Technische Universität Ilmenau

Fakultät für Informatik und Automatisierung

Fachgebiet Telematik/Rechnernetze

Betreuer: Dipl.-Inf. Thorsten Strufe

Hauptseminar im SS 2004

Thema

„Caching“

vorgelegt von:

Tellhelm, Karsten

Ehrenbergstrasse 37 (P 315)
98693 Ilmenau

Tel.: 0177/7901222

hrlm22@yahoo.de

Matrikel-Nr. 2828

Inhaltsverzeichnis

Abbildungsverzeichnis

1.Einführung

Der zunehmende Wandel bei der Anbindung von privaten Endnutzern an das Internet, führt auch zu einem Wandel bei dessen Nutzung. Hatten im Jahr 2000 erst 200.000 Einwohner Deutschlands einen Internet-Breitbandzugang, so waren es im Jahr 2001 bereits 780.000 Personen.[1] Das ergibt innerhalb eines Jahres eine Steigerung um 290%. Es ist zu erwarten, dass sich diese Entwicklung vorsetzen wird.
Durch den besseren Zugang der Nutzer zum Medium Internet ist es möglich, Inhalte anzubieten welche eine höhere Bandbreite benötigen. Somit avanciert das Internet mehr und mehr zur universellen Infrastruktur von Informationen verschiedenster Art, vor allem wegen kontinuierlicher Medien wie Audio und Video.
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. So soll der Weltweite Markt für Video-Streaming von derzeit 991 Millionen Dollar bis zum Jahr 2007 auf 4,5 Milliarden Dollar wachsen.[2]

1.1 Problemstellung

Die steigende Nutzung von Multimediadiensten führt zu einer stärkeren Belastung der Netzinfrastruktur, verbunden damit sind steigende Kosten der Internetbetreiber und die hohen Investitionen an Server- und Netzwerkhardware seitens der Serviceanbieter der Mediendienste.
Desweiteren ist die Qualität der übertragenen Video- und Audioströme in der Regel nicht zufriedenstellend. Die derzeitige Lage ist von hohen Wartezeiten zu Beginn der Wiedergabe, niedrige Bild- und Tonqualität, stockende Übertragungen und zwischenzeitliche Datenverluste geprägt.

---

[1] Vgl. Organisation für Wirtschaftliche Zusammenarbeit und Entwicklung /Statistische Veröffentlichung/ The development of Broadband access in OECD countries
[2] Vgl. Olsen, Fiutak / Streaming-Markt soll bis 2007 auf 4,5 Milliarden Dollar wachsen /
Stefanie Olsen, Martin Fiutak: Streaming-Markt soll bis 2007 auf 4,5 Milliarden Dollar wachsen
http://www.zdnet.de/news/business/0,39023142,39115951,00.htm Abruf: 2004-07-04

Außerdem ist der Nutzer speziell bei Multimediaangeboten, wie Audio und Video, an hohe Qualität seitens des Tons und des Bildes durch Radio und Fernsehen gewöhnt.

1.2 Zielsetzung

Der Einsatz von Caching-Strategien soll Abhilfe bei den oben genannten Problemen bieten. Im Gegensatz zum Cachen von Text- und Bilddaten, welches schon seit längerem betrieben wird, erfordert das zwischenspeichern von Medienströmen besonderer Technologien.
Der Grund hierfür liegt bei den besonderen Eigenschaften dieser Ströme. So ist zum Beispiel meist das vollständige Cachen, wegen der Objektgröße nicht möglich. Weiterhin stellen Medienströme strenge zeitliche Anforderungen an die Übertragung und Qualität.
Neue Ansätze welche sich an Punkten wie schneller Präfix-Transfer, Skalierbarkeit durch Segmentierung der Streaming-Objekte und Anfrage-Aggregierung orientieren müssen beachtet werden.
Auch den Aspekt besseren Komforts und einer erhöhten Funktionalität muss man mit einbeziehen (zum Beispiel Pause Funktion, Rückspulen).

1.2 Aufbau der Arbeit

Im Rahmen dieses Hauptseminars soll auf die verschiedenen Technologien, die beim Caching von Medienströmen zum Einsatz kommen, eingegangen werden. Im Anschluss an die Einleitung wird im zweiten Kapitel die Segmentierung als wichtige Grundlage des Caching erläutern. Im folgenden dritten Kapitel wird auf die Clientseitige Form des Cachen, das Prefix Cachen eingegangen. Das vierte Kapitel soll die Funktionsweise und den Aufbau des Proxy-Caching erklären. In Kapitell fünf wird dargelegt, welche Möglichkeiten des Cachings es in Peer to Peer Netzwerken gibt. Abschließend wird im Kapitell sechs die Hauptseminararbeit zusammengefasst und ein Fazit getroffen.

