Seminar

„Streamingtechnologien für Videodaten“

Universität Dortmund
Lehrstuhl Informatik I

Wintersemester 1999/2000

Betreuer:

Prof. Dr. Gisbert Dittrich
Dipl.-Inform. Jörg Westbomke

Teilnehmer:

Daniel Herche
Malte Hülder
Stefan Michaelis
Oliver Pauls
Jörg Röhrig
Björn Somberg
Konstantin Steuer
Dirk Vleugels
Gerd Zellmer

Vorwort

Elektronische Dokumente werden immer häufiger durch zeitbasierte Medienobjekte wie z.B. Videoclips angereichert. Selbst im WWW sind mittlerweile Videos komfortabel durch Streamingtechnologie verfügbar.

Die Streamingtechnologie ermöglicht es dabei, nach recht kurzer Pufferzeit auch lange Videos, an beliebigen Stellen beginnend, „direkt aus dem Netz heraus“ zu nutzen. Ein vollständiges Herunterladen des gesamten, zu betrachtenden Videos wird damit unnötig. Diese Funktionalität dürfte die Verwendbarkeit von Videos im Netz wesentlich steigern.

Dieses Seminar sollte Studierenden im Hauptstudium des Fachbereichs Informatik einen Überblick über die grundlegenden Techniken in diesem Bereich vermitteln und gleichzeitig existierende Softwarelösungen vorstellen.

Nach einer Einführung in das Themengebiet sowie der Darstellung von möglichen Einsatzgebieten wurden die Grundlagen der Datenübertragung im Internet erarbeitet. Aufbauend auf diesem Grundwissen wurden in Form von zwei Vorträgen die wesentlichen Spezifikationen (RTSP/RTP/RTCP) zur Übertragung von Videodaten per Streamingtechnologie erläutert. Die folgenden Vorträge stellen dann existierende Softwareprodukte vor, die den Einsatz von Videos im Internet per Streamingtechnologie ermöglichten.

Dortmund, den 18. Januar 2000

G. Dittrich J. Westbomke

Inhaltsverzeichnis

Überblick über Anwendungsfelder für Streamingtechnologien für Videodaten

Malte Hülder

FB Informatik, Uni Dortmund
malte@huelder.com

Zusammenfassung

Da dies die Ausarbeitung zum ersten Vortrag in der Seminarreihe ist, wird hier im Wesentlichen eine Einführung in das Thema gegeben. Da Streamingtechnologien ein zukunftsträchtiges Feld sind, wird zunächst auf die Charakteristika und die grundlegende Funktionsweise von Streaming eingegangen.

Weiterhin werden verschiedene Anwendungsfelder für Streamingtechnologien vorgestellt. Dabei wird nicht unbedingt unterschieden, ob es bereits Streaminganwendungen in diesem Gebiet gibt, oder ob es sich nur um Überlegungen handelt, dass Streaming in diesem Anwendungsfeld zum Einsatz kommen könnte.

Abschließend werden noch einmal die Anforderungen zusammengefasst, die Streaming an Netzwerkumgebung und Computer stellt. Da es sich hier nur um eine einleitende Übersicht handelt, werden verschiedene technische Details nich tiefergehend behandelt, sondern nur kurz umrissen, um zu einem späteren Zeitpunkt erläutert zu werden.

1 Charakteristika von Streamingtechnologien

Mit Streamingtechnologien können speicherintensive Daten wie z.B. Audio- und Videodaten übertragen werden. Gerade bei dieser Art von Daten, ist es wichtig, dass sie zusammenhängend und in der richtigen Reihenfolge beim Empfänger ankommen. Ein Tondokument kann zum Beispiel nur verstanden werden, wenn die einzelnen Laute in der richtigen Reihenfolge und flüssig hintereinander abgespielt werden. Das Netzwerkprotokoll IP (Internet Protocol) ist allerdings ein verbindungsloses und paketorientiertes Protokoll, das nicht nur die richtige Reihenfolge nicht garantieren kann, sondern auch nicht sicherstellen kann, dass die Daten überhaupt beim Empfänger ankommen. Streamingprotokolle, die auf IP aufsetzen, müssen sich also der Herausforderung stellen, eine solche gesicherte Übertragung zu ermöglichen.

Die Daten, die bei Streaminganwendungen übertragen werden, können entweder gespeichert auf einem Server zum Abruf bereit liegen oder *live* übertragen werden. Die letzte Variante ist besonders bemerkenswert, da hier ja bei Beginn der Übertragung weder die Länge, noch irgendein Dateiende feststeht. Live-Übertragungen ermöglichen es außerdem, kleinen Sendern oder Privatleuten Daten weltweit zu übertragen. Auf der anderen Seite ermöglichen sie es Nutzern, Informationen zu empfangen, die für sie sonst nur mit erheblich größerem Aufwand oder gar nicht erreichbar wären.

Streamingtechnologien können also den Kostenaufwand für Hardware reduzieren. Insbesondere muss ein Empfänger keinen Speicherplatz haben, um sich zum Beispiel einen ganzen Videofilm abzuspeichern, um diesen sehen zu können. Durch die Streamingtechnologie werden die Daten übertragen und praktisch sofort auf dem Monitor oder einem anderen Ausgabegerät dargestellt, ohne gespeichert zu werden. Mit entsprechenden Sicherheitsverfahren gekoppelt, bietet sich dieses Verfahren auch zur Sicherung von Urheberrechten an. Wenn ein Nutzer keine Daten speichern muss, ist es auch denkbar, dass er diese Daten auch gar nicht speichern kann. So würde es unmöglich, illegale Kopien oder Mitschnitte der Daten anzufertigen.

Wenn gerade davon gesprochen wurde, dass Streamingdaten nicht gespeichert werden, dann ist das natürlich nur zum Teil richtig. Die Daten, die über das Netzwerk ankommen, werden zur Verarbeitung in einem Puffer gespeichert. Dadurch liegt natürlich immer ein gewisser Teil der Daten beim Nutzer - im Vergleich zu den Gesamtdaten ist dies allerdings verschwindend gering. Diese Pufferung kann aber genutzt werden um Schwankungen in der Übertragungsqualität zu kompensieren. So kommen die Daten immer etwas früher an, als sie tatsächlich verarbeitet werden müssen. Sollten sich aufgrund von Schwankungen in der Netzauslastung einzelne Pakete verspäten, können sie so wieder in die richtige Reihenfolge gebracht werden, bevor diese Daten dann widergegeben werden müssen. Dies gilt auch für Pakete, die verloren gehen, so dass ihnen eine gewisse Zeit bleibt, um eine Neuübertragung zu ermöglichen.

2 Der Ablauf einer Streaming-Sitzung

Die zu empfangenden Daten müssen vom Absender auf einem Server bereitgestellt werden. Der Nutzer, der diese Daten empfangen möchte, startet ein Clientprogramm und gibt an, welche Daten er empfangen möchte. Das Clientprogramm baut eine Verbindung zum angegebenen Server auf, wobei es das Streamingprotokoll benutzt. Über dieses Streamingprotokoll antwortet der Server und schickt die Daten nach und nach zum Client. Die ersten ankommenden Daten werden in einen Pufferspeicher geschrieben, der relativ klein sein kann. Wenn dieser Puffer voll ist, wird mit der Widergabe begonnen. Die ersten Daten werden vom Anfang des Puffers ausgelesen und entfernt, neue Daten werden werden am Ende in diesen Puffer eingefügt. Wenn zwischenzeitlich die Netzkapazität etwas nachläßt, kann die Widergabe aufrecht erhalten werden, bis der Pufferspeicher leer ist. Wenn der Kapazitätseinbruch aber nur kurzzeitig war, dann kann der Puffer rechtzeitig wieder aufgefüllt werden und der Nutzer bekommt von diesen Schwankungen nichts mit.

Wenn der Nutzer z.B. in einem Videofilm an eine andere Stellen "spulen" will, dann kann er einfach die Stelle suchen, zu der er springen möchte. Das Clientprogramm verwirft die alten, gepufferten Daten und fordert die neuen an, die dieser Stelle im Film entsprechen. Die neuen Daten werden wieder gepuffert und widergegeben, sobald der Pufferspeicher voll ist. Dadurch benötigt ein Sprung innerhalb eines Films immer konstante Zeit, unabhängig davon, wie weit dieser Sprung ist. In einem Live-Stream ist es natürlich nicht möglich, hin und her zu springen, da die zukünftigen Daten ja noch gar nicht existieren und vergangene auch nicht gespeichert werden.

Zum Ende der Sitzung wird die Verbindung wieder abgebaut. Das kann passieren, weil die Widergabe der Daten am Ende angekommen ist, oder weil der Benutzer die Widergabe persönlich stoppt.

3 Anwendungsfelder für Streaming

Im folgenden sollen einige Anwendungsfelder für Streamingtechnologien erläutert werden. Dabei gibt es für einige Anwendungen bereits tatsächliche Umsetzungen, andere scheinen nur eine gute Anwendung für Streamingtechnologie zu sein, ohne dass dort bereits ein brauchbares Produkt verfügbar wäre.

3.1 Videoüberwachung

Durch starken Preisverfall wird die Verbreitung von Videokameras gefördert. Dies nicht nur aber im Besonderen auch für Kameras, die an einen handelsüblichen Computer angeschlossen werden können. Noch sind - zumindest im Bereich der "IBM-kompatiblen Computer" - Rechner mit einem Eingang für Videoquellen nicht der Standard, aber auch hier ermöglichen sinkende Preise vielen Benutzern einen entsprechend ausgestatteten Computer.

Die bereits "klassischen" WebCams liefern ein regelmäßig aktualisiertes Standbild von den Vorgängen im Einzugsbereich der Kamera. Mit dieser Technik sind Übertragungen von wenigen Bildern in der Minute möglich. Durch den Einsatz von Streamingtechnologien kann die Übertragungsrate deutlich höher ausfallen. Ein entsprechender Live-Stream benötigt weniger Speicherplatz als die Bilder einer WebCam, die ja in regelmäßigen Abständen in eine Internetseite eingebunden und auf dem entsprechenden Server gespeichert werden müssen.

Mit Hilfe dieser Technik und der Verbreitung durch das Internet ist es von nahezu überall möglich, Büroräume oder Privatwohnungen zu überwachen. Dazu kommen die extrem günstigen Verbindungskosten, die bei Verwendung der Internetübertragung anfallen.

3.2 Bildtelefonie

Um von der Videoüberwachung zur Bildtelefonie zu gelangen, ist weitere Multimedia-Hardware notwendig. Allerdings ist eine Soundkarte bei den meisten Computermodellen heute bereits Standard und auch Mikrofon und Lautsprecher oder Kopfhörer sind preisgünstig zu bekommen.

Natürlich gibt es hier bereits einige Anwendungen, die nicht auf Streamingtechnologie basieren, z.B. iVisit, NetMeeting oder die MBone Tools, die im Abschnitt 3.6 erläutert werden.

Die klassische Übertragung mittels ISDN-Leitungen ist besonders bei Überseeleitungen um ein Vielfaches teurer, als die bei der Verwendung von Streamingtechnologie zu Grunde liegende Internetverbindung. Allerdings muss man dafür das für Streaming typische Zwischenpuffern in Kauf nehmen und dementsprechend mit gewissen Verzögerungen rechnen. Dafür kann man aber flüssige, bewegte Bilder erhoffen. Wenn kleinere Zwischenpuffer nötig werden, könnte sich diese Verzögerungszeit auch weiter reduzieren.

3.3 Vide Conferencing

Von der Bildtelefonie ist die Erweiterung zum Videokonferenzsystem nur noch ein kleiner Schritt. Bei einer Konferenz müssen allerdings alle Video- und Audiodaten an alle Konferenzteilnehmer verteilt werden. Dies führt natürlich zu einem exponentiellen Anstieg der zu übertragenden Daten, wenn mehrere Teilnehmer zu einer Konferenz dazustoßen. Üblicherweise wird ein zentraler Knoten benutzt, der die Daten an alle Teilnehmer verteilt.

Die Übertragung über ISDN-Leitungen ist natürlich teurer als Internetverbindungen, besonders da hier ja für mehrere Teilnehmer Kosten für Ferngespräche anfallen können. Beim Einsatz von Streaming-

technologie ist natürlich auch wieder mit Verzögerungen durch das Zwischenpuffern zu rechnen. n-wender müßten sich vermutlich an einen neuen Kommunikationsstil gewöhnen, da es immer einig Zeit dauert, bis ein Gesprächspartner antwortet. Die geringen Kosten bei möglicherweise besserer Qualität und die weltweite Verfügbarkeit des Internets sollten aber genug Anreiz für eine Entwicklung auf diesem Gebiet geben.

Ein Beispiel für ein Internet Videokonferenzsystem, dass allerdings nicht auf Streaming beruht, sind die in Abschnitt 3.6 besprochenen MBone Tools.

3.4 Pressekonferenzen

Pressekonferenzen stellen ein Beispiel für die genannten Streaming-Anwendungen dar. Ähnlich einer anderen Live-Übertragung von bestimmten Ereignissen z.B. durch das Fernsehen, ist eine schnell Verbreitung der Informationen möglich. Durch Nutzung des Internets können Interessenten auf der ganzen Welt erreicht werden.

In Verbindung mit eine Videokonferenzsystem können nicht nur Informationen weitergegeben werden, sondern auch eine Rückkopplung und Nachfragen der Empfänger ist möglich. Viele klein Presseorgane können es sich nicht leisten, ihre Korrespondenten und Reporter überall auf der Welt zu verteilen. Mit einer solchen Technik wäre es somit auch diese Presseorgane möglich, eine eigene B e-richterstattung zu gewährleisten und Interviews mit entfernten Personen zu führen. Dies wäre auch ein Schritt zur Wahrung oder gar Ausweitung der Pressefreiheit.

3.5 CD- und Videovorstellungen (Werbung)

Werbung kann ein weiteres Beispiel sein. So wäre es denkbar, dass Plattenfirmen in Zukunft neue Produkte im Internet bewerben werden und einzelne Titel oder sogar ein ganzen Musikalbum zu Probe hören angeboten werden wird. Das gleiche kann natürlich auch mit Kinofilmen oder Videos gemacht werden. Durch die Streamingtechnologie werden die Anforderungen an den Computer des Konsumenten niedrig gehalten und evtl. eine illegale Kopie der Daten verhindert.

Wenn zusätzlich noch sichere Zahlungssysteme für das Internet vorliegen, ist es den Kunden sogar möglich, die getesteten Produkte auch sofort zu bestellen und zu bezahlen.

3.6 MBone Tools

Die Abkürzung MBone steht für "Multicast Backbone", also einen Teil des Internets, de Multicasting unterstützt. Die Idee de Multicasting besteht darin, gleiche Daten nicht mehrfach über dieselben Leitungen zu schicken. Wenn z.B. ein Teilnehmer in den USA an einer Videokonferenz mit mehreren E uropäern teilnimmt, dann muss sein Video- und Audiosignal an alle Teilnehmer in Europa über ein Transatlantikleitung geschickt werden. Beim Multicasting werden hingegen die Daten nicht für jeden Teilnehmer einzeln über di Transatlantikleitung geschickt, sondern nur einmalig und dann von eine Router in Europa an die Teilnehmer verteilt. Multicasting beruht also auf einer virtuellen Baumstru k-tur (siehe Abbildung 3-1).

Für Multicasting sind Internet-Adressen der Klasse D (224.0.0.0 bis 239.255.255.255) reserviert. Die Daten werden also vom Sender nicht direkt an den Empfänger gesendet, sondern an eine Multicast-Adresse. Die Empfänger müssen sich vorher für die Konferenz anmelden. Die Router wissen dann, dass eine Kopie der Daten für die entsprechende Multicast-Adresse an den angemeldeten Empfänger weitergeleitet werden soll.

