblock |
| | Number_block | Anzahl der Blöcke |

- Dateninformation des Medien lesen

| Message | OS_GETINF | Read media Info |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |
| | File_info | Dateiinformation |

- Dateninformation des Medien schreiben

| Message | OS_PUTINF | Write media Info |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |
| | File_info | Dateiinformation |

### 5.1.3.2 Messages vom Operating-System (OS) zu MS-Manager

- Mediendatei öffnen

| Message | OSRET_FILOPN | Open a file |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |
| | Confirm_OK | Bestättigung 1=TRUE, 0 = FALSE |

- Mediendatei Schließen

| Message | OSRET_FILCLS | Close a File |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |
| | Confirm_OK | Bestättigung 1=TRUE, 0 = FALSE |

- Datenblock lesen

| Message | OSRET_GETDAT | Read data block |
|---|---|---|
| Parameter | Stream_ID | Stream Nummer |
| | Size | Datenmengen |

| | | |
|---|---|---|
| | `Data` | Daten |

- Datenblock schreiben

| | | |
|---|---|---|
| Message | `OSRET_PUTDAT` | Stream data |
| Parameter | `Stream_ID` | Stream Nummer |
| | `Confirm_OK` | Bestättigung 1=TRUE, 0 = FALSE |

- Dateninformation des Medien lesen

| | | |
|---|---|---|
| Message | `OSRET_GETINF` | Read Info |
| Parameter | `Stream_ID` | Stream Nummer |
| | `File_name` | Dateiname |
| | `File_info` | Dateiinformation |

- Dateninformation des Medien schreiben

| | | |
|---|---|---|
| Message | `OSRET_PUTINF` | Stream data |
| Parameter | `Stream_ID` | Stream Nummer |
| | `Confirm_OK` | Bestättigung 1=TRUE, 0 = FALSE |

## 5.2 Literaturverzeichnis

[Bar98] Giovanni Nájera Barquero
Simulation von MPEG-Datenströmen : Parameterextraktion und Visualisierung, Studienarbeit am Institut für Nachrichtentechnik, Universität Karlsruhe, 1998

[Bar99] Giovanni Nájera Barquero
Implementierung eines SCSI-Protokolls, Diplomarbeit am Institut für Nachrichtentechnik, Universität Karlsruhe, 1999

[Bie1] Ernst Biersack, Frédéric Thiesse :
Statistical Admission Control in Video Servers with Variable Bit Rate and Constant Time Length Retrieval, Institut Eurécom 1996. 0-1089-6503/96

[Bie2] Ernst Biersack, Frédéric Thiesse, Christoph Bernhardt :
Constant Data Time Length Retrieval for Video Servers with Variable Bit Rate Streams, Institut Eurécom 1996. 0-8186-7436/96

[Fal99] Raul Ivo Faller
Client/Server-Modelle eines Medienservers, Diplomarbeit am Institut für Nachrichtentechnik, Universität Karlsruhe, 1999

[Fer99] R. Ferreira
Implementierung eines Bandarchives, Diplomarbeit am Institut für Nachrichtentechnik, Universität Karlsruhe, 1999

[Jos96] Joseph Y. Hui, Ezhan Karasan, Jun Li, Junbiao Zhang
Client-Server Synchronization and Buffering for Variable Rate Multimedia Retrievals, IEEE Journal on Selected Areas in Communications, Vol. 14 No1, January 1996

[Kal99] Michael Kaltenbach
Client-Modellierung in einem Medienserver-Simulationssystem, Studienarbeit am Institut für Nachrichtentechnik, Universität Karlsruhe, 1999

[Mil95] Milde, T.
Videokompressionsverfahren im Vergleich. Dpunkt, Verlag für digitale Technologie, Heidelberg, 1995.

[Pan99] HweelHwa Pang, Bobby Jose and M.S. Krishnan :
Reseource Schudeling In A High-Performance Multimedia Server. 1999. 1041-4347/99

[Ste99] Ralf Steinmetz
Multimedia Technologie, Grundlage, Komponenten und Systeme, Springer Verlag 1999.