2. Segmentierung des Datenstroms

Wie schon erläutert bestehen bei dem Cachen von Medienströmen zwei wichtige unterschiede zum Cachen von normalen statischen Webseiten. Zum einen erschwert die Grösse des Streams und auf der anderen Seite die Anforderung an das Timing das Zwischenspeichern.
So benötigt man bei einem Videofilm in akzeptabler Qualität ca. 500 kBit/s (Vollbilddarstellung). Bei einem Kinofilm von ungefähr 90 Minuten Länge werden dem nach mehrere hundert Megabyte Speicherkapazität gebraucht. Der Speicherplatz und der Zeitaufwand sind also zu groß um diesen komplett zu cachen. Ein wichtiger Ansatz um dieses Problem zu lösen ist die Segmentierung.

2.1 Ermittlung der Segmentgröße

Bei der Segmentierung von Medienströmen werden diese in einzelne Segmente unterteilt und unabhängig voneinander gecached.[3] Bei der Größeneinteilung der Segmente existieren unterschiedliche Ansätze.
Will man eine optimale Speicherausnutzung und Verwaltung des Caches garantieren, so nehmen die Segmente die Größe eines vielfaches der Größe eines Festplattenblockes des Caches an.
Ein anderer Ansatz geht von der über das Netz versendeten Paketgröße aus. Ein Segment ist nun so groß wie dieses Paket. Bei Verlusten einzelner Pakete, kann man dieses leichter kompensieren indem man das Segment erneut versendet. Im Gegensatz dazu müssten bei dem Speicherorientierten Ansatz, unter Umständen mehrere Segmente neu übermittelt werden. Bei diesem Ansatz ist also die Netzwerklast geringer und somit die Datenübermittlung schneller.

[3] Vgl. Hofmann, Eugene Ng, Guo, Paul, Zahng. /Caching Techniques for Streaming/

2.2 Einteilung der Segmente

Das völlig unabhängige speichern und ersetzen der eingeteilten Segmente ist nicht sinnvoll. Bei einer Anfrage an den Cache, nach einem Stream, kann man davon ausgehen das dieser in der Regel nicht vollständig vorliegt. Es sind nur einzelne Segmente gespeichert, fehlende Segmente müssen zeitaufwendig bei der Originalquelle angefordert werden. Es entsteht somit ein hoher Administrations- und Synchronisationsaufwand. Aufgrund von Synchronisationsproblemen wäre ein Abbruch der Übertragung durchaus möglich.
Um diesem Problem zu begegnen wird der Stream noch einmal in logische Abschnitte unterteilt (zum Beispiel einzelne Kapitel innerhalb eines Films). Diese logischen Abschnitte werden als „chunks“ bezeichnet. Ein chunk besteht aus einer fortlaufenden Sequenz von Segmenten. Ein solcher logischer Abschnitt wird unabhängig von anderen Abschnitten mittels der Prefix-Caching-Strategie gecached.[4]

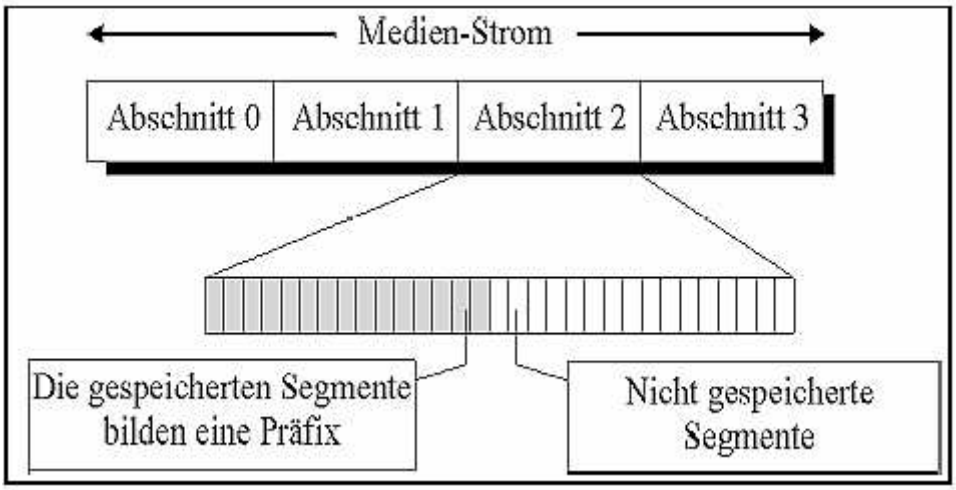

Abbildung 1: Einteilung eines Stroms[5]

Dabei müssen nicht alle Segmente eines chunks gecached sein. Gecachte Segmente eines logischen Abschnittes werden als Präfix bezeichnet. Das Speichern und ersetzen im Cache wird nie mit ganzen chunks durchgeführt sondern mit einzelnen Segmenten.

