Diplomarbeit

zum Thema

Admission Control für ein echtzeitfähiges Dateisystem

an der

Technischen Universität Dresden
Fakultät Informatik
Institut für Betriebssysteme, Datenbanken und Rechnernetze
Lehrstuhl für Betriebssysteme

Eingereicht von: Sven Rudolph
Geboren am: 28.4.1972
Geboren in: Radeberg
Matrikel-Nr.: 1017263
Eingereicht am: 31. Mai 1998

Verantwortlicher Hochschullehrer:
Prof. Dr. H. Härtig

Betreuer:
Dr. Claude-Joachim Hamann
Dipl.-Inf. Lars Reuther

Selbständigkeitserklärung

Hiermit erkläre ich, daß ich diese Arbeit selbständig und nur mit den zugelassenen und aufgeführten Hilfsmitteln erstellt habe.

Dresden, den 31. Mai 1998

Sven Rudolph

Danksagung

Ich möchte mich an dieser Stelle herzlich bei allen bedanken, die mich bei der Erstellung dieser Arbeit unterstützt haben. Insbesondere gilt mein Dank Lars Reuther, Dr. Hamann, Frank Mehnert und Prof. Härtig sowie der gesamten Betriebssystemegruppe der Fakultät Informatik.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Am Lehrstuhl Betriebssysteme der Technischen Universität Dresden wird im Rahmen des Projekts *Dresden Real-time Operating System (***DROPS***)* ein *echtzeitfähiges* Betriebssystem konstruiert, mit dem sich sowohl Echtzeit-Anwendungen (vorrangig aus dem Multimedia-Bereich) als auch klassische Nicht-Echtzeit(*time-sharing*)-Anwendungen auf einem Rechner gemeinsam nutzen lassen.

Multimedia bezeichnet die gemeinsame Nutzung verschiedener Medien-Typen (Video, Audio) auf einem System. Mit der zunehmenden Leistungsfähigkeit von Rechnern und Rechner-Netzen wachsen die Anwendungsmöglichkeiten und die Verbreitung dieser Technik. Multimedia-Anwendungen stellen dabei neue Anforderungen an Betriebssysteme. Um zum Beispiel einen Video-Strom ruckfrei und in hoher Qualität anzuzeigen, muß jedes einzelne Bild zum richtigen Zeitpunkt dargestellt werden. Die Verfügbarkeit der dafür erforderlichen Ressourcen kann nur garantiert werden, wenn das Betriebssystem die Ressourcen vorher für diesen Video-Strom reserviert hat.

Anwendungen, die zu bestimmten Zeitpunkten garantiert Dienste erbringen, werden als **Echtzeit**-Anwendungen bezeichnet. Der Echtzeit-Begriff wurde ursprünglich für Anwendungen der Steuerungstechnik verwendet: Wenn ein bestimmtes Ereignis eintritt, muß der Steuerrechner innerhalb einer bestimmten Zeit mit hundertprozentiger Zuverlässigkeit eine Reaktion veranlassen; anderenfalls treten schwerwiegende Schäden ein. Um dieses Ziel zu erreichen, werden spezielle Steuerrechner konstruiert, auf denen eine bestimmter, auf diesen Anwendungszweck zugeschnittener Satz von Programmen läuft.

Es gibt aber auch Unterschiede zwischen diesen Arten von Echtzeit-Anwendungen. So sind die Auswirkungen der Nichterfüllung eines Auftrags bei Multimedia-Anwendungen nicht so schwerwiegend: Wird bei einem Video ein Bild zu spät dargestellt oder ganz ausgelassen, ist das unschön, aber keine Katastrophe. Des weiteren ist bei Multimedia-Anwendungen die Reaktionszeit auf eingehende Ereignisse nicht ganz so wichtig. Deshalb wird dieses Anforderungsprofil als **weiche Echtzeit** und das der Steuerungstechnik als **harte Echtzeit** bezeichnet.

Um Echtzeit-Anforderungen zu erfüllen, muß das System sicherstellen, daß für jeden Auftrag die erforderlichen Ressourcen zum geforderten Zeitpunkt zur Verfügung stehen. Deshalb muß jeder Echtzeit-Auftrag vorher dem System bekanntgegeben werden. Das System überprüft, ob die für diesen Auftrag benötigten Ressourcen zum geforderten Zeitpunkt verfügbar sein werden. Wenn dem so ist, werden diese Ressourcen für diesen Auftrag reserviert, anderenfalls wird der Auftrag mangels Ressourcen abgelehnt. Dieser Vorgang der Ressourcen-Überprüfung und -Reservierung wird auch **Admission Control** genannt.

Als Teil von DROPS wird gegenwärtig ein **Echtzeit-Dateisystem** (*real time file system,* **RTFS**) entwickelt. In dieser Arbeit wird untersucht, welche Ressourcen dieses Dateisystem für verschiedene mögliche DROPS-Anwendungen benötigt und wie diese Anforderungen quantitativ dargestellt werden können. Auf Basis der im Rahmen von DROPS entwickelten Arbeiten zur echtzeitfähigen Ansteuerung von SCSI-Festplatten [Meh98] und zur Gesamtstruktur des RTFS und der Einbindung in DROPS [Reu98] wird die Admission Control entworfen und implementiert.

Gliederung

Das folgende Kapitel bietet einen Überblick über die wichtigsten Datenformate für Multimedia-Ströme und über bereits existierende Methoden zur quantitativen Beschreibung solcher Datenströme. Anschließend werden existierende Dateisysteme für Multimedia-Anwendungen vorgestellt, wobei der Schwerpunkt auf den zur Admission Control benutzten Verfahren liegt. In Kapitel 3 werden Anwendungsszenarien analysiert, und darauf aufbauend werden verschiedene Entwurfs-Alternativen für die Admission Control entwickelt und bewertet. In Kapitel 4 wird die Struktur der Implementierung beschrieben. Zum Abschluß wird der Stand der Arbeit zusammengefaßt und eingeschätzt und ein Ausblick auf weiterführende Arbeiten gegeben.

Kapitel 2

Stand der Technik

Nach einer Übersicht über das DROPS-Projekt werden die wichtigsten für die Speicherung und Übertragung von Multimedia-Strömen verwendeten Datenformate vorgestellt. Anschließend werden ausgewählte Multimedia-Dateisysteme und insbesondere ihre Mechanismen zur Admission Control und der Erfüllung der Zusagen untersucht.

2.1 DROPS

In diesem Abschnitt werden die Grobstruktur von DROPS und die hier benötigten Zwischenergebnisse des DROPS-Projekts vorgestellt.

2.1.1 Gesamtstruktur

DROPS [HBB 97, HBB 98] ist eine auf dem Mikrokern L4 aufbauende Betriebssystem-Umgebung, die die für Multimedia-Anwendungen nötige Echtzeit-Unterstützung bereitstellt. L^4Linux [Hoh96] ist eine Portierung von Linux auf L4; damit können Anwendungen ohne Echtzeit-Anforderungen in einer komfortablen Linux-Umgebung parallel zu den Echtzeit-Anwendungen auf dem selben Rechner laufen.

Das Mikrokern-Betriebssystem **L4** [Lie96, Lie95] stellt grundlegende hardware-unabhängige Abstraktionen (Adreßräume, Threads) zur Verfügung. Gerätetreiber werden außerhalb des L4-Kerns implementiert, des weiteren bietet L4 Mechanismen zur Auslagerung der Speicher- und Prozessorzeit-Verwaltung in Nutzerprozesse.

Die **Ressourcen-Verwaltung** steuert mittels dieser Mechanismen den Zugriff auf die von jedem Prozeß benötigten Ressourcen Hauptspeicher und CPU-Zeit. Dadurch wird die von L4 bereitgestellte strukturelle Entkoppelung der Prozesse auf diese Ressourcen ausgeweitet.

Die **Echtzeit-Komponenten** bilden den Kern des DROPS-Projekts: Sie steuern bestimmte Hardware-Komponenten so an, daß zeitliche Zusagen gemacht werden können. Momentan sind neben dem Echtzeit-Dateisystem Komponenten zur Datenübertragung über ATM-Rechnernetze und zur Ansteuerung der Grafikkarten in Arbeit.

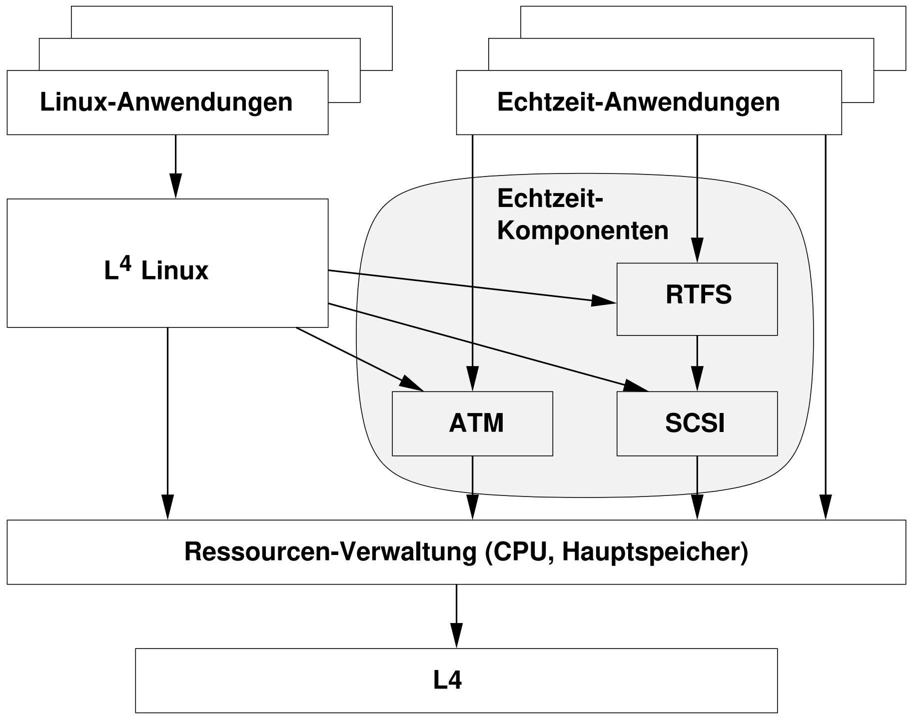

Abbildung 2.1: Systemarchitektur von DROPS.

Eine **Echtzeit-Anwendung** stellt eine bestimmte Echtzeit-Funktionalität bereit. Dafür kann sie auf die Echtzeit-Komponenten und die Ressourcenverwaltung zurückgreifen, um für bestimmte Teilaufgaben Echtzeitzusagen zu gewährleisten.

Damit die Linux-Anwendungen auch auf die von den Echtzeit-Komponenten verwalteten Geräte zugreifen können, soll L^4Linux für diese Geräte virtuelle Treiber bereitstellen. Diese Treiber stellen für eine Linux-Anwendung ein normales Linux-Gerät dar und realisieren die Funktionen, indem sie die Aufträge an die jeweilige Echtzeit-Komponente weiterleiten.

2.1.2 SCSI und Echtzeit

Das RTFS soll über den SCSI-Bus angeschlossene Festplatten so nutzen, daß feste Zeitzusagen gemacht werden können. Im folgenden wird gezeigt, was SCSI ist und wie die Echtzeiteigenschaften realisiert werden können.

SCSI (*Small Computer System Interface*) ist ein weitverbreitetes Bus-System zur Ansteuerung von Massenspeichern und anderen Geräten. Der SCSI-2-Standard [ANS] definiert die physischen Parameter (Stecker, Kabel-Eigenschaften, Spannungspegel), das Protokoll zum Bus-Zugriff (Arbitrierung, Priorisierung) und Befehlssätze zum Zugriff auf die Geräte. Jedes Gerät muß einen minimalen Befehlssatz beherrschen, darüber hinaus sind für verschiedene Gerätetypen (Festplatten, Bandlaufwerke, CD-ROM-Laufwerke) zusätzlich geforderte Befehle definiert.

Schwerpunkte bei der Entwicklung von SCSI waren:

Abstraktion der Schnittstelle
Durch die Definition von Gerätetypen werden die früher genutzten Hersteller-spezifischen Schnittstellen abgelöst. Alle Festplatten können unabhängig vom Hersteller angesteuert werden, und es ist für diese Geräte nur ein Bus notwendig.

Unterstützung intelligenter Geräte
Die Geräte werden unabhängig von ihrer physischen Realisierung angesprochen, so werden die Blöcke einer Platte nicht über die physischen Eigenschaften Scheiben-Nummer, Sektor-Nummer und Block-Nummer, sondern über eine einzige kontinuierlich vergebene logische Block-Nummer angesprochen. Im Rahmen der Schnittstellen-Bedingungen können die Geräte selbst über Zugriffs- und Cache-Algorithmen entscheiden.

Hoher Gesamt-Durchsatz
Erhält ein Gerät einen Auftrag, kann es den Bus freigeben und das Ergebnis später zurückgeben. Damit wird zum Beispiel während des Suchens eines Datenblocks auf der Festplatte der SCSI-Bus nicht blockiert, und er steht für Übertragung zu anderen angeschlossenen Geräten zur Verfügung. Da die Leserate einer Platte meist geringer als die Übertragungsrate des SCSI-Busses ist, können die Daten eines Leseauftrags in mehreren Teilen zum Rechner gesendet werden; die Platte gibt also zwischendurch den Bus frei (*Disconnect/Reconnect*).

Dagegen wurde die Unterstützung für Echtzeit-Ansprüche nicht berücksichtigt. Tatsächlich erschweren einige Mechanismen (Geräte mit Eigen-Intelligenz, gemeinsamer Bus mit statischer Priorisierung der Geräte) die Zusage maximaler Ausführungszeiten für einen bestimmten Auftrag.

Als Grundlage für das DROPS-Echtzeit-Dateisystem wurde in [Meh98] untersucht, wie man den SCSI-Bus und die angeschlossenen Festplatten ansteuern kann, um für eine möglichst hohe Gesamt-Datenrate zwischen SCSI-Hostadapter und Platten eine akzeptable maximale Ausführungszeit für SCSI-Aufträge zu erhalten.

Grundidee des entwickelten Verfahrens ist, durch Planung der Zugriffe auf den SCSI-Bus die Bearbeitungsdauer der Aufträge vorhersagbar zu gestalten. Dabei wird für jeden Auftrag die maximale Bearbeitungszeit (*worst case time*) eingeplant. Die Bearbeitung eines Auftrags unterteilt sich in drei Phasen:

1. Der Lese-Auftrag wird über den SCSI-Bus an die Platte gesendet.
2. Die Platte positioniert den Kopf auf den im Auftrag angegebenen Datenblock. Dafür wird der SCSI-Bus zunächst nicht benötigt.
3. Die Platte liest den Datenblock in den internen Cache ein und versucht gleichzeitig, die Daten über den SCSI-Bus an den Rechner zu liefern.

Da diese Platte während der zweiten Phase den SCSI-Bus nicht belegt, kann der Bus während dieser Zeit für andere Aufträge genutzt werden. Die Aufträge für verschiedene Platten werden entsprechend zeitversetzt abgeschickt, dadurch ergibt sich die in Abb. 2.2 gezeigte Struktur.

