REIHE INFORMATIK
13/94

Architektur und Implementierung eines Anwendungsprotokolls für digitale Filme

R. Keller
Universität Mannheim
L15,16
D-68131 Mannheim

Architektur und Implementierung eines Anwendungsprotokolls für digitale Filme

Ralf Keller

Praktische Informatik IV
Universität Mannheim
D-68131 Mannheim
keller@pi4.informatik.uni-mannheim.de

Zusammenfassung. Das Movie Transmission Protocol ist ein portables Anwendungsprotokoll zur Übertragung von kontinuierlichen Medien. Es ist vollständig implementiert und setzt auf den Internet-Transportprotokollen auf. Die Funktionalität der existierenden Transportschnittstellen muß MTP allerdings um vorausschauende Fehlerkorrektur und ratenbasierte Flußkontrolle erweitern. In dieser Arbeit werden nach einer Einführung in die Architektur des XMovie-Systems zunächst die funktionalen Einheiten von MTP vorgestellt. Danach wird der Dienst von MTP beschrieben und die Spezifikation des Protokolls skizziert sowie auf einige Implementierungsdetails eingegangen.

1 Einleitung

An der Universität Mannheim wird im Rahmen des XMovie-Projekts die digitale Filmübertragung als eine innovative Anwendung in Rechnernetzen untersucht [13]. XMovie definiert eine Architektur und Protokolle für ein verteiltes Multimedia-System, bestehend aus vernetzten heterogenen UNIX-Workstations (siehe Abb. 1).

Die digitalen kontinuierliche Medien (*continuous media*, CM) Film und Audio werden über ein digitales Hochgeschwindigkeitsnetz übertragen; die Filme werden in Fenstern des X-Window-Systems [22] angezeigt, und vorhandene Audioströme werden mit Hilfe des AudioFile-Systems abgespielt [17]. Das XMovie-System unterscheidet sich von anderen ähnlichen Systemen (z.B. Pandora [9]) dadurch, daß es keine spezielle Hardware zur Verarbeitung und Anzeige von Filmen erfordert. Die Verwendung von spezieller Hardware kann zwar die Leistung eines Systems kurzfristig steigern, schränkt aber die Flexibilität und Portabilität innovativer Multimedia-Systeme zu sehr ein.

Im Vergleich mit anderen Softwarelösungen für digitale Filme (z.B. der Berkeley MPEG-Player [20] oder QuickTime [1]) zeichnet sich XMovie durch die Netzwerkfähigkeit aus. Die Filme können anderswo im Netz gespeichert werden. Bis heute ist es nicht möglich, digitale Filme über Netzwerke online zu übertragen; selbst im *World Wide Web* (WWW, W3 [2]), das als Hypermedia-System auch Filme unterstützt, werden kontinuierliche Medien per FTP übertragen, zwischengespeichert und dann lokal abgespielt.

Ein wichtiger Teil der XMovie-Architektur ist deshalb auch das *Movie Transmission Protocol* (MTP), mit dem CM-Ströme isochron über Hochgeschwindigkeitsnetze übertragen werden können. Um diesen isochronen Übertragungsdienst in einer heterogenen Umgebung anbieten zu können, muß MTP die Funktionalität der existierenden Transportschnittstellen um vorausschauende Fehlerkorrektur und ratenbasierte Flußkontrolle erweitern [3, 25] sowie möglichst wenig

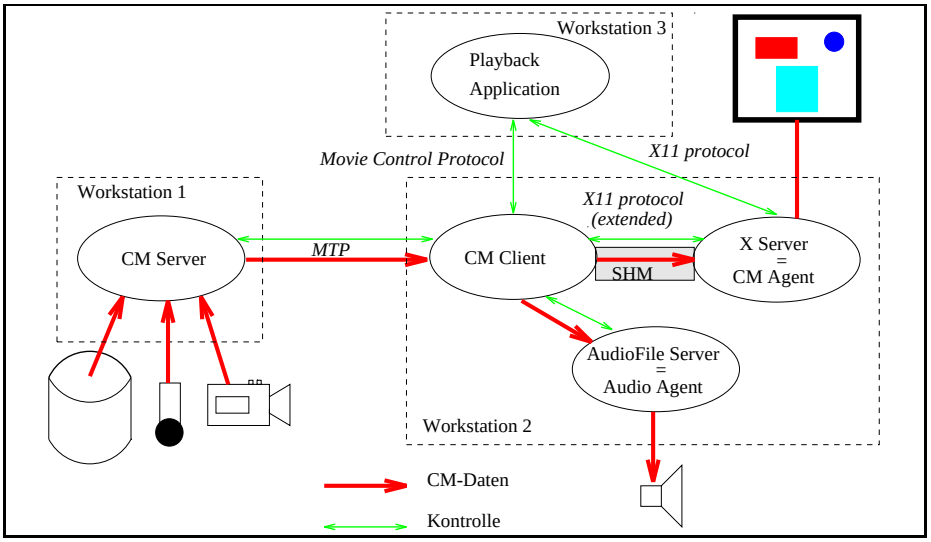

Abb. 1. Architektur von XMovie

Verzögerung und Verzögerungsvarianz dem CM-Strom hinzufügen. Zusätzlich soll MTP portabel und erweiterbar sein, um es leicht an neue Anforderungen und vor allem an neue Übertragungsdienste anpassen zu können.

Die Einordnung von MTP in das OSI-Referenzmodell zeigt Abb. 2. MTP ist ein Anwendungsschichtprotokoll mit integrierter Darstellungsfunktionalität und baut auf den Diensten der Internet-Transportprotokolle TCP und UDP auf [5].

Andere Protokolle zur Übertragung kontinuierlicher Medien wurden bereits definiert, z.B. [4, 7, 23, 29]. Diese sind jedoch alle der Transportschicht zuzuordnen und setzen nicht auf existierende Transportschnittstellen auf. Die Architektur von XMovie basiert dagegen auf gängigen Transportschnittstellen und verlagert die anwendungsspezifischen Funktionen in die MTP-Schicht. Dies verbessert die Portabilität des Systems erheblich, hat allerdings auch den Nachteil, daß wünschenswerte Funktionen eines Multimedia-Transportsystems wie z.B. Multicast nicht zur Verfügung stehen.

Dieser Artikel ist wie folgt strukturiert: In Kapitel 2 wird MTP vorgestellt. Dabei wird ausführlich auf die funktionalen Einheiten von MTP eingegangen. Kapitel 3 beschreibt den Dienst, den MTP anbietet. Die Protokollspezifikation und die Abbildung auf unterliegende Dienste wird in Kapitel 4 skizziert. MTP ist vollständig implementiert und einsatzfähig; Anmerkungen zur Implementierung enthält Kapitel 5. Eine Zusammenfassung und eine Beschreibung des aktuellen Status bilden den Abschluß dieser Arbeit.