[4] siehe hierzu 3. Kapitel „Prefix Caching“
[5] Hofmann, Eugene Ng, Guo, Paul, Zahng. /Caching Techniques for Streaming/

Existiert nun eine Anfrage an den Cache für einen bestimmten logischen Abschnitt, so darf dort kein Segment ersetzt oder gelöscht werden. Es besteht nur die Möglichkeit, das der Präfix dieses chunks vergrößert wird, bis zu dem Zustand wo der Präfix gleich dem chunk ist. Man geht davon aus, dass speziell in logischen Abschnitten auf die der Nutzer gerade zugreift, Funktionen wie Rückspulen oder Pause öfters benutzt werden.
Benötigt der Cache nun weiteren Speicherplatz um unser Präfix, des aktuell abgefragten chunks zu vergrößern, so löscht er Segmente innerhalb des Präfixes nicht abgefragter chunks. Es wird immer das zu letzt gespeicherte Segment innerhalb eines chunks gelöscht. Dies hat den Hintergrund, dass man davon ausgeht das ein Abschnitt meist vom Beginn an betrachtet wird. Sollte zu einem späteren Zeitpunkt ein Zugriff auf diesen logischen Abschnitt vorliegen, sind immer noch einige gespeicherte Segmente am Anfang vorhanden. Bei der Auswahl der chunks, von denen Segmente gelöscht werden können, gibt es verschiedene Möglichkeiten:

1. **Least Recently Used (LRU):** Hier wird der Zeitpunkt des letzten Zugriffs auf einen logischen Abschnitt mitgespeichert. Bei dem am längsten unbenutzten Abschnitt (least recently used – LRU) werden die Segmente gelöscht.

2. **Least Frequently Used (LFU):** Es werden zu jedem chunk die Anzahl der Zugriffe vermerkt. Bei dem chunk mit der geringsten Anzahl an Zugriffen wird mit der Löschung begonnen.

3. **First In First Out (FIFO):** In der Reihenfolge wie die Daten gespeichert werden, werden sie auch gelöscht. Der Algorithmus ist sehr schnell, wenn die Daten die gleiche Größe haben.

4. **Staging Hot Video Only (SHVO):** SHVO ist ein heuristischer Algorithmus, welcher speziell für Video-Proxys entwickelt wurde. Es wird ermittelt wie viele Gleichzeitige Zugriffe auf einen chunk existieren. Nur ausreichend Nachgefragte chunks werden gecached. Ist ein chunk „hot“, es existieren also sehr viele Zugriffe, so wird er komplett gecached. Bei Chunks mit wenigen parallelen Zugriffen werden Segmente gelöscht.

5. **Largest Bandwidth Reduction Ratio First (LBRRF)**: Dieser Algorithmus ist ebenfalls speziell für Videos entwickelt wurden. Hierbei wird nicht nur die Zugriffsstatistik vermerkt sondern auch die Charakteristik des chunks. Es wird die Wechselhaftigkeit der Bandbreitenanforderung ermittelt, um Festzustellen wie sich die obere Bandbreitenanforderung senken lässt. Gibt es bei chunks steile spitzen, so lohnt es sich diese Teile in den Cache abzulegen.

Die Größe eines Präfixes richtet sich nach verschiedenen Gesichtspunkten. Zum einen nach der Übertragungsgeschwindigkeit zwischen Cache und Client, als auch nach der Wiedergabeverzögerung beim Client. Das Präfix sollte groß genug sein, um die Verzögerungen zwischen Server und Cache und den Verlust einzelner Pakete zu kompensieren, wenn der Rest des Clips vom Server bezogen wird.
Geht man davon aus, das die Verzögerung, gemessen in Frame Slots, zwischen Cache und Client zwischen t(min) und t(max) liegt und die Startup Latency des Clients bei k ist, muss ein angemessener Cache dann {t(max) – k, 0} Bilder speichern.

2.3 Struktur der Segmente

Die Ströme werden meist in Files gecached. Solche Dateien beginnen mit einem Medienfileheader, welcher Medienspezifische Informationen und eine Tabelle, die Offsets auf die einzelnen vorhandenen Segmente speichert, enthält.
Weiterhin bestehen die Dateien aus nicht überlappenden Zeitsegmenten. Diese werden nach dem Medienfileheader gespeichert und besitzen zusätzlich selbst einen Header, welcher Informationen über Start- und Endzeitpunkt, Sequenznummer, Anzahl der Pakete und eine Indextabelle zum schnellen Zugriff auf die einzelnen Blöcke der entsprechenden Zeiger speichert. Danach folgen die eigentlichen Datenpakete in zeitlich geordneter Reihenfolge.
Es existiert somit eine Baumstruktur, wobei der Medienfileheader mit der Segmenttabelle die Wurzel und dessen Kinder die Zeitsegmente sind.