Jeder Auftrag wird in einen Slot eingeplant, die Slots der verschiedenen Platten sind zeitlich versetzt angeordnet. Die Dauer eines Slots ergibt sich aus der Dauer der drei Phasen, dabei ist für jede Phase

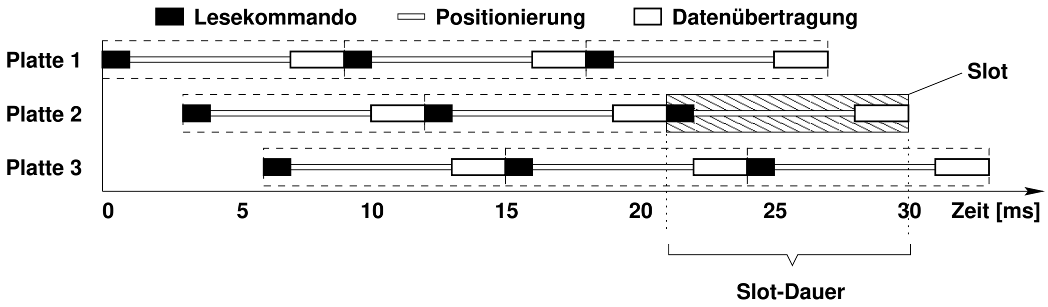

Abbildung 2.2: Periodisch versetztes Absetzen der Aufträge (nach [Meh98, Reu98]).

die maximal benötigte Zeit einzuplanen. Die maximale Anzahl der Platten, die mit diesem Verfahren angesteuert werden können, ergibt sich aus dem Quotienten der Slotdauer und der für die Phasen 1 und 3 benötigten Zeit.

In der Arbeit wurde weiter darauf eingegangen, daß sich die konkrete Abfolge verändert, wenn die Daten bereits im Cache der Festplatte vorliegen, aber es läßt sich für die maximale Ausführungsdauer eine obere Schranke angeben läßt, so daß das Modell weiterhin genutzt werden kann. Des weiteren wird gezeigt, daß sich das Verhalten prinzipiell nicht ändert, wenn Daten nicht nur von Platte gelesen, sondern auch auf die Platte geschrieben werden.

Ergebnis dieser Arbeit ist ein auf Basis dieses Modells arbeitender SCSI-Treiber, der als Echtzeit-Komponente für die Entwicklung des Echtzeit-Dateisystems genutzt werden kann.

2.1.3 Einheitliche Strom-Beschreibung

Um eine Admission Control durchzuführen, muß bekannt sein, zu welchem Zeitpunkt das Echtzeit-Dateisystem für die Bearbeitung eines Multimedia-Stroms Ressourcen benötigt. Dieser Bedarf läßt sich aus der Datenrate des Stroms zu diesen Zeitpunkten ableiten.

In DROPS sollen alle Echtzeit-Komponenten ein einheitliches Modell zur Beschreibung dieser Strom-Eigenschaften nutzen. Auf Basis dieser gemeinsamen Strombeschreibung wird in DROPS die Admission Control zwischen verschiedenen Echtzeit-Komponenten und Echtzeit-Applikationen ablaufen.

In [Ham97] wird aufbauend auf anderen Systemen, insbesondere aus dem Bereich der Ressourcen-Reservierung in Rechnernetzen, der Versuch unternommen, dieses einheitliche Modell für Strom-Beschreibungen in DROPS zu entwickeln. Dieses Modell beschreibt Ereignis-Ströme mit einem durchschnittlichen (T) und einem minimalen (D) Ereignis-Abstand. Die Parameter τ und τ' beschreiben den Bereich, in dem das Ereignis eintreten darf und somit die Schwankung der Datenrate; Ströme dieser Art werden als **schwankungsbeschränkte Ströme** bezeichnet.

In Abb. 2.3 wird ein Beispiel-Strom angegeben. Das erste Ereignis trifft genau zum geforderten Zeitpunkt ein, die folgenden beiden kommen verfrüht, und das vierte verspätet sich. Das dritte und vierte Ereignis nutzen jeweils die maximal zulässige Verfrühung beziehungsweise Verspätung aus. Das zweite ist so angeordnet, daß es möglichst dicht am dritten Ereignis liegt.

Wenn es nicht möglich ist, den minimalen Ereignis-Abstand exakt zu bestimmen, so kann man ihn auf 0 setzen. Dann entspricht die Beschreibung der des LBAP-Modells (Beweis in [Ham97]). Die

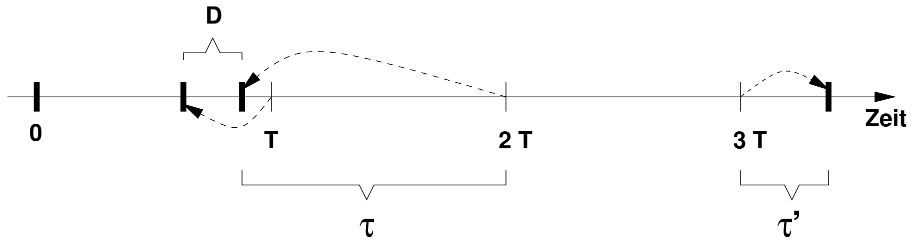

Abbildung 2.3: Schwankungsbeschränkter Strom mit konstanter Rate (nach [Ham97]).

durch dieses vereinfachte Modell enstehende Abweichung ist um so kleiner, je größer die maximale Ereignisdichte $1/D$ gegenüber der durchschnittlichen Ereignisdichte $1/T$ des Stromes ist.

Soll ein schwankungsbeschränkter Strom in einen Strom ohne Schwankung umgewandelt werden, muß die durch τ und τ' beschriebene Schwankung durch Pufferung ausgeglichen werden. Aus den Parametern des Stroms läßt sich die für die Pufferung benötigte Speichermenge berechnen.

Hauptproblem dieser Beschreibung schwankungsbeschränkter Ströme ist, daß die Ereignisse nicht unterschiedlich große Datenmengen repräsentieren können. Beim Lesen von Festplatte wird beispielsweise nicht immer die gleiche Datenmenge in einem Ereignis übertragen. Man könnte das Problem zu umgehen versuchen, indem man eine kleinste Dateneinheit, ein Bit, als ein selbständiges Ereignis auffaßt, aber dadurch wird wieder der minimale Zeitabstand zwischen zwei Ereignissen schwer bestimmbar und die Anzahl der Ereignisse wird sehr hoch. Es ist geplant, das Modell schwankungsbeschränkter Ströme weiterzuentwickeln, um dieses Problem zu lösen.

Ein anderer Ansatz ist, die akkumulierte Datenmenge über der Zeit anzutragen und einen Korridor für die Bandbreitenschwankungen vorzusehen. In Abb. 2.4 wird am Beispiel eines Stroms, der in Blöcken konstanter Größe von Festplatte gelesen und als kontinuierlicher Datenstrom weitergeleitet wird, das Grundprinzip dargestellt.

b bezeichnet die mittlere Bandbreite über den gesamten Stromverlauf. v (*Vorlauf*) bezeichnet die Datenmenge, die benötigt wird, bevor sie entsprechend der durchschnittlichen Bandbreite eingelesen worden wäre. Diese Menge muß vor dem Start des Stroms bereits eingelesen werden. n (*Nachlauf*) ist die maximale Datenmenge, die früher gelesen als benötigt wird. n und v sind jeweils Maximum und Minimum der Differenz zwischen der Funktion $b*t$ und der Funktion der akkumulierten Datenmenge; sie bestimmen die Breite des Korridors.

Durch dieses Verfahren ist einfach planbar, wieviel Speicher zum Puffern der Schwankungen gebraucht wird. Es ist unklar, ob dieses Modell als allgemeine Beschreibung für DROPS geeignet ist, jedoch läßt sich anhand dieses Modells die Planung im Dateisystem gut durchführen. Für die Kommunikation mit anderen Komponenten muß diese Darstellung in die gemeinsame DROPS-Strombeschreibung umgewandelt werden.

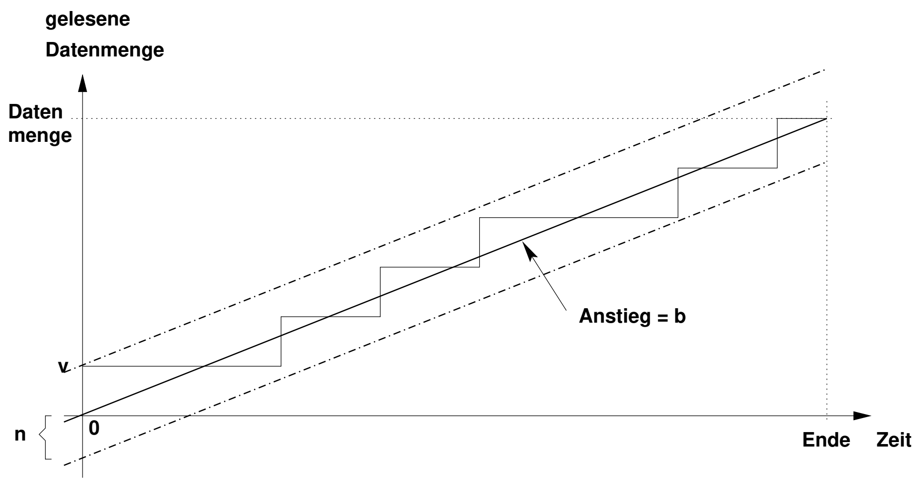

Abbildung 2.4: Strombeschreibung mittels Bandbreite und Schwankungs-Korridor.

2.1.4 Echtzeit-Dateisystem

In [Reu98] wurde die geplante Struktur des Echtzeit-Dateisystems beschrieben. Es setzt sich aus mehreren Teilen zusammen:

Datei- und Blockverwaltung

Dieser Teil realisiert den Zugriff auf die Dateien anhand der auf den Festplatten gespeicherten Filesystem-Strukturen. Die Abbildung der Datei auf einzelne Blöcke wird über Unix-ähnliche Inodes durchgeführt. Um auch bei hohen Abweichungen der Datenraten und der Dateigröße effizient arbeiten zu können, ist die Blockverwaltung nicht auf eine Blockgröße beschränkt.

Admission Control

Hier werden die Reservierung von Slots für den Plattenzugriff durchgeführt und die Reservierungs-Informationen gespeichert. Entwurf und Implementierung dieser Komponente sind das Thema der vorliegenden Arbeit.

Auftragsgenerierung

Aus den Reservierungs-Informationen der Admission Control werden die Aufträge an den SCSI-Treiber erzeugt und abgesendet.

Pufferverwaltung

Diese Komponente verwaltet die zur Datenübertragung zwischen den Komponenten und Applikationen benötigten Puffer.

Zum Zugriff auf die SCSI-Festplatten nutzt das Echtzeit-Dateisystem den oben vorgestellten SCSI-Treiber.

Das Dateisystem kann sowohl Echtzeit-Aufträge als auch Aufträge ohne Echtzeit-Anforderungen bearbeiten. Über die Nicht-Echtzeit-Schnittstelle können auch von L^4Linux Aufträge entgegengenommen und bearbeitet werden. Dazu muß ein Dateisystem-Treiber in L^4Linux implementiert werden, der die Aufträge an die Schnittstelle des Echtzeit-Dateisystems anpaßt.

2.2 Formate für Multimedia-Ströme

Um für einen Multimedia-Strom zu ermitteln, wann welche Datenmenge zur Verarbeitung benötigt wird, muß das Format dieses Datenstroms interpretiert werden. Auch hat die Wahl des Datenformats Einfluß auf die Schwankung der Datenmengen. Im folgenden werden Datenformate vorgestellt, die für die Darstellung von Multimedia-Daten in DROPS genutzt werden können.

2.2.1 MPEG-Standards

Die Moving Pictures Experts Group (MPEG) ist eine gemeinsame Arbeitsgruppe der International Organization for Standardization (ISO) und der International Electrotechnical Comission (IOC); sie entwickelt Standards für die Kodierung von Video- und Audio-Strömen.

MPEG-1 ist der älteste dieser Standards und besteht wiederum aus drei Teilen: MPEG-1 Video [MPFL97] spezifiziert den Aufbau eines komprimierten Video-Stroms und ein entsprechendes Dekodier-Verfahren und MPEG-1 Audio spezifiziert gleiches für Audio-Ströme. Den dritten Teil bildet der MPEG System Layer, der beschreibt, wie Audio- und Video-Ströme zu einem einzigen Multimedia-Strom zusammengefügt werden können.

MPEG-2 baut auf MPEG-1 auf. MPEG-2 Video ermöglicht höhere Bildgrößen und Datenraten. Für verschiedene Qualitätsanforderungen wurden Profile und Level definiert, die von MPEG-1-ähnlicher Qualität bis zu HDTV-Qualität reichen. Ein Profil beschreibt die genutzten Kompressionsverfahren und -parameter; das Level beschreibt die Bildgröße und die Bandbreite des kodierten Datenstroms.

Die Kodierverfahren sind prinzipiell asymmetrisch; der Aufwand zur Kodierung ist bedeutend höher als der zur Dekodierung. Die Dekodierung ist mit einfacher und preiswerter Hardware in hoher Qualität möglich, aber für die Kodierung wird üblicherweise spezielle, teure Hardware benutzt, um die benötigte Rechenleistung bereitzustellen. Alternativ verfügbare reine Software-Lösungen sind nicht in der Lage, die Ströme in Echtzeit zu kodieren.

Damit ist MPEG für Anwendungen mit wenigen Kodier- und häufigen Dekodier-Vorgängen (zum Beispiel Digitales Fernsehen, Distribution von Medien-Produkten), aber nicht für individuelle interaktive Kommunikation (zum Beispiel Video-Konferenzen) geeignet,

Eine für Video-Übertragungen sinnvolle Qualität wird durch eine Kombination von *main profile (MP)* und *main level (ML)* (zusammen: *MP@ML*) bereitgestellt.

Aufbau eines MPEG-Video-Stroms

Ein Video-Strom ist eine Folge von Einzelbildern. In einem MPEG-Video-Strom ist jedes Einzelbild in einem Frame kodiert; der Strom besteht aus einer Folge von Frames.

	Datenrate	Auflösung
MPEG-1 Audio (Layer 2)	128 KBit/s	32 . . . 48 kHz
MPEG-1 Video	1,5 MBit/s	352x240 Bildpunkte
MPEG-2 Video (MP@ML)	15 MBit/s	720x576 Bildpunkte

Tabelle 2.1: Typische Qualitätsparameter für MPEG-Ströme

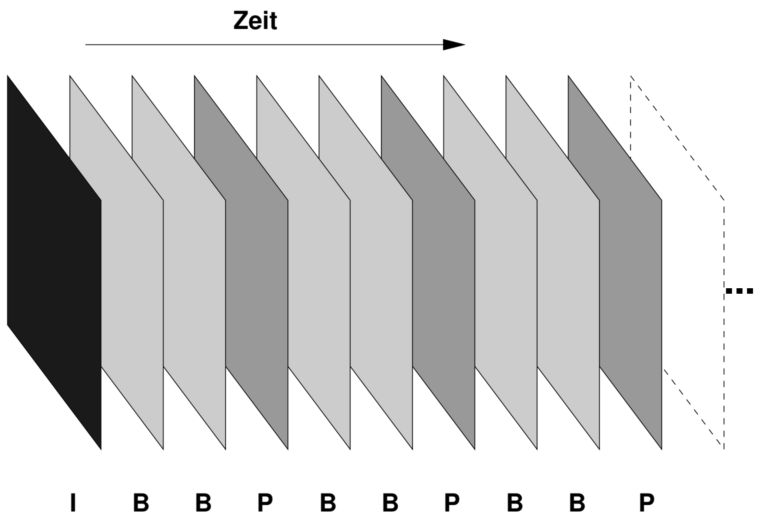

Abbildung 2.5: Typische Anzeige-Frame-Folge eines MPEG-Video-Stroms [MPFL97].