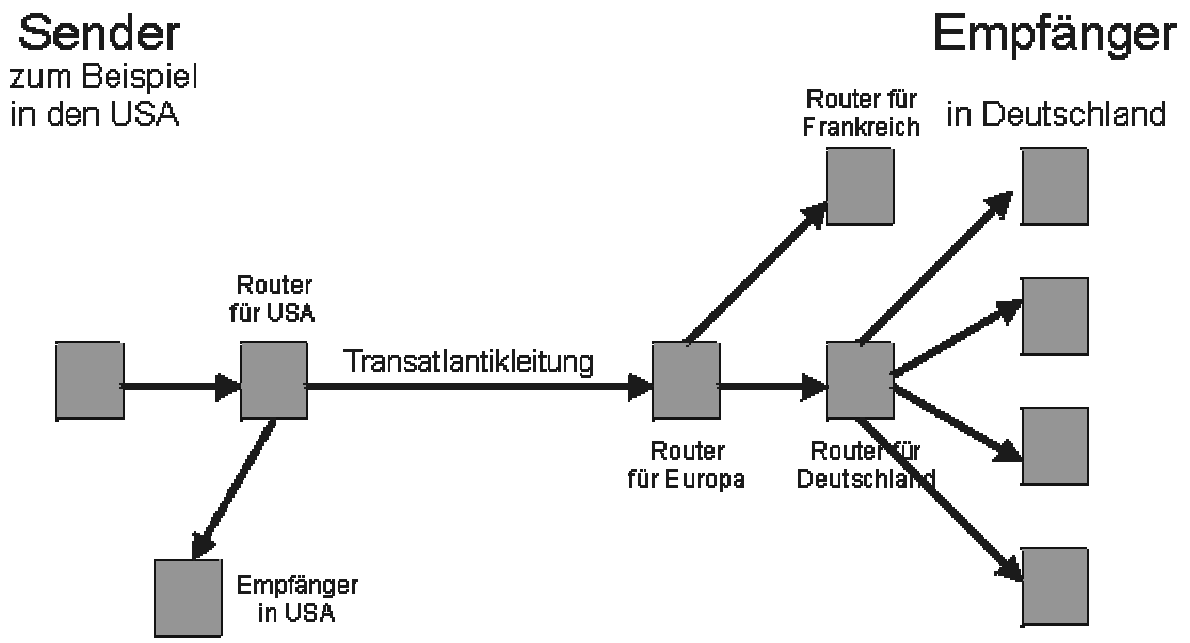


Abbildung 3-1 virtuelle Baumstruktur bei Multicasting

Für die Nutzung de MBone stehen verschiedene Programme zur Verfügung, die auch für diverse Plattformen angeboten werden. Im Folgenden werden die wichtigsten UNIX-Tools erläutert. Bei allen Programmen gibt es auch die Möglichkeit, die Datenübertragung mittels DES (Data Encription Standard) zu verschlüsseln. So können auch Konferenzen abgehalten werden, bei denen die Daten nur b estimmten Leuten (die den Schlüssel kennen) zugänglich sind.

Eine Konferenz besteht immer aus der gleichzeitigen Nutzung mehrerer der folgenden Programme.

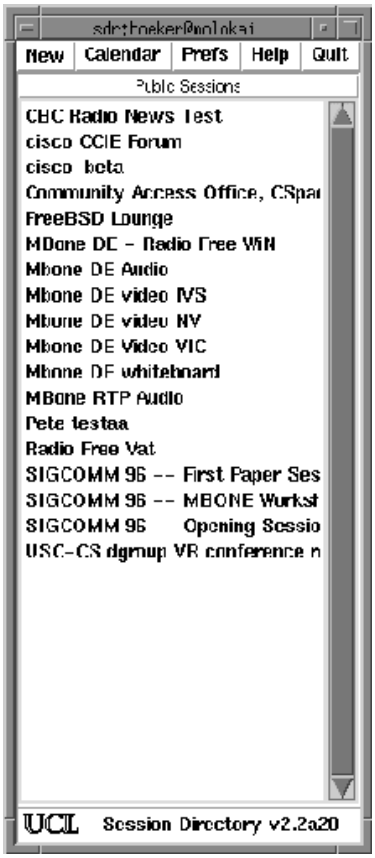
3.6.1 sdr = Session Directory

Das Session Directory enthält alle Informationen über angekündigte Konferenzen im MBone. Alle Konferenzen müssen z.B. mittels `sdr` angekündigt werden und bei Beendigung auch wieder aus dem Verzeichnis gelöscht werden. Mit dem Programm `sdr` kann man nicht nur Konferenzen anlegen und löschen sondern auch an bereits angekündigten Konferenzen teilnehmen.

Beim Start von `sdr` erhält man das Hauptfenster (Abbildung 3-2). Im Hauptfenster werden die bereits angekündigten Konferenzen angezeigt. Man kann mit dem Menüpunkt *New* eine neue Konferenz ankündigen oder durch Klicken auf einen Eintrag mehr Informationen über diese Konferenz abrufen. Im letzten Fall erscheint das Fenster Session Information (Abbildung 3-3).

In diesem Fenster erhält man genauere Informationen über Inhalt, Ziel und Zweck der Konferenz, über den Initiator der Konferenz und die Zeitdauer, für die die Konferenz angesetzt ist. Außerdem werden die Dienste, Formate und Multicast-Adressen angegeben, die dieser Konferenz zugeordnet sind. Durch einen Klick auf den *audio*-Button wird zum Beispiel das Audio Tool `vat` (siehe 3.6.2) gestartet. Durch Klicken auf den Button *Start All* werden alle verknüpften Dienste (hier: `vat`, `vic` und `wb`) gestartet. Beim Start eines oder mehrerer Dienste nimmt man automatisch an der Konferenz teil. Mit dem *Invite*-Button können noch andere Teilnehmer zur Konferenz eingeladen werden. Voraussetzung ist, das diese auch `sdr` gestartet haben. Mit *Dismiss* kann man dieses Fenster schließlich wieder schließen.

Beim Anlegen einer neuen Konferenz muss man diese ganzen Informationen natürlich mit angeben. In dem entsprechenden Fenster (ohne Abb.) gibt man insbesondere auch die verschiedenen Anwendu ngen an, die bei dieser Konferenz benutzt werden sollen.



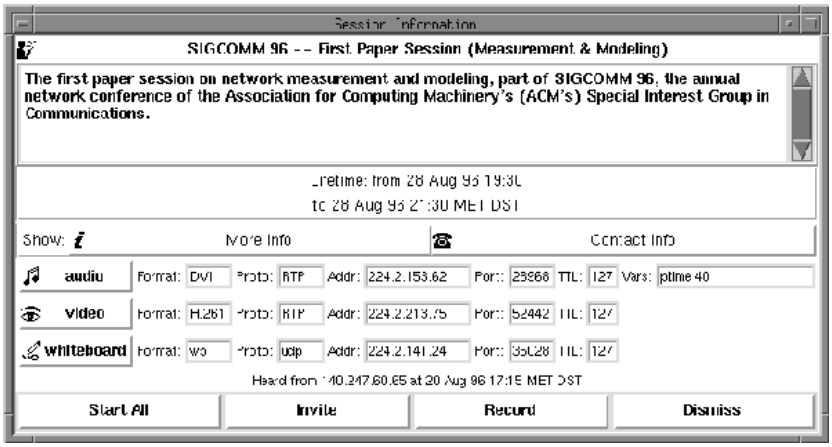


Abbildung 3-2 sdr Hauptfenster **Abbildung 3-3 sdr Session Information**

3.6.2 vat = Visual Audio Tool

Das Visual Audio Tool `vat` dient zur Übertragung von Sprache. Dabei kann es Telefonqualität err i-chen, allerdings ermöglicht es im Gegensatz zum normalen Telefon auch Gespräche mit mehreren Teilnehmern.

Im Hauptfenster von `vat` (Abbildung 3-4) werden alle Gesprächsteilnehmer aufgelistet. Es besteht di Möglichkeit Teilnehmer stumm zu schalten (angekreuzt, Teilnehmer 1 und 8). Teilnehmer, zu denen die Verbindung gestört ist werden ausgegraut (Teilnehmer 6 und 9) und gerade aktiv sprechende Tei l-nehmer werden mit einem Punkt und farbig hervorgehoben (Teilnehmer 10).

Über den *Menu*-Button kann man verschiedene Einstellungen zur Verbindung vornehmen, insbeso n-dere kann zwischen verschiedenen Audio-Formaten gewählt werden (PCM, DVI, GSM, LPC). Im Hauptfenster befinden sich neben der Teilnehmerliste noch die Aussteuerungsanzeige und die Lau t-stärkeregelung für Lautsprecher und Mikrofon. Um Rückkopplungen zu vermeiden, biete `vat` neben Vollduplex auch noch zwei Halbduplex-Modi an. In diesen Fällen schaltet die Nutzung des Mikrofons die Lautsprecher ab oder Töne aus den Lautsprechern verhindern eine Nutzung des Mikrofons.

3.6.3 vic = Video Conferencing Tool

Zur Bildübertragung bei einer Videokonferenz wird das Video Conferencing Tool `vic` benutzt. Ähn-lich wie bei `vat` werden die Teilnehmer im Hauptfenster (Abb. 3-5) angezeigt. Hier bekommt jeder Teilnehmer ein kleines Fenster, in dem seine Videodaten dargestellt werden. Bei Klick auf ein solches Fenster kann dieses Video in einem größeren Fenster angesehen werden. Es gibt auch die Möglichkeit, Teilnehmer "stumm" zu schalten, dann bleibt das letzte Bild dieses Teilnehmers stehen und weitere Bilder werden nicht mehr empfangen.

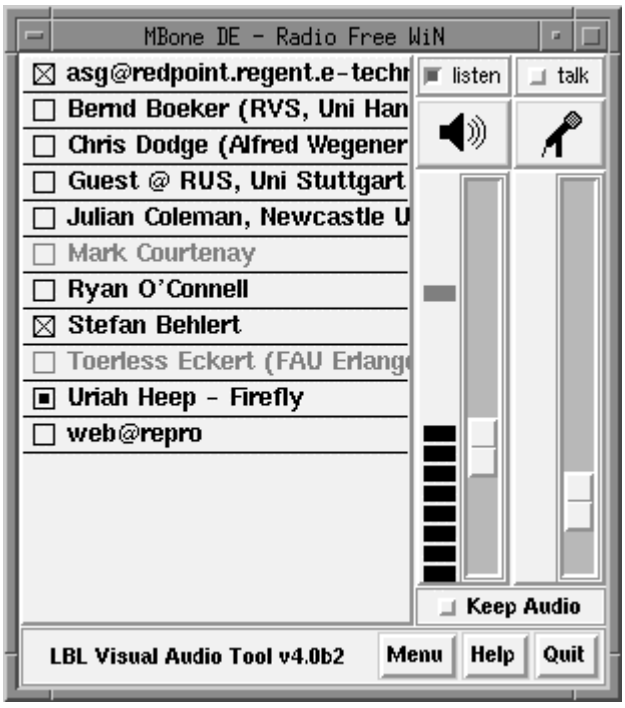


Abbildung 3-4 vat Hauptfenster

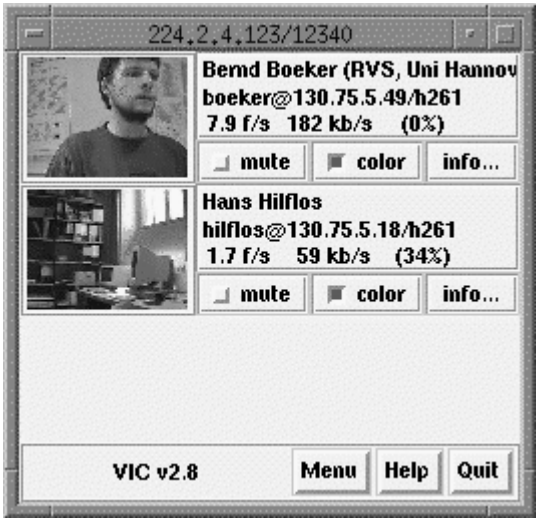


Abbildung 3-5 vic Hauptfenster

Über den *Menu*-Button kann wie bei `vat` verschiedene Einstellungen ändern, hier stehen auch wieder verschiedene Daten-Formate zur Auswahl (JPEG, H261, nv). Wird i nv-Modus gesendet, kann der Empfänger das Video auch in dem Programm `nv` anzeigen lassen.

Es gibt außerdem noch die Möglichkeit, sich Informationen über die Verbindungseigenschaften mit dem Knopf *info...* anzusehen. Bei guten Bedingungen kann `vic` bis zu 30 Bilder pro Sekunde übertr a-gen, in der Praxis liegt dieser Wert jedoch meistens unter 5 Bildern p. Sekunde.

3.6.4 wb = White Board

Das White Board (**Fehler! Verweisquelle konnte nicht gefunden werden.**) bietet die Möglichkeit, mit mehreren Teilnehmern einer Konferenz an einem Zeichenbrett oder einer Tafel zu arbeiten. Di Funktionen, die von `wb` zur Verfügung gestellt werden können, entsprechen denen, die man auch in einfachen Grafikprogrammen hat. Mit einfachen Zeichenstiften, Kästen, Kreisen und verschiedenen Farben kann man Illustrationen auf die Zeichenfläche malen. Über ein Textwerkzeug kann man auch Texte mit verschiedenen Schriften bearbeiten. Weiterhin stellt `wb` Funktionen zum Import von Text- und Postscript-Dateien zur Verfügung.

Wenn man `wb` für Vorträge oder Lehrveranstaltungen benutzen möchte, wäre es ungünstig, wenn alle Teilnehmer auf der Tafel herummalen könnten. Deshalb sollte man das White Board schreibgeschützt machen, dann können alle den Erklärungen des Vortragenden auch durch dessen Zeichnungen folgen, aber nur dieser kann etwas in `wb` zeichnen.

3.6.5 nt = Network Text Editor

Der Network Text Editor `nt` (Abbildung 3-7) bietet eine ähnliche Funktionalität wie `wb`, allerdings kann in `nt` nicht gezeichnet, sondern ausschließlich Text bearbeitet werden. Dazu wird der Text in Textblöcke aufgeteilt, die von allen Teilnehmern gleichzeitig interaktiv bearbeitet werden können. Allerdings dürfen zwei Teilnehmer nicht gleichzeitig dieselbe Zeile editieren.

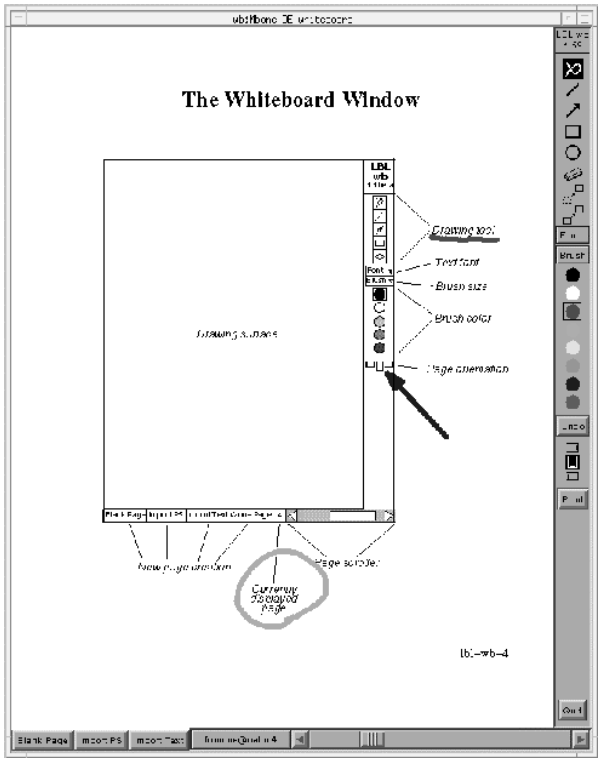


Abbildung 3-6 Das White-Board

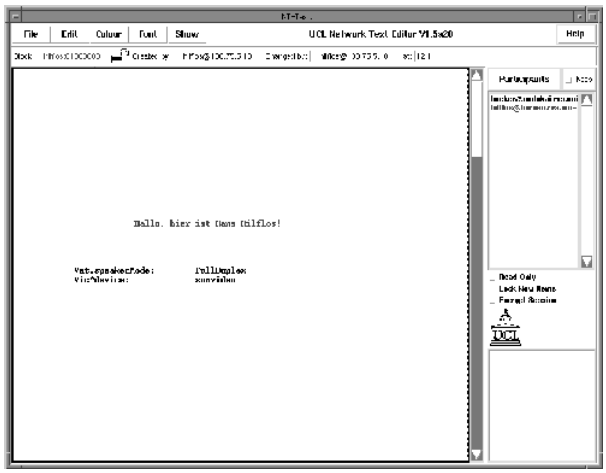


Abbildung 3-7 Der Network Text Editor

3.7 Vide on Demand

Vide on Demand ist ein Schlagwort, dass schon seit einigen Jahren immer wider in Verbindung mit digitalem Fernsehen und Pay per View fällt. Es bedeutet so viel wie Video auf Abruf. Nutzer eines Vide on Demand Dienstes können sich aus einem bestimmten Angebot zu einer beliebigen Zeit ein Video auswählen und es ansehen. Damit ist man völlig unabhängig vom Fernsehprogramm und Ang ebot und Vorstellungszeiten im Kino. Video on Demand lässt sich am ehesten noch mit dem Gang zu einer Videothek vergleichen, der alledings auch wieder nur zu bestimmten Öffnungszeiten möglich ist.

Das Internet kann durch die Streamingtechnologie eine neue Dimension für Video on Demand bringen. So können Filme weltweit auf Servern angeboten und ebenfalls weltweit von Nutzern abgerufen werden. Da die meisten Server 24 Stunden am Tag online sind, gibt es auch keine Probleme mit Öffnungszeiten mehr.