3. Prefix Caching

Beim Prefix Caching handelt es sich um eine Clientseitige Form des Cachens. Beim Prefix Caching werden nur wenige Sekunden des eingehenden Datenstroms in einen eigenen Puffer gespeichert (dejitter buffer). Dies hat den Sinn Abweichungen in der Übertragung (jitter) und Paketverluste (paket loss) auszugleichen.
Bei dem Auffüllen des dejitter buffers wird ein gewisses Zeitfenster benötigt (startup latency). Dies kann zu hohen Wartezeiten zu Beginn der Wiedergabe führen. Bei einem MPEG-2 Film mit guter Qualität, werden für 10 Sekunden dejitter buffer ca. 5-6 MBytes Speicherplatz benötigt. Geht man von einer Übertragungsgeschwindigkeit von 20-30kB\s aus, besteht also eine startup latency von 3-5 Minuten.
Speziell bei kürzeren Video- und Audioclips kann das als störend empfunden werden.
Durch Proxy Caches, die die ersten Sekunden eines Medienstroms zwischenspeichern, kann die Initiale Verzögerung erheblich vermindert werden.[6]
Ein weiteres Problem besteht bei dem erneuten Auffüllen des dejitter buffers. So kann es durch Paketverluste und Abweichungen in der Übertragung passieren, dass dieser „Aufgebraucht“ wird. Die Übertragung wird meist in diesem Fall gestoppt und erst fortgesetzt wenn der buffer wieder komplett aufgefüllt ist.

4. Proxy Caching

Unter Proxy Caches versteht man Netzwerkknoten, die Datenströme teilweise zwischenspeichern und für etwaige Anfragen vom Client bereithalten. In der Regel sind sie in der Infrastruktur näher an dem Client platziert als die eigentliche Datenquelle. Im Idealfall bezieht der Client also seinen Stream von dem Proxy Cache.

[6] Vgl. Sen, Rexford, Towsley /Proxy Prefix Caching/

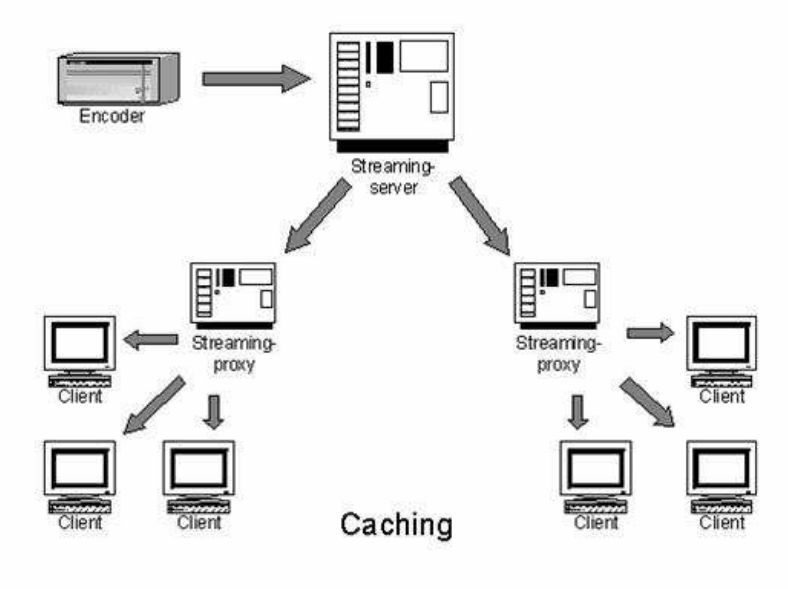

Abbildung 2: Proxy Cache, Netzwerktopologie

In der Praxis haben Proxy Caches eine Trefferquote von ca. 50%, das heißt jede zweite Anfrage kann von dem Proxy Cache erfolgreich beantwortet werden. Durch hierarchisches Caching (Proxy Cache für einen Proxy Cache) kann man diese Trefferquote nochmals verbessern. Hierfür wurde ein spezielles Protokoll entwickelt, das ICP (Internet Caching Protokoll). Proxy Caches schicken ihre Anfragen an benachbarte und in der Hierarchie höher stehende Proxy Caches. Ist das gewünschte Objekt vorhanden wird es übertragen, andernfalls wird weiter auf, in der Hierarchie höher stehende, Proxy Caches gesucht (bis zur Wurzel, dem Originalserver). Das Internet Caching Protokoll basiert auf UDP und ist somit relativ schnell, die entstehenden Performanceverluste durch die Mehrfachanfragen wird durch die erhöhte Trefferquote ausgeglichen.