In einem MPEG-Video-Strom werden meist drei verschiedene Frame-Typen genutzt:

I-Frame (*intra-coded*)
Ein I-Frame ist als eigenständiges Bild kodiert. Er läßt sich ohne Zuhilfenahme früherer oder späterer Bilder dekodieren.

P-Frame (*predictive-coded*)
Bei der Erstellung eines P-Frames werden Redundanzen relativ zum vorher auftretenden I- oder P-Frame ausgenutzt. Es werden also nur die Änderungen gegenüber diesen vorhergehenden Frames kodiert. Beim Dekodieren wird also zusätzlich dieser vorhergehende Frame (beziehungsweise das daraus dekodierte Bild) benötigt.

B-Frame (*bidirectionally predictive-coded*)
Beim B-Frame erfolgt die Kodierung ähnlich wie beim P-Frame, aber es können auch Bezüge auf den folgenden I- oder P-Frame auftreten.

Aufgrund der Abhängigkeiten zwischen den Frame-Typen wird zum Dekodieren von B-Frames der jeweils folgende P-Frame benötigt. Wenn ein MPEG-Video-Strom angezeigt werden soll, muß dieser

P-Frame bereits vor den direkt vor ihm anzuzeigenden B-Frames beim Dekoder vorliegen. Aus diesem Grund werden die Frames in einem MPEG-Strom nicht in der Reihenfolge, in der sie später angezeigt werden, sondern in der Reihenfolge, in der sie beim Dekoder benötigt werden, abgelegt.

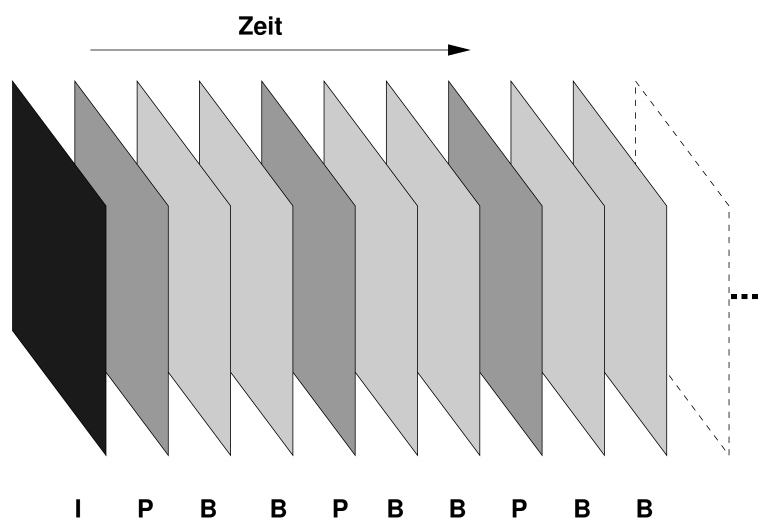

Abbildung 2.6: Typische Dekodier-Frame-Folge eines MPEG-Video-Stroms [MPFL97].

2.2.2 H.261

Die Empfehlung H.261 der ITU-T [ITU93] standardisiert ein Kodier- und Dekodier-Verfahren für audiovisuelle Dienste geringer Bandbreite (p * 64 kBit; ab p = 1). (Die ITU-T ist eine für Telekommunikation zuständige Unterorganisation der ITU (International Telecommunication Union) und in dieser Eigenschaft Nachfolger der CCITT.)

H.261 wurde primär für Video-Konferenz- und Video-Telefon-Anwendungen entwickelt, es kann aber bei geringen Qualitätsanforderungen auch für Fernseh-ähnliche Dienste genutzt werden. Im Unterschied zu MPEG ist es ein symmetrisches Kodier-Verfahren; der Aufwand zur Kodierung entspricht also ungefähr dem zur Dekodierung benötigten Aufwand. Damit ist die Durchführung der Kodierung preiswerter als bei MPEG und somit auch für indviduelle Kommunikation realisierbar.

Die minimale Bandbreite von 64 kBit entspricht der eines ISDN-Kanals, somit können bereits vorhandene Telekommunikationskanäle für Video-Telefonie und Video-Konferenzen genutzt werden.

H.261 auf PC-Hardware

In [Tur93] wird eine auf H.261 aufbauende Software-Lösung für Videokonferenzen über IP-Netze vorgestellt. Mit diesem System reicht eine Datenrate von 30 KByte/s bereits für eine Videokonferenz aus. Es werden spezielle Verfahren genutzt, um Jitter und Paket-Verluste, wie sie im Internet auftreten, zu kompensieren.

Diese Lösung ist nicht auf ISDN beschränkt, sondern läuft auf gängigen PCs und Workstations. Im Unterschied zu den ISDN-Anwendungen wird die H.261-Kodierung in Software vorgenommen. Damit ist H.261 ohne spezielle Hardware einsetzbar, und es ergeben sich neue Anwendungsgebiete.

2.2.3 Audio-Ströme in GSM

GSM (*Global System for Mobile Communications*) ist ein weltweit genutztes System zur digitalen mobilen Sprach-Übertragung (Telefonie) auf Basis von zellularen Funk-Netzen.

In diesem System werden die Audio-Daten digitalisiert und komprimiert über Funkverbindungen übertragen. Dabei steht für die Übermittlung der Daten nur eine sehr geringe Bandbreite zur Verfügung, somit müssen die Audio-Daten sehr gut komprimiert werden.

Es wurde ein Kodier-Verfahren entwickelt, daß darauf optimiert ist, Sprache in guter Qualität zu komprimieren. Durch diese Kompression wird eine Datenrate von nur 13 KBit/s benötigt.

Im Rahmen von DROPS ist denkbar, dieses Verfahren für Anwendungen zur Sprach-Kommunikation und -Speicherung zu nutzen.

2.3 Admission Control in Echtzeit-Dateisystemen

In diesem Abschnitt werden bereits existierende Multimedia-Dateisystem-Projekte vorgestellt und sowohl auf Schwächen als auch auf für DROPS interessante Lösungsansätze hin untersucht.

2.3.1 Tiger Video Fileserver

Der bei Microsoft Research entwickelte Tiger Video Fileserver [BBD 96, BFD97] ist ein verteilter, fehlertoleranter Echtzeit-Fileserver für Video-on-Demand-Anwendungen.

Ein Tiger-System enthält mehrere Speicher-Knoten, ein jeder besteht aus einem Rechner mit SCSI-Bus und mehreren SCSI-Festplatten. Diese Knoten sind über einen speziellen ATM-Switch verbunden, der diese Ströme auf die ausgehenden Leitungen switcht. Der Steuerung des Systems dient ein zusätzlicher Rechner, der über ein einfaches Netz mit den Speicher-Knoten gekoppelt ist.

Um die für ein bestimmtes Video nutzbare Bandbreite unabhängig von der Bandbreite einer einzelnen Platte zu machen, werden die Videos in Stücke von der Dauer einer Sekunde aufgeteilt und der Reihe nach rundum auf alle Platten aller Speicher-Knoten verteilt (*Striping*).

In Tiger müssen alle Ströme eine konstante Datenrate haben, und alle Ströme müssen die gleiche Datenrate haben. Der Quotient aus der Datenrate eines einzelnen Stroms und der Gesamtbandbreite des Platten-Systems ergibt die maximale Anzahl bedienbarer Datenströme.

Die Admission Control erfolgt, indem für jeden möglichen Datenstrom ein Eintrag in einem Zeitplan vorgesehen und bei der Anforderung eines Stroms belegt wird. (Dieser Plan dient zusätzlich der Organisation des Stripings).

Zur Erhöhung der Verfügbarkeit können die Daten auf anderen Speicher-Knoten gespiegelt werden. Der Inhalt eines Speicher-Knotens wird dabei auf n andere Speicher-Knoten gespiegelt; dadurch steigt bei Ausfall eines Speicher-Knotens die Last auf den spiegelnden Speicher-Knoten nur um $1/n$.

Der Tiger Video Fileserver hat eine recht feste Struktur, er wurde für eine bestimmte Anwendung konstruiert. Selbst für Video-on-Demand sind Flexibilitäts-Anforderungen denkbar, die der Tiger Video Fileserver nicht erfüllen kann (siehe 3.2.2). Aufgrund der statischen Architektur sind einfache Lösungen für Lastverteilung und Umgehung von Hardware-Ausfällen möglich.

2.3.2 UBC Distributed Continuous Media File System

Das UBC Distributed Continuous Media File System [NMH, MHN95] ist als File-Server für Audio- und Video-Ströme konzipiert. Im Unterschied zu anderen Projekten können die Ströme unterschiedliche Datenraten haben, und die Datenrate jedes einzelnen Stroms kann zeitlich schwanken.

Alle Ströme werden in Slots fester Länge (z.B. 500 ms) unterteilt. Die Datenmenge, die in den Slots gelesen werden soll, wird als Vektor gespeichert. Durch diesen Vektor werden somit die Bitraten-Anforderungen eines Stroms beschrieben.

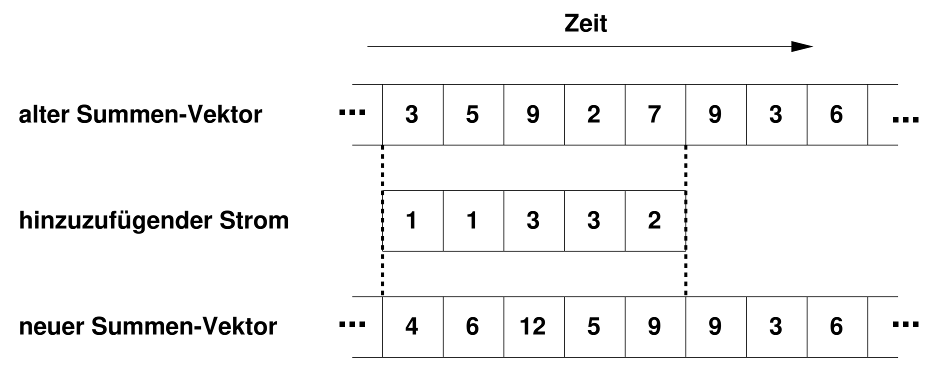

Abbildung 2.7: Admission Control im UBC Distributed Continuous Media File System [NMH].

Die Admission Control wird durchgeführt, indem der Vektor der bereits vorliegenden Anforderungen und der Vektor des neuen Stroms addiert werden. Übersteigt ein Element des Summenvektors das im System verfügbare Maximum, muß der neue Strom abgelehnt werden.

Diese Restriktion läßt sich mindern, indem man versucht, solche Konflikte durch vorausschauendes Lesen (*Read-Ahead*) aufzulösen. Liegen für einen bestimmten Slot zu viele Anforderungen vor, wird versucht, diese bereits in den vorherigen Slots zu erfüllen. Dadurch können lokale Lastspitzen ausgeglichen werden, und die zugesagte Auslastung wird erhöht (Abb. 2.8).

Durch die vorfristige Erfüllung der Anforderungen werden die Blöcke zeitiger gelesen als sie beim Client benötigt werden. Im einfachsten Fall wurde angenommen, daß auf den Clients kein Puffer-Speicher zur Verfügung steht. Für *Read-Ahead* wird so zusätzlicher Speicher auf dem Server benötigt. Ist der für diese Pufferung verfügbare Speicher erschöpft, können keine neuen Ströme mehr zugelassen werden. Deshalb muß jetzt zusätzlich die Ressource „Puffer-Speicher“ verwaltet werden. Analog zu den Vektoren für die Platten-Anforderungen wird ein Vektor für den genutzten Puffer-Speicher verwaltet.

Durch die Berücksichtigung variabler Datenraten nimmt das UBC Distributed Continuous Media File System eine besondere Stellung ein. Allerdings führt das Verfahren zur Strombeschreibung zu unnötig

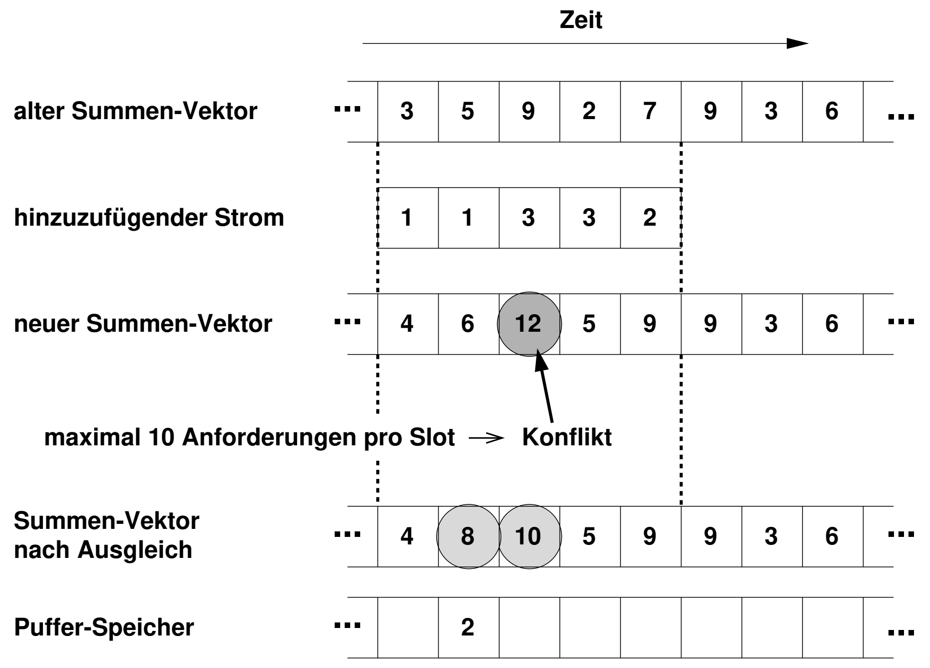

Abbildung 2.8: Ausgleich von Lastspitzen im UBC Distributed Continuous Media File System.

großen Datenmengen, und die systemweite Festlegung einer Periode bringt eine für einen einzelnen Strom willkürliche Aufteilung mit sich. Im Rahmen dieser Arbeit wird untersucht werden müssen, ob die vorgestellten Mechanismen auch im zu entwickelnden Dateisystem anwendbar sind und durch welche Techniken weiterführende Eigenschaften zu erreichen sind.

2.3.3 Kx-CMFS

In [Kli97] wird ein Continuous Media File System (Kx-CMFS) entworfen.

Die Übertragung eines Datenstroms teilt sich in eine Folge von Aufträgen an den SCSI-Treiber auf. Für die Einplanung dieser Auftragsfolgen werden zwei Alternativen erwogen:

mit vollständiger Liste der Aufträge

Für jeden zu übertragenden Multimedia-Strom erhält der SCSI-Treiber einer Liste der Aufträge. Ein einzelner Auftrag besteht dabei aus einem Zeitpunkt und aus der Nummer des Blocks, der zu diesem Zeitpunkt übertragen werden soll. Auf Basis dieser Listen kann der SCSI-Treiber dann ermitteln, ob dieser Strom zugelassen werden kann, und er kann die entsprechenden Ressourcen reservieren.

