Proxy Caching von Multimediaströmen

Seminararbeit

vorgelegt am

Lehrstuhl für Praktische Informatik IV
Prof. Dr. W. Effelsberg
Universität Mannheim

im

Januar 2001

von

Christian Hoffmann
aus Mannheim

Betreuer: Dipl. Wirtsch.-Inf. Volker Hilt

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Das Internet wird mehr und mehr zur universellen Infrastruktur für die Verteilung von Informationen verschiedenster Art, kontinuierliche Medien wie Audio und Video eingeschlossen. Dies zeigt sich sowohl in der steigenden Anzahl von Internetseiten, die sich auf Audio und Video stützen, als auch in der wachsenden Verfügbarkeit von kommerziellen Produkten zur Wiedergabe von Medienströmen. Die steigende Nutzung von Multimediadiensten führt nicht nur zu einer noch stärkeren Belastung der Netzinfrastruktur und somit zu immer höheren Kosten für die Internetbetreiber, sie fordert auch vom Serviceanbieter der Mediendienste hohe Investitionen in Server- und Netzwerkhardware. Letztendlich ist die vom Benutzer wahrgenomme Qualität von via Internet übertragenen Video- und Audioströmen in der Regel nicht zufriedenstellend. Hohe Wartezeiten zu Beginn der Wiedergabe, niedrige Bild- und Tonqualität, stockende Übertragungen und zwischenzeitliche Datenverluste sind im gegenwärtigen Internet an der Tagesordnung.

Der Einsatz von *Proxy Caches* verspricht Abhilfe. Proxy Caches sind Netzknoten, die Medienströme für Benutzer vorhalten und die sich näher am Client befinden als die eigentliche Datenquelle. Das Zwischenspeichern von Medienströmen erfordert aber im Gegensatz zu Caches für Text- und Bilddaten, die schon seit längerem durch diverse Produkte realisiert werden (vgl. z.B. [11]), den Einsatz besonderer Technologien, um den besonderen Charakteristiken von Medienströmen (wie z.B. die Größe, Echtzeit-Eigenschaften und die Anforderungen an die Qualität der Wiedergabe) Rechnung zu tragen.

Im Rahmen dieser Seminararbeit soll auf die verschiedenen Technologien, die beim Caching von Medienströmen zum Einsatz kommen, eingegangen werden. Zunächst wird die generelle Funktionsweise eines Proxy Caches vorgestellt (Abschnitt 2) und auf dessen Vorteile in Abschnitt 2.3 hingewiesen. In Abschnitt 3 werden die einzelnen in der aktuellen Literatur vorhandenen Technologien vorgestellt. Insbesondere wird auf Prefix Caching, Ring Buffers und Quality Adaptive Streaming eingegangen. Auf die im Augenblick am Markt vorhandenen, kommerzialisierten Caching Produkte wird in Abschnitt 4 eingegangen. Abschnitt 5 faßt diese Seminarbeit zusammen und gibt einen kurzen Ausblick auf zukünftige Trends des Proxy Cachings für Multimediaströme.

2 Generelle Funktionsweise und Infrastruktur

Proxy Caches sind Netzwerkknoten, die Multimediaströme (teilweise) zwischenspeichern und für den Client vorhalten. Sie befinden sich in der Netzwerkinfrastruktur näher am Client als die eigentliche Datenquelle, dem Server beim Anbieter des Dienstes. Anstelle die Daten vom eigentlichen Server zu beziehen, interagiert ein Client mit einem Proxy Cache der die Daten entweder im Hauptspeicher oder im Festplattenspeicher vorhält und an den Client überträgt. Abbildung 1 zeigt ein Beispiel für eine vereinfachte Netzwerktopologie. In diesem Falle gibt es im benutzten Auschnitt des Netzwerks nur einen isolierten Proxy Cache.

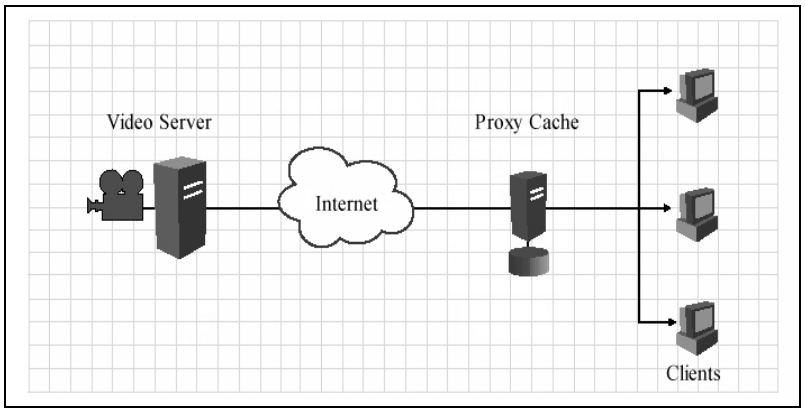

Abbildung 1: Beispiel einer Netzwerktopologie

Hofman et al. schlagen in [3] vor, eine Infrastruktur mit vielen untereinander kommunizierenden Proxy Caches aufzubauen. Hierbei tauschen die Caches untereinander Status-Informationen aus, wer welche Daten zwischengespeichert hat. Diese Date werden periodisch per Multicast an die anderen Proxies verteilt. Bei einer Anfrage eines Clients kann dann ein Proxy auswählen, von welcher Datenquelle der Client bedient werden kann. Dies entscheiden Kostenfunktionen, die sowohl die Auslastung der Netze als auch der einzelnen Proxies berücksichtigen und in die Entscheidung für einen Server mit aufnehmen.

2.1 Zwischenschalten eines Proxy Caches

Um zu verhindern, daß sich ein Client direkt mit dem Server verbindet, sondern zuerst mit einem Proxy in Verbindung tritt gibt es die folgenden Möglichkeiten:

- Clientkonfiguration: Eine Möglichkeit ist es, den Proxy in der Clientsoftware zu konfigurieren. Dazu muß die Adresse des Proxies in ein lokales Konfigurationsfile eingetragen werden. Hoher Adminstrationsaufwand, Fehleranfälligkeit und eine statische Umleitung ohne Berücksichtigung von eventuellen

Load-Balancing-Strategien sind die Nachteile einer client-seitigen Konfiguration.

- Layer 4/7 Switches: Diese Nachteile können durch dynamische Systeme umgehen werden. Es besteht die Möglichkeit sogenannte Layer 4 oder Layer 7 Switches einzusetzten. Layer 4/7 Switches ersetzen entweder die lokal vorhandenen Hubs/Switches oder befinden sich zwischen dem Access Point des Clients und dem ersten Router. Die Switches untersuchen eingehende Pakete und leiten diese bei Bedarf entweder an einen Proxy Cache oder an die urspünglich gewünschte Adresse weiter. Layer 4 Switches analysieren die eingehenden Pakete auf der Ebene des Transportprotokolls (Schicht 4 des OSI-Referenzmodells, s. [2]), die im TCP/IP Modell der TCP/UDP-Schicht entspricht (s. z.B. Leiner et al. in [6]). Dazu extrahiert der Switch Protokoll ID und Portnummer aus den Paketen und leitet diese dann an einen Proxy Cache weiter, falls es sich um Anfragen an Medienstromserver handelt. Diese Layer 4 Switches können ein einfaches Load-Balancing betreiben, z.B. durch Zufallszuteilung, Round-Robin oder durch Ermitteln des Proxies mit den wenigsten TCP/IP-Verbindungen. Layer 7 Switches hingegen arbeiten auf der Anwendungsschicht des OSI Modells (Schicht 7). Hier werden anwendungsspezifische Informationen aus den Paketen gewonnen, wie z.B. die URL in einem HTTP-Request, um zu entscheiden wohin die Pakete weitergeleitet werden sollen. So wird es möglich komplexere Load-Balancing Strategien zu realisieren.