2 MTP

MTP kann sowohl für reine Kontrollaufgaben als auch für eine kombinierte Kontrolle und Übertragung von CM-Strömen verwendet werden. In beiden Fällen wird derjenige MTP-Dienstbenutzer, von dem der Verbindungswunsch ausgeht,

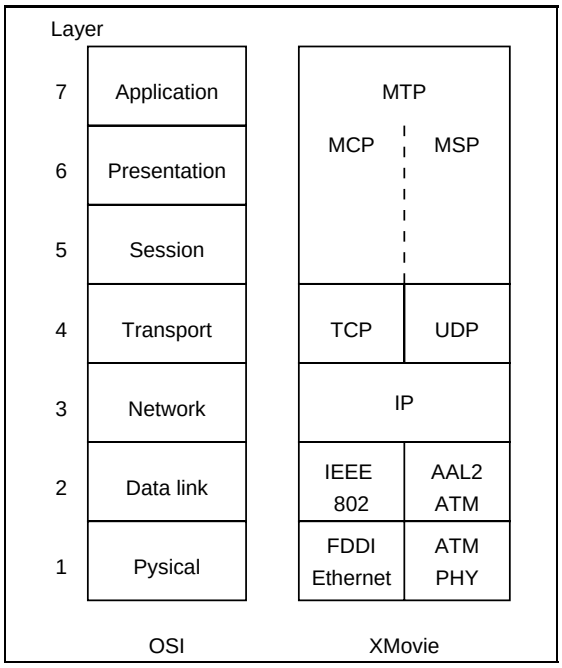

Abb. 2. Einordnung von MTP ins OSI-Referenzmodell

als Initiator und der andere als Responder bezeichnet. In Abb. 1 werden zwischen der *Playback Application* als Initiator und dem *CM-Client* als Responder nur Kontrollinformationen ausgetauscht. Zwischen dem *CM-Client* als Initiator und dem *CM-Server* als Responder werden dagegen sowohl Kontrollinformationen ausgetauscht als auch CM-Daten versendet.

MTP zerfällt in zwei Teile: das Kontrollprotokoll MCP (*Movie Control Protocol*) und das Stromprotokoll MSP (*Movie Stream Protocol*). Beide Teile setzen sich wiederum aus einzelnen funktionalen Einheiten zusammen (siehe Abb. 3).

Die Trennung in MCP und MSP ergibt aus zwei Gründen. Erstens stellen die Kontrolle und die Übertragung von kontinuierlichen Medien an die unterliegenden Dienste unterschiedliche Anforderungen. Ein Kontrollprotokoll benötigt einen zuverlässigen asynchronen Übertragungsdienst relativ geringer Bandbreite und stellt nur sehr geringe Anforderungen bezüglich der Verzögerungsvarianz. Demgegenüber fordert ein Stromprotokoll hohe Bandbreiten bei möglichst isochroner Übertragung mit minimaler Verzögerung, wobei Datenverluste oft toleriert werden können; ein fehlender Bildteil für die Dauer von 1/25 Sekunde wird nicht wahrgenommen. Die akzeptable Verlustrate ist jedoch anwendungsabängig und sollte regulierbar sein.

Zweitens ermöglicht die Trennung den Einsatz von MTP in unterschiedlichen Anwendungen. Ereignisgesteuerte Anwendungen, die die Benutzerschnittstelle und andere höhere Programmfunktionen unterstützen, müssen auf plötzliche Ereignisse reagieren. Sie werden normalerweise flexibel strukturiert und nicht auf Leistung hin optimiert. Demgegenüber werden mediumgesteuerte Anwendungen, in denen der größte Teil der Verarbeitung abläuft, auf Leistung hin optimiert. Ein Beispiel für eine solche ereignisgesteuerte Anwendung stellt die *Playback*

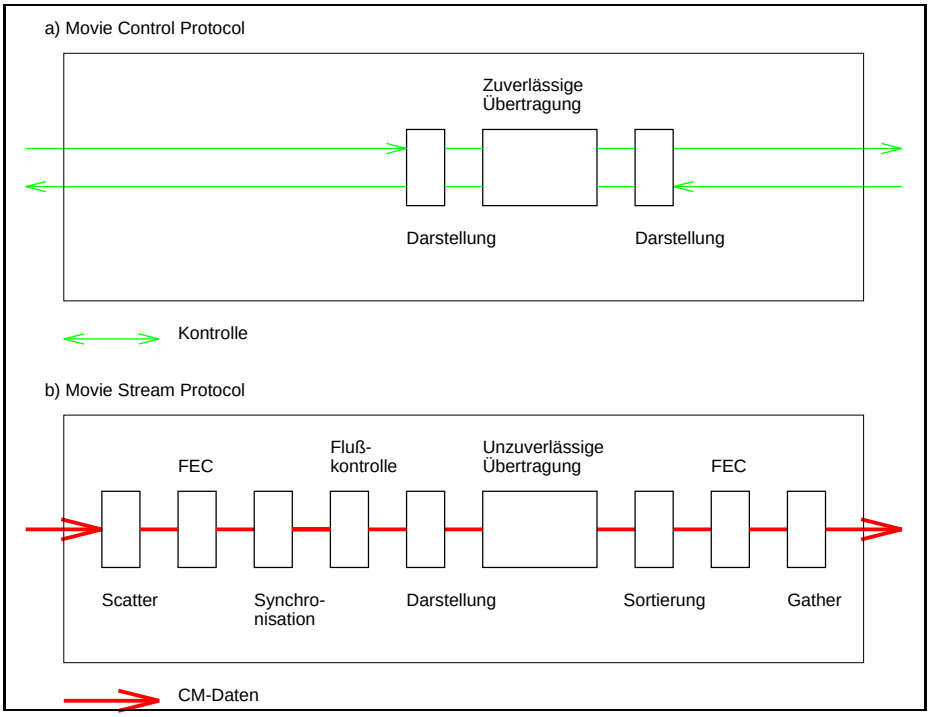

Abb. 3. Funktionale Einheiten von MCP und MSP

Application in Abb. 1 dar, die auch nur auf die MCP-Dienste zugreift. Der *CM-Client* und der *CM-Server* sind Beispiele für mediumgesteuerte Anwendungen, die auch den MSP-Dienst verwenden.

Die Internet-Protokollfamilie stellt für zuverlässige Übertragungen TCP zur Verfügung. Der von TCP bereitgestellte Dienst wird von MCP zur Übertragung seiner Protokolldateneinheiten verwendet. Für eine isochrone Übertragung kann TCP jedoch nicht eingesetzt werden, da seine Fehlerkorrektur- und Flußkontrollalgorithmen zu starken Verzögerungen und Verzögerungsschwankungen führen [14]. Daher verwendet MSP den unzuverlässigen Übertragungsdienst UDP zur Übertragung kontinuierlicher Datenströme.