Es wird in einer Beispielrechnung darauf hingewiesen, daß die Liste der Aufträge relativ groß werden kann (512 KByte für eine Datei von 2 GByte Größe) und daß die Algorithmen auf Basis dieser Listenstrukturen voraussichtlich viel Rechenzeit und Speicherplatz benötigen.

auf Basis maximaler Übertragungswerte

> Aufgrund obiger erwarteter Probleme wurde für das Kx-CMFS ein vereinfachtes Verfahren vorgesehen: Dem SCSI-Treiber wird nur die maximale Datenrate übergeben, und auf Grund dieser Maximal-Werte werden die Admission Control und die Ressourcen-Reservierung für den Strom durchgeführt. Durch dieses Verfahren wird der Aufwand für die Admission Control stark gesenkt, andererseits ist die Planung bedeutend ungenauer und verringert so die Menge zusagenfähiger Ressourcen.

Für das Kx-CMFS wurde die zweite Variante vorgesehen. Als Anforderungs-Parameter sollen die durchschnittliche Datenrate, die maximale Datenrate und die Burstlänge (maximale Länge eines Stromabschnitts, in dem die maximale Datenrate gefordert wird) genutzt werden.

Kapitel 3

Entwurf

Die bisher vorgestellten Verfahren zur Admission Control erscheinen für den Einsatz in DROPS ungeeignet. In diesem Kapitel werden verschiedene Anwendungsszenarien untersucht, um die Anforderungen zu konkretisieren. Darauf aufbauend werden verschiedene Ansätze entworfen und hinsichtlich ihrer Anwendbarkeit verglichen.

3.1 Entwurfsziele

Ziel dieser Arbeit ist die Entwicklung der Admission Control für das Echtzeit-Dateisystem von DROPS. Die Admission Control soll folgende Bedingungen erfüllen:

- In DROPS sollen gleichzeitig verschiedene Typen von Multimedia-Strömen verarbeitet werden. Dafür ist es notwendig, auf einem DROPS-System viele unterschiedliche Datenraten zu ermöglichen. Die Admission Control muß verschiedene Datenraten verwalten können.
- Das Filesystem und somit die Admission Control muß nur das Daten-Zeit-Verhalten eines Stroms, aber nicht das konkret benutzte Datenformat kennen. So ist es einfach möglich, neue Datenformate zu nutzen.
- Die Datenraten verschiedener Strömen können sich um Größenordnungen unterscheiden. Die Admission Control soll sehr große und sehr kleine Datenraten ermöglichen.
- Bei kleiner Datenrate sind theoretisch sehr viele Ströme möglich. Die Admission Control soll in der Lage sein, eine entsprechende Anzahl von Strömen zu verwalten.
- Obwohl Multimedia-Anwendungen der *weichen Echtzeit* zugeordnet werden, sollen Verluste soweit möglich vermieden werden, da das Dateisystem auch für andere Anwendungen nutzbar sein soll. Insbesondere soll die Admission Control keine stochastischen Methoden verwenden, sondern wirklich alle Ressourcen sicher einplanen.

3.2 Anwendungsszenarien

Im folgenden wird untersucht, welche Anforderungen an das Echtzeit-Dateisystem und die Admission Control sich aus verschiedenen Anwendungsszenarien ergeben.

3.2.1 Server für Digitales Fernsehen

Durch Kompressionsverfahren läßt sich die zur Übertragung von Bewegtbildern und Ton benötigte Datenmenge stark reduzieren. Deshalb wurde für die Weiterentwicklung der Fernsehtechnik der Übergang von analoger zu digitaler Datenübertragung vorgesehen.

Bei digitaler Übertragung lassen sich aus der Analog-Technik bekannte Qualitätsschwankungen aufgrund von Rauschen u.ä. vermeiden. Die Erhöhung der Bildqualität ist dank Kompression ohne eine teure Erhöhung der Übertragungsbandbreite möglich. Die freigewordene Bandbreite kann für zusätzliche Dienste genutzt werden.

Im einfachsten Fall ändert sich für den Endnutzer nichts. Die Fernseh-Programme werden digital eingespeist, über entsprechende Übertragungs-Kanäle (Funk, Satellit oder Kabel) übertragen und sind im Endgerät genauso wie konventionelle Fernsehprogramme empfangbar.

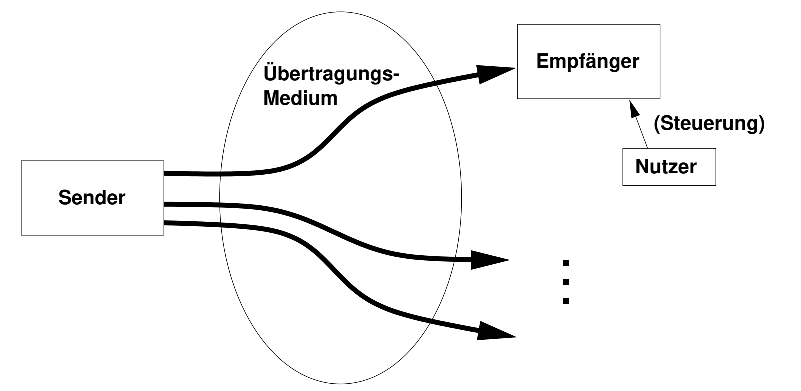

Abbildung 3.1: Datenfluß beim Digitalen Fernsehen.

Es erfolgt also keine Rückkopplung vom Empfänger zum Sender. Damit ist der Ressourcenbedarf des Servers nicht von der Anzahl der Empfänger, sondern nur von der Anzahl der zu verteilenden Programme abhängig. Das Kommunikationsmedium kann von allen Empfängern gleichzeitig genutzt werden, die benötigte Bandbreite ist ebenfalls von der Anzahl der Empfänger unabhängig.

Diese Anwendung hat eine relativ statische Struktur, sie könnte auch mit bereits existierenden Systemen realisiert werden. Da die benötigten Ressourcen nur von der Anzahl der benötigten Programme abhängig sind, kann die Ressourcenreservierung bereits zum Systemstart durchgeführt werden und muß nicht später verändert werden. Aufgrund der statischen Struktur kann die Anordnung der Daten auf den Festplatten gezielt auf die Strom-Eigenschaften zugeschnitten werden (siehe auch Tiger Video Fileserver 2.3.1).

3.2.2 Server für Video on Demand

Video on Demand unterscheidet sich von obigem Digitalen Fernsehen dadurch, daß der Endnutzer jetzt einen Kommunikationskanal zum Server hat. Durch diese Rückkopplung können die gesendeten Daten direkt auf ihn zugeschnitten werden.

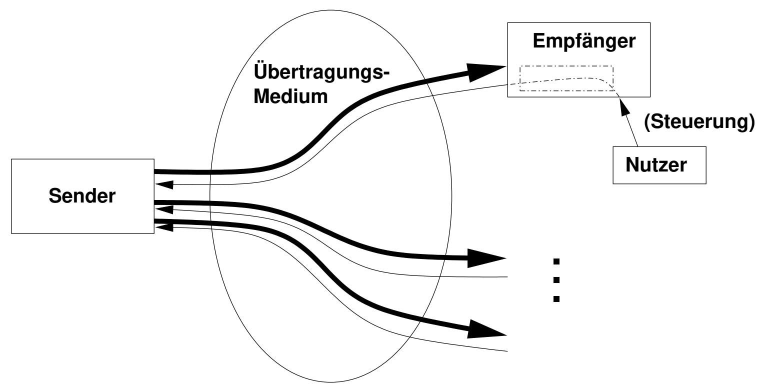

Abbildung 3.2: Datenfluß bei Video on Demand.

Dafür muß das Server-System bedeutend mehr Flexibilität bieten: Der Nutzer kann ein bestimmtes Video auswählen und den Startzeitpunkt frei wählen. Es muß dann nur dieses ausgewählte Video an den Empfänger gesendet werden; es existiert also für jeden Empfänger eine eigene Verbindung. Die Anforderungen an den Server sind dynamisch: es ist nicht vorhersehbar, welche Videos wann gesendet werden sollen. Um eine bestimmte Dienst-Qualität zu garantieren, muß die Möglichkeit bestehen, Dienst-Anforderungen des Nutzers zu verhandeln und gegebenenfalls abzulehnen.

Soll das Video-on-Demand-System in der Funktionalität einem Video-Player entsprechen, können während des Sendens eines Stroms Anforderungen zur Änderung des Stroms eintreffen:

Pause

Das Abspielen wird für einen bestimmten Zeitpunkt unterbrochen.

Der Empfänger wird nicht genügend Speicherkapazität haben, den kontinuierlich eintreffenden Video-Strom bis zum Ende der Pause zu puffern. Deshalb muß der Sender informiert werden, um für den Zeitraum der Pause das Senden ebenfalls zu unterbrechen. Wird die Pause beendet, wird der Sender benachrichtigt und setzt das Senden an der gleichen Stelle fort.

Position anwählen

Es soll eine bestimmte Stelle im Video angefahren werden, um anschließend dort mit dem Abspielen fortzufahren.

Es muß der Video-Server benachrichtigt werden, und er muß die Anforderung umsetzen.

Spulen mit Anzeige

Mit einem Video-Player ist es möglich, daß während des Positionierens die Bilder mit einer entsprechend der Spulgeschwindigkeit erhöhten Geschwindigkeit abgespielt werden.

Diese Funktion ist nicht trivial realisierbar. Würde man den Strom einfach mit mehrfacher Geschwindigkeit senden lassen, würde mehr Bandbreite im Server und im Übertragungsmedium benötigt als ursprünglich reserviert wurde. Soll diese Funktion bereitgestellt werden, muß sie gesondert konzipiert und implementiert werden.

Überlegungen zur Behandlung dieser Funktionen in anderen Dateisystem-Umgebungen sind in [KHS] und [Che95] zu finden.

Soweit möglich sollten diese Funktionen ohne eine erneute Verhandlung der Dienstqualität erfolgen. Eine solche Neuverhandlung beinhaltet das Risiko des Mißerfolgs. Nimmt man den Video-Player als Maßstab, ist ein Abbruch als Antwort auf eine der obigen Anforderungen für den Anwender unverständlich.

Dieses Szenario erfordert also bedeutend mehr Flexibilität vom Echtzeit-Dateisystem. Es ist nicht im voraus bekannt, welche Videos wann abgespielt werden sollen, und es könnte auch gleichzeitig mehrmals auf ein und dasselbe Video zugegriffen werden. Um die Funktionen eines Video-Players vollständig nachzubilden, ist zusätzlicher Aufwand nötig.

3.2.3 Server für Audio on Demand

Ein System für Audio on Demand entspricht im Prinzip dem oben für Video on Demand beschriebenen. Während ein Video-Strom (MPEG-2 MP@ML) eine Datenrate von 15MBit benötigt, genügen für einen Audio-Strom vergleichbar hoher Qualität (MPEG-Audio Layer 3) 128KBit/s. Ausgehend von diesen Zahlen könnte ein Server also über einhundertmal mehr Audio-Ströme senden.

Dieses Szenario stellt somit die Skalierbarkeit des Systems hinsichtlich der Gesamt-Anzahl bereitstellbarer Ströme auf die Probe.

3.2.4 Anrufbeantworter

Die Bandbreite von Audio-Strömen ist bedeutend geringer als die von Video-Strömen. Wenn die Audio-Qualität hochkomprimierender Verfahren (GSM, [Sco95], 13 KBit/s) ausreichend ist, können theoretisch statt eines einzigen MPEG-2-MP@ML-Stroms tausend Audio-Ströme bearbeitet werden.

Mit einem einzigen Rechner könnte man somit Systeme für einen großen Benutzerkreis betreiben, zum Beispiel einen firmenweiten Anrufbeantworter, der Anrufe aufzeichnen und später wiedergeben kann.

Dieses Szenario erhöht nochmals aufgrund der verringerten Bandbreite die Anzahl der in einem System möglichen Ströme. Zusätzlich ist es hier notwendig, in Echtzeit zu schreiben.

3.2.5 Videobearbeitung

Für die Bearbeitung von Videos müssen diese meist in unkomprimierter Form vorliegen. Echtzeit-Anforderungen beim Bearbeiten von Videos treten nur dann auf, wenn die Videos in ihrer wirklichen Abspielgeschwindigkeit bearbeitet werden sollen. Für ein Video in Fernseh-Qualität ergibt sich bei einer Auflösung von 720*576 Bildpunkten, einer Framerate von 30 Frames/s und einer Pixelgröße von 24 Bit eine Datenrate von 35 MByte/s. Wenn man einen Strom in einem Rechner bearbeiten will, müssen die Daten nicht nur zum Prozessor hin, sondern auch zurück fließen, man benötigt also die doppelte Bandbreite.

Die Bearbeitung von Videos stellt hohe Anforderungen an die Hardware. Deshalb ist es notwendig, das Dateisystem effizient zu gestalten und hohe Datenraten zu ermöglichen.

3.2.6 Schreiben von CDs

Ursprünglich wurde die CD als Speicher- und Vertriebs-Medium für Audio-Strömen entwickelt. Später wurde darauf aufbauend für die Datenspeicherung die CD-ROM entwickelt, und aufgrund der Anforderungen im EDV-Bereich wurden Techniken zum Schreiben von CDs entwickelt. Dazu werden speziell CD-R-Laufwerke und -Rohlinge benötigt; diese Rohlinge sind nur einmal beschreibbar.

Ein fertig beschriebener CD-R-Rohling soll sich wie eine klassische CD verhalten. Daraus erwachsen einige technische Restriktionen, die dazu führen, daß beim Schreiben einer CD die Daten mit konstanter Datenrate geliefert werden müssen und ein Abbrechen des Datenstroms den Rohling unbenutzbar und somit wertlos hinterläßt: also ein Echtzeit-Problem.

In der Praxis wird das Problem gelöst, indem man schnelle Rechner nimmt und hofft, daß der Ressourcen-Überschuß ausreicht. In DROPS kann man stattdessen die Daten als Echtzeitstrom von Platte gelesen und zum CD-Rekorder senden.

Dieses Szenario entspringt der Suche nach DROPS-Anwendungen, die nicht an Multimedia gekoppelt sind. Es ergeben sich keine neue Anforderungen an das Echtzeit-Dateisystem.

3.2.7 Zusammenfassung

Die vorgestellten Szenarien zeigen das Spektrum der Anforderungen an das Dateisystem. Die Bandbreiten liegen zwischen 13 kBit/s und mehr als 35 MByte/s. In den meisten Anwendungen fließen die Ströme kontinuierlich mit schwankender Datenrate.

3.3 Charakteristische Eigenschaften von Multimedia-Strömen

Um einen Strom unter Echtzeitbedingungen zu bearbeiten, müssen dafür im voraus Ressourcen reserviert werden. Um nicht unnötig Ressourcen zu verschwenden, sollen nur so viele Ressourcen belegt werden wie für den beabsichtigten Bearbeitungsvorgang erforderlich sind. In einem Dateisystem hängen die benötigten Ressourcen zu großen Teilen davon ab, wieviele Daten dieses Stroms in einem bestimmten Zeitbereich übertragen werden sollen. Deshalb ist es für die Reservierung notwendig, möglichst genau das Verhalten der Datenrate des Stromes zu beschreiben.

In [Ens94] wurden die Eigenschaften vom MPEG-Video-Strömen im Hinblick auf die Übertragung in ATM-Netzen untersucht. In dieser Arbeit wurden mit statistischen Methoden die Wahrscheinlichkeit eines Datenverlusts beim Multiplexen von MPEG-Video-Strömen über eine ATM-Verbindung ermittelt und die Ergebnisse mittels Simulation validiert. Die Videoströme wurden in ATM-Zellen aufgeteilt und es wurde die Wahrscheinlichkeit, daß aufgrund eines Pufferüberlaufs Zellen verlorengehen, in Abhängigkeit von der Größe der ATM-Zellen dargestellt. Für reale MPEG-Ströme genügt ein Puffer von 100 Zellen (mit je 47 Byte), um die Wahrscheinlichkeit des Zellverlusts auf unter 10^{-3} zu senken. Die benötigte Puffer-Größe hängt dabei von der maximalen Burstlänge ab.