- Domain Name Service: Eine weitere Möglichkeit besteht in der Ausnutzung des Domain Name Services (DNS), der eigentlich nur dazu benutzt wird Hostnamen in IP Adressen aufzulösen. Normalerweise wird ein Hostname immer statisch in eine IP Adresse aufgelöst. Es ist jedoch möglich die Abbildungstabellen von Hostnamen und IP-Adressen eines DNS-Servers dynamisch zu verändern. Dies kann benutzt werden, um den Netzverkehr vom Client zum Server an den Proxy umzuleiten. Da der DNS eigentlich nicht zur Umleitung von Clientanfragen gedacht ist, bringt diese „mißbräuchliche“ Benutzung einige Nachteile und Unwegsamkeiten mit sich. Will ein Client die IP-Addresse eines Servers erfahren, von dem er lediglich den Hostname kennt, so stellt dieser eine Anfrage an den lokalen DNS-Server. Da dieser den Hostnamen in der Regel nicht auflösen kann, muß er die Anfrage an den DNS-Server des Zielnetzes richten. Dieser kann jetzt die IP-Addresse liefern. Um den Zugriff auf den eigentlichen Server zu verhindern, liefert der DNS-Server nun die IP Addresse eines Proxies. Damit dies überhaupt sinnvoll ist, muß dieser DNS Server den für den Client besten Proxy bestimmen. Da dies in der Regel der geographisch nächste Proxy Cache ist, muß der Standort des Clients bekannt sein. Auf Grund des DNS Protokolls ist dem DNS-Server nur die IP-Addresse des lokalen DNS-Servers bekannt,

der die Anfrage des Clients weitergeleitet hat. Das Auffinden des nächsten Proxy Caches des Clients auf Basis der IP-Adresse des lokalen DNS-Servers stellt sich in der Praxis als schwierig und ungenau dar.

2.2 Übertragungsprotokolle und Wiedergabe eines Medienstroms

Damit Proxy Caches überhaupt mit einer großen Akzeptanz in der Praxis rechnen können, sollte ihr Einsatz keine Änderungen an existierenden Multimedia-Applikationen, bzw. Kommunikationsprotokollen erfordern. Obwohl einfache Applikationen für Medienströme auch mit HTTP möglich wären, haben sich in der Praxis die Protokolle RTP und RTSP faktisch zu den Standardprotokollen für die Übertragung von Medienströmen etabliert.

Das Real-Time Protocol (RTP) bietet die Basisfunktionalität für die Übertragung von Echtzeitdaten über das Internet [8]. Dazu definiert das Protokoll Pakete, in denen u.a. ein Zeitstempel, die Identität der Nutzdaten, die Identität der Datenquelle und Sequenznummern vorhanden sind. RTP kann sowohl über UDP als auch über TCP genutzt werden.

RTSP, das Real-Time Streaming Protocol, wird dazu benutzt, die Lieferung von Medienströmen zu kontrollieren [9]. Ähnlich der Funktionalität eines Videorekorders, implementiert RTSP das Starten, Anhalten, Vor- und Zurückspulen von Medienströmen. Typischerweise laufen RTSP Kommunikationen über TCP-Verbindungen, aber auch UDP ist möglich. Im Gegensatz zu HTTP ist RTSP nicht zustandslos, d.h. es werden Informationen über die Sitzungen vorgehalten. RTSP definiert verschiedene Nachrichten zum Aufbau und zur Kontrolle von Sitzungen. Die OPTIONS Nachricht wird benutzt, um Informationen über die Eigenschaften des Server zu bekommen, wie z.B. die von ihm unterstützten Protokollversionen. Die DESCRIBE Nachricht übermittelt die Eigenschaften eines bestimmten Medienstroms, wie z.B. die Wiedergaberate. Informationen über den zu benutztende Transportweg (TCP oder UDP und die entsprechende Portnummer) werden mittels der SETUP Nachricht übertragen. Die PLAY Nachricht startet die eigentliche Übertragung des Stroms, wobei mittels des RANGE Headers ein bestimmtes Intervall innerhalb des Stroms gewählt werden kann.

Der Zustand der einzelnen Verbindungen wird vom Server gesteuert. Befindet sich nun zwischen Server und Client ein Proxy, so muß dieser den Zustand der einzelnen Verbindungen zu den Clients koordinieren.
Im folgenden Beispiel wird der Austausch von RTSP Nachrichten zwischen Client, Proxy und Server für den Fall dargestellt, daß der Client die Daten nur teilweise oder gar nicht vom lokalen Cache bezieht. Zwischen Server und Proxy und Proxy und Client bestehen zwei völlig getrennte Verbindungen mit verschiedener Sitzungsnummer (*session identifier*–SID). Bei Weiterleitung von RTSP Nachrichten des Clients an den Server, muß der Proxy also Teile der Header austauschen. Der

Proxy leitet die OPTIONS, DESCRIBE und SETUP Nachrichten an den Server weiter und gibt die Antworten an den Client zurück, wobei er die CSeq (sequence number), Session und Transport-Headerinformationen entsprechend modifiziert. Geht nun eine PLAY Nachricht beim Proxy ein, der einen Teil der Daten des Stroms gecached, so kann er diese (in Form von RTP Paketen) an den Client zurücksenden. Danach leitet der Proxy die PLAY Nachricht weiter, die nun einen modizifizierten RANGE Header erhält, je nach dem, wie groß der im Zwischenspeicher vorgehaltene Clip war. Dieser Teil des Clips muß ja vom Server nicht mehr geliefert werden. Zum Beenden der Verbindung wird dann die TEARDOWN Nachricht des Clients an den Server mit geänderten Headerdaten durchgereicht.[1] Abbildung 2 zeigt den Verlauf der Kommunikation zwischen Client, Proxy und Server schematisch.