Innerhalb von MSP muß der Dienst von UDP allerdings erweitert werden, um den an MTP gestellten Anforderungen zu genügen. Dazu gehören eine vorausschauende Fehlerkorrektur (*Forward Error Correction*, FEC), eine ratenbasierte Flußkontrolle und eine Realzeitsynchronisation. Die Funktionsweise und das Zusammenwirken der funktionalen Einheiten von MCP und MSP wird in den folgenden Abschnitten erläutert.

2.1 Die funktionalen Einheiten von MCP

Das Movie Control Protocol (MCP) verwendet zwei funktionale Einheiten: die zuverlässige Übertragung und die Darstellungsumwandlung (siehe Abb. 3a).

Zuverlässige Übertragung. Diese Einheit stellt Dienstprimitive zum Aufbau und Abbau einer zuverlässigen Transportverbindung sowie zum Senden und Empfangen von Kontrolldaten bereit.

Darstellungsumwandlung. Die Darstellungsumwandlung sorgt dafür, daß beim Informationsaustausch zwischen heterogenen Systemen die Darstellungsart der Daten angepaßt wird. Zwischen den beiden MTP-Instanzen werden Daten mit den Typen Zeichenfolge und Integer ausgetauscht. Jedes Zeichen einer Zeichenfolge wird in einem Byte kodiert, und es wird der ASCII-Zeichensatz verwendet. Die Integer-Typen unterscheiden sich untereinander in der Anzahl der Bytes (1,2 oder 4). Auf den Systemen, auf denen XMovie zum Einsatz kommt, existiert als weiterer Unterschied in der Darstellungsart dieser Typen die Byte-Ordnung: Einige Maschinen kodieren Integer in der sogenannten Big-Endian Byteordnung, andere wiederum in der Little-Endian Byteordnung. Die Aufgabe der Darstellungsschicht beschränkt sich somit darauf, bei Bedarf die Byte-Ordnung von Integer-Werten zu verändern. Dies ist die gleiche Problemstellung wie beim X-Protokoll [22] des X-Window-Systems und bei XDR (*External Data Representation* [26]) und wesentlich einfacher als die Problemstellung in der Darstellungsschicht [10] des OSI-Referenzmodells.

Während des Verbindungsaufbaus teilt die MTP-Instanz des Initiators der MTP-Instanz des Responders ihre lokale Byte-Ordnung mit. Die PDUs beim Verbindungsaufbau werden per Voreinstellung in der Big-Endian Byte-Ordnung übertragen. Besitzt der Responder die gleiche Byte-Ordnung wie der Initiator, so wird der Dienst der Darstellungseinheit nicht benötigt. Anderenfalls muß die MTP-Instanz des Responders alle zu sendenden Daten in die Darstellungsart des Initiators und alle empfangenen Daten in ihre eigene Darstellungsart umwandeln. Diese Umwandlung wird bei MCP für alle Integer-Daten durchgeführt.

2.2 Die funktionalen Einheiten von MSP

Um den unzuverlässigen Datenübertragungsdienst von UDP an die Anforderungen von MTP anzupassen, verwendet das Movie Stream Protocol (MSP) eine Reihe von funktionalen Einheiten (siehe Abb. 3b), die im folgenden beschrieben werden.

Unzuverlässige Übertragung. Diese Einheit stellt Dienstprimitive zum Auf- und Abbau einer unzuverlässigen Transportverbindung sowie zum Senden und Empfangen von Benutzerdaten bereit. Bei UDP beschränkt sich der Verbindungsaufbau und -abbau auf das Öffnen bzw. Schließen von Verbindungsendpunkten (*Sockets*). Um den zu übertragenden Teil des kontinuierlichen Mediums nicht unnötig kopieren zu müssen, werden Header- und Nutzdaten in getrennten Puffern an die Senderoutine übergeben bzw. von der Empfangsroutine übernommen (*scatter/gather interface*, siehe auch [27]).

Darstellungsumwandlung. Diese funktionale Einheit ist mit der bei MCP beschriebenen vergleichbar. Bei MSP werden allerdings im Gegensatz zu MCP vom Responder nicht alle Daten, sondern nur die Header-Datenfelder angepaßt; die Daten der kontinuierlichen Medien werden von MSP nur durchgereicht.

Sortierung. An dieser Stelle werden die empfangenen Teile einer Einheit sortiert. Dabei werden für verlorene Teile Verlustmarken gesetzt und Duplikate verworfen.

Ratenbasierte Flußkontrolle. Bei der Übertragung kontinuierlicher Datenströme muß wie bei der Übertragung diskreter Daten darauf geachtet werden, daß der Empfänger nicht überflutet wird. Dabei müssen zwei Ebenen der ratenbasierten Flußkontrolle unterschieden werden: Die obere Ebene regelt die Rate, mit der das jeweilige Medium übertragen wird, auf einen Wert ein, den das Gesamtsystem verarbeiten kann. Kann z.B. die Übertragungskomponente 25 Einheiten pro Sekunde übertragen, die Dekodierkomponente aber nur 20 Einheiten pro Sekunde verarbeiten, so wird das Gesamtsystem auf 20 Einheiten pro Sekunde eingeregelt. Diese Regelung findet außerhalb von MSP statt und wird in [12] beschrieben. Innerhalb von MSP hat die Realzeitsynchronisation die Aufgabe, für die Einhaltung dieser Rate zu sorgen.

Die untere Ebene der ratenbasierten Flußkontrolle regelt die Übertragung von einzelnen Einheiten von der MSP-Instanz des Responders zu der des Initiators. Eine zu hohe Rate, d.h. ein zu schnelles Senden, würde zu Verlusten von Teilen einer Einheit führen. Diese Verluste können zwar oft durch die weiter unten beschriebene vorausschauende Fehlerkorrektur kompensiert werden, allerdings wird durch diesen zusätzlichen Verarbeitungsaufwand die Verzögerung und die Verzögerungsvarianz erhöht. Daher sollten möglichst wenig Verluste auftreten.

Um eine Angleichung der Sende- und Empfangsgeschwindigkeiten durchführen zu können, teilt die MSP-Instanz des Initiators der MSP-Instanz des Responders ihre Leistungsfähigkeit in Form eines Leistungsindexes mit. Die MSP-Instanz des Responders vergleicht diesen Leistungsindex mit seinem eigenen und stellt damit die Ratenkontrolle ein.

Diese Ratenkontrolle besteht im Einfügen von gewissen Wartezeiten, den sogenannten *interpacket gaps*, zwischen dem Senden der einzelnen Teile einer Einheit. Ist der Responder schneller als der Initiator, so wird nicht gewartet.