4.1 Verbindungsaufbau zum Proxy Cache

Es gibt unterschiedliche Arten wie es ermöglicht wird, dass sich der Client mit dem Proxy Cache verbindet und nicht bei der Originalquelle nachfragt.
Ein Ansatz ist die Clientkonfiguration. Man trägt hier die Adresse des Proxy Caches in ein lokales Konfigurationsfile der Clientsoftware ein. Ein großes Problem bei dieser

Methode besteht darin, dass durch Unwissenheit nicht immer der nächstliegende Proxy Cache angesprochen wird. Weitere Nachteile sind ein hoher Administrationsaufwand, Fehleranfälligkeit und eine statische Umleitung ohne Berücksichtigung der Load-Balancing-Strategien.

Solche Nachteile kann man mit dynamischen Systemen umgehen. Hierzu werden Layer 4 Switches oder Layer 7 Switches verwendet. Die Layer 4/7 Switches ersetzen entweder die lokal vorhandenen Hubs/Switches oder befinden sich zwischen dem Access Point des Clients und dem ersten Router. Eingehende Anfragen gehen zuerst über diese Switches, werden dort untersucht und dann nach bedarf an den Proxy Cache oder die gewünschte Original Adresse weitergeleitet. Layer 4 Switches analysieren die eingehenden Pakete auf der Ebene des Transportprotokolls (Schicht 4 des OSI-Referenzmodells)[7], die im TCP/IP Modell der TCP/UDP-Schicht entspricht[8]. Dazu extrahiert der Switch die Protokoll ID und Portnummer aus den Paketen und leitet diese dann an einen Proxy Cache weiter, falls es sich um Anfragen an den Streamingserver handelt. Diese Layer 4 Switches können ein einfaches Load-Balancing betreiben, zum Beispiel durch Zufallszuteilung, Round-Robin oder durch ermitteln des Proxies mit den wenigsten TCP/IP-Verbindungen.

Die Layer 7 Switches hingegen arbeiten auf der Anwendungsschicht des OSI Modells (Schicht 7). Hier werden spezifische Informationen über die Anwendung aus den Paketen gewonnen, um zu entscheiden wohin die Pakete weitergeleitet werden sollen. So wird es ermöglicht komplexere Load Balancing Strategien zu realisieren.

Ein weiterer Ansatz besteht in der Ausnutzung des Domain Name Services (DNS). Im Gegensatz zu der sonst üblichen Methode, dass ein Hostname immer statisch in einer IP-Adresse aufgelöst wird, verwendet man hier dynamische Abbildungstabellen von IP-Adressen und Hostnamen. Da der DNS hierfür eigentlich nicht vorgesehen ist, bringt diese Benutzung einige Nachteile mit sich. Will zum Beispiel ein Client die IP-Adresse eines Servers erfahren, von dem er lediglich den Hostname kennt, stellt er eine Anfrage an den lokalen DNS-Server. Dieser kann den Hostnamen in der Regel nicht auflösen, er muss also die Anfrage an den DNS-Server des Zielnetzes richten. Um den Zugriff auf den eigentlichen Server zu verhindern, liefert dieser DNS-Server nun die IP Adresse eines Proxy Caches. Dies sollte ein Proxy Cache sein, der für den Client einen schnellen

[7] Vgl. Day, Zimmermann /The OSI Reference Model/ In: Proceedings of the IEEE
[8] Vgl. Leiner, Cole, Postel, Mills /The DARPA Internet Protocol Suite/ IEEE Communications Magazine, Band 23/ S. 29-34

Zugriff garantiert. Das ist in der Regel der geographisch nächste Proxy Cache, dafür muss jedoch der Standort des Clients bekannt sein. Auf Grund des DNS Protokolls ist dem DNS-Server jedoch nur die IP-Adresse des lokalen DNS-Servers bekannt. Eine Ermittlung des besten Proxy Caches stellt sich also als schwierig dar.

4.2 Kommunikation mit dem Proxy Cache am Beispiel RTSP

Protokolle wie RTP (Real Time Protokoll) und RTSP (Real Time Streaming Protokoll) haben sich zu Standardprotokollen bei der Übermittlung von Medienströmen etabliert. Dabei wird das Real Time Streaming Protokoll dazu benutzt, die Lieferung von Medienströmen zu kontrollieren.[9] Es implementiert dabei Funktionalitäten wie das Starten, Anhalten, Vor- und Zurückspulen von Streams.
Das Real Time Streaming Protokoll definiert verschiedene Nachrichten (OPTIONS, DESCRIBE, SETUP, PLAY, TEARDOWN) zum Aufbau und zur Kontrolle von Sitzungen.