Um Video- bzw. Kinofilme über das Internet zu sehen muss natürlich auch noch das Problem des r heberrechts gelöst werden. Schließlich ist es nicht Interesse der Filmindustrie, wenn jedermann die Filme kostenlos betrachten und evtl. auch kopieren kann.

Insgesamt bietet Streaming Video im Internet einige Vorteile zum klassischen Kauf- oder Leihvideo:

- Keine Öffnungszeiten
- Weltweite Verfügbarkeit, d.h. es ist auch möglich ausländische Filme zu sehen, die sonst im eigenen Land nicht auf den Markt kämen.
- Streaming Video erlaubt beliebige Sprünge innerhalb eines Films, ohne spulen zu müssen, d.h. konstante Sprungzeiten unabhängig von der Sprungweite

Lediglich der Speicherplatz der Server und die Netzbandbreite schränken das Angebot sowie den Z ugriff noch ein. In Zeiten sinkender Kosten für Speicherplatten, ist es nicht mehr undenkbar ganze Kinofilme auf einem Server zum Abruf bereitzulegen. Das Problem der Netzbandbreite läßt sich aller-

dings nicht ohne Weiteres lösen. Die Investition in leistungsfähigere Netzwerkverbindungen ist teue und zeitaufwendig. Bei dem derzeitigen Wachstum der Internetnutzer wird auch die Belastung der einzelnen Verbindungen immer größer. Deshalb muss der Ausbau der Netzkapazität in erster Linie der Beibehaltung heutiger Zugriffsgeschwindigkeiten dienen und kann erst in zweiter Linie diese dann s ogar ausbauen.

Beispiele für Streaming Video on Demand sind:

ein Filmbericht zum 36. Todestag von J.F. Kennedy

http://www.necnews.com/vidload/?vidname=/1999/11/22/0700lc02

oder die Nikolausvorlesung 1999 am Fachbereich Informatik der Uni Dortmund

http://stl-www.cs.uni-dortmund.de/Multimedia/Niko.ra

3.8 Pay per View

Pay per View ist ein weiteres Stichwort, dass gerne im Zusammenhang mit Video on Demand und digitalem Fernsehen genannt wird. Pay per View bedeutet, dass jede Sendung einzeln bezahlt werden muss. Dadurch bezahlt der Anwender nur das, was er auch wirklich sehen will und sieht. Pay per View stellt somit das Gegenteil zu klassischen Rundfunkgebühren dar, bei denen alle Teilnehmer den gleichen Betrag bezahlen, unabhängig davon, wieviel und welche Sendungen sie tatsächlich sehen.

In Verbindung mit Video on Demand und Internet-Streamingtechnologien könnte man sich eine Art "virtuell Videothek" oder ein "virtuelles Kino" vorstellen. Der Konsument besucht die Internetseiten des Videothekbetreibers und wählt einen Film aus, den er gerne sehen möchte. Gegen Zahlung eines Entgeltes wird ihm der Film auf dem Streamingserver freigeschaltet.

Um eine solche Idee zu verwirklichen sind natürlich einfache und sicherere Zahlungssysteme für das Internet notwendig. Desweiteren muss die Kapazität des Betreiberservers entsprechend viele Anfragen bearbeiten können und die Netzwerkleistung muss sicherstellen, dass die Daten beim Kunden zuverlässig ankommen. Wenn der Kunde für den Dienst bezahlt, erwartet er natürlich auch, dass er ein qualitative Gegenleistung bekommt, d.h. ein ruckelfreies Bild in guter Auflösung und Schärfe.

3.9 Digitales Fernsehe

Bereits in weniger als zehn Jahren (2008) wird das analoge Fernsehen in der heutigen Form abg eschafft. Ein solch gravierender Schritt muss verschiedene Gründe haben:

i.) durch das digitale Übertragungsverfahren wird Bandbreite gespart, daher sind in Zukunft mehr ere digitale Kanäle möglich, wo bisher nur ein analoger Kanal übertragen werden konnte

ii.) durch die vielen privaten Rundfunkanstalten ist die Nachfrage nach Übertragungskanälen extre gestiegen

iii.) durch die digitale Übertragung ist eine bessere Bild- und Tonqualität möglich

iv.) die Einführung von Video on Demand und Pay per View wird gefördert

Für das digitale Fernsehen sind speziell Dekoder notwendig oder spezielle Dekoder, die anstelle des herkömmlichen analogen Programmtuners einen entsprechenden Dekoder bereits eingebaut haben. Solche Dekoder gibt es bereits heute als proprietäre Standards der Kirchgruppe mit den Sendern DF1 und Premiere World. Im Gegenzug soll es den offenen Standard der MHP (Multimedia Home Plattform) geben, die voraussichtlich im Februar 2000 verabschiedet wird. Auf der Basis der MHP entwi kkelt die „Deutsche TV Plattform“ aus ARD, ZDF, RTL , SAT1, Deutsche Telekom, Nokia und Bundeswirtschaftsministerium ein Dekoderbox, die ab Herbst 2000 v rfügbar sein soll.

Anstelle des Dekoders für den Fernseher ist es natürlich auch denkbar, mit dem Computer solche digitalen Programme zu empfangen. Als Dekoder kommt dann eine spezielle Software zum Einsatz, die das Signal umwandelt und direkt auf dem Monitor anzeigen kann. Dadurch wird der mobile Empfang von Fernsehprogrammen wesentlich vereinfacht, da dazu dann ein einfacher Laptop zum Einsatz kommen könnte.

Eine andere Form von digitalem Fernsehen stellt Streaming Video dar. So ist es natürlich auch möglich, Nachrichten oder andere Sendungen live an meinem Computer zu sehen, die von irgendwo auf der Welt eingespielt werden.

Ein Beispiel dafür ist ein Live-Stream vom BBC WorldService:

http://news.bbc.co.uk/olmedia/video/now45.ra

3.10 Radio- und Fernsehsender für das Internet

Durch die Möglichkeit, ihre Sendungen als Live-Stream über das Internet zu verbreiten, gibt es Radio- und Fernsehsender, die auf diese Weise weltweit empfangen werden können. Weil die benötigte Hardware sehr preiswert ist und Programmplätze im Kabel, Satellit oder terrestrischer Antenne sehr knapp sind, ist es sogar möglich, dass sich Radio- oder Fernsehstationen bilden, die gar nicht auf herkömmlichem Wege zu empfangen sind. Mit geringeren Kosten und geringerem Aufwand erreichen solche Stationen eine größere, nämlich weltweite Verbreitung.

Aufgrund der heutigen technischen Bedingungen sind aber alle diese heutigen Streamingtechnologien noch nicht von einer so hohen Qualität, dass sie eine echte Alternative zu herkömmlichen Übertragungstechniken darstellen.

4 Zusammenfassung

Streamingtechnologien ermöglichen eine Übertragung von speicherintesiven Daten z.B. für Audio- und Videoanwendungen, ohne dabei zu große Anforderungen an die Empfänger zu stellen. Durch Pufferspeicher wird auch die schwankende Netzlast aufgefangen. Durch das universelle Prinzip und die verschiedenen Auftretensformen von Audio- und Videodaten sind auch sehr viele Anwendungen für Streaming denkbar.

Allerdings sind die bis heute entwickelten Techniken noch nicht ausreichend. Die Bilder bei Videoübertragung sind extrem klein und unscharf. Trotzdem kommt es immer wieder zu Ruckeln und Ausfällen, weil die Netzbandbreite nicht ausreicht, um die entsprechende Menge an Daten zu liefern. Lediglich bei reinen Audio-Streaming Anwendungen kann man sich mit Telefonqualität vorerst zufrieden geben.

Um die Leistung von Streamingtechniken zu verbessern, muss man verschiedene Anforderungen erfüllen:

- Gute Kompressionsverfahren müssen entwickelt werden, da ein Echtfarbenbild mit Fernsehauflösung (768 x 576 Pixel) bereits 1,3 MByte Speicherplatz belegt. Bei 25 Bildern/Sekunde macht dies ca. 32,5 MByte/Sek. bzw. 260 Mbit/Sek. Damit sind bereits schnell Intranetleitungen (100 MBit) überfordert, ein ISDN-B-Kanal schafft gerade einmal 8 KByte/Sek.
- Zusätzlich zur Kompression müssen die Datenraten der Netzwerke erhöht werden, da eine Kompression um einen Faktor von über 4000 nicht realistisch erscheint. Selbst bei einer ISDN-Anbindung erhält man oft nicht die volle mögliche Übertragungsrate, weil viele Netze überlastet sind.

- Durch Multicasting könnte die allgemeine Netzlast gesenkt werden, da gleiche Daten nicht unnötigerweise mehrfach durch das gleiche Netzsegment geschickt werden.
- Es muss Möglichkeiten geben, die verfügbaren Datenraten konstant zu halten. Dann kann sich di Streamingsoftware mit Ihrer Puffergröße an der verfügbaren Rate orientieren und notfalls qualitativ schlechtere Kompressionsverfahren nutzen, um wenigstens die Funktionalität in gewisse Maße zu erhalten.
- Die Übertragung muss zuverlässiger werden. Bei Paketverlustraten von bis zu 50 % kann man keine brauchbare Qualität mehr erwarten. Natürlich hängt ein großer Teil der Verluste auch mit der Überlastung der Netze zusammen. Dadurch kommen vereinzelte Pakete zu spät und müssen verworfen werden.

In IPv6 sind zwei weitere Aspekte berücksichtigt, die bei der Lösung der genannten Probleme helfen können:

- Qualit of Service: Mit diesem Feature soll es möglich sein, bestimmten Daten oder Nutzern Vorrang einzuräumen. Es könnte dann die Möglichkeit geben, verschiedene Internetverträge anzubieten, wobei durch höhere Gebühren auch eine bessere Dienstqualität erkauft wird. Es könnten auch zeitkritische Daten, wie z.B. Streaming-Daten, an Routern Vorfahrt vor weniger kritischen Daten bekommen, um möglichst kurze Verzögerungen zu gewährleisten.
- Statische Routen: Ein Prinzip des Internets ist es, Daten von einem Router zum nächsten zu schicken, bis sie ihr Ziel erreicht haben. Manchmal stehen aber mehrer Router und somit mehrere verschiedene Wege zum Ziel zur Auswahl. Unter Umständen nehmen zwei Pakete dann verschiedene Wege und benötigen unterschiedlich lange Zeit - eine falsche Paketreihenfolge und Paketverlust können die Folge sein. Mit statischen Routen kann der Weg, den ein Paket zum Ziel nimmt, genau festgelegt werden.

5 Literaturverzeichnis

[Brau 80] Brauer, W. (Ed.):
Net Theory and Applications
LNCS Vol. 84, Springer Verlag 1980

[Brau 84] Brauer, W. : How to play the token game
Newsletter 16, SIG "Petri Nets and Related System Models", p. 3-13, 1984

[BrRR 87a] Brauer, W. - Reisig, W. - Rozenberg, G. (Eds.):
Petri Nets: Central Models and their Properties
LNCS Vol. 254, Springer Verlag 1987

[Dünh 98] Dünhölter, K.:
http://members.aol.com/xmldoku/syntax.htm [Stand: 4.11.1998]

Grundlegende Netzwerkdienste/ -protokolle

Jörg Röhrig

FB Informatik, Uni Dortmund
J.Roehrig@uni-duisburg.de

Zusammenfassung

Diese Seminarausarbeitung soll dazu dienen, die grundlegenden Kenntnisse der Datenübertragung im Internet zu vermitteln.

Dazu wird als erstes die Entwicklung des Internets betrachtet, wobei die Entstehungsgeschichte des Netzes von besonderem Interesse ist. Da dieser Sachverhalt für den strukturellen Aufbau des heutigen Internets verantwortlich ist und somit auch für die Art der Datenübertragung.

Danach werden die verschiedenen Übertragungsprotokolle des Internets, wie IP, TCP und UDP vorgestellt und deren Arbeitsweise erläutert. Die Hauptziele dieser Betrachtung sind das Verständnis des theoretischen Aufbaus dieser Protokolle und die daraus resultierende paketorientierte Sichtweise des Internets.

Als nächstes wird das sogenannte “Routing” behandelt, also wie die Datenpakete im Internet ihren Weg von einem Sender zum Empfänger finden. In diesem Zusammenhang wird auch auf die unterschiedlichen IP-Klassen und Netzwerkarchitekturen, wie LAN, MAN und WAN, eingegangen. Desweiteren wird der Router, der für die eigentliche Weiterleitung der Datenpakete im Internet verantwortlich ist, kurz vorgestellt.

Zum Abschluß der Ausarbeitung wird noch ein Blick in die nahe Zukunft des Internets geworfen. Die sogenannte MBone-Technologie (Multicast-Backbone), soll dem Internet den Weg für weltweites Video eröffnen.

1 Wie alles begann

Der Grundstein des heutigen Internets wurde bereits im Jahre 1968 gelegt. Das US–Verteidigungsministerium, „Department of Defence“ (DoD), vergab an die „Advanced Research Projects Agency“ (ARPA) den Auftrag, ein Computernetzwerk zu entwickeln, das unverwundbar gegen feindliche Angriffe sein sollte. Das Netzwerk sollte auch dann noch weiterarbeiten, wenn ganze Teile, z.B. durch eine Bombe oder Sabotage zerstört wurden.

Bereits Ende 1969 entwickelte sich aus diesem Bestreben das ARPANet, welches im Anfangsstadium lediglich aus 4 Knotenrechnern bestand, den sogenannten „Interface Messages Processors“ (IMP). Diese IMP´s (heute: Router) waren untereinander mit einer 50kbps Leitung verbunden, wie es die folgende Abbildung zeigt.

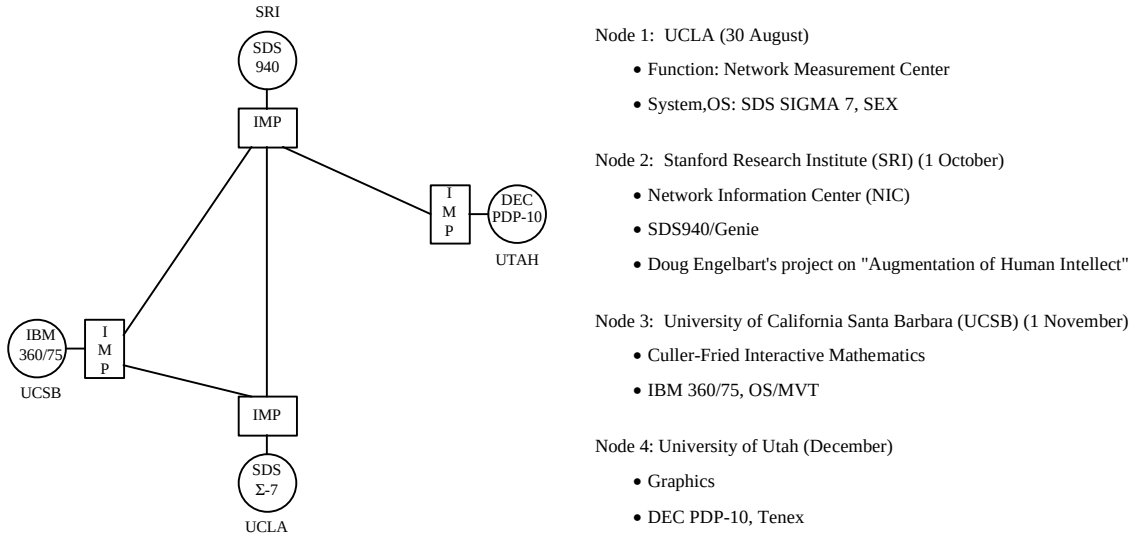


Abbildung 1-1: ARPANet Dezember 1969 [Grom 99]/[Zako 99]

Als Netzwerkprotokoll kam das „Network Control Protocol“ (NCP) zum Einsatz, das erste Host-to-Host Protokoll.

Im Laufe der folgenden Jahre stieg die Zahl der Rechner im ARPANet rasant an, zumindest für damalige Verhältnisse, so daß im Jahre 1972, als das ARPANet auf der „International Conferences on Computer Communications“ (ICCC) der Öffentlichkeit vorgestellt wurde, dieses bereits aus 40 IMP´s bestand.

Im Jahre 1973 wurde an der Standford University das „Internet-Programm“ gegründet. Ziel dieser Internet Working Group war es, die Grundsätze zur Verbindung unabhängiger Netzwerke zu erarbeiten. Durch die stark anwachsende Rechnerzahl im ARPANet wurde sehr schnell klar, daß das NCP am Ende seiner Leistungsfähigkeit angekommen war. Aus diesem Grunde begannen Bob Kahn (DARPA) und Vincent Cerf (Standford University) im Jahre 1973 mit dem Entwurf eines Net-to-Net Verbindungsprotokolls, welchem sie den Namen „Transmission Control Protocol“ (TCP) gaben. Mit diesem Protokoll wurde erstmals die Möglichkeit geschaffen, eine Host-to-Host Verbindung, über das lokale Netzwerk hinaus, zu einem Rechner in einem anderen Netzwerk aufzubauen. Durch die Entwicklung dieses Protokolls konnten nun verschiedene andere Netze mit dem ARPANet verbunden werden, wie in Abbildung 1-2 zu sehen ist.