Die Wartezeiten können sehr klein sein. Werte unter einer Millisekunde sind oft ausreichend. Die Systemuhren auf unseren Workstations arbeiten allerdings oft mit Granularitäten über diesen Bereich, und Systemaufrufe zum Warten sind oft erst im Bereich über 10 Millisekunden exakt. Das von uns realisierte Verfahren beruht auf dem Prinzip des aktiven Wartens. Dazu ermitteln wir die Puffergröße für einen Systemaufruf, der möglichst exakt 100 Mikrosekunden dauern soll. Dadurch lassen sich durch unsere Wartefunktion auch sehr kurze Wartezeiten (alle Vielfachen von 100 Mikrosekunden) realisieren.

Aus dieser Puffergröße wird quasi als Nebenprodukt der Leistungsindex abgeleitet.[1] In Tabelle 1 sind als Beispiel die gemessenen minimalen Granularitäten von verschiedenen Systemuhren und die ermittelten Leistungsindizes von verschiedenen Rechnern angegeben.

Die Sun SPARCstation und die IBM RS/6000 enthalten Hardware-Register und spezielle Betriebssystemfunktionen, um eine Auflösung der Systemuhr im Mikrosekundenbereich zu erreichen. Für verschiedene andere Betriebssysteme, wie z.B. Ultrix und OSF/1 von DEC, sind solche Betriebssystemerweiterungen bereits verfügbar [18]. Allerdings sind auf allen unseren Maschinen die Wartefunktionen des Betriebssystems erst ab Wartezeiten von mehreren Millisekunden exakt.

Realzeitsynchronisation. Diese funktionale Einheit hat die Aufgabe, das Senden von Einheiten u bei der MSP-Instanz des Responders mit der von der oberen

[1] Der Leistungsindex ergibt sich als Puffergröße, ganzzahlig geteilt durch die Dauer des Systemaufrufs in Mikrosekunden (100).

Tabelle 1. Granularität der Systemuhren und Leistungsindizes

Workstation	Betriebssystem	Granularität der Systemuhr [μs]	Leistungsindex
DECstation 5000/133	Ultrix V4.3	3906	3
Sun SPARCstation 10/41 MP	Solaris 2.3	3	100
IBM RS/6000 340	AIX V3.2	22	116
DEC AXP 3000/800S	OSF/1 V1.3	976	428

Ebene der Flußkontrolle eingestellten Rate zu synchronisieren. Bei einer gegebenen Rate R ist jeder Einheit eine Gesamtzeit $t/u = 1/R$ zugeordnet,[2] in der die Einheit in der Zeit s/u übertragen werden muß. Diese Zeit s/u soll so klein wie möglich gehalten werden, damit die Restzeit p/u für die weitere Verarbeitung des CM-Stroms beim Initiator (z.B. zum Dekomprimieren, Filtern, und Darstellen) möglichst groß ist (siehe Abb. 4).

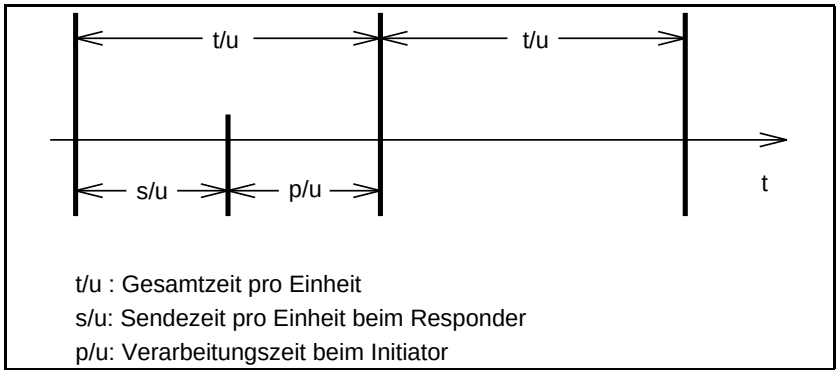

Abb. 4. Synchronisation mit der Realzeit

Wird MCP zu früh mit der Übertragung einer Einheit beauftragt, so wird bis zum nächsten Intervall t/u gewartet. Wird MCP sehr oft zu spät mit der Übertragung beauftragt, so ist dies ein Zeichen dafür, daß die von der oberen Ebene der Flußkontrolle geforderte Rate nicht eingehalten werden kann. Es ist allerdings die Aufgabe der oberen Ebene der Flußkontrolle, geeignete Maßnahmen wie die Reduzierung der Rate anzuordnen. MSP überwacht nur diesen Vorgang und kann auf Anforderung eine Rate empfehlen.

Vorausschauende Fehlerkorrektur. MSP unterteilt zu übertragende Einheiten (*Units*) in kleinere Fragmente, da die unterliegende Dienstschnittstelle nur Benutzerdatenfelder beschränkter Größe annimmt.[3] Die Fragmente werden vor der Übertragung noch um Header-Datenfelder ergänzt.

[2] Die Zeiteinheit t/u wird auch als Periode bezeichnet.

[3] Diese Grenze ist implementierungsabhängig und auf unseren Maschinen sehr unterschiedlich.

Da es bei der Übertragung zu Paketverlusten kommen kann und nicht alle Verluste tolerierbar sind, kann von der MSP-Instanz des Responders in den Datenstrom Redundanz in Form von zusätzlichen Fragmenten eingefügt werden, die aus den gegebenen Fragmenten durch geeignete Operationen berechnet werden. Diese Redundanz wird von der MTP-Instanz des Initiators dazu verwendet, um verlorene Fragmente einer Einheit zu rekonstruieren.

Die mathematischen Grundlagen dieses Verfahren zur vorausschauenden Fehlerkorrektur, genannt AdFEC (*Adaptable Forward Error Correction*), werden in [14] und [15] erläutert. Da AdFEC paketorientiert arbeitet, unterscheidet es sich wesentlich von Verfahren zur Korrektur von Bit- und Bytefehlern (siehe z.B. die Arbeiten von Biersack [3]). Die Bitfehlerwahrscheinlichkeit ist in glasfaserbasierten Netzen wie FDDI sehr gering, und aufgetretene Bitfehler werden schon von den unterliegenden Schicht erkannt; fehlerhafte Pakete werden dabei verworfen. Verluste treten vor allem durch Pufferüberläufe in Zwischen- und Endsystemen auf.