Im folgenden werden Ergebnisse eigener Messungen dargestellt, um zu untersuchen, welches Schwankungsverhalten von Multimedia-Strömen in DROPS-Anwendungen zu erwarten ist und welche Auswirkungen sich daraus ergeben.

3.3.1 Untersuchung von MPEG-Video-Strömen

Da keine Videos im MPEG-Format vorlagen, wurden verschiedene Videos eingelesen und mit dem MPEG-2-Encoder der MPEG Software Simulation Group [1] kodiert. Die Videos wurden mit einer Auflösung von 200*320 Bildpunkten und einer Framerate von 25 Bildern je Sekunde aufgenommen. Die Ergebnisse von zwei Videos werden hier vorgestellt: Video A ist ein Konzertmitschnitt einer Rockband, Video B ist ein Actionfilm. (Zusätzlich wurde Video B mit der Shareware-Version eines Encoders von Xing kodiert, das Ergebnis ist Strom B'.) Bei den ausgewählten Videos ändern sich die Bildinhalte teilweise stark; es wäre also zu erwarten, daß die Datenrate dementsprechend stark schwankt.

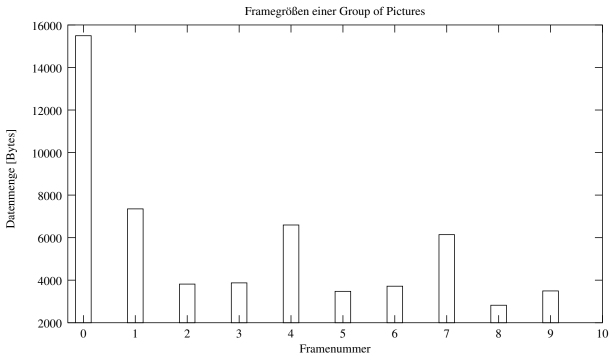

Abbildung 3.3: Beispiel: Größe der Frames

Da die die I-, P- und B-Frames unterschiedlich viele Information beinhalten, sind die entsprechenden kodierten Frames unterschiedlich groß. In Abb. 3.3 sind die ersten Frames von Strom B dargestellt. Erwartungsgemäß ist der I-Frame am größten, und die B-Frames sind am kleinsten.

Wenn man jedoch jeweils die Gesamtdatenmenge der Framefolgen (*group of pictures, GOP*) betrachtet, die mit einem I-Frame beginnen und vor dem nächsten I-Frame enden, ergibt sich dagegen eine relativ geringe Schwankung der Datenrate.

Allerdings enthält Strom A ein paar GOPs, deren Größe deutlich über dem Durchschnitt liegen. Es ist zu beobachten, daß dafür die benachbarten GOPS bedeutend kleiner sind. Der MPEG-Encoder hat also bereits versucht, die Bandbreiten-Schwankung innerhalb kurzer Zeit auszugleichen.

	GOP-Anzahl	GOP-Länge	b (Byte/GOP)	n (Byte)	v (Byte)
Strom A	291	12	552404	349864	508728
Strom B	291	24	1105133	121081	3971
Strom B'	169	15	494913	39084	47697

Tabelle 3.1: Pufferbedarf für die Video-Ströme

[1] http://www.mpeg.org/MSSG/

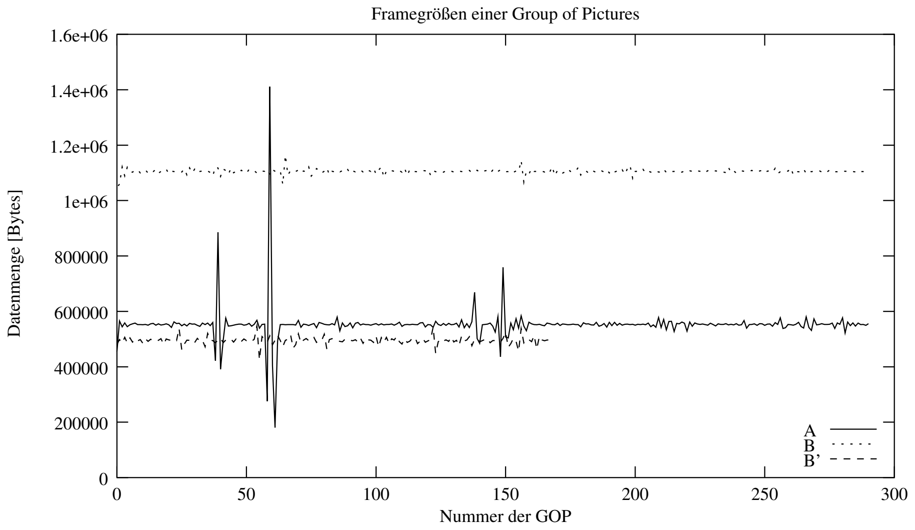

Abbildung 3.4: Gemessene Größen der GOPs

Stellt man diese Videos als Ströme konstanter Bandbreite mit Schwankungskorridor (2.1.3), ergeben sich die in Tabelle 3.1 angegebenen Werte.

Bei Video B werden also weniger als 130 KByte benötigt, um die Schwankungen der Datenrate auszugleichen. Damit ist es sinnvoll, diese Ströme als schwankungsbeschränkte Ströme zu modellieren. Bei Video A sind durch vereinzelt auftretende Spitzen insgesamt fast 900 KByte nötig. Insgesamt ist aber trotzdem zu beobachten, daß die Schwankungen relativ zur Datenmenge des Gesamtstroms sehr gering sind.

Dieses Verhalten kann man damit erklären, daß der MPEG-Encoder versucht, einen Strom relativ konstanter Datenrate zu erzeugen. MPEG-1 wurde größtenteils für die Speicherung von Videos auf CDs genutzt. CDs haben eine konstante Datenrate, und so ist es naheliegend, den Encoder daraufhin zu parametrisieren.

Das MPEG-Kodierverfahren müßte eigentlich bei geringen Bild-Änderungen wenige Daten und bei großen Änderungen mehr Daten pro Frame erzeugen, um eine konstante Bildqualität zu erzeugen. Offensichtlich wird dies nicht getan, wenn der Encoder Ströme relativ konstanter Datenrate erzeugt. Stattdessen wird die Qualität (insbesondere Detail-Auflösung) der Bilder der verfügbaren Bandbreite angepaßt. Man geht also davon aus, daß diese Qualitätseinbußen in diesen Anwendungen akzeptabel sind.

Will man Videos über Rechnernetze übertragen, erfordert es zusätzlichen Aufwand, um Videos mit schwankenden Datenraten zu übertragen (siehe [Ens94]). Eine naheliegende Lösung ist, auch in dieser Anwendung auf höchste Qualität zu verzichten.

Es ist prinzipiell möglich, daß sich Anwendungen etablieren werden, die qualitätskonstant kodierte MPEG-Ströme mit stärker schwankender Datenrate nutzen. Die aktuellen Entwicklungen im Bereich der Video-Speicherung auf CDs höherer Kapazität lassen dies zumindest für den sehr bedeutenden Consumer-Bereich als unwahrscheinlich erscheinen.

3.3.2 Andere Formate für Multimedia-Ströme

Andere Video- und Audio-Formate werden ebenfalls in Systemen mit konstanter Datenrate eingesetzt:

H.261
H.261 wurde für Video-Konferenzen über ISDN entwickelt. Ein ISDN-Kanal hat eine konstante Bandbreite von 64 kBit/s.

GSM-Kodierung
GSM stellt für die Audio-Ströme eine konstante Datenrate von 13kBit/s bereit.

Die konstanten Datenraten sagen genaugenommen nichts über die Schwankung aus, aber da sowohl Video-Konferenzen als auch Telefonieren interaktive Anwendungen sind, ist gewährleistet, daß die Daten mindestens mehrmals pro Sekunde gesendet werden und die Schwankung der Datenmenge somit gering ist.

3.3.3 Schlußfolgerung

Die Datenraten-Schwankung der betrachteten Formate für Multimedia-Ströme ist ausreichend klein, so daß sie durch den Hauptspeicher des Rechners ausgeglichen werden kann. Damit ist das Modell schwankungsbeschränkter Ströme prinzipiell anwendbar.

Auch das als in dieser Hinsicht als kritisch vermutete MPEG-Video-Format hat zumindest in der momentan genutzten Form nur eine geringe Schwankung der Datenrate. (Wenn dies kurzfristig nicht gilt, ist es immer noch möglich, den Strom aus mehreren Teilströmen unterschiedlicher Datenrate zusammenzusetzen.)

3.4 Admission Control mittels Festplatten-Simulation

Um eine möglichst hohe zugesagte Übertragungs-Datenrate von einer Festplatte zu erreichen, müßte man möglichst viel über das Übertragungs-Verhalten der Festplatte wissen. In [WGPW95] werden verschiedene Verfahren zur Ermittlung von Parametern von SCSI-Festplatten vorgestellt. Diese Verfahren werden validiert, indem man SCSI-Festplatten mit diesen ermittelten Parametern und Test-Datenströmen in Software simuliert (siehe auch [RW94] und [WGP94]) und die Ergebnisse vergleicht. Diese Messungen zeigen, daß sich die Parameter von Festplatten mit sehr guter Genauigkeit und akzeptablem Zeitaufwand ermitteln lassen.

Für die Admission Control könnte man sich diese Verfahren zunutze machen, indem man alle Aufträge im voraus in einem Simulator ausführen läßt und anhand dessen entscheidet, ob diese Aufträge durchführbar sein werden.

Allerdings läßt dieser Ansatz ähnliche Probleme wie in 2.3.3 beschrieben erwarten: Es ergibt sich ein hoher Rechenzeit- und Speicher-Bedarf, um alle Aufträge zu speichern und bei jeder Admission-Control-Anforderung die Simulation für den gesamten Zeitlauf zu durchlaufen. Deshalb wurde dieser Ansatz im Rahmen dieses Projekts nicht weiter verfolgt.

3.5 Admission Control auf Bandbreiten-Basis

Aus der Analyse des Admission-Control-Verfahrens des UBC CMFS (siehe 2.3.2) ergab sich die Idee, die Datenmengen-Vektoren durch Polynome genähert darzustellen.

Nutzt man algorithmisch leicht handhabbare Polynome nullten Grades, wird ein Strom durch eine Folge von Teilströmen mit konstanter Datenrate dargestellt. Da Multimedia-Ströme meist ohnehin eine relativ konstante Datenrate haben, ist es nicht notwendig, zu Polynomen höheren Grades überzugehen.

Im einfachsten Fall wird ein MPEG-Strom durch ein einziges Segment dargestellt. In dieser Darstellung geht aber das Schwankungsverhalten verloren, man müßte deshalb jedes Strom-Segment als Strom mit Schwankungskorridor darstellen.

Die Admission Control besteht dann darin, die Bandbreiten der verschiedenen Zeitabschnitte zu addieren und den für Schwankungsausgleich vorgesehen Speicher entsprechend zu vergrößern. Zusätzlich kann man kurzzeitige Bandbreiten-Engpässe durch vorausschauendes Lesen (*Read-Ahead*), also durch mehr Speicher, ausgleichen.

Vorteile dieser Darstellung sind der geringere Speicherbedarf und die Einfachheit der Operationen. Der Nachteil dieses Verfahrens ist, daß die Ebene des Plattenzugriffs gar nicht berücksichtigt wird. Damit könnten im ungünstigsten Fall alle Ströme zur gleichen Zeit einen Plattenzugriff durchführen. Will man dieses Problem durch vorausschauendes Lesen (*Read-Ahead*) lösen, muß man pro Strom eine Slot-Länge im voraus planen. Sind sehr viele Ströme im System, ist dieses Verfahren ressourcenaufwendig.

3.6 Admission Control mittels Slot-Reservierung

Um die Probleme der bisherigen Admission-Control-Verfahren zu vermeiden, wurde ein Verfahren entwickelt, das für jeden Strom im voraus feste Slots zum Zugriff auf die Festplatte (siehe 2.1.2) zuordnet . In diesem Verfahren wurde aber die Liste der Slots periodisch gestaltet, so daß die sich ergebende Beschreibung nur wenig Speicherplatz beansprucht und mit geringem Rechenaufwand handhabbar ist.

3.6.1 Binäre Bus-Aufteilung

Entsprechend dem Slot-Modell zur Aufteilung des SCSI-Busses (2.1.2) haben alle Slots für den Zugriff auf eine SCSI-Festplatte die gleiche Länge. Auf dieser Basis läßt sich der Maximal-Datenstrom in zwei Datenströme halber Bandbreite aufteilen, indem abwechselnd ein Slot dem ersten Teilstrom und der nächste Slot dem zweiten Teilstrom zugeordnet werden (Abb. 3.5).

Wendet man dieses Verfahren auf einen so erhaltenen Teilstrom an, erhält man wiederum zwei Ströme, von denen jeder die halbe Bandbreite des ursprünglichen Stroms hat.

3.6.2 Abbildung auf binären Baum

Jeder Strom läßt sich in zwei Teilströme aufteilen. Dieser Sachverhalt läßt sich durch einen binären Baum darstellen. Der oberste Knoten repräsentiert einen Strom mit maximaler Bandbreite, dieser Strom wird in zwei Teilströme, Strom A und Strom B, unterteilt (Abb. 3.6).

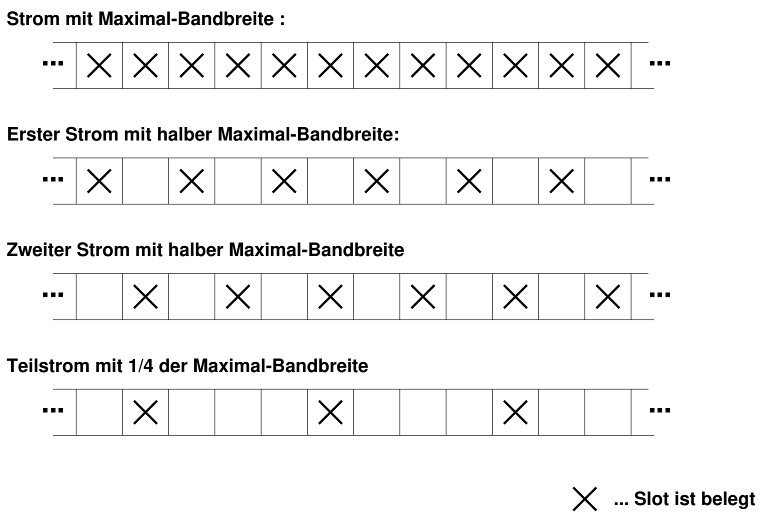

Abbildung 3.5: Aufteilung eines Stroms in Teilströme

Die Blatt-Knoten (*leaf nodes*) dieses Baumes repräsentieren nicht weiter unterteilte Ströme. Alle diese Ströme haben disjunkte Slot-Mengen.

Der zur Verfügung stehende Zeitraum ist für alle Aufträge gleich lang. Für die Ausführung der Aufträge wurde eine virtuelle Zeit, die Auftrags-Zeit, definiert.