Den großen Durchbruch schaffte TCP jedoch erst im Jahr 1983, da viele wissenschaftliche Einrichtungen Unix-Rechner verwandten und mit dem Erscheinen des damals neuen Unix 4.2 BSD (Berkley System Distribution) stand TCP erstmals kostenlos zur Verfügung. Dieser Umstand hatte zur Folge, daß sich TCP nun sehr schnell verbreitete und de facto zum Standard erklärt wurde.

Kurz darauf kam auch in die Netzstruktur Bewegung. Das ARPANet, das inzwischen unter die Schirmherrschaft der „Defence Communications Agency“ (DCA) gestellt worden war, wurde aufgeteilt in einen militärischen Teil, das MILNet und das weiterhin öffentlich zugängliche ARPANet. Die beiden Netze wurden mittels Gateways von einander getrennt, um das MILNet vor Angriffen aus dem ARPANet zu schützen. Diese Gateways werden in Abb. 1-2 durch eine senkrechte Linie symbolisiert.

Mitte der 80er Jahre schlossen sich immer mehr Netzwerke an das ARPANet an, wie z.B. das NSFNet der „National Science Foundation“. Womit dann das Zeitalter des Internets begann.

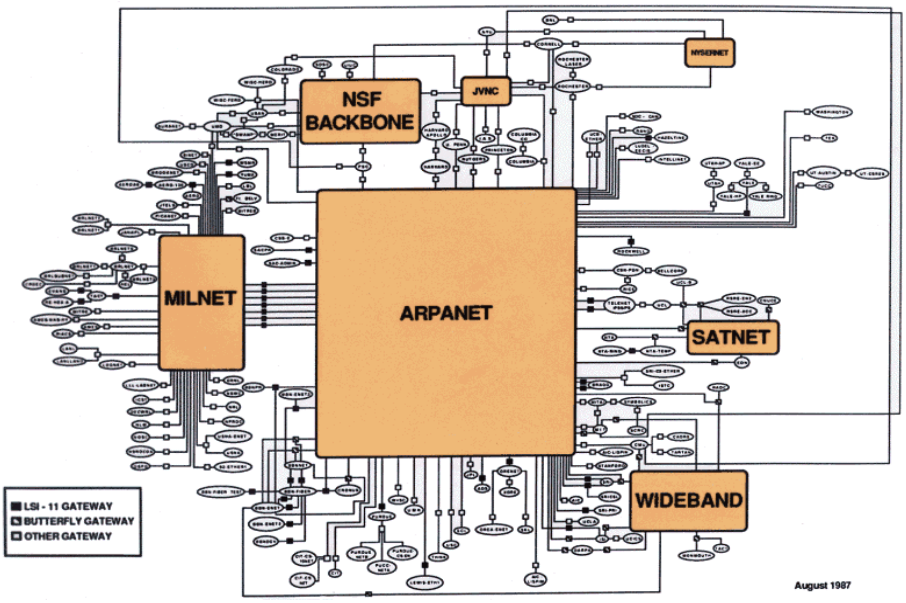


Abbildung 1-2: ARPANet August 1987 [Dodg 99]

2 Die Internetprotokolle

2.1 Die Schichtmodelle

2.1.1 ISO- OSI Referenzmodell

Das OSI-Schichtmodell (Open System Interconnection) der „International Standards Organisation“ (ISO) wird zur Beschreibung der Funktionsweise bzw. zur Realisierung von Netzwerkprotokollen verwendet. Wie in Abbildung 2-1 zu sehen ist, umfaßt das Modell insgesamt 7 Schichten (Layers).

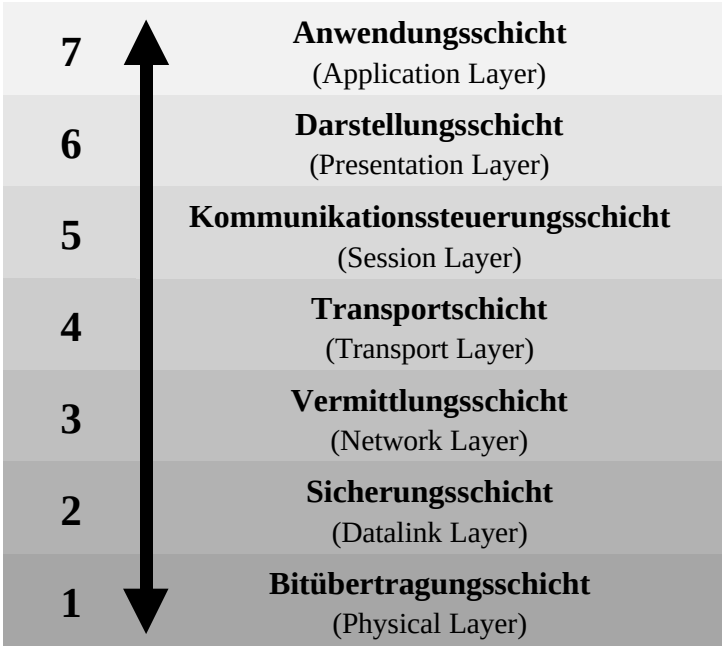


Abbildung 2-1: ISO-OSI Schichtmodell [Nico 96]

Durch die hierarchische Anordnung der Schichten im OSI-Modell wird eine Reduktion der Komplexität der Kommunikation erreicht. Jede Schicht bietet eine Dienstleistung an, die vollkommen unabhängig von der Implementation der übrigen Schichten ist. Einzig die Schnittstellendefinition, Import- und Exportparameter, müssen den benachbarten Schichten bekannt sein.

Soll also ein Datenpaket an einen Rechner im Netzwerk übermittelt werden, durchläuft das Paket beginnend mit der Anwendungsschicht (7) nach einander alle Schichten bis zur Bitübertragungsschicht (1). Auf der Seite des empfangenden Rechners erfolgt dann die selbe Prozedur, mit dem Unterschied, daß die Schichten diesmal in umgekehrter Reihenfolge durchlaufen werden.

Zwischen je zwei Schichten (Sender- und Empfängerschicht) besteht auf jeder Ebene eine Art Pseudoverbindung. Zum Beispiel versieht die Sicherungsschicht des Senders die zu übertragenden Daten mit zusätzlichen Bits zur Fehlererkennung. In der entsprechenden Schicht des Empfängers werden diese Bits dann ausgewertet und von den eigentlichen Daten wieder getrennt.

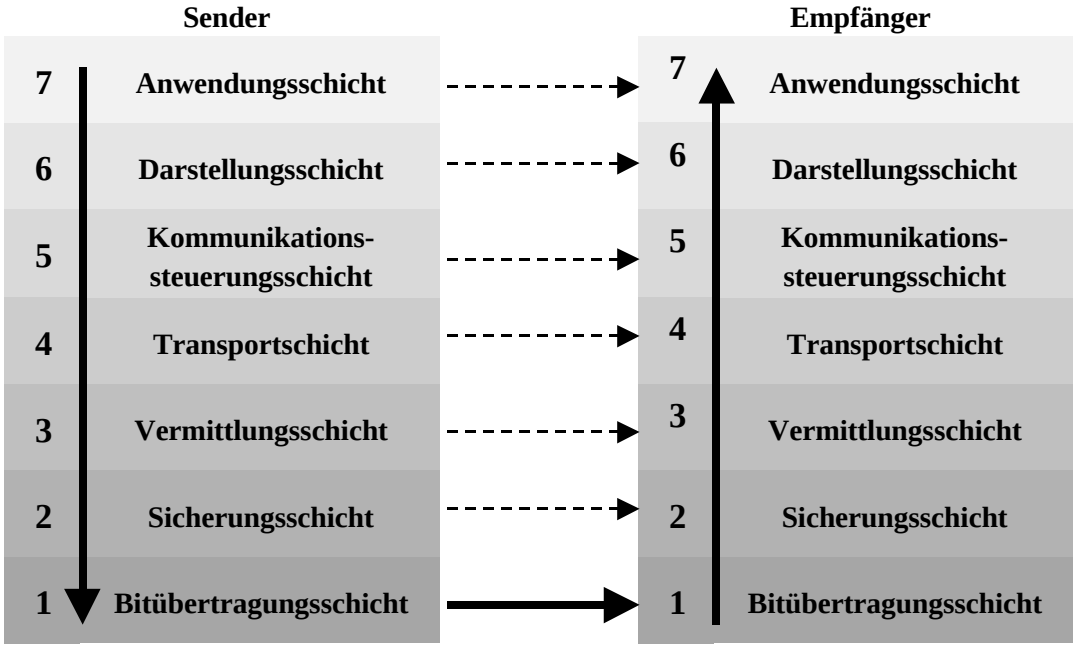


Abbildung 2-2: Arbeitsweise des OSI-Modells

Die Dienste der jeweiligen Schichten im OSI-Modell sind folgendermaßen definiert.

(1) **Bitübertragungsschicht:**
Definiert die physikalischen Eigenschaften der Übertragungswege, wie z.B. Telefonkabel, Koaxkabel oder Lichtwellenleiter usw.

(2) **Sicherungsschicht:**
Diese Schicht soll für eine fehlerfreie Übertragung der über die physikalische Verbindung empfangenen/gesendeten Daten sorgen. Ihre Aufgabe besteht also in der Erkennung und Vermeidung von fehlerhaften Datenpaketen.

(3) **Vermittlungsschicht:**
Dient zur Verwaltung von Verbindungen zwischen den höheren Schichten und den Rechnern im Netzwerk. Diese Verbindungen können dabei entweder verbindungslos oder verbindungsorientiert sein. Desweiteren kümmert sich diese Schicht um die Wegwahl (Routing) im Netz.

(4) **Transportschicht:**
Diese Schicht ist für die Qualität der Verbindung (Quality of Service, QoS) verantwortlich, wie z.B. Datendurchsatz, Fehlerwahrscheinlichkeit usw.

(5) **Kommunikationssteuerungsschicht:**
In dieser Schicht wird der Dialog zwischen zwei Anwendungen verwaltet, dazu gehört unter anderem der Auf- bzw. Abbau von Verbindungen, aber auch die Art des Dialogs (voll- oder halbduplex).

(6) **Darstellungsschicht:**
Diese Schicht ist für die Syntax der Daten zuständig, wie Format (Komprimierung) und die Art der Kodierung (Zeichensatz, Verschlüsselung)

(7) **Anwendungsschicht:**
Dient zur Definition der Kommunikationsschnittstellen für die Anwendungen, die auf diese Schicht zugreifen um das Netz zu nutzen. Meist ist die Anwendungsschicht jedoch herstellerspezifisch.

2.1.2 Das TCP/IP Modell

Der Vergleich des im letzten Abschnitt vorgestellten ISO-OSI Modells mit dem TCP/IP Schichtmodell zeigt einige Unterschiede auf. Diese sind hauptsächlich durch die Tatsache begründet, daß zur Entwicklungszeit des TCP/IP Protokolls das OSI-Modell noch nicht existierte.

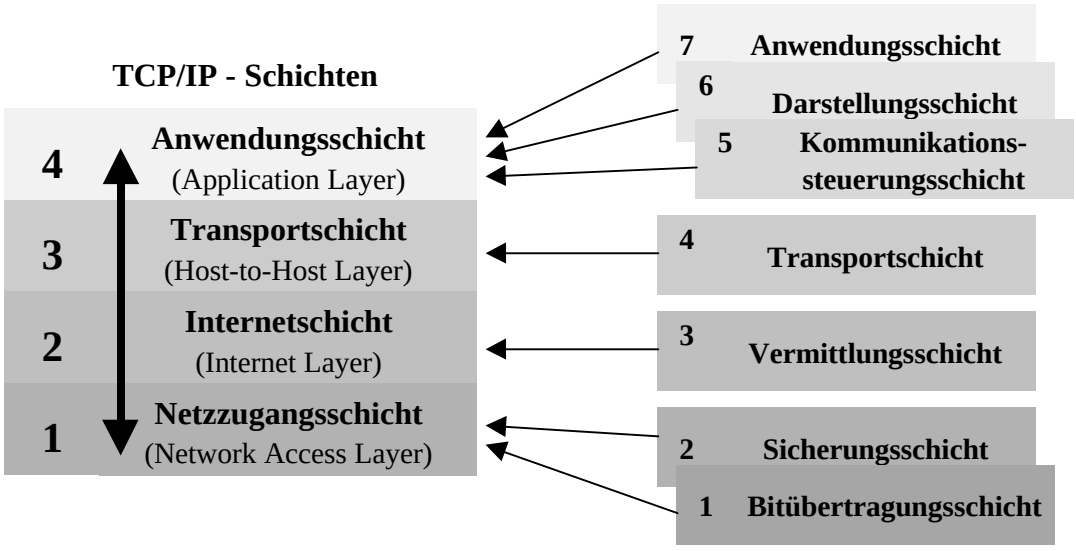


Abbildung 2-3: TCP/IP vs. ISO-Modell [Nico 96]

Wie in der Abbildung 2-3 zu sehen ist, enthält das theoretische TCP/IP-Schichtmodell im Gegensatz zum OSI-Modell nur 4 Schichten, da einige Schichten des ISO-Modells zusammenfallen.

(1) **Netzzugangsschicht:**
Diese Schicht ist für den Transport der IP-Datengramme durch das Netzwerk verantwortlich. Wie diese Übertragung im Einzelnen zu erfolgen hat, ist nicht genau spezifiziert. Zum Einsatz kommen Protokolle wie z.B. X.25 oder Ethernet.

(2) **Internetschicht:**
Zu den Aufgaben dieser Schicht, die das „Internet Protokoll“ (IP) übernimmt, gehört die Adressierung von Datenpaketen (Datengramme) und das Fragmentieren eines Datenpaketes in mehrere einzelne Datengramme, wenn deren Paketlänge die durch die „Maximum Transmission Unit“ (MTU) festgelegte Länge übersteigt.

(3) **Transportschicht:**
In dieser Schicht können zwei verschiedene Protokolle zum Einsatz kommen. Zum Einen ist dies das „Transmission Control Protocol“ (TCP), welches verbindungsorientiert ist und eine Fehlerkorrektur bietet. Zum Anderen kann das „User Datagram Protocol“ (UDP), welches verbindungslos ist und nur über eine Fehlererkennung verfügt, eingesetzt werden. Die Aufgabe ist jedoch in beiden Fällen die gleiche, es soll eine Host-to-Host (Ende-zu-Ende) aufgebaut werden.

(4) **Anwendungsschicht:**
Stellt die Anwendungssoftware bereit, wie z.B. FTP, TELNET, SMTP usw. und bildet somit die Schnittstelle zum Benutzer.

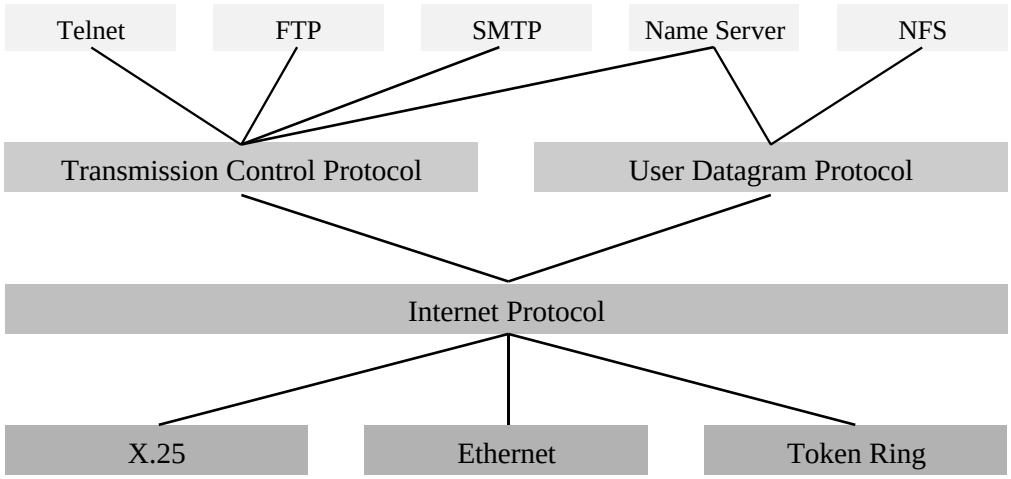


Abbildung 2-4: Architekturbeispiel [BrHo 95]

Die Arbeitsweise des TCP/IP-Schichtmodells läßt sich am einfachsten durch einen Vergleich mit der normalen Briefpost erklären. Sei also das Datenpaket, welches von einem Rechner zu einem anderen Rechner im Netzwerk übertragen werden soll, ein Brief, der von einem Absender zu einem Empfänger geschickt werden soll.