Die Menge der eingefügten Redundanz kann an die Erfordernisse des zu übertragenden Datenstroms angepaßt werden. In der jetzigen Implementierung von AdFEC können zu n gegebenen Teilen, $n \in \{1, 2, 3\}$, jeweils m redundante Teile, $m \in \{1, 2\}$, generiert werden, wobei $n \geq m$. So ist es z.B. sinnvoll, in einem aus JPEG-Einzelbildern [21] zusammengesetztem Film (Motion-JPEG) regelmäßig ein Bild mit fehlersichernder Redundanz zu übertragen, um eine gewisse Qualität zu garantieren. In MPEG-kodierten Filmen [16] kann es eine gute Strategie sein, I-Frames mit einer sehr großen, P-Frames mit einer mittleren und B-Frames ohne fehlersichernde Redundanz zu übertragen.

Scatter/Gather. Die Scatter-Einheit teilt eine CM-Einheit in Fragmente fester Größe auf. Diese Größe ist abhängig von der Zielmaschine und ist ebenfalls Gegenstand der Verhandlung beim Verbindungsaufbau zwischen den MSP-Instanzen des Initiators und des Responders. Die Gather-Einheit fügt empfangene und rekonstruierte Teile wieder zu einer Einheit zusammen. Sollte die Einheit nicht vollständig sein, so wird dies dem Initiator mitgeteilt; dieser kann dann darüber entscheiden, ob die unvollständige Einheit für ihn von Nutzen ist oder nicht.

3 Dienstdefinition

Zur Übertragung und Kontrolle von kontinuierlichen Medien in einer heterogenen Umgebung stellt MTP einen wohldefinierten Dienst und ein wohldefiniertes Protokoll bereit. Dabei ist auch exakt spezifiziert, welche Parameter die Dienstgüte (*Quality of Service*, QoS) regeln.

3.1 Hierarchie der Dienstprimitive

Die Dienstprimitive von MTP lassen sich verschiedenen Abstraktionsebenen zuordnen und bilden somit eine Hierarchie (siehe Abb. 5). Die oberste Ebene (*MTP connection regime*) enthält Dienstprimitive für einen normalen Verbindungsauf- und -abbau sowie für einen abrupten Verbindungsabbau. In der nächsten Ebene (*MTP open regime*) lassen sich Film öffnen und schließen. Die unterste Ebene (*MTP control regime*) beinhaltet Dienstprimitive zur Kontrolle des Filmablaufs und zur Übertragung eines CM-Stromes. Innerhalb der zwei unteren Ebenen

kann auch noch auf ein Dienstprimitiv zur Steuerung von Abspielparametern zugegriffen werden.

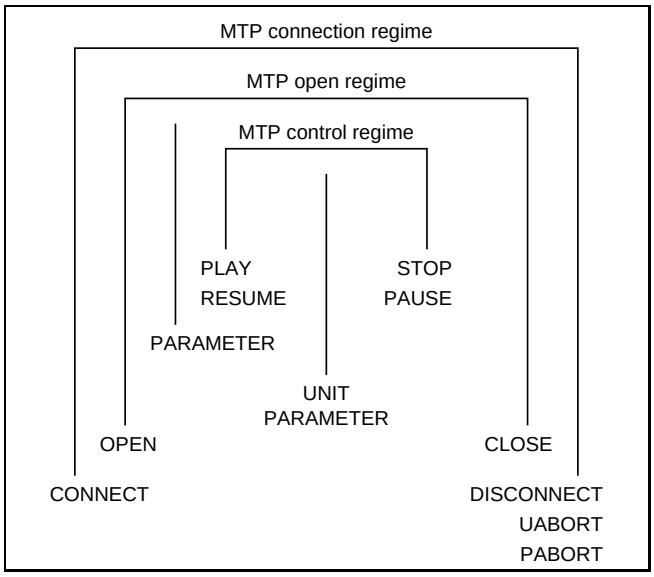

Abb. 5. Hierarchie der MTP-Dienstprimitive

Die einzelnen Dienstprimitive von MTP werden im folgenden Abschnitt kurz beschrieben. Dabei wird auch immer angegeben, wer den Dienst aufrufen kann.

3.2 Dienstprimitive

MTP connection regime. Vier Dienste sind mit dieser Ebene verbunden. Eine Verbindung zwischen Initiator und Responder wird mit einem CONNECT aufgebaut. Beim Verbindungsaufbau legt der Initiator fest, ob nur die Dienste von MCP oder auch die Dienste von MSP benötigt werden. Außerdem teilt die MTP-Instanz des Initiators der MTP-Instanz des Responders ihre eigene Byte-Ordnung mit sowie, falls die Strom-Dienste benötigt werden, auch die maximale PDU-Größe und ihren Leistungsindex. Einen ordentlichen Verbindungsabbau erreicht der Initiator durch ein DISCONNECT. Einen abrupten Verbindungsabbau können sowohl einer der beiden Dienstnutzer (UABORT) als auch der Diensterbringer (PABORT) durchführen.

MTP open regime. Mit dem OPEN-Dienst versucht der Initiator, eine von ihm spezifizierte CM-Quelle zu öffnen. Dadurch wird eine MSP-Verbindung zwischen den MTP-Instanzen des Responders und des Initiators geöffnet, falls nicht nur der MCP-Dienst beim Verbindungsaufbau angefordert wurde. Mit dem CLOSE-Dienst wird diese Quelle wieder geschlossen.

MTP control regime. Um den Ablauf des CM-Stromes zu steuern, stellt MTP vier Dienste und zur Übertragung des CM-Stromes einen Dienst zur Verfügung.

Mit dem PLAY-Dienst startet der Initiator den Strom. Wurde der MSP-Dienst beim Verbindungsaufbau aktiviert, so wird der CM-Strom, aufgeteilt in Einheiten, mit dem UNIT-Dienst übertragen. Der Initiator kann den Strom zwischenzeitlich anhalten (PAUSE) und anschließend wieder anlaufen lassen (RESUME). Beide Dienstnutzer können den STOP-Dienst benutzen; der Initiator zu jeder Zeit, solange der Film läuft, und der Responder, um dem Initiator das Ende des CM-Stromes (z.B. das Dateiende bei einem gespeichertem Film) mitzuteilen.

Ablaufparameter. Die Parameter, die den Ablauf eines CM-Stromes regeln, kann der Initiator mit dem PARAMETER-Dienst sowohl vor als auch während des Ablaufs des CM-Stromes mit dem Responder aushandeln. Dabei kann der Initiator angeben, mit welcher Rate (*period*) und welcher Geschwindigkeit (*mode*) der Strom ablaufen soll. Er kann sowohl die Richtung (*direction*) als auch den Ausschnitt (*section*) und die Qualität (*quality*) des CM-Stromes auswählen. Er kann die Zuverlässigkeit (*reliability*) der Übertragung bestimmen und auswählen, wieviele Einzelbilder (*frames per unit*) bzw. Audiosamples (*samples per unit*) des CM-Stromes zu einer Einheit zusammengefaßt werden. Alle diese Ablaufparameter sind mit Standardwerten vorbelegt; so gilt z.B., daß normalerweise pro Einheit nur ein Bild übertragen und die niedrigste Zuverlässigkeitsstufe (keine Redundanz) verwendet wird.