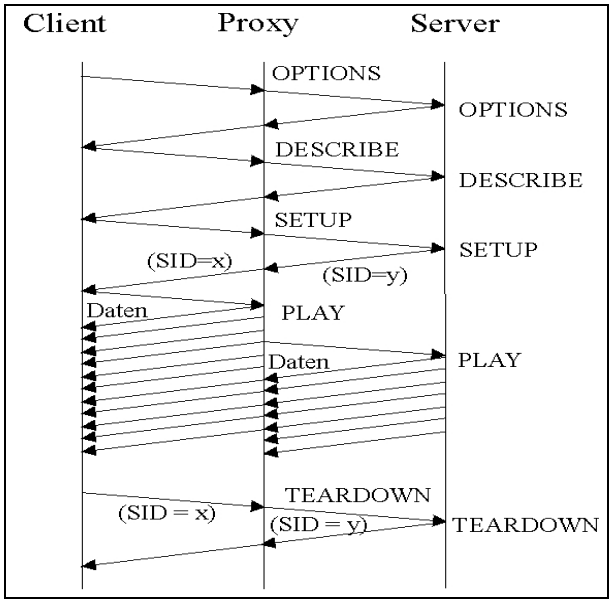

Abbildung 2: RTSP/RTP Kommunikation von Client/Proxy/Server (Quelle [4])

Die RTP Daten des Stroms können sich aus unterschiedlichen Quellen zusammensetzen. Beispielsweise ist es möglich, daß der Beginn eines Videofilms aus dem Zwischenspeicher des Proxys kommt, der Rest aber vom Server bezogen wird. Inbesondere RTP Headerfelder wie Sequenznummer und Zeitstempel sind also bei zwei verschiedenen Quellen nicht im Einklang und müssen vom Proxy modifiziert werden, damit ein sinnvoller RTP Strom zusammengesetzt wird und zum Client hin übertragen werden kann, ohne daß dieser den Übergang von einer Datenquelle auf die andere bemerkt.

[1]Da die RTSP Implementierung von Real Networks proprietäre Header mit verschlüsselten Feldern enthält, muß die OPTIONS, DESCRIBE und SETUP Nachricht immer direkt an den Server weitergeleitet werden, selbst wenn der Clip gänzlich im Cache vorhanden ist.

2.3 Bewertung

Vom Einsatz von Proxy Caches versprechen sich alle am Einsatz von Multimediaströmen beteiligten Parteien erhebliche Vorteile. Interessant für Internet Service Provider (ISP) ist die geringere Belastung der Netzwerkinfrastruktur, da Clients, die gleichzeitig denselben Videoclip sehen die Daten vom Proxy beziehen und somit die Daten zwischen Server und Proxy nur einmal übertragen werden.
Proxy Caches stellen somit eine billige Alternative zum Ausbau von Netzkapazitäten und zur Einführung von Quality of Service Unterstützung dar, da sie in herkömmliche Best-Effort Netzwerke integrierbar sind und keine grundlegenden Veränderungen der Netzwerkinfrastruktur zur Folge haben.
Der Endbenutzer hat den Vorteil der geringeren Startzeit der Wiedergabe (start-up-latency) und verbesserter Wiedergabequalität, durch verminderte Paketverluste, da sich der gewünschte Inhalt nun näher am Client befindet.
Solche Caching Services, die den Inhalt aller potentiellen Video- und Audioinhalte des Internets zwischenspeichern und deren Kosten hauptsächlich von ISPs getragen werden, nennt man „content-consumer oriented services“, da sie hauptsächlich einen Mehrwert für die Konsumenten darstellen.
Aber auch der Anbieter von Multimediainhalten profitiert vom Einsatz von Proxy Caches, da zwischen Proxy und Server weniger Verbindungen entstehen und somit die Serverlast erheblich reduziert wird. Anbieter profitieren auch von der höheren Kundenzufriendheit, die sich aus kürzeren Startupzeiten und besseren Wiedergabequalitäten ergibt. Deshalb tendieren in letzter Zeit Content Provider dazu, solche Caches von speziellen Firmen (Content Delivery Service Providers CDSPs) bereitstellen zu lassen. Diese kehren das normale Caching Geschäftsmodell um, da hierbei der Content Provider die Kosten übernimmt, damit er - möglichst nahe am Enduser - die Kunden in besserer Qualität erreicht. Hierbei werden solche Proxy Caches bei einer Vielzahl von ISPs installiert (vgl Hofmann und Sbnanin [4]).

Wird zur Anbindung eines Proxy Caches ein Layer 4/7 Switch benutzt, so kann neben den Anschaffungskosten eine geringerwertig schlechtere Performance für alle statischen Webinhalte als Nachteil angesehen werden. Die hierbei auftretende, zusätzliche Latenzzeit kann jedoch wohl vernachlässigt werden. Der erhöhte Administrationsaufwand wird jedoch durch die erzielten Performancegewinne für die Wiedergabe von Medienströmen sicher wettgemacht.

Eine quantitative Analyse der Performancegewinne ist nur im praktischen Einsatz zu bestimmen, da diese stark vom tatsächlich beobachteten Benutzerverhalten, von der Konfiguration der Proxy Caches, der eingesetzten Netzwerkinfrastruktur und von den zu übertragenden Medienströmen abhängt.

3 Technologien

Um das Caching von Multimediaströmen effizient zu realisieren, sind Technologien erforderlich, die auf die besonderen Eigenschaften der Medienströme eingehen. Das Segmentieren von Medienströmen ermöglicht das Caching trotz ihrer erheblichen Größe. Diese Technologie wird in Abschnitt 3.1 vorgestellt. Auf eine Verminderung der benötigten Bandbreite bei der Übertragung von Medienströmen geht Abschnitt 3.2 ein. Abschnitt 3.3 beschreibt die Kontrolle der Datentransferrate zum Client und die somit mögliche Reduzierung der initalen Verzögerung. Proxy Caches können durch die Ausnutzung der besonderen Kodierungsverfahren der Ströme die Qualität der zu übertragenden Clips verbessern. Dies beschreibt Abschnitt 3.4.

Diese hier beschrieben Technologien können als orthogonal angesehen und unabhängig von einander behandelt werden. Eine mögliche Implementierung eines Caches muß somit nicht alle Aspekte umsetzen, sondern kann nur einzelne Teile realisieren. Insbesondere eine Umsetzung des Verfahrens zur Qualitätsanpassung kann als optional angesehen werden.

3.1 Segmentierung der Medienströme

Zwei fundamentale Unterschiede zwischen Medienströmen und statischen Daten wie Text und Bildern sind deren Größe und die Anforderungen an das Timing. Diese beiden Unterschiede erschweren das Caching von Audio- und Videoclips. Ein zwei Stunden langer MPEG-I Film benötigt ca. 1,4 GByte Speicher. Das Caching ganzer Filme würde also den limitierten Speicher eines Proxy Caches zu sehr in Anspruch nehmen und nur wenige Filme könnten zwischengespeichert werden. In der Praxis werden deshalb in einem Proxy Cache nur kurze Teile eines Films gespeichert.
Die clientseitige Software zur Wiedergabe von Medienströmen speichert zunächst in einem Puffer einige Sekunden der eingehenden Daten, um Abweichungen in der Übertragung (*jitter*) und einzelne Paketverluste (*paket loss*) ausgleichen zu können (der sogenannte *dejitter buffer*). Insbesondere das Auffüllen des dejitter buffers erzeugt beim Client eine hohe Wartezeit zu Beginn der Wiedergabe, was insbesondere bei sehr kurzen Video- oder Audioclips als sehr störend empfunden wird. Durch Proxy Caches, die die ersten Sekunden eines Medienstroms zwischenspeichern, kann die initiale Verzögerung (*startup latency*) erheblich vermindern. Diese Methode wird *prefix caching* genannt [10].