Der Absender, der den Brief verschicken will, versieht diesen mit der Adresse des Empfängers (**Anwendungsschicht**). Danach wirft er den Brief in einen Briefkasten und übergibt diesen somit der Post zum Transport (**Transportschicht**). Der Briefkasten wird von einem Postboten geleert und die Postsendungen nach Zielorten sortiert. Zum Transport wird der Brief dann an ein passendes Verkehrsmittel, z.B. die Bahn, übergeben (**Internetschicht)**. Bei diesem Transport kann es natürlich vorkommen, daß das Transportmittel mehrmals gewechselt wird (**Netzwerkschicht**). Am Zielort angekommen, werden alle Postsendungen vom Transportmittel abgeholt und an einen Postboten übergeben. Dieser sortiert die Sendungen und verteilt diese in die Fächer der entsprechenden Auslieferungsbezirke (**Internetschicht**). Der Zusteller, der für die Auslieferung des betrachteten Briefes verantwortlich ist, wirft diesen in den Briefkasten des Empfängers (**Transportschicht**). Der Empfänger, der inzwischen seinen Briefkasten geleert hat, kann nun den Brief lesen und gegebenenfalls darauf antworten (**Anwendungsschicht**).

2.2 Die Internetschicht

2.2.1 IP-Header

Die Hauptaufgabe des Internet-Protokolls ist die Adressierung und die Fragmentierung von Datenpaketen. Dazu wird der sogenannte IP-Header konstruiert und vor das eigentliche Datenpaket gehängt. Wie in Abbildung 2-5 zu sehen ist, besteht der IP-Header aus Datenfeldern, die alle zur Übermittlung des Datenpaketes erforderlichen Daten enthalten. In Analogie zur Briefpost, kann man den IP-Header mit einem Briefumschlag vergleichen.

Beim Internet-Protokoll handelt es sich um ein *verbindungsloses* Protokoll (connection less), das heißt, es wird keine Host-to-Host Verbindung aufgebaut. Die Datenpakete werden nur ins Netz geworfen und müssen ihren Weg selbst finden (Routing). Daher wird IP auch als *unzuverlässig* (unreliable) bezeichnet. Die fehlerfreie Übertragung zum Zielhost wird somit nicht garantiert. Dies ist jedoch kein Nachteil, wie man im ersten Augenblick annehmen könnte. Sondern hat den Vorteil, daß IP dadurch effizienter arbeitet. Daraus folgt aber, daß sich eine höhere Schicht um die Verläßlichkeit der Datenübertragung kümmern muß. Wie im Kapitel 2.3 zu sehen ist, übernimmt dies das TCP-Protokoll.

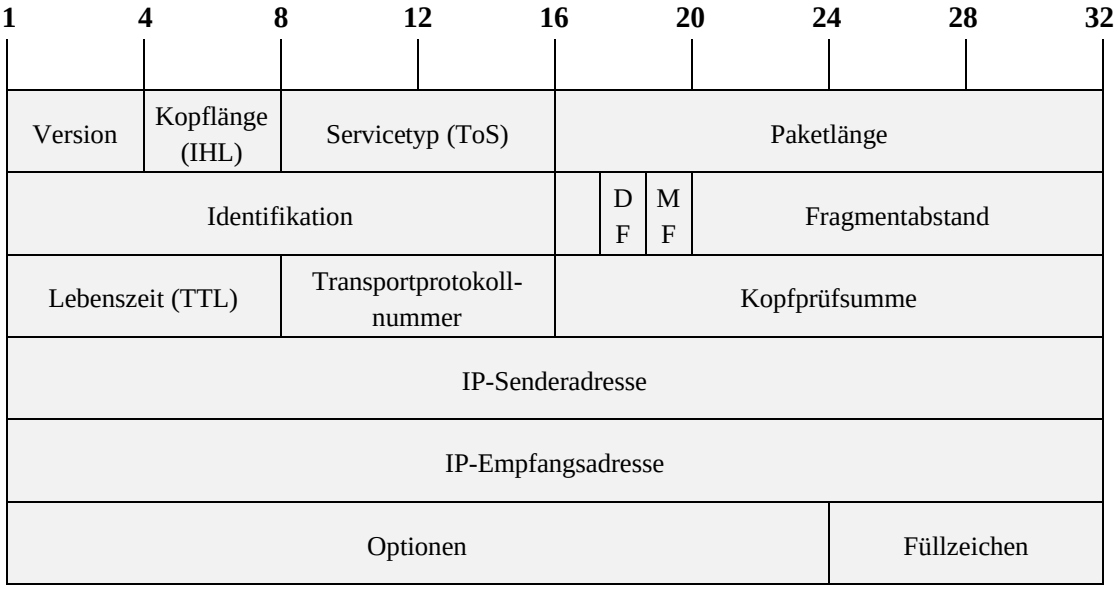


Abbildung 2-5: Der IP-Header [BrHo 95]

- **Version:**

  Dieses Feld gibt die Version des verwendeten IP-Protokolls an. Zur Zeit wird noch Version 4 benutzt, jedoch befindet sich die IP Version 6 schon in der Erprobung.

- **Kopflänge (Internet Header Length):**

  Gibt die Länge des Protokollkopfes (Header) an, da diese wegen des variablen Options-Feldes nicht konstant ist. Die Länge entspricht dabei der Anzahl von 32-Bit-Wörtern im Header. Die kleinste mögliche Länge ist somit 5, wenn es kein Options-Feld gibt. Der Maximalwert liegt bei 15, was insgesamt 60 Bytes entspricht.

- **Servicetyp (Type of Service):**

  In diesem Feld können spezielle Kriterien in Bezug auf die Zuverlässigkeit und die Geschwindigkeit angegeben werden.

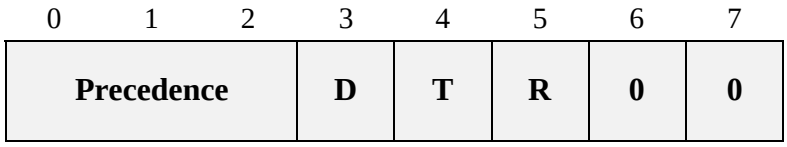


Abbildung 2-6: Servicetyp-Feld (ToS)

Die Bits des Servicetyp-Feldes haben folgende Bedeutung:

Bits 0-2: **Precedence** (Vorrang)

Bit 3: 0 = Normal **D**elay (Verzögerung) 1 = Low **D**elay

Bit 4: 0 = Normal **T**hroughput (Durchsatz) 1 = High **T**hroughput

Bit 5: 0 = Normal **R**eliability (Zuverlässigkeit) 1 = High **R**eliability

Bits 6-7: Reserviert für spätere Benutzung, daher immer 0

Precedence-Bits nach RFC 791

111 - Network Control	011 - Flash
110 - Internetwork Control	010 - Immediate
101 - CRITIC/ECP	001 - Priority
100 - Flash Override	000 - Routine

Standardmäßig (Default) werden alle 7 Bits auf Null gesetzt. Die Precedence-Bits geben die Dringlichkeit eines Datengramms an, die in 7 Stufen von 0 (Normal) bis 7 (Netzwerkkontrolle) reicht. Diese drei Bits werden jedoch nur vom US-Verteidigungsministerium (DoD) genutzt, so daß sie im Internet keine Rolle spielen. Die Bits 3-5 sollen dem Router bei der Wahl des richtigen Wegs zur Weiterleitung des Datengramms helfen. Jedoch werden auch diese Bits nur selten ausgewertet.

- **Paketlänge (Total Length):**

  Gibt die Gesamtlänge (Header + Daten) des Datengramms in Bytes an. Da das Feld eine Größe von 16-Bit hat, liegt die maximale Länge des Headers bei 2^{16}, was 65.535 Bytes entspricht. Jedoch ist dieser Wert nur als theoretisch anzusehen, da nach RFC 791 nur 576 Byte spezifiziert sind. Was aber nicht heißen soll, daß in der Praxis nicht auch mal längere Datenpakete verwendet werden.

- **Identifikation (Identification):**

  Anhand dieses Feldes kann ein Zielhost ein fragmentiertes Datengramm anderen Fragmenten des gleichen Datenpaketes zuordnen. Dabei bekommt jedes dieser Fragmente die gleiche Identifikationsnummer. Die Vergabe dieser Nummer übernimmt in der Regel die Transportschicht.

- **DF-/MF-Flag:**

  Beim gesetzten DF-Flag (Don´t Fragment) darf das Datenpaket auf keinen Fall fragmentiert werden. Dies kann aber zur Folge haben, daß Datengramme, deren Länge größer als 567 Bytes sind, unter Umständen nicht verarbeitet werden können und somit verworfen werden.

  Das Setzen des MF-Flags (More Fragments) teilt dem Zielhost mit, daß es sich bei dem Datengramm um ein Fragment handelt und daß noch weitere Fragmente folgen werden. Beim letzten

Fragment ist das MF-Bit nicht mehr gesetzt und signalisiert damit, daß alle Fragmente übermittelt wurden.

- **Fragmentabstand (Fragment Offset):**

  In diesem Feld wird jedem Fragment eines Datenpaketes eine fortlaufende Nummer zugeordnet. Damit wird es dem Zielhost ermöglicht, die Fragmente in der richtigen Reihenfolge zusammensetzen zu können. Auf Grund der Länge des Feldes (12 Bit) liegt die maximale Anzahl von Fragmenten bei 4096. Bei nicht fragmentierten Paketen wird grundsätzlich eine Null angegeben.

- **Lebenszeit (Time to Live):**

  Das Feld gibt die maximale Lebensdauer eines Datengramms an. Dies soll verhindern, daß unzustellbare Datengramme endlos in einem Netzwerk umherirren. Der Startwert beträgt 255 Sekunden. Bei jedem Erreichen eines Netzwerkknotens (Router) wird dieser Wert um mindestens 1 verringert werden, je nachdem wie lange das Datengramm zwischengespeichert wird. Daraus ergibt sich, daß ein Paket nach Erreichen von 255 Netzwerkknoten verworfen wird. Sollte dieser Fall eintreten, erhält der Sender eine Fehlermeldung in Form einer ICMP-Nachricht (Internet Control Message Protocol).

- **Transportprotokollnummer (Protocol):**

  Enthält die Nummer des Transportschichtprotokolls, an welches das Datenpaket weitergeleitet werden soll. Die Numerierung der Protokolle ist nach RFC 1700 einheitlich für das ganze Internet standardisiert, z.B. ICMP=1, TCP=6, UDP=17 usw.

- **Kopfprüfsumme (Header checksum):**

  In diesem Feld wird eine Prüfsumme gespeichert, die jedoch nur die Felder des IP-Headers berücksichtigt, die Nutzdaten des Datengrammes werden nicht geprüft. Da sich das Feld „Lebenszeit“ (Time to Live) bei jedem Netzwerkknoten (Router) verändert, muß auch die Prüfsumme jedesmal neu berechnet werden. Zur Berechnung der Prüfsumme wird das 1er-Komplement aller 16-Bit Halbworte addiert und aus der Summe wird dann wiederum das 1er-Komplement gebildet. Zur Erzeugung des 1er-Komplementes werden bei einer Dualzahl alle Stellen negiert und das höchste Bit als Vorzeichen interpretiert.

- **IP-Sendeadresse (Source Address):**

  Enthält die Internet-Adresse des Senders als 32-Bit Wort.

- **IP-Empfangsadresse (Destination Address):**

  Hier steht die Internet-Adresse des Empfängers.

- **Optionen (Options):**

  Das Optionen-Feld dient zur Aufnahme von zusätzlichen Informationen, z.B. um Anpassungen an die Anforderungen eines höheren Protokolls zu ermöglichen. Das hat den Vorteil, daß selten benutzte Optionen keinen unnötigen Speicherplatz im Header durch ein festes Feld in Anspruch nehmen. Außerdem können auf diese Art, jederzeit weitere Optionen in die Spezifikation des IP-Protokolls aufgenommen werden, ohne daß dessen Struktur verändert werden muß. Jede Option beginnt mit einem Byte, das Aufschluß darüber gibt, um welche Option es sich handelt. Danach kann noch ein weiteres Byte folgen, welches auch der Identifikation dient. Im Folgenden können auch noch ein oder mehrere Datenbytes für die Option angehängt sein. Zur Zeit sind folgende Optionen verfügbar:

  i.) *End of Option List*

  Kennzeichnet das Ende der Optionsliste.

ii.) No Option

Dient zum Auffüllen von Bits zwischen zwei Optionen.

iii.) Strict Source - Routing

Schreibt dem Datengramm genau vor, über welchen Weg es das Netzwerk zum Zielhost zu durchlaufen hat. Dieser Pfad wird durch die IP´s der Router vorgegeben.

iv.) Loose Source - Routing

Analog „Strikt Source – Routing“, jedoch ist es dem Datengramm zwischen den vorgeschriebenen IP`s auch noch erlaubt, andere Router zu durchlaufen.

v.) Record Route

Bewirkt, daß alle auf dem Pfad durchs Netzwerk durchlaufenden Router, ihre IP-Adresse an das Ende des Optionen-Feldes anhängen. Diese Funktion ist jedoch mit Vorsicht zu genießen, da, wie bereits erwähnt, die Länge des Feldes auf 40 Bytes begrenzt ist.

vi.) Timestamp

Arbeitet ähnlich der „Record Route“ Operation. Es wird jedoch neben der IP auch die Uhrzeit des Besuchs eines Routers gespeichert. Dadurch kann man feststellen, welche Zeit für die Übertragung des Datengramms benötigt wurde. Auch hier muß die Länge des Optionsfeldes berücksichtigt werden.

vii.) Security

Gibt an, wie geheim ein Datengramm einzustufen ist. Die Option ist hauptsächlich für das Militär gedacht, wird aber nur selten von Routern beachtet.

- **Füllzeichen (Padding):**

Da das Optionen-Feld eine variable Länge hat und die Länge des TCP-Headers immer ein vielfaches von 32 Bit sein muß, wird mittels Füllzeichen (Nullen) ausgeglichen.

2.3 Die Transportschicht

2.3.1 TCP (Transmission Control Protocol)

Das TCP Protokoll ist ein *verläßliches* (reliable) und *verbindungsorientiertes* (connection oriented) Protokoll. Es sorgt dafür, daß die Daten fehlerfrei zum Empfänger übermittelt werden. Dazu wird, ähnlich wie beim Internet-Protokoll, ein Header verwendet der vor das Datenpaket gehängt wird. Dieser Header besteht aus einer Reihe von Datenfeldern, die alle für den Empfänger relevanten Informationen enthalten. Diese Datenpakete, die aus dem TCP-Header und den Nutzdaten bestehen, werden als Segmente bezeichnet.

Um dem Empfänger die Möglichkeit der Fehleranalyse zu ermöglichen, verfügt jedes Segment über eine Prüfsumme. Wenn ein Segment korrekt empfangen wurde, wird dieser Empfang durch ein Segment des Empfängers bestätigt. Dieses Segment hat die Funktion einer Quittung, die dem Sender den fehlerfreien Empfang bestätigt. Sollte diese Quittung den Sender jedoch nicht in einer festgelegten Zeitperiode erreichen, wird das nicht korrekt übermittelte Segment erneut an den Empfänger übertragen. Dadurch kann es natürlich passieren, daß der Empfänger ein Segment mehrfach erhält. Dies hat jedoch keine negativen Auswirkungen auf die Datenkonsistenz, da jedes Segment über eine eindeutige Sequenznummer verfügt, anhand welcher alle Duplikate ermittelt werden können. Die mehrfach empfangenen Segmente werden einfach vernichtet.

Neben der korrekten Übermittlung der Daten ist TCP auch für die Weiterleitung der empfangenen Daten an die entsprechende Anwendung verantwortlich. Dies geschieht mittels der Portnummer, die im Feld Empfänger-Port des TCP-Headers vermerkt ist. Da dieses Feld 16 Bit groß ist, liegt die theoretische Anzahl von Portnummern bei 65.535. Für die wichtigsten Anwendungen sind diese Portnummern nach RFC 1700 standardisiert, z.B. FTP = 21, Telnet = 23 usw.. Durch die Portnummer zusammen mit der IP-Adresse kann ein Anwendungsprozeß eindeutig angesprochen werden, diese Kombination wird auch als Socket bezeichnet.