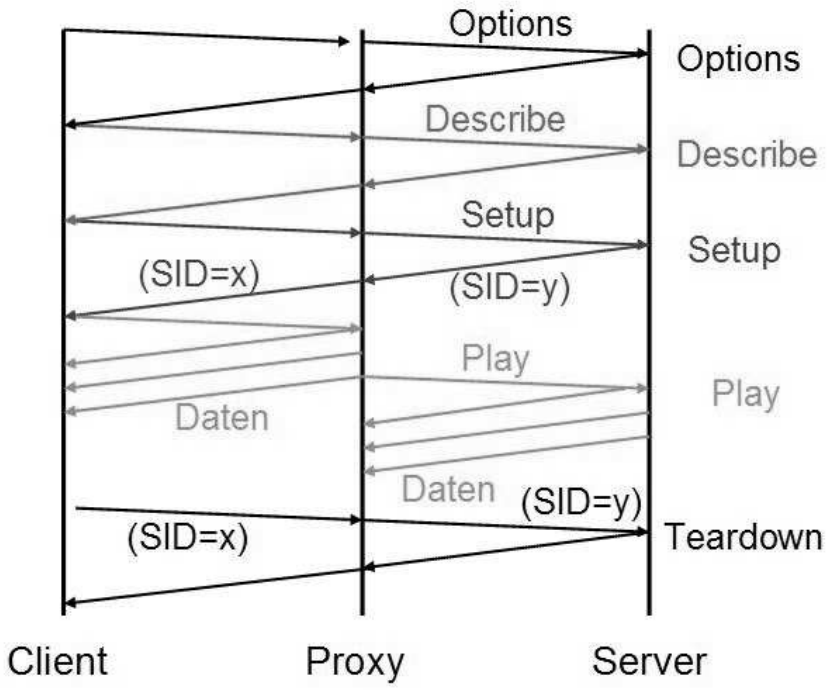

Abbildung 3: RTSP, Kommunikation Client-Proxy[10]

[9] Schulzrinne, Rao, Lanphier /Real Time Streaming Protocol/ Internet Request for Comments
[10] Hofmann, Sbnani /Streaming and Broadcasting over the Internet/ Proceedings of the IEEE ATM 2000

Die OPTIONS Nachricht wird dazu benutzt, um Informationen über die Eigenschaften des Servers zu bekommen, wie z.B. die von ihm unterstützten Protokollversionen. Bei der DESCRIBE Nachricht werden die Eigenschaften eines Streams (zum Beispiel Wiedergaberate) übermittelt. Woben man bei der SETUP Nachricht die Information über den zu benutzenden Transportweg (TCP oder UDP) erhält. Durch die PLAY Nachricht startet man die Wiedergabe, mittels eines RANGE Headers kann man dabei ein bestimmtes Intervall des Stroms anwählen. 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. Zwischen Server und Proxy sowie Proxy und Client bestehen zwei völlig getrennte Verbindungen mit verschiedener Sitzungsnummer (Session identifier, SID). Bei der Weiterleitung von RTSP Nachrichten des Clients an den Server, muss der Proxy also Teile der Headers austauschen. Der Proxy leitet die OPTIONS, DESCRIBE und SETUP Nachrichten an den Server weiter und gibt die Antworten an den Client zurück. Dabei modifiziert er entsprechend die Sequenz Nummer, Session- und Transport-Headerinformationen. Geht nun eine PLAY Nachricht bei dem Proxy ein, der einen Teil des Streams gecached hat, so kann er die gecachten Daten an den Client zurücksenden. Danach leitet der Proxy die PLAY Nachricht weiter, mit einem veränderten RANGE Header (je nach dem, wie groß der im Zwischenspeicher vorgehaltene Clip war). Der Teil welcher bereits im Proxy gecached ist, muss nicht mehr vom Server geliefert werden.
Zum Beenden der Verbindung sendet der Client eine TEARDOWN Nachricht an den Server, welche mit geänderten Header vom Proxy durchgereicht wird.

4.3 Ringbuffer

Ein Proxy Cache kann Anfragen seitens der Clients nach einen Stream nicht nur aus dem Cache sondern auch aus einem Ringbuffer bedienen. Erhält der Proxy Cache zum ersten Mal eine Anfrage nach einem speziellen Stream von einem Client, so fordert er diesen vom Server an. Gleichzeitig legt der Proxy Cache einen Ringbuffer im

Hauptspeicher an und hinterlegt dort die nachgefragten Daten. Die Anfrage des Clients wird nun aus diesem Buffer bedient.
Trifft nun eine zweite Anfrage nach dem Stream bei dem Proxy Cache ein, so wird diese aus dem Ringbuffer bedient, so lange die temporäre Distanz zwischen der ersten und zweiten Anfrage nicht zu groß ist. Gemeint ist damit, dass die Vorhaltezeit der Daten im Buffer größer sein muss, als die Differenz zwischen den zweiten und ersten Anfragezeitpunkt. Liegt kein Ringbuffer mehr vor, muss er erneut angelegt werden. Die Daten dazu werden entweder vom Server angefordert oder wenn möglich aus dem eigenen Cache bezogen. Es muss dabei sichergestellt werden, dass es während der Wiedergabe zu keinem Bufferüberlauf oder zu einem Bufferunterlauf kommt. Dies kann für alle Clients sichergestellt werden, wenn es für den ersten Client und für den letzten Client garantiert wird. Um einen Bufferüberlauf zu verhindern, muss sich der Garbage Collector immer hinter dem letzten Client befinden. Einen Bufferunterlauf verhindert man, indem die Datenquelle immer vor der Anfrage des ersten Clients die Datenpakete anliefert.[11]