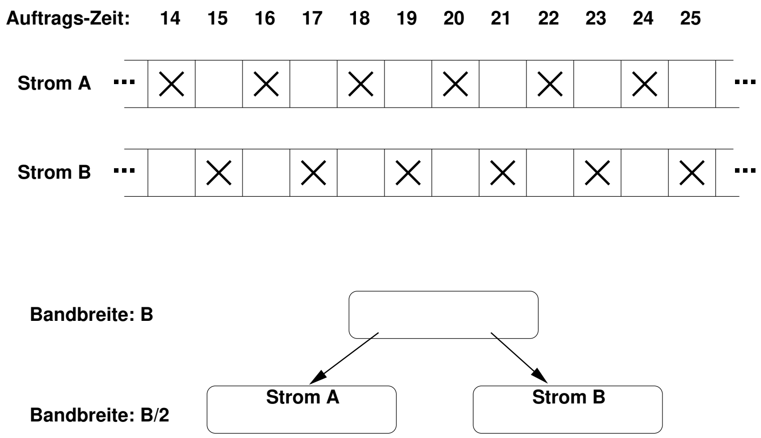

Abbildung 3.6: Abbildung auf binären Baum

Durch die Struktur des binären Baums läßt sich ein großer Bandbreiten-Bereich abdecken. Konzeptionell gibt es keine kleinste Bandbreite.

3.6.3 Strom-Adressen

Um für jeden Strom festzulegen, zu welchen Zeitpunkten seine Aufträge erfüllt werden sollen, muß ein Zusammenhang zwischen der Lage seines Knotens im binären Baum und den Erfüllungszeitpunkten erstellt werden. Die Lage eines Knotens wird durch seine Adresse bestimmt. Im folgenden wird definiert, wie ein Strom im binären Baum adressiert wird.

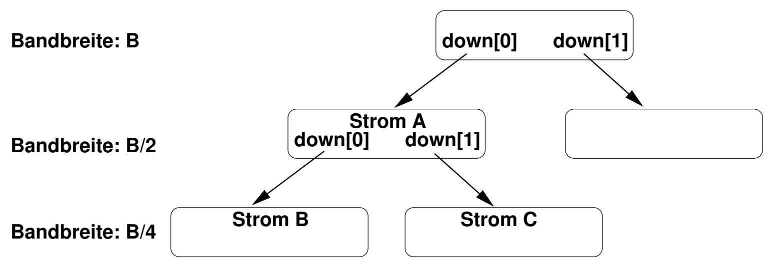

Abbildung 3.7: Adressierung von Strömen im binären Baum

In Abb. 3.7 werden zwei Ströme mit jeweils einem Viertel der maximalen Bandbreite dargestellt. Strom A ist ein Strom halber Bandbreite. Er wird wiederum halbiert, indem seine Aufträge abwechselnd den Strömen B und C zugeordnet werden.

Ordnet man die Verweise auf die halbierten Ströme in einem Feld an, stellen die 0 und die 1 den Index dar, der in diesem Feld auf den linken beziehungsweise den rechten Strom verweist. (Im folgenden wird dieses Feld `down` genannt, `down[0]` verweist auf den linken Unter-Strom und `down[1]` auf den rechten.)

Im Baum ist die Lage eines Elements durch die Indizes bestimmt, durch die er vom obersten Element erreicht wird. Die Folge dieser Indizes wird als **Strom-Adresse** bezeichnet. Strom A hat die Adresse “0”, Strom B die Adresse “00” und Strom C die Adresse “01”. Für das oberste Element, das den Strom voller Bandbreite repräsentiert, ist die Adresse die leere Folge “”.

Eine Strom-Adresse kann beliebig lang sein, es lassen sich also Ströme beliebig kleiner Bandbreite adressieren.

3.6.4 Strom-Adresse und Erfüllungszeitpunkt

Das Datei-System muß von der Admission Control wissen, welcher Strom zu einem bestimmten Zeitpunkt zu bearbeiten ist. Dazu muß es möglich sein, die durch die Blatt-Knoten dargestellten nicht weiter unterteilten Ströme eineindeutig auf einen Erfüllungszeitpunkt abzubilden.

In Abb. 3.6 wurde ein Strom in zwei Teilströme aufgeteilt, die über die Indizes 0 und 1 vom obersten Knoten erreicht werden.

Da die Aufträge der beiden Teilströme jeweils abwechselnd ausgeführt werden, werden alle Aufträge des einen Stroms zu geradzahligen Zeitpunkten und die des anderen zu ungeradzahligen Zeitpunkten ausgeführt. Es wird festgelegt, daß der linke Strom zu einem geradzahligen Zeitpunkt beginnt. Damit werden die Aufträge zum Zeitpunkt $t \bmod 2 + i$ ausgeführt, wobei t die Zeit und i die Adresse des Stroms ist.

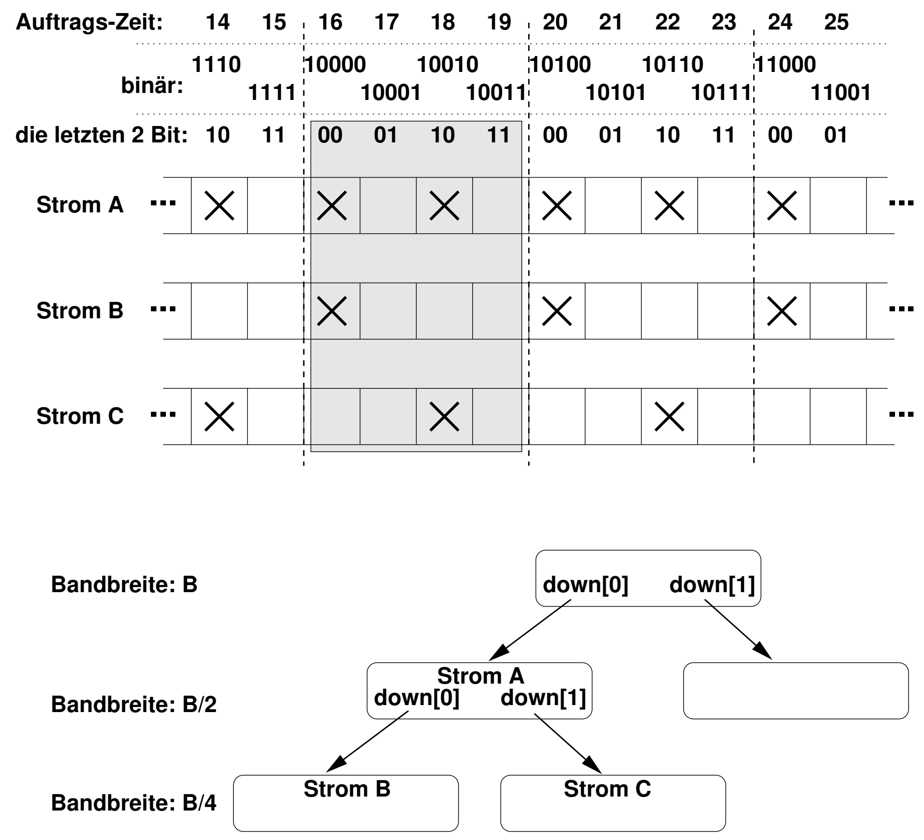

Abbildung 3.8: Zusammenhang zwischen Strom-Adresse und Erfüllungszeitpunkt

Im folgenden soll dieser Zusammenhang für mehrfach halbierte Bandbreiten verallgemeinert werden. In Abb. 3.8 wurde ein Strom A halber Bandbreite in zwei Ströme B und C mit viertel Bandbreite aufgeteilt.

Wenn keine kleineren Bandbreiten-Bruchteile im System vorkommen, wiederholt sich der Plan zyklisch nach vier Zeiteinheiten. Die letzten zwei Bit der Binär-Darstellung der Auftragszeit wiederholen sich ebenfalls in diesem Zyklus.

Die Adressen und Zeitpunkte der Ströme sind:

	Adresse	Zeitpunkt mod 4 (binär)
	“”	00, 10, 10, 11
Strom A	“0”	00, 10
Strom B	“00”	00
Strom C	“01”	10

Für einen Strom voller Bandbreite ist die aktuelle Zeit irrelevant; für ihn muß jeder Slot benutzt werden, Bei halber Bandbreite entscheidet das letzte Bit der Zeit, bei Strömen mit einem Viertel der

Bandbreite die letzten beiden Bits. Bei jeder Halbierung wird also ein weiteres Bit der Zeit für die Entscheidung, ob der Slot für diesen Strom genutzt wird, signifikant.

Bei der Halbierung wird der Strom-Adresse eine Ziffer angefügt. Genau diese Ziffer wird den signifikanten Bits der Zeit vorangestellt, um die neue Folge signifikanter Bits zu erhalten. Dadurch entsteht die Folge signifikanter Bits aus der Strom-Adresse, indem man die Reihenfolge der Ziffern der Strom-Adresse umkehrt.

Durch diese Methode sind jedem Element im binären Baum die Zeitpunkte für die Auftrags-Erfüllung zugeordnet. Für jeden nicht weiter unterteilte Knoten (Blatt-Knoten) ist sichergestellt, daß nur dieser Strom zu diesen Zeitpunkten den Slot nutzen kann. Die inneren Knoten (die Knoten, die keine Blatt-Knoten sind) stellen keine wirklich nutzbaren Ströme dar. sondern sie drücken nur aus, daß dieser Strom teilweise genutzt wird.

Das Dateisystem sendet die so ermittelten Aufträge mit festen Erfüllungs-Zeitpunkten an den SCSI-Treiber. Anhand der Zeitpunkte garantiert der SCSI-Treiber, daß jeder Auftrag spätestens zum angegebenen Zeitpunkt erfüllt wurde.

3.6.5 Zeitlich begrenzte Ströme

Bisher wurde nicht berücksichtigt, daß die Ströme nicht unendlich lang sind, sondern einen Anfangs- und einen Endzeitpunkt haben. Zwei Ströme können die gleiche Strom-Adresse haben, wenn die Anfangszeit des einen größer als die Endzeit des anderen ist.

Deshalb wird jedem Element des binären Baums ein Gültigkeits-Bereich zugeordnet. Wenn zwei Ströme unterschiedlicher Dauer einen gemeinsamen übergeordneten Strom haben, muß dieser hinsichtlich seiner Dauer aufgeteilt werden, so daß ein Teil dieses Stroms die gemeinsamen Dauer abdeckt und andere Teile die übrigen Zeitbereiche.

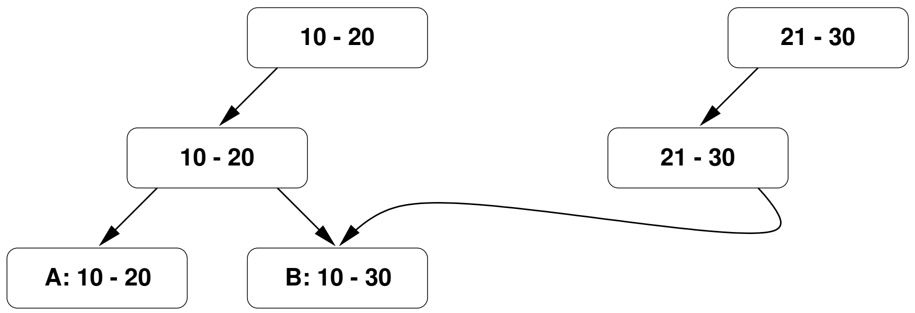

Abbildung 3.9: Binäre Bäume für zeitlich begrenzte Ströme

In 3.9 wird die sich ergebende Struktur für zwei Ströme A und B mit gleichem Anfangs-, aber unterschiedlichem Endzeitpunkt dargestellt.

Wenn man von den die maximale Bandbreite repräsentierenden Elementen ausgeht, sind sie weiterhin die Wurzel-Elemente binärer Bäume, und diese Bäume haben gemeinsame Teilbäume. Beginnt man stattdessen unten mit den Blatt-Elementen, die die Ströme A und B repräsentieren, sind diese ebenfalls Wurzel-Elemente von Bäumen, die gemeinsame Teilbäume haben. Von oben nach unten sind die

Bäume entsprechend der Bandbreite aufgeteilt, von unten nach oben entsprechend der abgedeckten Zeitbereiche. Ein jedes Element stellt eine größere Bandbreite als ein darunterliegendes Element dar, und sein Gültigkeitsbereich ist der des darunterliegenden Elements oder ein Teilbereich davon.

3.6.6 Durchführung der Admission Control

Das Einplanen eines neuen Stroms geschieht in folgenden Schritten:

1. Finden einer passenden freien Strom-Adresse

   Für den neuen Strom wird ein freier Platz im Baum gesucht, der die benötigte Bandbreite hat und den geforderten Zeitraum vollständig abdeckt. Gibt es mehrere Möglichkeiten, muß eine davon ausgewählt werden; gibt es keine, ist die Admission Control für diesen Strom beendet und es wird eine Fehlermeldung zurückgegeben.

2. Erstellen des Blatt-Knoten

   Für den Strom wird ein Blatt-Knoten mit den geforderten Parametern erstellt.

3. Einbinden des neuen Elements in den Baum

   Gibt es Knoten doppelter Bandbreite, die den Gültigkeitsbereich des neuen Blattknoten teilweise abdecken, werden sie an diesen Zeitgrenzen aufgeteilt. Anschließend wird der neue Knoten mit seinen übergeordneten Knoten verbunden.

Wird beim Suchen der freien Strom-Adresse keine ausreichend langer Bereich gefunden, kann man versuchen, die benötigte Bandbreite durch mehrere Teilabschnitte zusammenzusetzen.

3.6.7 Ströme „beliebiger“ Bandbreite

In einem vielfältig nutzbaren Dateisystem werden die Bandbreiten der meisten Ströme nicht exakt ein Zweierpotenzen-Bruchteil der maximalen Bandbreite sein. Im folgenden wird gezeigt, wie solche Ströme gehandhabt werden können.

Reservierung „beliebiger“ Bandbreiten

Liegt die Bandbreite eines zu reservierenden Stroms knapp über einem Zweierpotenz-Bruchteile der maximalen Bandbreite reserviert, werden beinahe 50% der Bandbreite verschwendet.

Um diese Verluste zu verringern, kann man den Strom aus Teilströmen verschiedener Bandbreiten zusammensetzen. Allerdings werden für jeden Teilstrom sowohl für die Baum-Struktur als auch für die Pufferung zusätzliche Ressourcen benötigt. Deshalb sollten nur wenige Teilströme genutzt werden.

Anpassung der Transfer-Bandbreite

Ein anderer Ansatz zur Bereitstellung einer bestimmten Bandbreite besteht darin, die reservierten Slots nicht vollständig zu nutzen.

Wird für einen Strom eine bestimmte Bandbreite reserviert, heißt das, daß für diesen Strom eine dieser Bandbreite entsprechende Anzahl von Slots verfügbar ist. Diese Slots müssen aber nicht für diesen Strom genutzt werden. Eine geringere Bandbreite als die reservierte läßt sich erreichen, indem in entsprechendem Abstand Slots nicht genutzt werden.

Diese ausgelassenen Slots sind für die Echtzeit-Anwendungen verloren, sie können aber vom SCSI-Treiber für vorliegende Nicht-Echtzeit-Aufträge verwendet werden.

Praktisch wird man die beiden Methoden kombinieren, um eine bestimmte Bandbreite bereitzustellen.