Zugriffskontrolle. Offensichtlich muß in einem verteilten System der Zugriff auf Ressourcen wie CM-Quellen geregelt sein. Die Details der Zugriffskontrolle und der Authentifizierung sind jedoch nicht Gegenstand dieser Arbeit.

3.3 Dienstgüte

Mehrere Parameter in MTP regeln die Dienstgüte, die den Benutzern geboten wird. Dabei existiert keine 1:1-Beziehung zwischen diesen Parametern und den von Henkel und Stüttgen in [8] erweiterten QoS-Parametern Durchsatz, Zuverlässigkeit, *Burstiness*[4], Übertragungsverzögerung, Startverzögerung und Verzögerungsvarianz. So wird sowohl durch eine höhere Rate, gemessen als Periode (*period*), als auch durch bessere Qualität (*quality*) des CM-Stromes der Durchsatz erhöht. Beispiele für typische Werte sind 1/25 Sekunde pro Bild für die Periode und der Q-Faktor in JPEG [21] zur Einstellung der Bildqualität. Desweiteren wird durch ein Variieren der Werte für *frames per unit* bei Video und *samples per unit* bei Audio die Burstiness des CM-Stromes verändert. In Tabelle 2 wird neben diesen Beziehungen zwischen den erweiterten QoS-Parametern und den QoS-Parameter in MTP auch angegeben, mit welchem Dienst die einzelnen Parameter ausgehandelt werden.

Da MTP einen isochronen Übertragungsdienst für CM-Ströme bereitstellt, existiert kein QoS-Parameter für die Verzögerungsvarianz. Die durch die Übertragung und die weitere Verarbeitung (z.B. Dekompression) entstehende Verzögerungsvarianz wird vom Dienstnutzer aus dem CM-Strom entfernt. Dies ist ebenfalls Aufgabe der oberen Ebene der Flußkontrolle und wird in [12] beschrieben.

[4] Unter Burstiness verstehen wir das Verhältnis zwischen der maximalen und der durchschnittlichen Datenrate.

Tabelle 2. Dienstgüteparameter in MTP

Erweiterte QoS-Parameter	QoS-Parameter in MTP	Ausgehandelt mit Dienstprimitiv
Durchsatz	period	PARAMETER
	quality	PARAMETER
Zuverlässigkeit	reliability	PARAMETER
Burstiness	frames per unit	PARAMETER
	samples per unit	PARAMETER
Übertragungsverzögerung	delay	OPEN
Startverzögerung	start offset	OPEN
Verzögerungsvarianz	—	—

4 Protokollspezifikation

Da MTP seinen Dienst in einer verteilten heterogenen Umgebung anbietet, muß das Protokoll präzise definiert sein. In diesem Abschnitt beschreiben wir die MTP-Protokolldateneinheiten (*protocol data units*, PDUs), ihre Abbildung auf unterliegende Dienste und das Zustandsübergangsdiagramm für die korrekte Reihenfolge der PDUs.

4.1 Protokolldateneinheiten von MTP

Die PDUs von MTP sind als C-Datenstrukturen beschrieben. Eine komplette Auflistung aller PDUs kann aufgrund der beschränkten Seitenanzahl dieser Arbeit nicht gegeben werden. Die PDUs von MCP werden auf den zuverlässigen Übertragungsdienst von TCP, die MSP-PDUs auf den unzuverlässigen Übertragungsdienst von UDP abgebildet. Die Zuordnung von PDUs zu den Dienstprimitiven und die Abbildung der einzelnen PDUs auf die Dienste der Transportschicht zeigt Tabelle 3.

Eine Besonderheit stellen die beiden PDUs PingRequest (PNGRQ) und PingResponse (PNGRP) dar. Mit diesen wird beim Öffnen eines CM-Stromes durch einen mehrmaligen Austausch die Übertragungsverzögerung zwischen den MTP-Instanzen des Initiators und des Responders bestimmt, falls der MSP-Dienst aktiviert ist. Die gemessene Verzögerung wird mit der vom Initiator geforderten Verzögerung verglichen und das Maximum der beiden Werte wird dem Responder mitgeteilt. Dieser kann den Wert des Verzögerungsparameter weiter erhöhen, aber nicht herabsetzen. Der neue Wert für die Verzögerung wird dem Initiator übermittelt, der dann entscheidet, ob diese Verzögerung für ihn akzeptabel ist.

4.2 Zustandsübergangsdiagramm

Neben den exakten PDU-Formaten und den Abbildungen auf tiefere Dienste muß eine Protokollspezifikation auch die erlaubten Ereignisfolgen angeben. Wir verwenden ein konventionelles Zustandsübergangsdiagramm, um das dynamische Verhalten des MTP-Protokoll zu beschreiben. Da die Zustandsübergangsdiagramme von Initiator und Responder sehr ähnlich sind, geben wir hier nur das Zustandsübergangsdiagramm des Initiators an (siehe Abb. 6).

Tabelle 3. MTP-Protokolldateneinheiten

Dienstprimitiv	PDU	Übertragungsdienst
CONNECTreq	CONRQ	TCP
CONNECTresp	CONRP	TCP
DISCONNECTreq	DISRQ	TCP
DISCONNECTresp	DISRP	TCP
OPENreq	OPNRQ	TCP
OPENresp	OPNRP	TCP
CLOSEreq	CLSRQ	TCP
PLAYreq	PLYRQ	TCP
STOPreq	STPRQ	TCP
PAUSEreq	PAURQ	TCP
RESUMEreq	RESRQ	TCP
PARAMETERreq	PARRQ	TCP
PARAMETERresp	PARRP	TCP
UABORTreq	UABRQ	TCP
—	PNGRQ	UDP
—	PNGRP	UDP
UNITreq	UNTRQ	UDP

5 Implementierung

Die Implementierung von MTP erfolgte in C++ auf vier unterschiedlichen Systemplattformen gleichzeitig, um schon von Anfang an ein Höchstmaß an Portabilität zu garantieren. Dabei wurden nur Systemaufrufe verwendet, die auf allen Plattformen verfügbar sind. Diese Plattformen sind:

- DECstation unter Ultrix V4.3
- IBM RS/6000 unter AIX V3.2
- Sun SPARCstation unter Solaris 2.3
- DEC AXP unter OSF/1 V1.3

Viele der funktionalen Einheiten von MTP wurden als eigene C++-Klassen realisiert; dadurch wird die Implementierung leichter wartbar und erweiterbar. Um eine Schnittstelle zu ST-II [28] zu definieren, könnte z.B. sehr einfach eine neue Klasse für die CM-Datenübertragung von der existierenden Klasse abgeleitet werden. Die Änderungen in der MTP-Klasse wären dabei minimal.