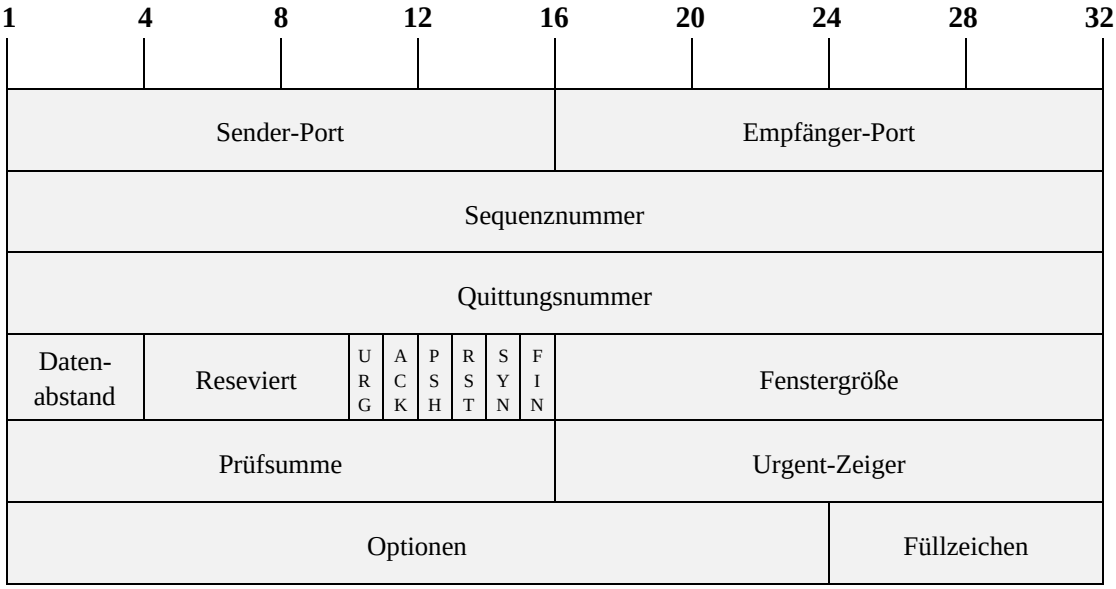


Abbildung 2-7: TCP-Header [BrHo 95]

- **Sender-Port (Source Port):**

  Gibt die Absenderadresse des Prozesses bzw. Dienstes an. Für die wichtigsten Anwendungen gibt es fest definierte Portnummern, z.B. FTP = 21, Telnet = 23, SMTP = 25 usw.

- **Empfänger-Port (Destination Port):**

  Bestimmt den Prozeß auf dem Zielhost, an den die Daten übermittelt werden sollen.

- **Sequenznummer (Sequence Number):**

  Die Sequenznummer gibt die Position des Segmentes im Datenstrom an. Zu Beginn der Übermittlung wird diese initial erzeugt und an den Zielhost übermittelt. Dieser wiederum erzeugt selbst eine Quittungsnummer, die er dann zur Bestätigung an den Absender zurückschickt. Nachdem die Verbindung steht, wird bei jedem zu übertragenden Segment die Sequenznummer um die Anzahl der erhaltenen Bytes erhöht.

- **Quittungsnummer (Acknowledgement Number):**

  Die Quittungsnummer gibt an, welche Bytes der Zielhost als nächstes empfangen wird. Damit weiß der Sender, bis zu welcher Stelle die Daten beim Empfänger korrekt angekommen sind.

- **Datenabstand (Data Offset):**

  Gibt die Länge des TCP-Headers, anhand der Anzahl enthaltener 32-Bit-Wörter an. Diese ist erforderlich, da die Länge des Optionen-Feldes variabel ist.

- **Reserviert (Reserve):**

  Enthält 6 reservierte Bits für eine spätere Nutzung. (Zur Zeit standardmäßig auf Null gesetzt)

- **URG (Urgent):**

  Bei gesetztem Bit wurde das Urgent-Zeiger-Feld verwendet und muß somit ausgewertet werden.

- **ACK (Acknowledge):**

  Wenn das Bit gesetzt ist, enthält das Quittungsnummer-Feld einen gültigen Eintrag. Anderenfalls wir der Eintrag nicht beachtet.

- **PSH (Push):**

  Bei gesetztem Bit wird veranlaßt, daß die Daten nicht gepuffert, sondern sofort an die Anwendung weitergeleitet werden.

- **RST (Reset):**

  Ist das RST-Bit = 1, dann wird die Verbindung zurückgesetzt. Dies geschieht z.B., wenn es bei der Übertragung zu einem Fehler gekommen ist oder wenn dem Aufbau einer Verbindung nicht zugestimmt wird.

- **SYN (Synchronisation):**

  Wird beim Aufbau einer Verbindung auf 1 gesetzt, zusätzlich ist das ACK-Flag = 0. Zur Bestätigung der Verbindung wird dem Zielhost eine Antwort mit gesetzten SYN- und ACK-Flag geschickt.

- **FIN (Final):**

  Signalisiert, daß die Datenübermittlung beendet ist und die Verbindung abgebaut wird.

- **Fenstergröße (Window Size):**

  Über dieses Feld wird die Flußkontrolle von TCP gesteuert. Dies geschieht mittels eines Schiebefensters (Sliding Window) mit variabler Größe. Durch die Angabe der Fenstergröße teilt der Empfänger dem Sender mit, über wieviel Puffer er verfügt, um empfangene Segmente zwischen zu speichern. Um ein Überlaufen (overflow) dieses Puffers zu verhindern, sendet der Empfänger eine Quittung mit einer Fenstergröße von Null. Um die Übertragung nach dem Stop wieder in Gang zu setzen, wird eine weitere Quittung mit einer Fenstergröße ungleich Null gesendet.

- **Prüfsumme (Checksum):**

  Die Prüfsumme dient zur Fehlererkennung und wird aus dem TCP-Header, den Daten und dem Pseudo-Header gebildet. Diese Summe wird auf die gleiche Art und Weise wie die Prüfsumme beim IP-Header erzeugt.

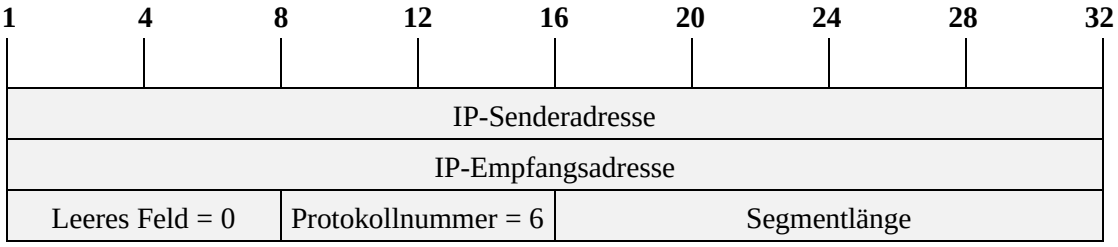


Abbildung 2-8: Pseudo-Header

Der Pseudo-Header dient nur zur Berechnung der Prüfsumme und wird danach sofort wieder verworfen. Die Benutzung der IP-Adressen im Pseudo-Header ist eigentlich eine Verletzung der Protokollhierarchie, da diese nur von den Internet-Schichten verwendet werden dürfen.

- **Urgent-Zeiger (Urgent Pointer):**

  Dieses Feld kennzeichnet Daten als besonders dringlich. Dies könnte z.B. ein Interrupt bzw. Break in einer Telnet-Sitzung sein. Durch Addition des Urgent-Zeigers und der Sequenznummer erhält man einen Zeiger auf das letzte dringliche Byte.

- **Optionen (Options):**

  Dient zur Angabe von Funktionen, die durch die anderen Felder im TCP-Header nicht abgedeckt werden. Zur Zeit sind folgende drei Optionen für dieses Feld vorgesehen.

  i.) *End of Option List*

  Kennzeichnet das Ende der Optionsliste

  ii.) *No-Operation*

  Dient zum Auffüllen von Bits zwischen zwei Optionen

  iii.) *Maximum Segment Size*

  Dient dazu, eine größere Anzahl von Nutzdaten in einem Segment festzulegen. Dadurch kann die Anzahl der zur Übertragung benötigten Segmente verringert werden, was sich günstig auf den Overhead auswirkt, der durch den TCP-Header verursacht wird. Der Default Wert liegt bei 536 Bytes.

- **Füllzeichen (Padding):**

  Da das Optionen-Feld eine variable Länge hat und die Länge des TCP-Headers immer ein vielfaches von 32 Bit sein muß, wird mittels Füllzeichen (Nullen) aufgefüllt.

Beim TCP-Protokoll handelt es sich, wie schon erwähnt, um ein verbindungsorientiertes Protokoll. Aus diesem Grunde wird die logische Verbindung zwischen Sender und Empfänger über ein sogenanntes „Three way handshake“ aufgebaut. Zum Aufbau der Verbindung übermittelt der Sender dem Empfänger ein erstes Segment, das noch keine Nutzdaten enthält. Dieses Segment enthält eine Sequenznummer, mit welcher die Übertragung startet. Außerdem ist in diesem Segment das SYN-Flag gesetzt, welches den Wunsch des Verbindungsaufbaus anzeigt. Sollte der Empfänger diesem Wunsch nachkommen, so sendet er zur Bestätigung des Verbindungsaufbaus ein Segment, bei dem das SYN- und ACK-Flag gesetzt wird. Desweiteren enthält dieses Segment eine vom Empfänger vergebene Sequenznummer, und im Quittungsnummern-Feld steht die um 1 erhöhte Sequenznummer des Senders. Nach Empfang dieses Segmentes durch den Sender verschickt dieser wiederum eine Bestätigung. Jedoch darf diese bereits Nutzdaten enthalten. Bei diesem Segment ist nur noch das ACK-Flag gesetzt und das Sequenznummern-Feld enthält die Quittungsnummer des Empfängers. Zusätzlich wird im Quittungsnummern-Feld die um 1 erhöhte Sequenznummer des Empfängers eingetragen.

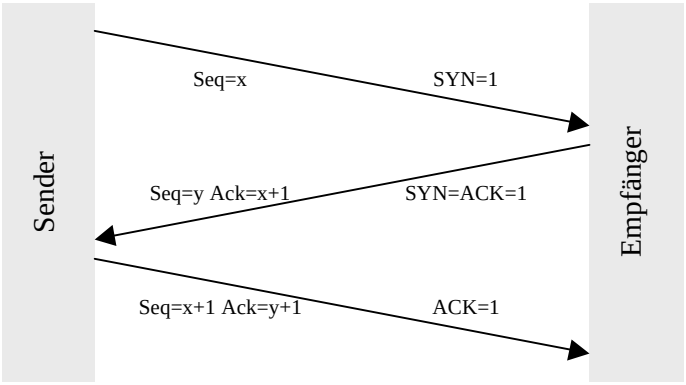


Abbildung 2-9: Verbindungsaufbau „Three Way Handshake“ [Hart 97]

Der eigentliche Datenaustausch findet, wie schon erwähnt, nach dem „Sliding Windows“ Prinzip statt. Der Sender schickt dem Empfänger so viele Bytes, wie in dem Fenstergröße-Feld im TCP-Header angeben ist, und wartet dann auf die Bestätigung des Empfängers, bevor die nächsten Bytes abgesandt werden. Nach erfolgreicher Datenübermittlung muß die Verbindung zwischen beiden Rechnern wieder getrennt werden. Dies geschieht, wie der Verbindungsaufbau, mittels eines „Three way handshake“. Der Rechner (A), der die Verbindung trennen will, sendet dem anderen Rechner (B) ein Segment, in dem das FIN- und ACK-Flag gesetzt ist und alle bis zu diesem Zeitpunkt empfangenen Daten bestätigt werden. Nach Erhalt dieses Segmentes sendet auch der Rechner (B) ein Segment, in dem er alle bis dahin empfangenen Daten quittiert. Auch hier ist das FIN- und ACK-Flag gesetzt. Wie sollte es anders sein. Wird dieser Erhalt von Rechner (A) durch ein ACK bestätigt, so ist die Verbindung abgebaut.

2.3.2 UDP (User Datagram Protocol)

Neben dem TCP-Prokoll gehört auch das „User Datagram Protokoll“ zur Transportschicht. Es handelt sich dabei um ein *unzuverläßliches* (unreliable) und *verbindungsloses* (Protokoll). Die Bezeichnung „unzuverlässig“ soll nicht, wie man vermuten könnte, bedeuten, daß die Daten nicht korrekt übermittelt werden, sondern bezeichnet die Tatsache, daß durch das Protokoll nicht garantiert wird, daß die Daten beim Empfänger auch ankommen. Wenn die Daten den Empfänger jedoch erreicht haben, kann deren Korrektheit mittels der Prüfsumme überprüft werden. Sollte diese Prüfung jedoch negativ ausfallen, dann sind die Daten auf jeden Fall verloren, da sie kein zweites Mal übermittelt werden.

Abbildung 2-10: UDP-Header [Hart 97]

Wie man in Abbildung 2-10 sehen kann, enthält der UDP-Header nur die absolut erforderlichen Einträge. Die Portnummern sind genauso definiert wie beim TCP-Protokoll und dienen zur Adressierung der richtigen Sender- und Empfänger-Prozesse. Die UDP-Länge gibt die Länge des gesamten Segmentes an, also inklusive UDP-Header. Die Prüfsumme ist im Gegensatz zu den anderen Feldern optional. Soll auf die Prüfsumme verzichtet werden, wird in das Prüfsummen-Feld eine Null eingetragen.

Auf Grund der guten Effizienz des UDP-Protokolls ist es prädestiniert für die Datenübermittlung bei Realtime-Anwendungen, wie z.B. Sprache oder Video. Hierbei kommt es meist weniger auf die Korrektheit der Daten an, als vielmehr auf die möglichst schnelle Übertragung. Dabei muß jedoch in aller Regel ein gewisser Prozentsatz der Segmente korrekt empfangen werden, anderenfalls wird die Verbindung wegen der schlechten Übertragungsqualität unterbrochen. Ein weiterer Einsatzbereich von UDP ist die Übermittlung kleiner Datenmengen, für die der Aufbau einer Verbindung mittels „Three way handshake“ zu aufwendig (zeitintensiv) ist, z.B. ist dies beim „Domain Name Service“ (DNS) der Fall.

3 Routing in TCP/IP-Netzwerken (Internet)

In der bisherigen Betrachtung wurde nur auf die Arbeitsweise des Senders und Empfängers eingegangen. Daher soll im Folgenden die eigentliche Datenübermittlung im Internet näher betrachtet werden.

3.1 IP-Adressen

Wie im letzten Kapitel erläutert, werden beim Internet-Protokoll die Daten in Form von Datengrammen übertragen. Damit die Datengramme von einem Host zu einem anderen Host gelangen können, muß jeder Host im Netzwerk (Internet) eindeutig identifizierbar sein. Aus diesem Grunde verfügt jeder Host über mindestens eine sogenannte IP-Adresse. Diese IP-Adressen haben eine Länge von 32Bits und werden zur besseren Lesbarkeit in vier durch Punkte getrennte in Dezimalschreibweise angebenen Oktette dargestellt. Die IP-Adresse setzt sich aus zwei Teilen zusammen. Der erste Teil ist die Netzadresse (netid) und der zweite Teil ist die Interface-Adresse (hostid). Dabei gibt die Netzadresse das Netzwerk an, in dem sich der Host befindet, und die Interface-Adresse dient dann zur Identifizierung des Hosts in dem Netzwerk.

Wenn man sich nochmals das Beispiel Briefpost/TCP-Protokoll vor Augen hält, würde die Netzadresse der Postleitzahl und die Interface-Adresse der Straße mit Hausnummer des Adressaten entsprechen.

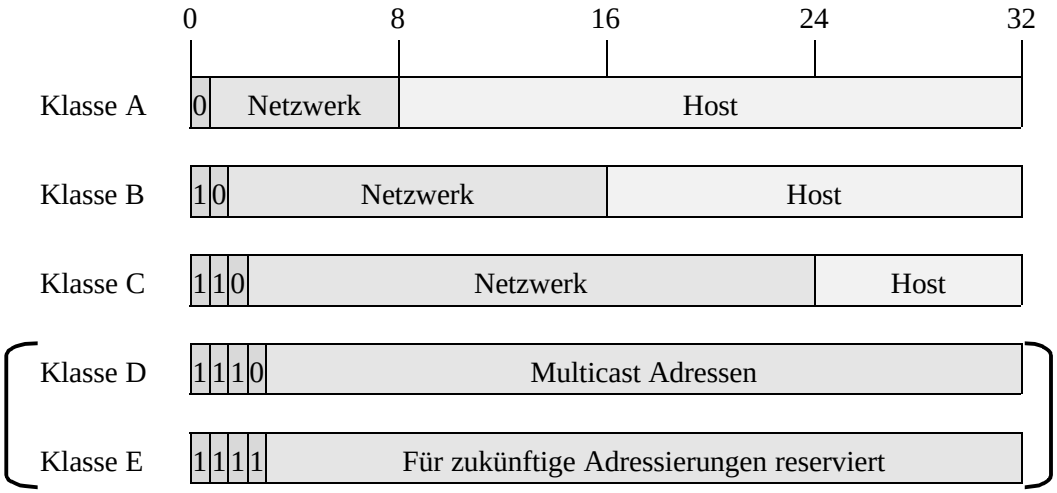


Abbildung 3-1: Adressklassen [Hart 97]

Wie in Abbildung 3-1 zu sehen, ist der Adressraum der Internet-Adressen in fünf Klassen unterteilt. Von diesen Klassen sind aber nur die Klassen A, B und C von praktischer Bedeutung. Die Unterscheidung der Klassen erfolgt mit den vordersten Bit(s) der IP-Adresse.