3.6.8 Planung des Startzeitpunkts

Wurde einmal eine Slot-Adresse für einen Strom bestimmt, sind damit auch die möglichen Zeitpunkte für den Start des Stroms festgelegt. Beträgt zum Beispiel die Bandbreite $1/64$ der maximalen Bandbreite und die Slotdauer 50 ms, so kann der entsprechende Strom nur nach jeweils 3,2 s gestartet werden. Für Video-Anwendungen ist das wahrscheinlich akzeptabel, aber da sich diese Zeit pro Bandbreiten-Halbierung verdoppelt, ergeben sich bei Audio-Strömen sehr hohe mögliche Verzögerungen.

Ein Lösungsansatz wäre, den ersten Block bereits bei der Admission Control einzulesen. Dann könnte der Strom jederzeit gestartet werden, allerdings wird für den zusätzlichen Block mehr Pufferspeicher benötigt.

Wenn noch genug Bandbreite im System frei ist, könnte man im Rahmen der Admission Control mehrere zeitversetzte Ströme belegen und nach dem Start die nicht benötigten Slots freigeben.

3.6.9 Nutzung mehrerer Festplatten

Die bisher vorgestellte Reservierungs-Struktur ist für nur eine Festplatte vorgesehen. Werden mehrere Platten genutzt, muß für jede Platte eine extra Reservierungs-Struktur genutzt werden. Soll ein Strom bearbeitet werden, der auf mehrere Platten verteilt ist, muß das Dateisystem die Teilströme bei den entsprechenden Platten reservieren.

Wird die Transfer-Bandbreite durch Auslassen von Slots angepaßt, muß das Dateisystem bereits beim Anlegen die Datei diese so auf die Platten verteilen, daß beim Auslassen eines Slots auch die zugehörige Platte ausgelassen wird.

3.6.10 Zusammenfassung

Das entworfene Verfahren kann die an das DROPS-Echtzeitdateisystem gestellten Forderungen hinsichtlich Flexibilität und Eignung für verschiedene Anwendungsszenarien sowie hinsichtlich Zuverlässigkeit der Auftragserfüllung erfüllen.

Allerdings ist eine enge Integration mit den anderen Komponenten des Dateisystems notwendig, um die Möglichkeiten auszuschöpfen.

Mit dem angegebenen Verfahren lassen sich auch andere Ressourcen verwalten. Wenn der SCSI-Bus einen Engpaß darstellen würde, könnte man ihn zusätzlich durch eine Reservierungs-Struktur darstellen und für jeden Platten-Slot zusätzlich einen Bus-Slot belegen. Es könnte auch möglich sein, den PCI-Bus damit zu steuern; dafür muß man ihn in Slots fester Länge aufteilen.

Kapitel 4

Implementierung

Es wurde die in Kapitel 3 entworfene Slot-basierte Admission Control implementiert. Die implementierten Funktionen wurden in einer Bibliothek zusammengefaßt. Zusätzlich wurden Test-Programme implementiert, um die Korrektheit und Robustheit der Implementierung zu prüfen.

Als Implementierungssprache wurde C++ gewählt. Folgende Eigenschaften sprechen dafür:

strengere Syntax-Prüfung
: ANSI C++ hat gegenüber C strengere Anforderungen an die Typ-Kompatibilität. Darauf aufbauend führt der zur Verfügung stehenden Compiler (GCC) für C++-Programm eine umfangreiche Überprüfung durch.

zusätzliche Funktionalität
: C++ stellt neue Mechanismen zur Verfügung. In der Implementierung werden Konstruktoren/Destruktoren und Templates benutzt, und es werden für manche Klassen Operatoren definiert.

STL
: STL (Standard Template Library) ist eine Bibliothek generischer Datentypen und Algorithmen. Sie stellt verschiedene Container-Typen und dazu passende Algorithmen bereit, die durch Nutzung von Templates nicht nur auf vordefinierte Datentypen anwendbar sind.

Neue Begriffe

Im Entwurf ist die Bandbreite eines Stroms ein Zweierpotenzen-Bruchteil der maximalen Bandbreite. Ein solcher Strom entspricht einer Folge von Slots, die in einer Zweierpotenz entsprechenden Abständen aufeinanderfolgten und durch eine Strom-Adresse bestimmt waren.

Die Reservierung von Bandbreite erfolgt in Zweierpotenz-Bruchteilen in einem binären Baum. Ein jedes Element dieses Baums wird im folgenden als **Slot** bezeichnet, es wird durch die **Slot-Adresse** adressiert. Damit entspricht eine solche Slot-Adresse der im Entwurf angegebenen Strom-Adresse.

Analog zu den Blatt-Knoten des binären Baums werden die nicht weiter unterteilten Slots als **Blatt-Slots** bezeichnet.

Ein **Strom** ist im folgenden dagegen eine Menge mehrere Slots, die vom Dateisystem zusammenhängend angefordert wurden. Ein Strom kann aus mehreren Slots verschiedener Bandbreite und Gültigkeitsdauer zusammengesetzt sein.

4.1 Slot-Adressen

Im Entwurf werden diese Adressen als Folge der Ziffern 0 und 1 dargestellt.

Um die Slot-Adresse zu speichern und die Operationen mit ihr effizient zu gestalten, wird diese Folge als `unsigned int` dargestellt, wobei mit dem höchsten Bit begonnen wird. Die Adresse "0" wird durch `0x00000000` und die Adresse "1" durch `0x80000000` dargestellt. In dieser Form können aber zum Beispiel die Adressen "0" und "00" nicht unterschieden werden, deshalb muß zusätzlich gespeichert werden, wieviele Bits für die Angabe der Adresse benutzt werden sollen.

Durch die Bandbreite eines Slots wird angegeben, wie tief er im binären Baum liegt. Damit ist dann auch angegeben, wie lang die Adresse eines Slots ist. Die Bandbreite wird ebenfalls als `unsigned int` dargestellt [1], wobei `0x80000000` die höchste Bandbreite repräsentiert und jede Halbierung der Bandbreite der Halbierung dieses Wertes entspricht.

```
class ac_slot_addr {
  unsigned int a;
  ac_bandwidth bw;

public:
  ac_slot_addr up();

  ac_slot_addr down(int last_bit);
  ac_slot_addr neighbour();

  bool operator==(const ac_slot_addr b);
};
```

Abbildung 4.1: Definition der Slot-Adresse (Ausschnitt)

Durch diese Darstellungsform lassen sich die Algorithmen zur Manipulation von Slot-Adressen mittels bitweiser logischer Verknüpfungen implementieren. Soll zum Beispiel die Adresse des darüberliegenden Slots bestimmt werden, muß nur das Bit, das der aktuellen Bandbreite entspricht, gelöscht werden (Abb. 4.2).

Auf den verwendeten Systemen ist `unsigned int` 32 Bit groß, dadurch ist die kleinste darstellbare Bandbreite $\frac{1}{2^{31}}$ der maximalen Bandbreite. Wäre die maximale Bandbreite 1 GByte/s, wäre die kleinste Bandbreite 0,5 Byte/s. Diese Spannbreite sollte praktisch keine Beschränkung darstellen; anderenfalls wäre sie mit zusätzlichem Implementierungsaufwand umgehbar.

[1] Die Bandbreite ist als extra Typ definiert, der sich zu `unsigned int` konvertieren läßt.

Um mehrere Slot-Adressen zu einer Menge zusammenzufassen, wurde `ac_slot_addr_set` definiert. Implementiert ist es auf Basis des STL-Templates `list<>`.

```
ac_slot_addr ac_slot_addr::up() {

  assert (! bw.is_full());

  return ac_slot_addr(a & ~ (bw << 1), bw << 1);
}
```

Abbildung 4.2: Implementierung von `ac_slot_addr::up()`

4.2 Reservierungs-Struktur

Die Reservierungs-Struktur besteht aus ineinander verschlungenen binären Bäumen (siehe Abb. 3.9). Die Elemente des Baums sind die Slots, die die Slotadresse und den Anfangs- und Endzeitpunkt für die Gültigkeit dieses Slots enthalten. Nur die Blatt-Slots repräsentieren einen eigenständig nutzbaren Strom. Die anderen Slots besagen nur, daß Teile dieses Slots bereits genutzt werden.

Die binären Bäume werden implementiert, indem jeder Slot Zeiger auf darüber- und darunterliegende (hinsichtlich der Bandbreite) und vorhergehende und nachfolgende (hinsichtlich der Zeit) Slots enthält (Abb. 4.3).

```
class ac_slot {

  ac_slot_addr addr;
  ac_time begin, end;
  ac_bandwidth used_bw;
  ac_bandwidth used_bw_fracts;
  ac_slot *down[2], *up;
  ac_slot *next, *prev;
};
```

Abbildung 4.3: Definition eines Slots (Ausschnitt)

Die Verweise sollen folgende Bedeutungen haben:

`up` ist genau dann NULL, wenn die Bandbreite dieses Slots maximal ist. Anderenfalls verweist `up` auf den Slot mit gleicher Anfangszeit, der die doppelte Bandbreite hat und dessen Adresse sich ergibt, indem man die letzte Ziffer der Slot-Adresse entfernt.

`down` ist ein Feld von zwei Verweisen. Sind diese nicht NULL, zeigen sie auf die beiden Teilslots halber Bandbreite. Die Adressen dieser Teilslots entstehen, indem man an die Slot-Adresse den Index in dem Feld anhängt. Die Teilslots müssen mindestens den Zeitbereich dieses Slots abdecken.

`next` zeigt, wenn es nicht NULL ist, auf einen Slot, der die gleiche Slotadresse hat, aber zeitlich später als der erste Slot ist. Seine Anfangszeit muß also größer als die Endzeit des ersten Slots sein. Wenn der Wert von `next` NULL ist, müssen andere Verfahren genutzt werden, um den zeitlich folgenden Slot zu finden.

`prev` zeigt, wenn es nicht NULL ist, auf den zeitlich vorangehenden Slot mit gleicher Adresse; es ist somit das Pendant zu `next`.

Die Reservierungs-Struktur wird im Objekt `ac_tree` zusammengefaßt. Als Start dient ein Verweis auf den zeitlich ersten Slot maximaler Bandbreite. Zusätzlich sind in `ac_tree` alle Funktionen zusammengefaßt, die mehr als einen einzelnen Slot betreffen.

4.3 Ablauf einer Reservierung

Wenn das Dateisystem einen neuen Echtzeit-Strom bereitstellen will, muß es zuerst entscheiden, ob und welche Zweierpotenz-Bruchteile der maximalen Bandbreite genutzt werden sollen. Dann versucht das Dateisystem, einen passenden freien Slot zu finden. Dazu werden mittels

```
ac_slot_addr_set find_free_slots(ac_bandwidth bw,
    ac_time begin, ac_time end);
```

alle Slot-Adressen der in dem entsprechenden Zeitraum freien Slots gesucht. Aus dieser Menge kann man dann entweder die erstbeste Slot-Adresse nehmen, oder man versucht mittels eigener Kriterien eine besonders gut passende Slot-Adresse zu finden.

Wenn keine passenden Slots gefunden wurden, kann man noch mal versuchen, den geforderten Zeitbereich durch mehrere Ströme zusammenzustellen. Mit

```
ac_slot_addr_set find_first_free_slots(ac_bandwidth bw,
    ac_time begin);
```

erhält man alle zu einem bestimmten Zeitpunkt freien Slots zurück.

Wurden nicht genügend passende freie Slots gefunden, ist die Admission Control fehlgeschlagen und das Dateisystem kann den Strom nicht wie gewünscht bearbeiten.

Anderenfalls kann man anschließend den Slot in die Reservierungs-Struktur eintragen:

```
ac_slot *reserve_slot(ac_slot_addr a,
    ac_time begin, ac_time end);
```

Intern erfolgt die Reservierung eines Slots stets in zwei Schritten. Zuerst wird sichergestellt, daß der entsprechende Slot mit genau den geforderten Zeitpunkten existiert, und anschließend werden er und seine darüberliegenden Slots durchlaufen, und es werden die entsprechenden Einträge aktualisiert.

`reserve_slot` wird nur für Blatt-Slots genutzt: es legt das `ac_slot`-Element für den neuen Slot an und initiiert anschließend das Markieren der darüberliegenden Slots mittels:

```
void mark_upper_slots(ac_slot *slot, ac_slot *leaf_slot);
```

`mark_upper_slots` versucht ebenfalls, den aktuellen und alle darüberliegenden Slots zu markieren. Vorher muß es aber wieder sicherstellen, daß die darüberliegenden Slots die geforderten Anfangs- und Endzeitpunkte haben. Da diese Slots keine Blatt-Slots mehr sind (wären sie es, könnte nicht ein darunterliegender Slot reserviert werden), müssen diese Slots zeitlich aufgeteilt und bei Bedarf erstellt werden.

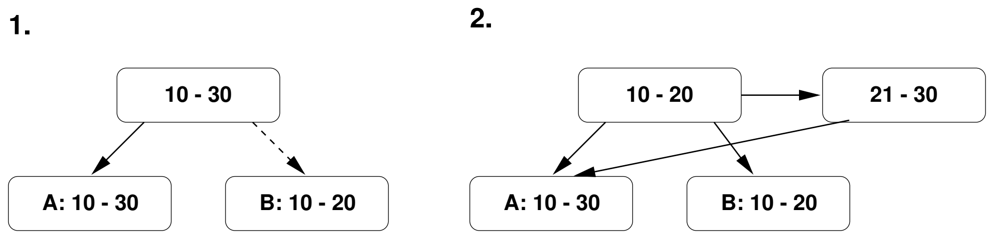

Abbildung 4.4: Beispiel: Der obere Slot muß gesplittet werden.

In Abb. 4.4 ist ein Beispiel angegeben: Slot A existiert bereits, und es kommt Slot B hinzu, der eher als A endet. Der darüberliegende Slot wird zeitlich aufgespaltet, und die entstehenden Slots werden über `prev` und `next` verbunden. Wenn dann B seine darüberliegenden Slots markiert, wird nur der Slot bearbeitet, der den gleichen Zeitraum wie B hat.

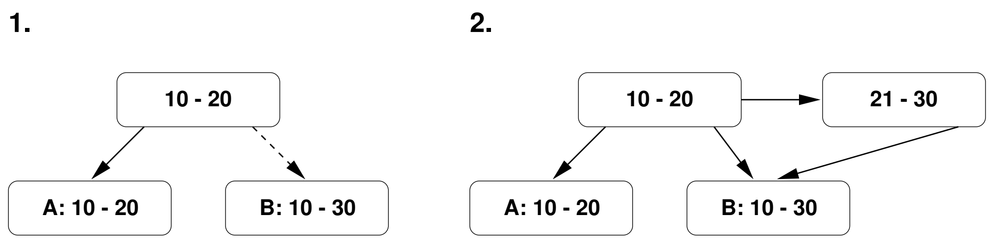

Abbildung 4.5: Beispiel: Es muß ein zeitlich folgender darüberliegender Slot erstellt werden.

Ein anderer Fall ist in Abb. 4.5 dargestellt. Der hinzukommende Slot B endet später als Slot A. Es muß ein darüberliegender Slot erstellt werden, der dem bereits existierenden zeitlich folgt, bevor das Markieren durchgeführt werden kann.

Weitere Fälle ergeben sich, wenn die Anfangszeitpunkte differieren.

Das Aktualisieren der Reservierungs-Struktur beim Reservieren eines neuen Slots erfolgt also dadurch, daß abwechselnd die direkt darüberliegenden Slots erstellt oder gesplittet und anschließend markiert werden. Diese Funktionen rufen sich gegenseitig rekursiv auf. Die Abbruchbedingung ist erreicht, wenn die Slots maximaler Bandbreite bearbeitet wurden.