Bei der Entwicklung von Kommunikationssystemen muß die Anzahl der Kopieroperationen minimiert werden [27]; dies gilt insbesondere bei der Verarbeitung von CM-Strömen, da dabei sehr große Datenmengen sehr schnell bewegt werden müssen. Innerhalb von MTP werden die CM-Daten beim Senden nicht und beim Empfangen nur einmal beim Zusammenfügen der Fragmente zu einer Einheit kopiert. Dazu wurden alle Modulschnittstellen so ausgelegt, daß nur Verweise auf die Daten weitergegeben werden. Da wir nicht wie in [24] eine Erweiterung des Betriebssystems vornehmen können, um ein effizientes Pufferverwaltungssystem einzubetten, müssen die zwischen funktionalen Einheiten ausgetauschten Puffer sehr sorgfältig verwaltet werden.

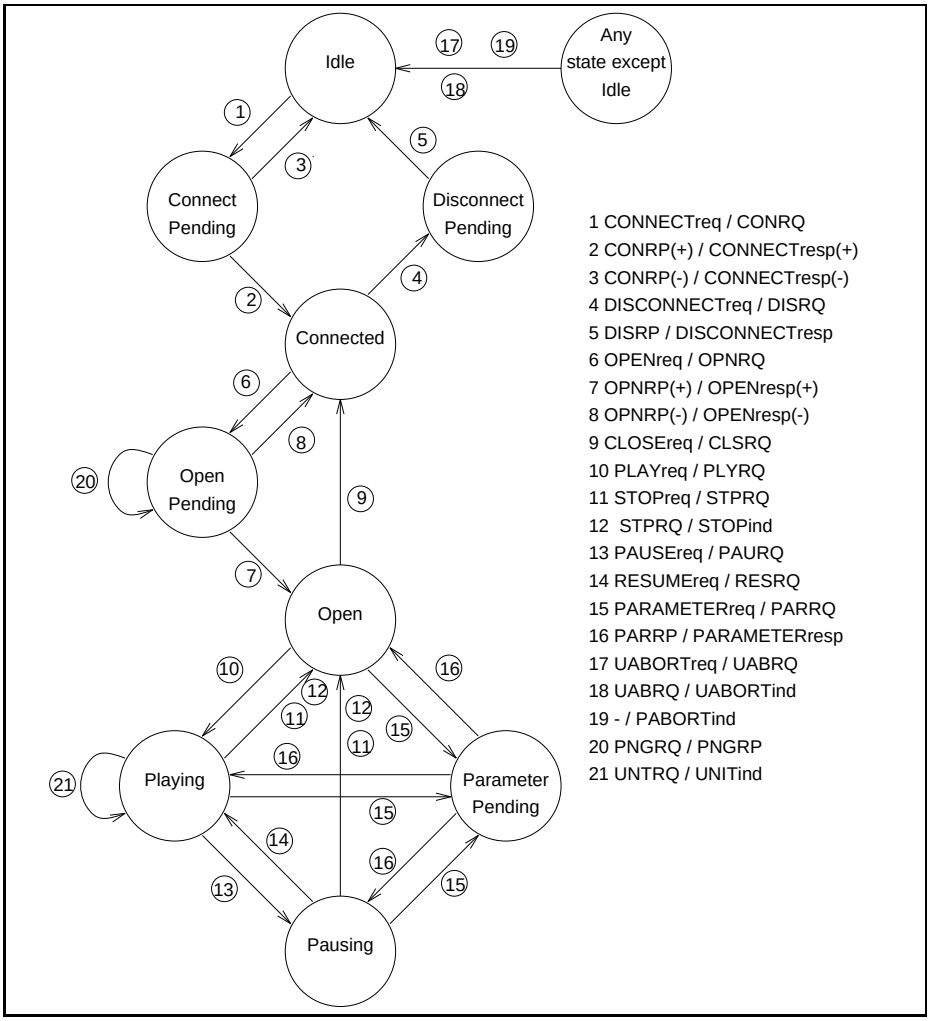

Abb. 6. Zustandsübergangsdiagramm des MTP-Initiators

6 Status und Ausblick

Mit MTP haben wir ein portables Steuerungs- und Übertragungsprotokoll für CM-Ströme vorgestellt. MTP ist vollständig implementiert. Zur Zeit setzen wir MTP zur Übertragung von CM-Strömen über FDDI und Ethernet ein. Da MTP auf den Diensten der Internet-Transportprotokolle aufbaut, wird eine Anpassung an ATM sehr leicht fallen, da die Internet-Protokolle mit als erstes über ATM verfügbar sein werden [19]. Im weiteren Ausbau planen wir außerdem eine Anpassung des MSP-Teils von MTP an AAL2, den *ATM Adaption Layer* für einen isochronen verbindungsorientierten Dienst mit variabler Bitrate [6].

MTP realisiert den CM-Strom-Dienst von MCAM (*Movie Control, Access, and Management* [11]), einem Anwendungsschichtprotokoll zur Steuerung von, zum Zugriff auf und zur Verwaltung von Filmen in einem verteilten System.

MCAM bildet somit ein Anwendungsdienstelement, das den Dienst von MTP verwendet, um seinen eigenen Dienst anbieten zu können.

Der Quellcode von MTP ist frei verfügbar. Momentan passen wir verschiedene Klienten und Server, die im XMovie-Projekt entwickelt worden sind, an das neue MTP an. Eine vollständige Version von MTP wird zusammen mit diesen Klienten und Servern auf unserem FTP-Server bereitgestellt.[5]

In ersten Versuchen mit der Übertragung von verschränkten Audio/Video-Ströme mit MTP wurde der Audio-Anteil der Ströme gegenüber dem Video-Anteil besonders priorisiert. Es wurden Bilder beim Senden und Empfangen verworfen, um selbst bei langsamen Rechnern noch eine akzeptable Audio-Qualität zur ermöglichen. Es muß allerdings noch genauer untersucht werden, wie sich verschränkte Ströme durch AdFEC optimal sichern lassen. Speziell für unterschiedlich kodierte Filme (z.B. MPEG-, Motion-JPEG-, H.261-kodiert) arbeiten wir zur Zeit an der Einstellung der fehlersichernden Redundanz, insbesondere bei wechselnder Netzbelastung.