4.4 Qualitätsanpassung

Bei hierarchisch kodierten Streams kann man den Aspekt ausnutzen, dass diese aus verschiedenen Ebenen bestehen. Wobei die unterste Ebene den Stream in niedrigster Qualität speichert und alle folgenden Schichten eine Qualitätsverbesserung darstellen.[12]
Existiert nun eine Anfrage von einem Client nach einem Stream, welcher im Cache noch nicht vorliegt, wird dieser vom Proxy angefordert und an den Client weitergeleitet. Dabei speichert der Proxy den Stream in seinen Cache zwischen. Durch Übertragungsfehler und Schwankungen in der Bandbreite zwischen Proxy und Server fehlen mitunter ganze, höhere Schichten bzw. einzelne Pakete der unteren Schichten. Bei einer zweiten Anfrage von einem anderen Client können diese fehlenden Stücke für die gewünschte Qualität erneut angefordert werden. Erst dieser Client profitiert also von der Qualitätsverbesserung.

[11] Vgl. Bommaiah, Guo, Hofmann, Paul /Design and Implementation of a Caching System/
[12] Vgl. Rejaie, Yu, Handley, Estrin /Multimedia Proxy Caching Mechanism/

Nun ist es jedoch auch möglich, dass dieser Client eine höhere Wiedergabebandbreite hat, als der im Cache vorgehaltene Stream. Der Proxy verlangt nun vom Server die fehlenden Schichten zur Anpassung der Qualität an die Wiedergabebandbreite des neuen Clients. Die neuen Schichten werden ebenfalls zwischengespeichert, so dass bei mehreren Anfragen die Qualität nach und nach steigt.

5. Ausgewählte Beispiele des Caching bei Peer to Peer

Beim Caching in Peer to Peer Netzwerken gelten veränderte Herausforderung im Gegensatz zum Caching von Multimedia Daten. Den Aspekt der strengen zeitlichen Anforderung kann man hier weitgehend außer Acht lassen. Bei dem Austausch von Daten in Peer to Peer Netzwerken, ist es zum Beispiel weniger relevant ob ein Datenpaket aufgrund von Paketverlusten mit Verzögerrungen empfangen werden kann. Empfinden wir es bei einem Videostream als störend, wenn der Film deswegen ruckelt, so wird es bei dem Peer to Peer kaum wahrgenommen wenn die Daten mit wenigen Sekunden Verzögerung eintreffen.
Als problematisch stellt sich jedoch die Tatsache der fehlenden Contiunität der Verfügbarkeit der Quelldaten heraus. So ist es leicht möglich dass ein Peer, von welchem man sein Zielobjekt, bezieht offline geht.

5.1 PeerCache

Bei PeerCache handelt es sich um ein Produkt der Firma Joltid. PeerCache wird bei dem Provider in das Netzwerk integriert, und sichert die am stärksten frequentierten Peer to Peer Daten vorübergehend. Laut Angaben der Firma soll den P2P hostenden Internet-Service-Providern (ISP) bis zu 25% weniger Datentransfer zugemutet und den Clients eine schnellere Downloadrate garantiert werden. Die Software wird also direkt auf einem Server beim Internet-Provider installiert. Sie fängt alle P2P-Downloads ab, um zu überprüfen, ob sie vielleicht schon zwischengespeichert sind. Wenn mehrere lokale

Nutzer die gleichen Daten vom Server anfordern, reduziert die Software das Datenvolumen zu entfernt gelegenen Netzwerksegmenten stark. Problematisch ist zu erkennen welche Teile des Datenstroms aus Internet-Downloads wirklich vorgehalten werden müssen. Das Problem dabei ist, dass Peer to Peer Datenverkehr oft getarnt ist. Leider ist aufgrund der Aktualität dieses Problems nicht zu ergründen wie dies hier gelöst wurde.