Hofman et al erweitern in [3] das prefix caching dahingehend, daß Medienströme in Segmente unterteilt werden, die unabhängig voneinander gecached und ersetzt werden können. Wäre z.B. S die Größe eines Festplattenblocks des Proxy Caches, dann könnte die Segmentgröße ein Vielfaches von S sein, um den Festplattenplatz effizient zu verwalten. Das vollig unabhängige Zwischenspeichern und Ersetzen von Segmenten hat jedoch einen großen Nachteil. Kommt eine An-

frage eines Clients nach einem Strom an den Proxy, so ist der Strom i.d.R. nicht vollständig im Cache, sondern nur einzelne Segmente davon. Der Proxy müßte nun die fehlenden Segmente entweder vom Server oder von anderen Proxies beziehen. Da die „Löcher“ im Strom an den unterschiedlichsten Stellen auftreten können, hätte der Proxy einen sehr hohen Administrations- und Synchronisierungsaufwand, die Gefahr eines Abbruchs der Übertragung auf Grund von Synchronisierungsproblemen wäre groß. Deshalb wird der Medienstrom nochmals in logische Abschnitte (*chunks*) unterteilt. Ein Abschnitt ist somit eine fortlaufende Sequenz von Segmenten. Jeder Abschnitt wird unabhängig von anderen chunks gecached und zwar mittels einer *prefix caching* Strategie. Das heißt also:

- Die gecachten Segmente eines Abschnitts bilden immer einen Präfix.
- Die kleinste Einheit des Cachings und der Ersetzung ist ein Segment.
- Wenn auf ein Segment eines Abschnitts zugegriffen wird, kann kein Segment des Abschnitts ersetzt werden.
- Wenn ein Segment eines Chunks vom Ersetzungsalgorithmus gewählt wird, so wird immer das letze gecachted Segment des Chunks entfernt (also vom Ende des Präfixes).

In Abbildung 3 erkennt man die einzelnen Abschnitte und die Segmente eines Medienstroms. Man sieht, daß die dunkler gefärbten Segmente eines chunks einen Präfix bilden.

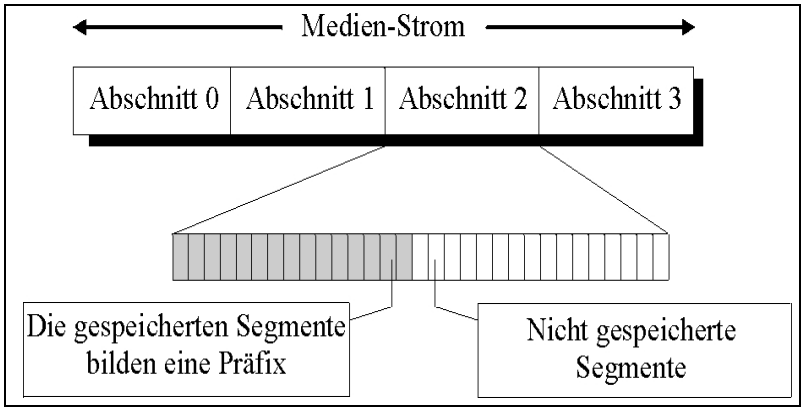

Abbildung 3: Einteilung eines Stroms in Abschnitte und Segmente (Quelle [3])

Es wird somit nicht nur der Beginn des Medienstroms gecached, sondern auch die wichtigen Abschnitte innerhalb häufig angefragter Medienströme, z.B. wenn diese explizite Sprungstellen aufweisen, wie z.B. Kapitel einer Vorlesung, o.ä.

Um Speicherbereiche wieder freizugeben kann für jeden Chunk der Zeitpunkt des letzten Zugriffs mitgespeichert werden. Der am längsten unbenutzte Abschnitt (least recently used - LRU) kann dann vom Ende her gelöscht werden, falls der

Speicher im Proxy zu Ende geht. Man geht hierbei also davon aus, daß Medienclips immer von Beginn an betrachtet werden.

Die Größe des Prefixs hängt sowohl von der Netzwerkperformanz zwischen Server und Proxy, als auch von der gewünschten Wiedergabeverzögerung beim Client ab. Der Prefix sollte groß genug sein, um die Verzögerungen zwischen Server und Proxy und den Verlust einzelner Pakete zu absorbieren, wenn der Rest des Clips vom Server bezogen wird. Um 5 Sekunden eines qualitativ hochwertigen MPEG-2 Clips zu speichern, werden nur circa 2,5 bis 3 MBytes an Pufferspeicher benötigt, was bei heutigen Preisen für Festplatten- und RAM-Speicher als angemessen erscheint. Angenommen die Verzögerung zwischen Server und Proxy liegt zwischen d_{min} und d_{max} gemessen in Bildeinheiten („frame slots“) und die gewünschte Startup Latency des Clients sei s, dann muß der Proxy einen Prefix von mindestens $max\{d_{max} - s, 0\}$ Bildern speichern.
Proxy Caches sollten unabhängig von der Kodierung des Medienstroms sein. Deshalb speichern sie direkt die Informationen der RTP Pakete der Medienströme auf den vorhanden Festplatten. Dazu werden die eigentlichen RTP Daten („payload“) und minimale Headerinformationen gespeichert, wie Payloadtyp, Sequenznummer, Zeitstempel, das Markerbit und RTP Header Erweiterungen. Mittels dieser gespeicherten Informationen können dann neue RTP Ströme bei der Übertragung zum Client erzeugt werden. Der Cache muß den eigentlichen Inhalt der RTP Packete somit nicht verstehen und ist unabhängig von der verwendeten Kodierung der Payload.
Medienströme werden beispielsweise in Files gecached (vgl. z.B. [1]). Diese Files werden wie folgt strukturiert:

- Medienfileheader: Jedes gecachted File beginnt mit einem Header, der die medienspezifischen Informationen beinhaltet. Diese Informationen werden in der Regel vom Medienserver durch einen RTSP DESCRIBE Request bezogen. Weiterhin enthält der Header eine Tabelle, die Offsets auf die einzelnen vorhandenen Segmente speichert.

- Nicht-überlappende Zeitsegmente: Nach dem Fileheader werden die Segmente gespeichert, die wiederum einen kurzen Header besitzen, der Informationen über das Segment wie Start- und Endezeitstempel, Sequenznummer und Anzahl der RTP Packete speichert. Die eigentlichen Datenpackete folgen dann in zeitlich geordenter Reihenfolge. Der Segmentheader verfügt auch über eine Indextabelle, die zum schnelleren Zugriff auf die einzelnen Blöcke die entsprechenden Zeiger speichert.

Diese Files realisieren logisch betrachtet also einen Baum, dessen Wurzel der Fileheader mit der Segmenttabelle darstellt und dessen Kinder die einzelnen Zeitsegmente sind.
Um schnell auf diese Files zugreifen zu können, hält der Proxy eine globale Indextabelle, die die Medienobjekte mit den Files mittels einer Hashtabelle korreliert.