Literatur

1. Apple Computer, Inc. *Inside Macintosh: QuickTime.* Addison-Wesley Publishing Company, Reading, MA, March 1993.
2. Tim Berners-Lee, Robert Cailliau, Ari Luotonen, Henrik Frystyk Nielsen und Arthur Secret. The World-Wide Web. *Communications of the ACM*, 37(8):76–82, August 1994.
3. Ernst W. Biersack. Performance Evaluation of Forward Error Correction in ATM Networks. *Computer Communication Review*, 22(4):248–257, October 1992.
4. Andrew Campbell, Geoff Coulson, Francisco Garcia und David Hutchinson. A Continuous Media Transport and Orchestration Service. *Computer Communication Review*, 22(4):99–110, October 1992.
5. Douglas E. Comer. *Internetworking with TCP/IP. Vol I: Principles, Protocols, and Architecture.* Prentice-Hall International Editions, Englewood Cliffs, 1991. 2nd ed.
6. Martin de Prycker. *Asynchronous transfer mode: Solution for Broadband ISDN.* Ellis Horwood Limited, 1993. 2nd ed.
7. Luca Delgrossi, Christian Halstrick, Ralf Guido Herrtwich und Heinrich Stüttgen. HeiTP – A Transport Protocol for ST-II. In: *Proceedings of Globecom 92*, Orlando, Florida, 1992.
8. Lutz Henckel und Heiner Stüttgen. Transportdienste in Breitbandnetzen. In: W. Effelsberg, H.W. Meuer und G. Müller, Hrsg., *Kommunikation in Verteilten Systemen, – Grundlagen, Anwendungen, Betrieb,* GI/ITG-Fachtagung, Mannheim, Informatik-Fachberichte 267, Seiten 96–111. Springer-Verlag, Berlin Heidelberg, 1991.
9. Andy Hopper. Pandora – an experimental system for multimedia applications. *ACM Operating Systems Review*, 24(2), April 1990.
10. Information processing systems – Open Systems Interconnection – Connection Oriented Presentation Service Definition and Protocol Specification. International Standard ISO 8822/23, 1988.
11. Ralf Keller und Wolfgang Effelsberg. MCAM: An Application Layer Protocol for Movie Control, Access, and Management. In: *Proceedings of ACM Multimedia 93 (Anaheim, CA, USA, August 1-6, 1993)*, Seiten 21–29. ACM, New York, 1993.
12. Ralf Keller, Wolfgang Effelsberg und Bernd Lamparter. Performance Bottlenecks in Digital Movie Systems. In: Doug Shepherd, Gordon Blair, Geoff Coulson, Nigel

[5] `URL= ftp://pi4.informatik.uni-mannheim.de/pub/XMovie/`

Davies und Frankie Garcia, Hrsg., *Network and Operating System Support for Digital Audio and Video,* 4th International Workshop, NOSSDAV'93, Lancaster, U.K., November 1993, Lecture Notes in Computer Science 846, Seiten 161–172. Springer-Verlag Berlin Heidelberg, 1994.

13. Ralf Keller, Wolfgang Effelsberg und Bernd Lamparter. XMovie: Architecture and Implementation of a Distributed Movie System. Technical Report TR-94-012, Praktische Informatik IV, Universität Mannheim, September 1994. URL= ftp: //pi4.informatik.uni-mannheim.de /pub/techreports/tr-94-012.ps.gz.
14. Bernd Lamparter. *XMovie: Digitale Filmübertragung in Rechnernetzen*. Dissertation, Praktische Informatik IV, Universität Mannheim, April 1994.
15. Bernd Lamparter, Otto Böhrer, Wolfgang Effelsberg und Volker Turau. Adaptable Forward Error Correction for Multimedia Data Streams. Technical Report TR-93-009, Praktische Informatik IV, Universität Mannheim, 1993. URL= ftp: //pi4.informatik.uni-mannheim.de /pub/techreports/tr-93-009.ps.gz.
16. Didier Le Gall. MPEG: A Video Compression Standard for Multimedia Applications. *Communications of the ACM*, 34(4):46–58, April 1991.
17. Thomas M. Levergood, Andrew C. Payne, James Gettys, G. Winfield Treese und Lawrence C. Stewart. AudioFile: A Network-Transparent System for Distributed Audio Applications. In: *Proceedings of USENIX Summer 1993 Technical Conference (Cincinnati, Ohio, USA, June 21–25, 1993)*, Seiten 219–236. USENIX Association, 1993.
18. David L. Mills. Precision Synchronization of Computer Network Clocks. *Computer Communication Review*, 24(2):28–43, April 1994.
19. Peter Newman. ATM Local Area Networks. *IEEE Communications Magazine*, 32(3):86–98, March 1994.
20. Ketan Patel, Brian C. Smith und Lawrence A. Rowe. Performance of a Software MPEG Video Decoder. In: *Proceedings of ACM Multimedia 93 (Anaheim, CA, USA, August 1-6, 1993)*, Seiten 75–82. ACM, New York, 1993.
21. William B. Pennebaker und Joan L. Mitchell. *JPEG still image data compression standard*. Van Nostrand Reinhold, New York, 1993.
22. Robert W. Scheifler und James Gettys. *X Window System: The Complete Reference to Xlib, X Protocol, ICCCM, XLFD*. Digital Press, 1990. 2nd ed.
23. Henning Schulzrinne und Stephen Casner. RTP: A Transport Protocol for Real-Time Applications. Internet Engineering Task Force, INTERNET-DRAFT, 1993. URL= ftp: //ftp.internic.net /internet-drafts/draft-ietf-avt-rtp-04.ps.
24. Ralf Steinmetz. Multimedia Operating Systems: Resource Reservation, Scheduling, File Systems, and Architecture. Technical Report 43.9402, IBM European Networking Center, Heidelberg, 1994.
25. W. Timothy Strayer, Bert J. Dempsey und Alfred C. Weaver. *XTP: The Xpress Transfer Protocol*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1992.
26. Sun Microsystems, Inc. XDR: External Data Representation Standard. Network Working Group, Request for Comments 1014, Sun Microsystems, Inc., 1987.
27. Liba Svobodova. Implementing OSI Systems. *IEEE Journal on Selected Areas in Communications*, 7(7):1115–1130, September 1989.
28. Claudio Topolocic. Experimental Internet Stream Protocol, Version 2 (ST-II). Network Working Group, Request for Comments 1190, University of Southern California, CA, October 1990.
29. Bernd Wolfinger und Mark Moran. A Continuous Media Transport Service and Protocol for Real-Time Communication in High Speed Networks. In: P. Venkat Rangan, Hrsg., *Network and Operating System Support for Digital Audio and Video,* Third International Workshop, La Jolla, California, USA, November 1992, Lecture Notes in Computer Science 712, Seiten 169–182. Springer-Verlag, Berlin Heidelberg, 1993.

Dieser Artikel wurde mit dem LaTeX Makro-Paket und dem LLNCS-Style formatiert.