4.4 Schnittstelle zum Datei-System

Die Schnittstelle wird von zwei Objekten bereitgestellt: Die **Planungstabelle** faßt mehrere Festplatten zusammen und stellt für diese eine von der Reservierungs-Struktur unabhängige Speicherung für die Reservierungsdaten bereit. Dadurch könnten Reservierung und Planung in unterschiedlichen Threads oder Tasks liegen. Ein **Strom** ist eine Menge von Slots, die aus Sicht des Dateisystems einer Datei zugeordnet ist.

4.4.1 Strom

Ein Strom kann aus mehreren Slots zusammengesetzt sein, außerdem kann er auf mehrere Festplatten verteilt sein. `ac_stream` faßt diese Angaben zusammen und stellt Methoden bereit, um Reservierungen vorzunehmen. Ein Strom ist fest mit einer Planungs-Tabelle verbunden, dadurch wird beim Reservieren eines Slots für einen Strom auch der entsprechende Eintrag in der Planungstabelle vorgenommen.

4.4.2 Planungs-Tabelle

Wenn einmal die Admission Control für einen Strom erfolgreich verlaufen ist, benötigt das Dateisystem regelmäßig Informationen, zu welchem Zeitpunkt welche Aufträge erfüllt werden sollen. Dazu wurde eine von der Reservierungs-Struktur unabhängige Planungs-Tabelle definiert.

Wenn das Dateisystem einen Strom reserviert, erhält es eine Slot-Adresse. Diese Slot-Adresse und der Gültigkeitsbereich werden in die Planungstabelle eingetragen. Damit wird die Reservierungs-Struktur nur zur Einplanung eines neu hinzukommenden Stroms benötigt. Ist das einmal erfolgt, kann das Dateisystem nur mit Kenntnis der Planungs-Tabelle die Aufträge für den Strom absenden.

Des weiteren enthält die Planungs-Tabelle die Informationen über die Festplatten. Die bisher beschriebene Reservierung erfolgt für jede Platte einzeln, und die entsprechenden Strukturen werden in der Planungs-Tabelle zusammengefaßt.

`list<ac_slot_range>` beschreibt die Liste der Slots, die für diese Platte vorliegen. Um festzustellen, welcher Slot als nächster zu bearbeiten ist, müßte man im ungünstigsten Fall die ganze Liste durchsuchen. Dieser Aufwand läßt sich erheblich verringern, indem man für die Slot-Addressen mit größeren Bandbreiten einen binären Baum implementiert. Diesen Baum kann man auf einen Vektor abbilden, der die einzelnen Aufträge in Listen enthält (Abb. 4.7).

```
class ac_slot_range {
 public:
  ac_slot_addr addr;
  ac_time begin, end;
  ac_stream *stream;
};

class ac_schedule {
  vector<int> abs_bw;
  vector<ac_tree> tree;
  vector< list<ac_slot_range> > slots;

 public:
  // : get the to-be-done slot range
  ac_slot_range *get_slot_range(int disk, ac_time time);
};
```

Abbildung 4.6: Definition der Planungs-Tabelle (Ausschnitt)

4.5 Testen

Die Reservierung der Slots ist der komplexeste Teil der Implementierung. Sie ist dadurch gekennzeichnet, daß der Anfangs- und Endzustand klar definiert sind. Um Fehlerquellen frühzeitig zu erkennen, werden möglichst häufig die verwendeten Datenstrukturen auf Konsistenz überprüft.

Typgebundene Bedingungen
: Manchmal lassen sich für einen Typ Bedingungen erfüllen, die praktisch jederzeit ohne Kenntnis der Umgebungsverhältnisse erfüllt sein müssen. Solche Bedingungen wurden in der Methode `check()` der jeweiligen Klassen zusammengefaßt. Bei der Slot-Adresse wird zum Beispiel überprüft, daß die aufgrund der Bandbreite nicht nutzbaren Bits auch wirklich den Wert 0 haben.

Situationsgebundene Bedingungen
: Meist ergeben sich die Bedingungen, die an ein Objekt zu stellen sind, erst aus der Situation, in der es angewendet werden soll. Diese Bedingungen werden an der entsprechenden Stelle überprüft.

Es wurden verschiedene Testdatensätze erstellt. Wenn Fehler auftraten, wurde ein möglichst kleiner Testdatensatz erstellt, mit dem sich der Fehler reproduzieren ließ. Zusätzlich wurden umfangreiche Testdatensätze erzeugt, die dazu dienen, diese Fehler zu provozieren.

4.6 Ermittlung der Strombeschreibung

Es wurde ein Programm implementiert, daß aus einer MPEG-Video-Datei die Parameter b, v und n der in Abb. 2.4 angegebenen Strombeschreibung ermittelt.

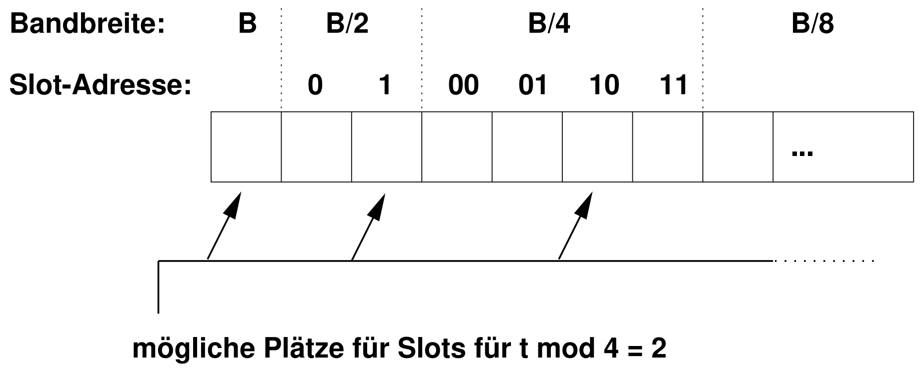

Abbildung 4.7: Optimierungsmöglichkeit für Planungs-Tabelle

Dieses Programm muß die Datei zweimal lesen: Zuerst wird die mittlere Bandbreite b ermittelt, und anschließend wird für jeden Frame überprüft, ob die anfänglich mit 0 initialisierten Werte v und n durch den aktuellen Frame überschritten werden.

Das implementierte Programm benötigt Linux (oder L^{4}Linux), es ist nicht direkt auf L4 nutzbar.

Kapitel 5

Zusammenfassung und Ausblick

In dieser Arbeit wurden Entwurf und Implementierung der Admission Control des Echtzeit-Dateisystems für DROPS beschrieben.

Die Admission Control versucht, Flexibilität hinsichtlich Stromtyp und Bandbreite mit hundertprozentiger Zuverlässigkeit einmal getroffener Zusagen zu verbinden. Dazu wird die Bandbreite eines Busses wiederholt halbiert (binärer Baum), und die entstandenen Bruchteile erhalten einen festen, eindeutigen Zeitpunkt für die Erfüllung ihrer Aufträge zugeordnet.

Dadurch ist ein großer möglicher Bandbreiten-Bereich abgedeckt, aber es werden zusätzliche Mechanismen auf der Ebene des Dateisystems nötig, um nicht nur Zweierpotenzbruchteile, sondern feiner abgestufte Bandbreiten zu erhalten.

Die existierenden Echtzeit-Dateisysteme bieten nicht die Flexibilität des hier gewählten Ansatzes, und traditionelle Filesystem können keine zuverlässigen Zusagen machen. Die in dieser Arbeit entworfenen Mechanismen versuchen, beides zu erfüllen, aber das ist nur teilweise gelungen: Die Flexibilität wird in ein relativ starres Schema eingeordnet, um vorhersagbar zu sein. Dadurch entsteht eine neue Komplexität, die auch andere Teile des Dateisystems beeinflußt.

Für weitere Arbeiten ergeben sich folgende Aufgaben:

- Implementierung der Ermittlung von Stromeigenschaften für neue Multimedia-Formate. Momentan wird nur MPEG-Video unterstützt, und die Implementierung stellt keine Schnittstelle für die Programmiersprache C bereit.

- Anpassung an die zu entwickelnde gemeinsame Strombeschreibung in DROPS. In DROPS sollen die Komponenten zur Admission Control untereinander eine gemeinsame Form der Strombeschreibung nutzen. Die Admission Control muß dieses Format nutzen können, um Reservierungen mit diesen Komponenten verhandeln zu können.

- Weiterentwicklung der Implementierung der Admission Control. Insbesondere muß das Zusammensetzen von Strömen aus Teilströmen unterstützt werden.

- Vollständige Anpassung des Dateisystems an die von der Admission Control bereitgestellten Schnittstellen.

Die vorliegende Implementierung der Admission Control zeigt, daß die entworfenen Konzepte implementierbar sind und funktionieren. Es wurden noch nicht alle Möglichkeiten zur Steigerung der Performance und der zusagbaren Bandbreite ausgenutzt. Ob diese Verbesserungen im Rahmen der entworfenen Konzepte möglich sind oder ob stattdessen diese Konzepte grundlegend überarbeitet werden müssen, könnte Gegenstand neuer Arbeiten sein.

Anhang A

Glossar

Adreßraum Menge der Adressen, auf die man zugreifen kann.

Arbitrierung Protokoll zum Zugriff auf gemeinsam genutzte Medien.

ATM *Asynchronous Transfer Mode*. Hochgeschwindigkeits-Netz (ca. 155 MBit/s) für Sprache, Daten und Multimedia.

Echtzeit Anforderung, daß ein bestimmter Dienst zu einem vorher festgelegten Zeitpunkt erbracht sein muß.

ISDN *Integrated Services Digital Network*. Dienstintegrierendes Netz niedriger Bandbreite (64 kBit/s).

Priorisierung Durchsetzen des Vorrangs beim Zugriff auf gemeinsam genutzte Betriebsmittel.

time-sharing Es existiert keine *Echtzeit*-Anforderung, stattdessen wird die Rechenzeit gleichmäßig auf die anstehenden Programme verteilt.

Literaturverzeichnis

[ANS] ANSI WORKING GROUP X3T9. Small Computer System Interface -2 (SCSI-2) draft. zu finden unter ftp://ncrinfo.ATTGIS.COM/pub/standards/io/scsi2/

[BBD 96] BOLOSKY, William J. ; BARRERA, III, Joseph S. ; DRAVES, Richard P. ; FITZGERALD, Robert P. ; GIBSON, Garth A. ; JONES, Michael B. ; LEVI, Steven P. ; MYHRVOLD, Nathan P. ; RASHID, Richard F.: Distributed Schedule Management in the Tiger Video Fileserver / Microsoft Research, Advanced Technology Division. 1996 (MSR-TR-96-09). – Technical Report

[BFD97] BOLOSKY, William J. ; FITZGERALD, Robert P. ; DOUCEUR, John R.: Distributed Schedule Management in the Tiger Video Fileserver. **In:** *Proceedings of the 16th ACM Symposium on Operating Systems Principles*, 1997

[Che95] CHEN, Huang-Jen: *A Disk Scheduling Scheme and MPEG Data Layout Policy for interactive Video Access from a Single Disk Storage Device*, Boston University College of Engineering, Diplomarbeit, 1995

[Ens94] ENSSLE, Jürgen: Modelling and Statistical Multiplexing of VBR MPEG Compressed Video in ATM Networks. **In:** *Proceedings of the 4th Open Workshop on High Speed Networks*, 1994, S. 59–67

[Ham97] HAMANN, Claude-Joachim: On the Quantitative Specification of Jitter Constrained Periodic Streams. **In:** *Proceedings of MASCOTS*, 1997

[HBB 97] HAMANN, Claude-Joachim ; BAUMGARTL, Robert ; BORRISS, Martin ; HÄRTIG, Hermann ; REUTHER, Lars: Dresden Real Time Operating System - ein Überblick. **In:** *Wissenschaftliche Beiträge zur Informatik, TU Dresden* (1997), Nr. 1, S. 5–14

[HBB 98] HÄRTIG, Hermann ; BAUMGARTL, Robert ; BORRISS, Martin ; HAMANN, Claude ; HOHMUTH, Michael ; MEHNERT, Frank ; REUTHER, Lars ; SCHÖNBERG, Sebastian ; WOLTER, Jean. DROPS - OS Support for Distributed Multimedia Applications. 1998

[Hoh96] HOHMUTH, Michael: *Linux-Emulation auf einem Mikrokern*, TU Dresden, Diplomarbeit, 1996

[ITU93] ITU-T, TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU. H.261 , Video Codec for audiovisual Services at p * 64 kbits. 1993

[KHS] KENCHAMMANA-HOSEKOTE, Deepak R. ; SRIVASTAVA, Jaideep: I/O Scheduling for Digital Continuous Media - Part II: VCR-like Operations / Department of Computer Science, University of Minnesota. – Technical Report

[Kli97] KLIX, Thomas: *Multimedia-Dateisystem auf L4*, Technische Universität Dresden, Diplomarbeit, 1997

[Lie95] LIEDTKE, Jochen: On μ-Kernel Construction. **In:** *Proceedings of the 15th ACM Symposium on Operating System Principles (SOSP)*, 1995

[Lie96] LIEDTKE, Jochen. L4 Reference Manual for 486, Pentium and Pentium Pro. zu finden auf http//os.inf.tu-dresden.de/L4/l4refx86.ps.gz: IBM Watson Technical Report. 1996

[Meh98] MEHNERT, Frank: *Ein zusagenfähiges SCSI-Subsystem für DROPS*, Technische Universität Dresden, Diplomarbeit, 1998

[MHN95] MAKAROFF, Dwight J. ; HUTCHINSON, Norman C. ; NEUFELD, Gerald W. The UBC Distributed Continuous Media File System: Internal Design of Server. 1995

[MPFL97] MITCHELL, Joan L. (Hrsg.) ; PENNEBAKER, William B. (Hrsg.) ; FOGG, Chad E. (Hrsg.) ; LEGALL, Didier J. (Hrsg.): *MPEG Video Compression Standard*. Chapman & Hall, New York, NY, 1997

[NMH] NEUFELD, Gerald ; MAKAROFF, Dwight ; HUTCHINSON, Norm. The Design of a Variable Bit Rate Continuous Media Server

[Reu98] REUTHER, Lars: *Entwicklung eines echtzeitfähigen Dateisystems*, Technische Universität Dresden, Diplomarbeit, 1998

[RW94] RUEMMLER, Chris ; WILKES, John: An Introduction to Disk Drive Modeling. **In:** *IEEE Computer* (1994), March

[Sco95] SCOURIAS, John. Overview of the Global System for Mobile Communications. zu finden auf http://ccnga.uwaterloo.ca/˜jscouria/GSM/index.html. 1995

[Tur93] TURLETTI, Thierry: H.261 Software Codec for Videoconferencing over the Internet / Institut National de Recherche en Informatique et en Automatique. 1993 (1834). – Technical Report

[WGP94] WORTHINGTON, Bruce L. ; GANGER, Gregory R. ; PATT, Yale N.: Scheduling Algorithms for Modern Disk Drives. **In:** *Procedings of the ACM Sigmetrics Conference*, 1994

[WGPW95] WORTHINGTON, Bruce L. ; GANGER, Gregory R. ; PATT, Yale N. ; WILKES, John: On-Line Extraction of SCSI Disk Drive Parameters. **In:** *Procedings of the ACM Sigmetrics Conference*, 1995