3.2 Zusammenführung mehrerer Anfragen

Anfragen von Clients nach Medienströmen können sich in drei Dimensionen unterscheiden:

1. Medium: Die Clients fragen verschiedene Ströme an
2. Anfragezeitpunkt: Die Anfragen kommen zu unterschiedlichen Zeitpunkten an
3. Abspielbereich: Anfragen beziehen sich auf unterschiedliche Intervalle eines Medienstroms

Der Unterschied zweier Anfragen an ein Medium in Bezug auf Anfragezeitpunkt und Abspielbereich wird durch die „temporäre Distanz“ gemessen. Sie ist die Zeitdifferenz zwischen zwei Anfragen nach dem gleichen Datenpaket. Will Anfrage r_1 zum Zeitpunkt t_1 einen Clip vom Zeitpunkt p_1 bis zum Ende spielen und Anfrage r_2 zum Zeitpunkt t_2 einen Clip von p_2 bis zum Ende spielen, so ist die temporäre Distanz der Anfragen: $\Delta_{(r_1,r_2)} = (t_2-p_2)-(t_1-p_1) = (t_2-t_1)-(p_2-p_1)$ Ziel eines Proxy Caches ist es, die Anzahl von Anfragen an einen Server gering zu halten, um Netzwerk und Server möglichst wenig zu belasten. Hierzu bedient ein Proxy Cache so viel Anfragen wie möglich aus seinen eigenen Speicher- und Diskressourcen.

Verschiedene Anfragen, deren temporäre Distanz nicht allzu groß ist werden aus einem sogenannten Ringbuffer bedient. Erhält ein Proxy Cache zum ersten Mal eine Anfrage von einem Client nach einem Clip, so gibt dieser die Anfrage an den Server weiter, der sodann mit der Übertragung des Clips zum Cache beginnt. Dieser legt einen Ringbuffer im Hauptspeicher an, gibt die Daten an den Client weiter und füllt gleichzeitg den Buffer mit den erhaltenen Daten. Erhält der Proxy eine weitere Anfrage nach dem gleichen Clip und sind die Daten noch im Ringbuffer erhältlich, so wird diese Anfrage direkt aus dem Ringbuffer bedient.

In Abbildung 4 fragt Client C_1 den Beginn eines Clips zum Zeitpunkt t_1 an. Dieser Request r_1 wird zum Server weitergeleitet. Nach Erhalt der ersten Daten vom Server legt der Proxy einen Ringbuffer B für Δ_b Sekunden an. Zum Zeitpunkt t_2 verlangt C_2 den gleichen Clip durch Request r_2. Ist t_2 nahe genug an t_1, d.h. ist die temporäre Distanz $\Delta_{(r_1,r_2)} = t_2 - t_1 \leq \Delta_b$, so ist der Anfang des Stroms zum Zeitpunkt t_2 immer noch im Buffer. r_2 kann also direkt aus dem Buffer bedient werden (s. Abbildung 4a). Ist $\Delta_{(r_1,r_2)} > \Delta_b$, so ist der Anfang des Clips nicht mehr im Ringbuffer und der Proxy legt einen neuen Buffer B' der Größe Δ_b an und füllt den Buffer entweder durch gecachte Daten von Festplatte oder durch einen neuen Request wieder vom Server (s. Abbildung 4b). Der Ringbuffer kann als Fenster auf einen Clip betrachtet werden, das sich ständig zum Ende des Clips hin versetzt. Der Proxy muß sicherstellen, daß es während der Wiedergabe weder zu einem Bufferüberlauf, noch zu einem Bufferunterlauf kommt. Dies kann für alle Clients sichergestellt werden, wenn es für den ersten Client und für den letzten

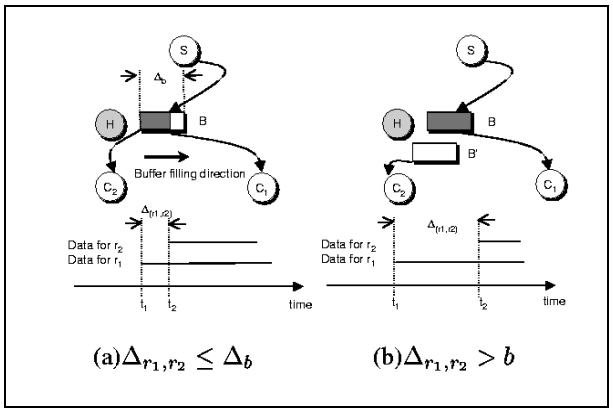

Abbildung 4: Allokierung des Ringbuffers für zwei Anfragen r1 und r2 (Quelle [1])

Client garantiert wird. Dazu kann eine Liste aller Clients eines Buffers gehalten werden, die nach der Position des nächsten zu übertragenden Paketes sortiert wird. Ein Bufferüberlauf kann verhindert werden, wenn sicher ist, daß sich der Garbage Collector immer hinter dem letzten Client befindet. Ein Bufferunterlauf wird verhindert, wenn eine Datenquelle (sei es Festplatte oder Netzverbindung zum Server) immer vor die Position des ersten Clients Daten anliefert. Welcher Client den letzten bzw. den ersten Client darstellt kann sich während der Wiedergabe laufend ändern, da sich die Wiedergaberate und andere Parameter ändern können. Diese Liste kann für den Garbage Collector benutzt werden, der einen Teil des Buffers erst dann löscht, wenn er zum letzten Client übertragen wurde. Ein alternativer Ansatz wäre das herkömmliche Referenzenzählen, d.h. für jeden Block wird die Anzahl der interessierten Clients gezählt und bei der Übertragung an den Client wird diese Zahl dekrementiert. Bei dynamisch hinzukommenden und wegfallenden Clients hätte dies den Nachteil die Anzahl Referenzen aller Blocks zu aktualisieren. Dieser Overhead entfällt bei der Listensteuerung des Garbage Collectors.

3.3 Kontrolle der Datentransferrate

Nach Abschicken einer Anfrage eines Clients an den Server, beginnt dieser normalerweise die Daten mit einer Abspielrate von r Bytes/Sekunde an den Client zurückzusenden. Hierbei wird zunächst der lokale Buffer von K Sekunden eines Clients gefüllt, um Netzwerkstörungen zu vermeiden bzw. zu vermindern. Wenn ein Client nun von einem Proxy bedient wird, der mit der Rate r Bytes/Sekunde dem Client die Daten zusendet, so wird der Buffer schneller gefüllt als bei einer direkten Verbindung zum Server auf Grund der kürzeren Netzwerkverbindung und der geringeren Netzauslastung. Der Proxy hat sogar weiter die Möglichkeit, die Daten, die der Client benötigt um seinen Playback-Puffer zu füllen mit einer

höheren Geschwindigkeit zu senden um die Wartezeit beim Abspielbeginn weiter zu verringern. In Abbildung 5 werden zwei Szenarien dargestellt. Im Szenario (a) ist ein Client und ein Server beteiligt, im Fall (b) ist ein Proxy an der Übertragung beteiligt. Die Verzögerung zwischen Client und Proxy sei d_1 und zwischen Proxy und Server d_2. Einfachheitshalber sei die Verzögerung zwischen Client und Server $d_1 + d_2$. Ziel des Proxies ist es die Startup-Latency zu vermindern. Diese ist im Falle ohne Proxy $L_0 = 2(d_1 + d_2) + K$. Der Faktor zwei ergibt sich hierbei aus der Roundtrip-Zeit.