So ist das erste (signifikanteste) Bit der Klasse A immer gleich Null. Für die Netzadresse (netid) bleiben somit 7 Bit übrig, so daß theoretisch Werte im Bereich 0 bis 127 möglich sind. Die Adressen 0 und 127 haben jedoch eine Sonderfunktion. Somit liegen die IP-Adressen im Bereich 1.0.0.0 bis 126.255.255.255. Bei der Klasse B sind die beiden ersten Stellen gleich 1 0. Die Netzadresse umfaßt 14 Bits und die Interface-Adresse 16 Bits. Die möglichen Adressen liegen in dem Bereich 128.0.0.0 bis 191.255.255.255. Die Klasse C enthält die größte Anzahl von Netzwerken, dafür ist aber die Anzahl der Hosts pro Netz auf 254 beschränkt. Der gültige Adressbereich reicht von 192.0.0.0 bis 223.255.255.255.

Die folgenden beiden Klassen D und E unterscheiden sich von den bisher betrachteten Klassen. Bei der Klasse D sind die ersten 4 Bits gleich 1 1 1 0. Sie enthält Multicast-Adressen, die es ermöglichen, mehrere Rechner gleichzeitig anzusprechen.

Für die Klasse E, bei der die ersten 4 Bits gleich 1 1 1 1 sind, gibt es derzeit noch keine Verwendung. Daher sind die IP-Adressen 240.0.0.0 bis 255.255.255.255 für zukünftige Adressierungen reserviert.

Klasse	Netze	max. Hosts pro Netz
A	126	16.777.214
B	16.363	65.534
C	2.097.151	254

Abbildung 3-2: Adressraum der IP-Klassen

Alle Hosts, deren IP-Adresse mit der gleichen Netid beginnen, müssen sich auch im gleichen Netzsegment befinden. Somit müßte ein Klasse A Netz über 16 Millionen Rechner verwalten. Da dies nicht praktikabel ist, gibt es die Möglichkeit, IP-Adressen zu maskieren. Dabei werden Teile der Hostid zur Erweiterung der Netid verwendet. Um nun festzustellen, welche Bits der IP-Adresse zur Netid und welche zur Hostid gehören, verwendet man die sogenannte „Subnet Mask“. In der Subnet Mask sind alle Bits der Netid mit 1 und die Bits der Hostid mit Null maskiert. Für ein Klasse B Netz entspricht die „Default Subnet Mask“ in Dezimalschreibweise 255.255.0.0. Da die Subnet Mask bitweise arbeitet, kann die Anzahl der Subnetze und der darin enthaltenen Anzahl von Hosts den Erfordernissen angepaßt werden.

3.2 Netwerkarchitektur

Im letzten Abschnitt wurden Netzwerke anhand ihrer IP-Klasse (A, B, C) unterschieden. Jedoch ist diese Charakterisierung normalerweise unüblich. Vielmehr unterscheidet man Netzwerke durch deren Größe und Einsatzgebiete.

- Das „Lokal Area Network“ (LAN) bezeichnet kleine lokale Netzwerke, deren Ausdehnung auf einige 100m beschränkt ist. Meist haben diese Netze aber eine relativ hohe Datenübertragungsrate. Die Netzwerkgröße entspricht also denen von kleineren und mittleren Betrieben.
- Die nächst größere Netzwerkklasse ist das „Metropolitan Area Network“ (MAN), dessen Ausdehnung im Bereich 5km bis 50km liegt. Diese Netze gehören nicht unbedingt einer einzelnen Organisation an, sondern können auch ein Zusammenschluß mehrerer Organisationen sein, die sich die Infrastruktur des MANs teilen.
- Die größte Netzwerkklasse bildet das „Wide Area Network“ (WAN), dessen Größe uneingeschränkt ist und somit den ganzen Globus umspannen kann. Dies können z.B. Netzwerke von Internet-Providern oder von großen Firmen sein, die weltweit tätig sind. Auch das Internet selbst fällt in diese Klasse und bildet somit das größte WAN.

3.3 MAC-Adressen

Wie am Anfang des Kapitels erläutert, verfügt jeder Host über eine IP-Adresse, anhand derer ein Rechner im Netzwerk (Internet) eindeutig identifiziert werden kann. Bei dieser Adresse handelt es sich jedoch nur um eine logische Adresse der Internet-Schicht (Internet-Layer). Die eigentliche Adressierung erfolgt über eine 48-Bit lange Adresse der Netzwerkkarte. Diese sogenannten MAC-Adressen (Media Access Control) sind wie die IP-Adressen eindeutig definiert. Damit dies gewährleistet ist, wird von der IEEE (Institute of Electrical and Electronic Engineers) für jede einzelne Netzwerkkarte eine einmalige MAC-Nummer vergeben.

Die Umsetzung der IP-Adresse in die MAC-Adresse erfolgt in der Netzzugangsschicht (Network Access Layer) normalerweise mittels des „Address Resolution Protocol“ (ARP). Jeder Host verfügt daher über eine Tabelle, die sogenannte „ARP Cache Table“, die für jede im lokalen Netzsegment enthaltene IP-Adresse die dazugehörige MAC-Adresse enthält. Da sich MAC- und IP-Adressen in einem Netzwerk verändern können, z.B. durch Tausch einer Netzwerkkarte, ist die ARP-Tabelle dynamisch angelegt. Jeder Eintrag wird 20 Minuten nach dem letzten Zugriff gelöscht, so daß es bei Verbindungsaufnahme vorkommt, daß zur angegebenen IP-Adresse kein Eintrag in der ARP-Liste vorhanden ist. In diesem Fall wird an alle Rechner ein Datenpaket gesandt, das die entsprechende IP-Adresse enthält. Sollte es im Netzwerk einen Host mit dieser IP-Adresse geben, so schickt dieser als Bestätigung seine MAC-Adresse.

3.4 Router

Wenn nun ein Host A einem anderen Host B ein Datenpaket schicken will, gibt es zwei Fälle, die zu unterscheiden sind. Beim ersten Fall befindet sich der Zielhost B im gleichen Netzwerksegment wie der Sendehost A. Zur Übertragung des Datenpaketes genügt es also, wenn der Host A das Datenpaket an die MAC-Adresse des Hosts B schickt. Die Übertragung erfolgt somit analog dem TCP/IP-Schichtmodell in Abbildung 3-3.

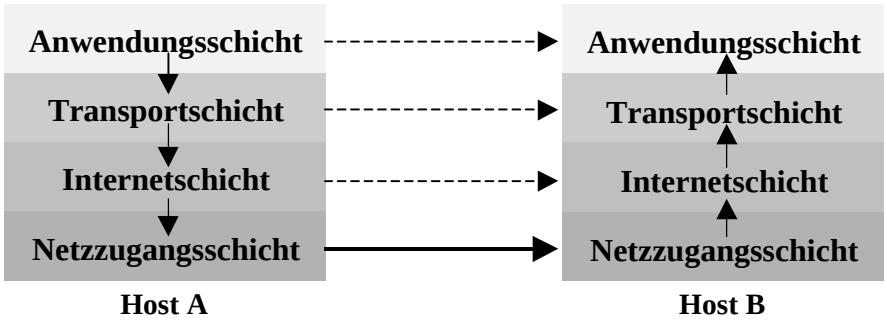


Abbildung 3-3: TCP/IP-Übertragung

Beim zweiten Fall befindet sich der Zielrechner B hingegen in einem anderen logischen Netzsegment. Somit gibt es für den Host A keine Möglichkeit, diesen direkt mittels der MAC-Adresse anzusprechen. Daher wird das Datenpaket an den nächsten Router weitergeleitet, der dann für den weiteren Transport des Datenpaketes an den Zielhost B verantwortlich ist.

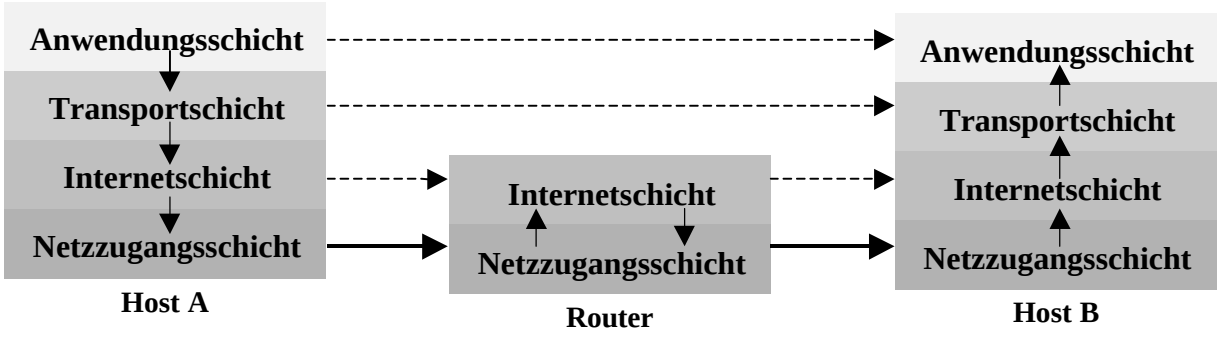


Abbildung 3-4: TCP/IP-Routing [Hart 97]

Auch beim Router gibt es wiederum zwei Fälle. Im ersten Fall kennt der Router die MAC-Adresse des Zielhosts B und kann somit das Datenpaket an den Empfänger weiterleiten. Sollte dies nicht der Fall sein, wird das Datenpaket an den nächsten Router weitergeleitet, usw. Solange bis der Zielhost B erreicht oder die Lebenszeit (TTL) des Datenpaketes abgelaufen ist.

Unter dem Begriff „Routing“ versteht man also die zielgerichtete Weiterleitung eines Datenpaketes von einem Host zum anderen Host, wobei sich die Hostrechner in unterschiedlichen logischen Netzwerksegmenten befinden können.

Wie sieht nun so ein Router aus? Der Hauptunterschied zwischen einem normalen Host und einem Router ist, daß dieser an mindestens zwei Netzwerksegmenten angeschlossen ist. Jedes dieser Netzsegmente ist dabei über eine separate Netzwerkkarte mit dem Router verbunden. Dies hat zur Folge, daß ein Router über mehrere MAC-Adressen und somit auch über mehrere IP-Adressen verfügt. Damit hängt die IP-Adresse des Routers davon ab, aus welchem Netz er angesprochen wird.

Damit der Router seine Aufgabe der IP-Weiterleitung (IP-Forwarding) erfüllen kann, verfügt dieser über sogenannte Routing-Tabellen, in denen er nachschaut, an welchen benachbarten Router bzw. Zielhost die Daten weitergeleitet werden müssen.

Man unterscheidet bei der Routenberechnung zwei verschiedene Arten:

- Beim *statischen Routing* werden alle Routen fest vorgegeben, so daß bei Veränderungen in der Netzstruktur, die Routing-Tabellen entsprechend verändert werden müssen.
- Bei der zweiten Variante, dem *dynamischen Routing*, werden die Routing-Tabellen in gewissen Zeitabständen automatisch an Veränderungen in der Netzstruktur angepaßt.

Zur Aktualisierung der Routing-Tabellen beim dynamischen Routing, werden spezielle Routing-Protokolle eingesetzt, wie z.B. das „Router Information Protocol“ (RIP).

Ausführliche Informationen zum Thema Routing-Protokolle sind z.B. in [StKu 99] zu finden.

4 Multicast-Backbone (MBone)

4.1 Kommunikationsmodelle

Bei der Übermittlung von Daten unterscheidet man drei verschiedene Kommunikationsmodelle.

1. Bei der *Unicast*-Übertragung besteht zwischen dem Empfänger und dem Sender eine Host-to-Host Verbindung. Diese Art der Übertragung hat z.B. bei einer Videoübertragung zur Folge, daß jedem Teilnehmer (Empfänger) ein eigener Datenstrom übermittelt werden muß. Dadurch entsteht ein großes Datenaufkommen, welches hohe Anforderungen an die Bandbreite des Netzwerkes stellt.

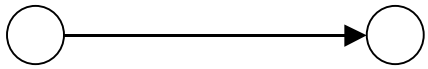


Abbildung 4-1: Unicast-Übertragung

2. Eine weitere Möglichkeit der Datenübertragung ist das sogenannte *Broadcast*. Dabei werden die Daten an jeden im Netzsegment erreichbaren Host gesandt. Diese Form der Übertragung reduziert zwar den Datenstrom auf ein Minimum, jedoch werden grundsätzlich alle Hosts angesprochen, also auch die jenigen, die Daten nicht auswerten. Aus diesem Grunde ist die Broadcast-Übertragung nur in lokalen Netzsegmenten möglich.

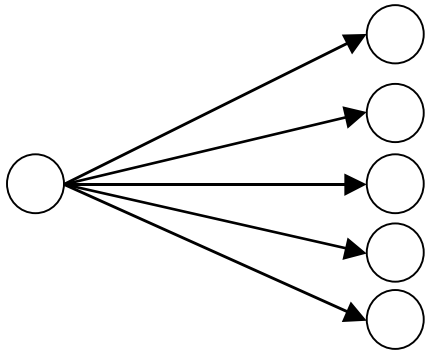


Abbildung 4-2: Broadcast-Übertragung

3. Bei der *Multicast*-Übertragung werden die Daten analog der Broadcast-Übertragung nur mittels eines Datenstroms übertragen. Jedoch werden nur die Hosts angesprochen, für die die Daten auch bestimmt sind. Dadurch ist die Multicast-Übertragung nicht auf ein Netzsegment beschränkt, sondern kann wie Unicast auch Netzwerk übergreifend verwendet werden.

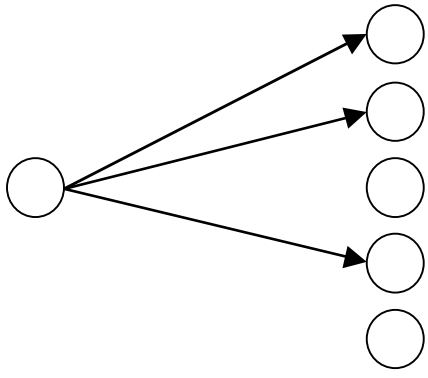


Abbildung 4-3: Multicast-Übertragung

4.2 Die Funktionsweise der Multicast-Übertragung

Wie gerade gesehen, läßt sich z.B. eine weltweite Videoübertragung nur mittels Multicast effizient durchführen. Aus diesem Grunde wird seit geraumer Zeit ein Multicast-Netz, der sogenannte Multicast-Backbone (MBone) aufgebaut. Obwohl die erste offizielle MBone-Übertragung bereits im Jahre 1992 stattfand, ist diese Art der Übertragung immer noch nur von einer Minderheit nutzbar. In Deutschland ist die Multicast-Übertragung hauptsächlich noch auf das Universitätsnetz beschränkt, siehe Abb. 4-4. Jedoch steigt die Anzahl der Internet-Nutzer, die am MBone hängen, rasant an und verdoppelt sich etwa alle 8 Monate.

Abbildung 4-4: MBone in Deutschland (Stand 7/1999) [Heil 99]

Wie funktioniert nun also die Multicast-Übertragung?

Der Versand eines Multicast-Paketes funktioniert analog der Übertragung von normalen IP-Paketen. Mit dem Unterschied, daß das Multicast-Paket von einer Gruppe von Netzwerkendpunkten empfangen wird, statt von einem einzelen Host. Wie schon in Kapitel 3.1 zu sehen war, enthält die IP-Adressklasse D mit den IP-Adressen von 224.0.0.0 bis 239.255.255.255 spezielle Multicast-Adressen.

Bei der Übertragung müssen zwei Fälle unterschieden werden. Im ersten Fall befinden sich alle Empfänger im lokalen Netzsegment des Senders. Somit können alle Hosts, wie schon erwähnt, mittels ihrer MAC-Adresse direkt angesprochen werden. Für die Multicast-Übertragung gibt es spezielle MAC-Adressen, die im Bereich von 01-00-5E-00-00-00 bis 01-00-5E-7F-FF-FF liegen, wobei die letzten 23 Bits dieser Adressen der IP-Multicast-Adresse entsprechen. Damit die IP-Pakete das Netzwerk nicht verlassen, wird im Lebenszeit-Feld (TTL) des IP-Headers eine Null eingetragen, dadurch wird sichergestellt, daß die IP-Pakete bei Erreichen eines Routers vernichtet werden.