5.2 Freenet

Bei Freenet werden Dateien über binäre Schlüssel identifiziert, die durch Anwendung einer Hash-Funktion erzeugt werden. Bei einer Anfrage seitens eines Peers, nach dieser Datei werden naheliegende Nodes kontaktiert. Ist diese dort nicht enthalten wird die Suche auf die nächstliegenden erweitert. Dies passiert so lange bis das Suchobjekt gefunden wurde (bzw. eine timeout erscheint).
Die gesuchten Daten werden nun an den Peer gesendet. Jeder Node, welcher zwischen dem Quellrechner und den suchenden Peer liegt, speichert die Daten zwischen. Wenn diese Daten längere Zeit nicht nachgefragt werden, werden sie wieder gelöscht. Bei erneuten Anfragen werden sie jedoch weiter vorgehalten.
So kommt es zu einer Spezialisierung des Netzes. Intention ist es, dass sich meist die Interessensgruppen für bestimmte Informationen auf einen Gebiet ballen. Durch ein Zwischenspeichern der Daten auf Nodes, die zwischen Sender und Empfänger liegen wird eine Verschiebung der vorgehaltenen Daten, hin zum Empfänger realisert.
Weiter entfernt liegende Nodes die ebenfalls diese Information zwischen gespeichert haben, löschen sie nach einem gewissen Zeitfenster wieder. Schließlich bleibt das gecachte Objekt nur auf den Nodes liegen, welche oft Anfragen nach den Daten haben. Diese sind in der Regel die am schnellsten zu erreichenden Nodes.

6. Fazit

Die bereits Angesprochne Entwicklung hin zu einen höheren Bedarf an Übertragungsbandbreite und einer besseren Übertragungsqualität, kann nur bedingt durch einen besseren Ausbau der Netze begegnet werden. Die Anzahl der Breitbandanschlüsse bei Privatpersonen steigt zwar ständig, jedoch kann dieser Fakt das Problem nicht lösen. Im Gegenteil, verstärkt sich dadurch die Anfrage nach Angeboten wie zum Beispiel Livestreams, Video on Demand und Audiostreams.
Der Ansatz der Caching Strategien gehen in die richtige Richtung. Speziell beim Proxy Caching wird die Netzlast stark verringert. Existieren ohne Proxy Caches viele parallele Anfragen an den Streamingserver, welche das Netz unnötig belasten und die der Server nur mit kostenintensiver, leistungsfähiger Hardware befriedigen kann ist es mit dem Proxy Cache möglich, Traffic und Kosten einzusparen. Die derzeitige Trefferquote von 50% bei Proxy Caches verspricht sicher noch eine Steigerung in diesem Bereich. Neue intelligente Algorithmen (wie zum Beispiel Staging Hot Video Only, Largest Bandwidth Reduction Ratio First) beim cachen und löschen von Segmenten könnten da Abhilfe schaffen. Auch die verbesserte Funktionalität durch das Caching sollte man nicht außer betracht lassen. Aufgrund dieser Tatsachen wird das Caching bei Multimediastreams, sich bald ähnlich stark durchsetzen, wie bei dem statischen Cachen von Internetseiten. 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 Caching Produkte in ihren Portfolios.

Auch das Cachen bei Peer to Peer Netzwerken scheint dringend Notwendig. Es wird geschätzt das 70% des Internettraffics durch solche Netze verursacht wird (Tendenz steigend!). Konzepte wie das Peer Caching sind interessante Lösungen für dieses Problem. Solche Ansätze können sicherlich Verbesserung versprechen, allerdings darf man hier die rechtliche Seite nicht aus den Augen verlieren. So gibt es bereits jetzt Bedenken wegen dem Uhrheberrecht und bereits installierte Systeme wurden aus Angst vor Klagen wieder deaktiviert. Bei der Entwicklung neuerer Ansätze muss man

deswegen eng mit juristischen Experten zusammenarbeiten um nicht an den Gesetzen vorbei zu entwickeln.

7.Literaturverzeichnis

Bommaiah, Guo, Hofmann, Paul /Design and Implementation of a Caching System/
Ethendranath Bommaiah, Katherine Guo, Markus Hofmann, 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, Mai2000

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

Hofmann, Sbnani /Streaming and Broadcasting over the Internet/ Proceedings of the IEEE ATM 2000
Markus Hofmann, Krishan Sbnani: Streaming and Broadcasting over the Internet.
In: Proceedings of the IEEE ATM 2000, Heidelberg, Deutschland, Juni 2000.

OECD /The development of Broadband access in OECD countries/
Organisation für Wirtschaftliche Zusammenarbeit und Entwicklung: Statistische Veröffentlichungen

Olsen, Fiutak / Streaming-Markt soll bis 2007 auf 4,5 Milliarden Dollar wachsen /

Stefanie Olsen, Martin Fiutak: Streaming-Markt soll bis 2007 auf 4,5 Milliarden Dollar wachsen
http://www.zdnet.de/news/business/0,39023142,39115951,00.htm
Abruf: 2004-07-04

Rejaie, Yu, Handley, Estrin /Multimedia Proxy Caching/
Reza Rejaie, Haobo Yu, Mark Handley, 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

Schulzrinne, Rao, Lanphier /Real Time Streaming Protocol/ Internet Request for Comments
H. Schulzrinne, S. Casner, R. Frederick, V. Jacobsen. RTP: A Transport Protocol for Real-Time Applications. Januar 1996

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