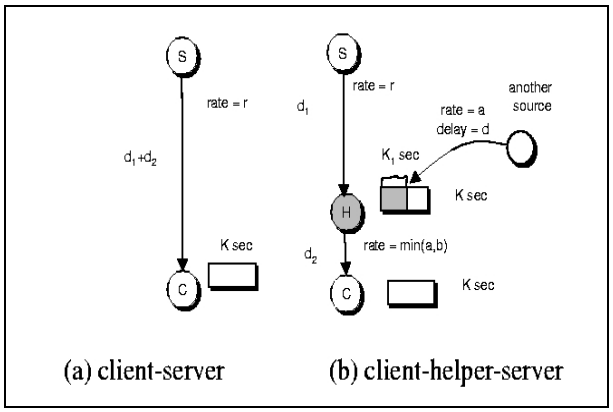

Abbildung 5: Übertragungsrate zwischen Client und Proxy (Quelle [1])

Im Szenario (b) hat der Proxy K_1 Sekunden der Daten in seinem Buffer, wobei $0 \leq K_1 \leq K$. Die Puffergröße ist unabhängig von K, allerdings sind nur die ersten K Sekunden wichtig für die Startup-Latency. Bis der Request beim Proxy ankommt vergehen d_2 Sekunden. Sobald der Proxy den Request erhält, startet er zwei Prozesse gleichzeitig. Er sendet die ersten K_1 Sekunden der Daten so schnell die Bandbreite es erlaubt an den Client. Bei einer Übertragungsrate von b Bytes/Sekunde dauert dies $(K_1 r)/b$ Sekunden. Außerdem muß der Proxy die verbleibenden $K - K_1$ Sekunden von einer anderen Datenquelle anfordern, sei es der Festplatte, einem anderen Proxy oder dem Server. Ist die einfache Verzögerung zwischen dieser Datenquelle und dem Proxy d und sei die Übertragungsrate zwischen Proxy und der Datenquelle a, so dauert es $2d$ Sekunden bis das erste Byte der Datenquelle beim Proxy ankommt. Die Zeit, die für beide Prozesse gebraucht wird ist also $max(K_1 r/b, 2d)$.
Sobald die beiden Prozesse beendet sind, kann der Proxy die verbleibenden $K - K_1$ Sekunden der Daten an den Client senden. Hierbei wird der Buffer mit einer Rate von a gefüllt und mit einer Rate von b geleert. Um einen Pufferunterlauf zu verhindern, muß die tatsächliche Rate zum Versenden der Daten auf $min(a, b)$ gesetzt werden. Daraus ergibt sich, daß zum Übertragen dieser Daten eine Zeit gebraucht wird, die sich aus $d_2 + (K - K_1)r/min(a, b)$ ergibt. Die gesamte Startup-Latency ist dann:

$L_1 = d_2 + max(K_1 r/b, 2d) + d_2 + (K - K_1)r/min(a, b)$
Im Falle, daß der Helper keine Daten gecached hat und als externe Datenquelle den Server benutzt, gilt $K_1 = 0, d = d_1, a = r$ und $min(a, b) = min(r, b) = r$. Dann ergibt sich eine Startup-Latency von $L_1 = 2d_1 + 2d_2 + K$, also das Ergebnis von L_0.
Der einzige vom Proxy beinflußbare Parameter zur Verminderung der Startup-Latency ist die Datentransferrate zwischen Proxy und Client. In einem Beispiel, bei dem der Proxy die verbleibenden $K - K_1$ Sekunden der Daten vom Server bezieht wird $a = r$ und $d_1 = 10ms, d_2 = 1ms$ und $K = 5s$ gesetzt. Variiert wird das Verhältnis der Datentransferraten b und r (also der Datenrate zwischen Client und Proxy und Server und Proxy) zwischen 1 und 10 und die Größe der gecachten Daten K_1 zwischen 0 und 5 Sekunden. Ist $K_1 = 0$, dann ist die Latency genauso groß wie beim direkten Bezug der Daten vom Server. Ist $K_1 \geq 1$, dann werden K_1 Sekunden der Daten mit der Rate b übertragen und $K - K_1$ Sekunden mit der Rate r. Mit fixem K_1 zeigt Abbildung 6, daß sich die Startup-Latency verringert, wenn b/r vergrößert wird und bei fixem r/b nimmt die Latency ab, wenn K_1 vergrößert wird.

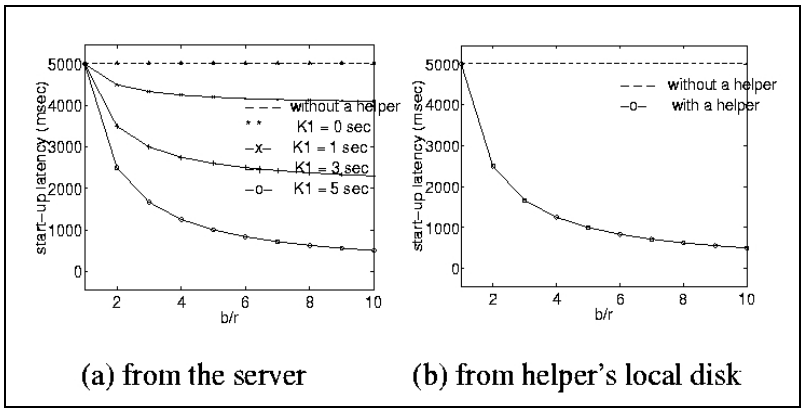

Abbildung 6: Startup-Latency für verschiedene Datenquellen (Quelle [1])

3.4 Qualitätsanpassung

Rejaie et al. schlagen in [7] vor, die Eigenschaften von hierarchisch kodierte Medienströme auszunutzen, um ein Caching mit sich anpassender Qualität zu ermöglichen. Hierbei wird ausgenutzt, daß sich ein Medienstrom aus mehreren Ebenen zusammensetzt. Die unterste Ebene speichert den Inhalt in minimaler Qualität, während die aufsetztenden Schichten optionale Qualitätsverbesserungen bieten. Das folgende Beispiel erklärt, wie ein Cache mit sich anpassender Qualität funktioniert.
Ein Client fordert ein Videoclip an, der sich augenblicklich noch nicht im Cache

befindet (sogenannter „cache miss“). Der Strom wird ausgehend vom Proxy vom Server angefordert, hier zwischengespeichert und so wie er vom Proxy erhalten wird direkt an den Client weitergegeben. Der Client profitiert also noch nicht von der Qualitätsverbesserung. Durch Übertragungsfehler und Schwankungen in der Bandbreite zwischen Proxy und Server fehlen mitunter ganze, höhere Schichten, bzw. einzelne Pakete der unteren Schichten (s. Abbildung 7)

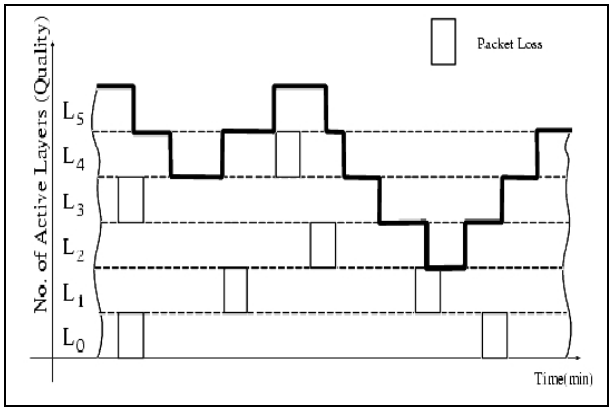

Abbildung 7: Beispiel eines Stroms im Cache mit heterogener Qualität (Quelle [7])

Erfolgt eine zweite Nachfrage nach dem gleichen Strom, kann der Cache nun den Client aus den zwischengespeicherten Daten bedienen („cache hit“). Hierbei sind zwei Szenarien denkbar:

1. Die durchschnittliche Bandbreite der Wiedergabe ist kleiner oder gleich der gespeicherten Bandbreite.

2. Die durchschnittliche Bandbreite der Wiedergabe ist größer der gespeicherten Bandbreite.

Obwohl im ersten Szenario die gespeicherte Bandbreite größer ist, können trotzdem im gespeicherten Strom einzelne Teile der für diese Qualität gewünschten höheren Schichten fehlen, da sie durch Qualitätsanpassungen des Servers bei der Übertragung zum Proxy nicht gesendet wurden. Der Proxy verlangt zur Qualitätsverbesserung die fehlenden Teile der Schicht und „repariert“ die durch Paketverlust aufgetreten Fehler. In Abbildung 8 sind zum Beispiel im Intervall $[t_2, t_3]$ die Schicht 2 und im Intervall $[t_1, t_4]$ die Schicht 3 betroffen.
Abbildung 9 zeigt das 2. Szenario, in dem nicht nur niedrigere Schichten repariert werden, sondern auch höhere Schichten vom Proxy angefordert werden, sobald es die höhere Bandbreite erlaubt. Alle vom Server angeforderten Segmente werden vom Proxy zwischengespeichert, so daß die Qualität der zwischengespeicherten Clips von Anfrage zu Anfrage steigt.
Problematisch hierbei ist, daß der Proxy zwei unsynchronisierte Verbindungen

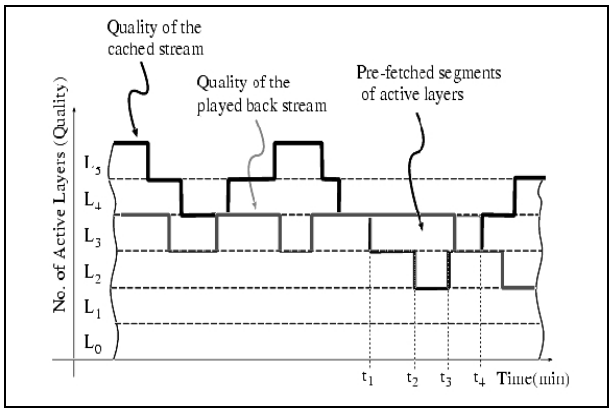

Abbildung 8: Lieferung niedriger Bandbreite aus dem Cache (Quelle [7])

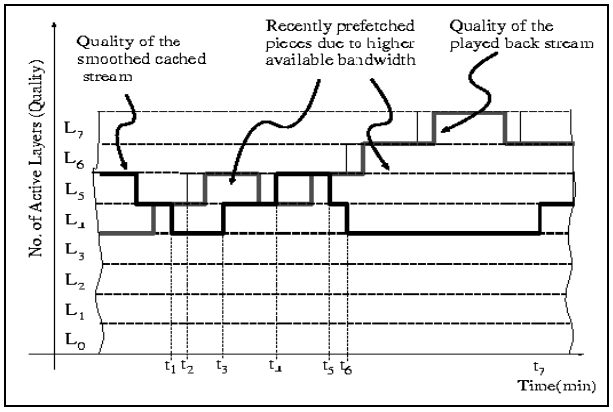

Abbildung 9: Lieferung höherer Bandbreite aus dem Cache (Quelle [7])

halten muß, nämlich die Verbindung zum Client und die Verbindung zum Server. Das Abholen eines fehlenden Segments vom Server verlangt mindestens die Zeit einer vollen Übertragung (Roundtrip Time - RTT). Der Proxy muß also die fehlenden Teile des Stroms anfordern und erhalten, bevor die Wiedergabe zum Client die entsprechende Stelle erreicht. Da die Qualität der Übertragung zum Server schwanken kann, ist es für den Proxy nicht möglich apriori zu wissen, welche Schichten er zusätzlich anfordern kann. Es liegt also ein gewisser Tradeoff vor: je früher der Proxy das fehlende Segment anfordert, desto ungenauer ist die Vorhersage, wieviele Schichten er erhalten kann, aber desto größer ist die Chance, daß die Pakete den Proxy rechtzeitig erreichen. Erreichen die Pakete den Proxy zu spät, so kann dieser sie zwar nicht mehr an den Client senden, kann sie aber trotzdem für nächste Wiedergaben zwischenspeichern. Um den Wiedergabestrom und den Strom vom Server miteinander zu synchronisieren, holt sich der Proxy die Daten vom Server in einem sogenannten „prefetch window“, das mit der gleichen Geschwindigkeit wie der Wiedergabestream vorwärts gleitet. Zum Wiedergabe Zeitpunkt t_p untersucht der Proxy das Intervall $[t_p + T, t_p + T + \delta]$ und identifi-

ziert alle fehlenden Segmente in diesem Intervall. Der Proxy sendet dann einen einzigen Request an den Server um die fehlenden Segemente zu erhalten. Der Server sendet die Segmente nach ihrer Priorität, d.h. nach Schichtnummer. Zuerst die unteren wichtigeren Schichten, dann die zusätzlichen. Erhält ein Server eine zweite Anfrage, bevor die erste vollständig abgearbeitet ist, so wird der Rest der ersten Anfrage ignoriert und mit der Abarbeitung der zweiten Anfrage begonnen. Im Gegensatz zu herkömmlichen Caches, die Seiten entweder ganz oder gar nicht aus dem Cache löschen, falls sie nicht mehr benötgt werden, kann bei Medienströmen der Speicher differenzierter freigegeben werden. Um Speicher effizient zu verwalten, werden bei diesem Ansatz die als nicht speichernswert angesehen Schichten gelöscht. Es ist sinnvoll, zunächst die höherwertigen Schichten ganz zu löschen und zwar ausgehend vom Ende zum Anfang hin und dann mit niedrigen Schichten fortzufahren, so daß sie für Clients mit niedriger Bandbreite sehr lange zur Verfügung stehen.

4 Verfügbare Produkte

Obwohl die Technologien rund um das Caching von Medienströmen noch Gegenstand aktueller Forschungsbemühungen sind, können am Markt schon kommerzielle Lösungen für Audio- und Videocaches erworben werden. Allerdings gewähren die Anbieter aus Gründen des angespannten Wettbewerbs in diesem Bereich nur selten tiefere Einblicke in die eingesetzten Technologien. Grundsätzlich lassen sich reine Softwarelösungen von spezialisierten Hardwareboxen abgrenzen.
Lucent Technologies Inc., die offizieller Caching-Partner von Real Networks Inc. wurden, bieten mit Imminet Webcache S100 eine in Hardware realisierte Cachinglösung, die im 1. Quartal 2001 verfügbar sein wird. Sie wird das segmentierte Caching und die Requestzusammenführung unterstützten.
Auch DynaCache von InfoLibria Inc. bietet Hardwareboxen an, die Multmediaströme nahe am Enduser cachen. Als OEM Partner von Nortel Networks Ltd. wird deren Produkt von einem führenden Hersteller von Netzwerktechnologie gefördert. DynaCache kann sowohl als Proxy im Browser des Endusers konfiguriert, sowie als transparenter Layer 4/7 Switch eingesetzt werden.
Mit EdgeAdvantage bietet auch Akamai Technologies Inc. eine Platform für das Zwischenspeichern von Multimediaströmen an.
Die Lösung von CacheFlow Inc. unterstützt neben den Protokollen von Real Networks auch Microsofts Windows Media sowie MP3 und MPEG. CacheFlow arbeitet augenblicklich an der Integration von Apples QuickTime.
Eine Softwarelösung für das Betriebssystem Solaris bietet Inktomi Inc. mit dem Traffic Server Media-IXT an. Mehrere Trafficserver können zusammen wirken und so eine hierarchisch angeordnete Cachinginfrastruktur aufbauen. Des weiteren können Medienströme mit der sogenannten Inktomi Content Delivery Suite auch schon im Vorhinein an die einzelnen Caches verteilt werden. Neben Real Networks RTSP/PNA wird auch Microsoft Windows Media unterstützt.
Eine weitere Softwarelösung bietet Real Networks Inc. mit dem RealSystem Proxy8 an. Die Caching Software läuft augenblicklich sowohl unter SUN Solaris als auch unter Microsoft Windows NT/2000 und Linux. Neben Requestzusammenführung bietet Proxy8 auch das Caching von Medienströmen an. Allerdings realisiert der Proxy kein Prefix-Caching, sondern die Ströme werden komplett gecached. Das Weiterleiten der Medienströme kann bei diesem Produkt sowohl mit Unicast als auch mittels IP Multicast erfolgen, was eine weitere Einsparung an benutzter Bandbreite ermöglichen kann. Unterstützt werden nur die Protokolle RTSP und PNA.

5 Zusammenfassung und Ausblick

Der Einsatz von Proxy Caches wird eine Möglichkeit bieten, den immer größer werdenden Bedarf an Bandbreite und Übertragungsqualität zu mildern. Der Einsatz von Hochgeschwindigkeitsnetzen bis zum Enduser, wie zum Beispiel der Ausbau der DSL Netze, verstärkt diesen Trend noch weiter und erzeugen zusätzliche Nachfrage nach Audio- und Videoübertragungen, die durch diese Technologien erst wirklich interessant werden.
ZD Net hat das Jahr 2001 als das Jahr der Streaming Media bezeichnet und die Internet Research Group prophezeit dem Markt der „streaming media services“ einen 20fachen Wachstum bis zum Jahr 2004 und ein Marktvolumen von $2,5 Milliarden (s. [5]). Der Einsatz von Proxy Caches, die streaming media unterstützen wird bis dahin so selbstverständlich sein wie der Einsatz von Proxies für statische Inhalte wie HTML-Seiten und Bilder. Alle großen Netzwerkspezialisten (wie Nortel Networks Inc. und Lucent Technologies Inc.) und die auf Caching spezialisierten Unternehmen (wie Akamai Inc. und Inktomi Inc.) verfügen schon über entsprechende Produkte in ihren Portfolios.
Die vorliegende Seminararbeit hat die Vorteile des Einsatzes von Proxy Caches und die wichtigsten Technologien, die hierbei zum Einsatz kommen vorgestellt. Noch nicht alle möglichen Technologien haben sich in verfügbaren Produkten wiedergefunden, insbesondere der Einsatz des Quality Adaptive Streaming konnte sich noch nicht in kommerziellen Entwicklungen wiederfinden.
Was der Einsatz von Proxy Caches dem Endbenutzer qualitativ bringen wird, werden Studien des praktischen Einsatzes noch belegen müssen. Qualitative Aussagen über die Kostenersparnis gegenüber einem weiteren, breitbandigeren Ausbau der Netzwerkinfrastruktur kann sich jedoch leicht messen lassen und wird sowohl ISPs als auch Content Provider vom Einsatz von Proxy Caches überzeugen.

Literatur

[1] Ethendranath Bommaiah, Katherine Guo, Markus Hofmann und Sanjoy Paul. Design and Implementation of a Caching System for Streaming Media over the Internet. In: *Proceedings of the Sixth IEEE Real Time Technology and Applications Symposium (RTAS 2000)*, Washington, VA, USA, Mai 2000.

[2] J.D. Day and H. Zimmermann. The OSI Reference Model. In: *Proceedings of the IEEE*, Band 71, Seiten 1334–1340, Dezember 1983.

[3] Markus Hofmann, T.S. Eugene Ng, Katherine Guo, Sanjoy Paul und Hui Zhang. Caching Techniques for Streaming Multimedia over the Internet. Technical Report BL011345-990409-04TM, Bell Laboratories, April 1999.

[4] Markus Hofmann und Krishan Sbnani. Streaming and Broadcasting over the Internet. In: *Proceedings of the IEEE ATM 2000*, Heidelberg, Deutschland, Juni 2000.

[5] CacheFlow Inc. Streaming Media with CacheFlow. `http://www.cacheflow.com/files/whitepapers/streaming_wcf_v2_wp.pdf`, November 2000.

[6] B.M. Leiner, R. Cole, J. Postel und D. Mills. The DARPA Internet Protocol Suite. *IEEE Communications Magazine*, Band 23, Seiten 29–34, März 1985.

[7] Reza Rejaie, Haobo Yu, Mark Handley und Deborah Estrin. Multimedia Proxy Caching Mechanism for Quality Adaptive Streaming Applications in the Internet. In: *Proceedings of IEEE Infocom'2000*, Tel-Aviv, Israel, März 2000.

[8] H. Schulzrinne, S. Casner, R. Frederick und V. Jacobsen. RTP: A Transport Protocol for Real-Time Applications. Internet Request for Comments 1889, Januar 1996.

[9] H. Schulzrinne, A. Rao und R. Lanphier. Real Time Streaming Protocol (RTSP). Internet Request for Comments 2326, April 1998.

[10] Subhabrata Sen, Jennifer Rexford und Don Towsley. Proxy Prefix Caching for Multimedia Streams. In: *Proceedings of IEEE Infocom'99*, Seiten 1310–1319, New York, USA, März 1999.

[11] D. Wessels. Squid Web Proxy Cache. http://www.squid-cache.org, 2000.