Beim zweiten Fall liegt mindestens ein Empfänger der Multicast-Übertragung außerhalb des Netzsegments des Senders. Damit die Datenpakte das Netzsegment verlassen können, muß die Lebenszeit (TTL) entsprechend groß genug gewählt werden. Die Weiterleitung der Daten erfolgt dann mittels Multicast-fähigen Routern (MRouter). Dabei kann es vorkommen, daß sich zwischen zwei MRoutern ein normaler Router befindet, der von Multicast keine Ahnung hat. Dann wird das sogenannte IP-Tunneling verwendet, das es erlaubt, die existierende Internet-Struktur zu nutzen. Zu diesem Zweck

wird das Multicast-Paket mit einem weiteren IP-Header versehen, der als Zieladresse den nächsten MRouter enthält. Bei diesem angekommen, wird der IP-Header wieder entfernt und die weitere Übertragung erfolgt dann wiederum über MRouter.

Eine weitere Besonderheit bildet die Live-Übertragung via Multicast, da die Empfänger die, zu einer Gruppe gehören, variieren können. Wenn sich ein Host zu einer Live-Übertragung anmeldet und somit einer bestimmten Multicast-Gruppe beitritt, dann versendet der MRouter des lokalen Netzsegmentes mittels des „Internet Group Management Protocol“ (IGMP) einen „Host Membership Report“, falls nicht bereits ein anderer Host im gleichen Netzsegment die Multicast-Übertragung empfängt. Es entsteht ein sogenannter „Spanning-Tree“, wie in Abb. 4-5 zu sehen ist, der nur die Netzsegmente mit dem Muticast-Stream versorgt, in dem sich mindestens ein angemeldeter Host befindet. Um den Aufbau des Spanning-Tree zu ermöglichen kommen spezielle Routing-Protokolle zum Einsatz, wie das „Distance Vector Multicast Routing Protocol“ (DVMRP), welches mit dem „Router Information Protocol“ (RIP) verwandt ist. Nähere Informationen zu diesem Thema stehen in [AnSc 98].

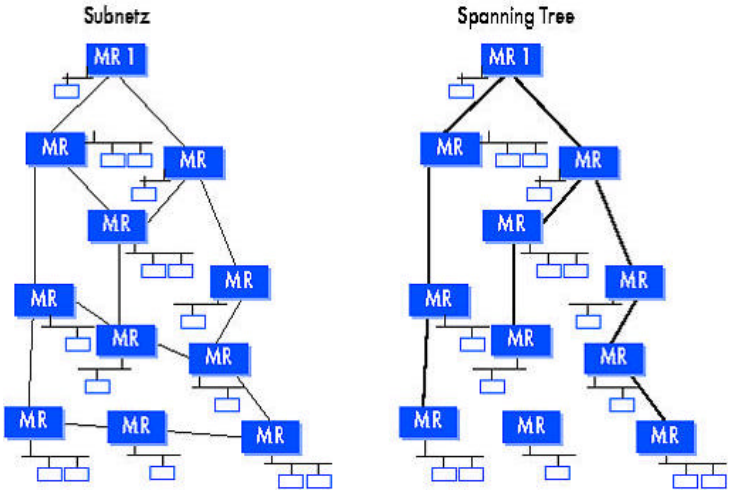


Abbildung 4-5: Spanning-Tree [AnSc 98]

4.3 Quality of Service (QoS)

Unter dem Begriff „Quality of Service“ werden die Übertragungsqualitäten einer Verbindung zusammengefaßt:

- Datendurchsatz (Bandbreite)
- Fehlerfreie Übertragung
- Geschwindigkeit (zeitlich)
- Ausfallsicherheit

Wenn man die Übertragung in TCP/IP-Netzwerken unter dem Aspekt „Quality of Service“ (QoS) betrachtet, stellt man fest, daß mit dem heute verwendeten Internet-Protokoll Version 4 (IPv4) keinerlei Verbindlichkeiten garantiert werden. Wie gesehen ist die Multicast-Übertragung schon ein Schritt in die richtige Richtung, jedoch wird auch hierbei nur die vorhandene Infrastruktur besser genutzt, was die Wahrscheinlichkeit einer guten Übertragung erhöht. Ein weitere Verbesserung wird das in der Entwicklung befindliche Internet-Protokoll Version 6 (IPv6) bringen, welches von sich aus bereits in der Lage ist, Multicast zu übertragen. Außerdem stellt es eine Möglichkeit bereit, IP-Datenpakete mit einer Priorität zu versehen, wodurch z.B. die Übertragung von Realtime-Anwendungen im Vergleich zur normalen Übertragung beschleunigt werden kann.

5 Literaturverzeichnis

[AnSc 98] Frank-Uwe Andersen, Jürgen Schmidt:
Weltweit Video – MBone: Multimedia im Internet
c´t – Magazin für Computertechnik, Ausgabe 21/98, S. 262ff

[Bogn 99] Istvan Bognar:
Das MultiMedia Seminar Sommersemester 1998 - Routing im Internet (Einführung)
http://www.informatik.uni-tuebingen.de/~bognar/MMseminar/struktur.html

[BrHo 95] Kai Brauer, Friedhelm Hosenfeld:
Kommunikation ohne Grenzen, TCP/IP: Informationsübermittlung im Internet
c´t – Magazin für Computertechnik, Ausgabe 12/95, S.330ff

[Dodg 99] Martin Dodge:
An Atlas of Cyberspaces - Historical Maps of ARPANET and the Internet
http://www.cybergeography.org/atlas/historical.html

[From 98] M.Fromme:
Internet-Protokoll: Multicast, QoS, IPv6

[Gade 95] Frank Gadegast:
Diplomarbeit: TCP/IP-basierte Dienste zur Speicherung von Multimedia-Daten
http://www.powerweb.de/phade/diplom/

[Gill 97] Andreas Gillhuber:
Kurze Wege durchs Netz, Schichtenweise – das OSI-Modell
c´t - Magazin für Computertecknik, Ausgabe 11/97, S.334ff

[Grom 99] Gregory R. Gromov:
The Roads and Crossroads of Internet History
http://www.netvalley.com/netvalley/intval1.html

[Hart 97] Mike Hartmann:
Institut für Informatik und Gesellschaft – Telematik
Die TCP/IP Protokoll Suite
http://www.iig.uni-freiburg.de/~mhartman/tcpip/index.html

[Heil 99] Peter Heiligers:
MBone-DE Adressbereich : B-WIN Regionen Map November 1997
http://www.mbone.de/maps/mbone-de-07-99.jpg

[Holt 97] Heiko Holtkamp:
Einführung in TCP/IP
http://www.rvs.uni-bielefeld.de/~heiko/tcpip/

[Kuri 96] Jürgen Kuri:
Wenn der Postmann zweimal klingelt-Namen und Adressen im TCP/IP-Netzwerk und im Internet
c´t - Magazin für Computertecknik, Ausgabe 12/1996, S.334ff

[Kuri 97] Jürgen Kuri:
Da geht´s lang! – Routing, oder: wie die Daten im Internet ihren Weg finden
c´t - Magazin für Computertecknik, Ausgabe 6/1997, S.380ff

[KuRe 98] Jürgen Kuri, Jörg Rech:
Das Netz aufs Bit geschaut – Protokollanalyse (nicht nur) im Windows-Netzwerk
c´t - Magazin für Computertecknik, Ausgabe 20/1998, S.222

[Mock 96] Andreas Mock:
Diplomarbeit: Recherchen im Internet
http://www.wiso.uni-augsburg.de/edvbeauf/studinfo/mock/public_html/diplom/inhalt.html

[Müll 99] Stefan Müller:
Klausurersatzleistung: Thema "Klassifizierung von Netzwerken"
http://www.gs-lehnin.pm.bb.schule.de/projekt/klassi1.html

[Nico 96] Michael Nicolai:
TCP/IP-Grundlagen
http://www.ruhr-uni-bochum.de/~rothamcw/Lokale.Netze/tcpip.html

[Post 81] J. Postel:
Network Working Group – Request for Comments: 795
http://andrew2.andrew.cmu.edu/rfc/rfc795.html

[StKu 99] Benjamin Stein, Jürgen Kuri:
Reisebüro - Routing und Internet-Zugang fürs LAN mit Windows
c´t - Magazin für Computertecknik, Ausgabe 8/1999, S.214ff

[Ullr 97] Mike Ullrich:
Teleseminar – Thema 1: Einführung in das Internet
http://www.stud.uni-karlsruhe.de/~uo01/d/teleseminar/index.html

[Zako 99] Robert H. Zakon:
Hobbes' Internet Timeline v4.2
http://www.isoc.org/zakon/Internet/History/HIT.html

Real Time Streaming Protocol

Konstantin Steuer

FB Informatik, Uni Dortmund
Konstantin.J.Steuer@ruhr-uni-bochum.de

Zusammenfassung

Zur Zeit gewinnen kontinuierliche Medien wie Video und Audio aber auch Live - Konferenzen immer mehr an Bedeutung. Das hat zu Folge, daß diese Medien auch übers Internet angeboten werden. Bis vor kurzem gab es nur die Möglichkeit, solche Dateien einfach im Ganzen zu Übertragen um sie anschließend abzuspielen. Um das zu ändern, entwickelte man Protokolle die es erlauben diese Quellen live zu nutzen. Dabei müssen zeitkritische Vorgänge verwaltet werden, denn eine Verzögerung bei der Übertragung oder der Verlust einiger Pakete zur erheblichen Störungen führen können. Real Time Streaming Protokoll trägt nicht selber die benötigten Daten, sondern stellt einen Steuerungsmechanismus zur Verfügung, das von Multimediaprogrammen benutzt wird um den Datenfluß zu kontrollieren. Damit sind Funktionen wie Play, Stop, Forward, die für diese Medien üblich sind, realisierbar.

In diesem Dokument wird die Funktionsweise und der Einsatz von RTSP in der Übertragung von kontinuierlichen Medien vorgestellt. Dabei wird auf wesentliche Mechanismen und Prinzipien dieses Protokolls eingegangen.

1 Einleitung

Der Trend zur multimedialen Information gewinnt ständig an Bedeutung. Printmedien bzw. Texte werden durch Video und Audio ergänzt oder sogar verdrängt. Natürlich ist diese Entwicklung am Internet oder allgemein an den Datennetzen nicht spurlos vorbeigegangen. Man findet daher immer mehr dieser Medien, die in den Netzen angeboten werden, sei es Schulungsmaterialien in internen Firmennetzen oder Radio- oder Fernsehsendungen im Internet.

Welche Anwendungen auch immer dabei benutzt werden, sie haben ein gemeinsames Problem. Die Daten müssen kontinuierlich übertragen werden. Das ist nur durch die Kombination verschiedener Protokolle möglich. Die Streaming-Technologie setzt dabei auf vier zentrale Protokolle: Resource Reservation Protocol (RSVP), das Netzwerkressourcen für den Datenstrom reserviert, Realtime Transport Protocol (RTP), das für den Transport der Daten sorgt, Synchronized Multimedia Integration Language (SMIL) für die Streambeschreibung und Formatierung und schließlich Realtime Streaming Protocol (RTSP), das unter der Federführung von Realnetworks und Netscape entstanden ist.

Real Time Streaming Protocol ist ein Protokoll der Anwendungsschicht. Seine Aufgabe besteht darin die Übertragung kontinuierlicher Medien wie Audio- oder Videodateien zu steuern. Damit ist nicht die Steuerung der eigentlichen Datenübertragung wie z.B. erneute Anforderung eines Datenpakets bei Verlust gemeint, obwohl dies im Notfall auch von RTSP übernommen werden kann. Viel mehr fungiert RTSP, als eine Art Fernsteuerung für Multimediaserver [ScRL 98a]. Dabei ermöglicht dieses

Protokoll Funktionen wie Play, Forward oder Stop zu realisieren, die Anwendungen eine direkte Kontrolle über die Medien auf den Servern geben.

In den folgenden Kapiteln werde ich grundlegende Eigenschaften, Prinzipien und Mechanismen, die sich hinter RTSP verstecken, vorstellen.

2 Eigenschaften

In der Literatur werden viele Eigenschaften und Merkmale dieses Protokolls erwähnt wie z.B. in [ScRL 98a]. Ich möchte jedoch auf wenige wichtige eingehen, die RTSP am besten charakterisieren.

- Protokollunabhängigkeit
  Wie schon oben angesprochen gehört RTSP der Anwendungsschicht an. Diese Schicht bittet eine sehr abstrakte Sicht auf die zu manipulierenden Daten. Um das zu gewährleisten, muß sich RTSP niederer Protokolle bedienen, die die eigentliche Arbeit leisten. RTSP ist in dieser Hinsicht sehr flexibel. Es stützt sich nicht auf bestimmte Protokolle. So können die Daten z.B. mit Hilfe von unzuverlässigen Übertragungsprotokollen wie UDP sowie durch extra für Real Time - Medien entworfene Protokolle wie RTP übermittelt werden.
- Erweiterbarkeit
  RTSP verfügt über eine große Anzahl an Parameter und unterstützenden Methoden. Nichtsdestotrotz steht es dem Benutzer frei, neue zu definieren. Damit ist RTSP innerhalb einer Version ausbaufähig und erlaubt es schneller auf neue Entwicklungen zu reagieren. Um jedoch Konflikte zu vermeiden, die daraus resultieren, daß die Partner einige Parameter oder Funktionen nicht unterstützen, verfügt RTSP über einen Kontrollmechanismus. Damit können nicht implementierte Methoden bzw. Parameter von Server oder Client verworfen werden, so daß die Kommunikation gegebenenfalls auf gemeinsamer Basis stattfinden kann. Wie solche Definitionen im einzelnen aussehen, wird in [ScRL 98a] beschrieben.
- Interaktion zwischen Client und Server
  Die bisherigen Protokolle unterstützen eine einseitige Kommunikation. Der Client fordert vom Server Daten an. Daraufhin werden diese ausgeliefert und der Server wartet auf weitere Instruktionen. RTSP dagegen, propagiert eine beidseitige Kommunikationsmöglichkeit. Somit kann auch der Server Anfragen an den Client senden. Dieses erlaubt dem Server beispielsweise die vom Client unterstützten Methoden zu erfahren oder während der Übertragungszeit die Auslieferungsadressen zu ändern. Die zweite Option ermöglicht eine Streuung der Daten über mehrere Server, worauf ich noch später eingehen werde.
- Bandbreitenunabhängigkeit
  Kontinuierliche Medien tragen große Datenmengen. Aufgrund dessen, können sie nicht im Ganzen übertragen werden, da damit die Fehleranfälligkeit steigt. In diesem Fall behelft man sich des **Streaming – Prinzips** [Reib 99, Down 99].
  Dabei werden die Dateien in kleine Pakete aufgespalten, die auch bei niedrigen Bandbreiten zuverlässig übertragen werden können. Diese Daten werden wie andere Dateien auf einem Server abgelegt oder aber von einem Media Server verteilt. Sind dann auf Empfängerseite genügend Pakete eingetroffen, kann z.B. der Player mit dem Abspielen beginnen. Auf seiten des Anwenders passiert vereinfacht folgendes: Der Player spielt das erste Paket ab, dekomprimiert das zweite Paket, während im Hintergrund ein drittes Paket eintrifft.
  Eine Mindestanforderung an die Bandbreite bleibt aber bestehen. D.h. auch bei der Anwendung dieses Prinzips, kann die Bandbreite nicht ins Bodenlose fallen. Ansonsten kann es passieren, daß

die ohnehin kleinen Pakete so langsam übertragen werden, daß deren Wiedergabe völlig inakzeptable Ergebnisse liefert.

- Multiserverfähigkeit
  Schon vorhergehend habe ich angedeutet, daß die Informationen nicht unbedingt auf einem Server abgelegt werden müssen. So können verschiedene Dateien aber auch deren Bruchstücke über mehrere Server verteilt werden. Dabei merkt der Anwender nichts vor all dem. RTSP ermöglicht es nämlich die Daten zur Laufzeit von anderen Orten zu ordern oder verschiedene Quellen zu synchronisieren. So können z.B. zu einer bestimmten Videoquelle gleichzeitig verschiedene Audioquellen angeboten werden um so die Informationen in verschiedenen Sprachen zu Verfügung zu stellen.
- Aggregationskontrolle
  Eine Multimediapräsentation kann mehrere Quellen enthalten, die gleichzeitig oder zeitversetzt wiedergegeben werden. Eine Möglichkeit diese Präsentation abzuspielen, ist es jede einzelne Datei anzusprechen und zu steuern. Dies ist jedoch mit großen Aufwand verbunden. RTSP bietet hierzu eine Lösung, die Aggregationskontrolle. Alle wichtigen Informationen, meist die zeitliche Anordnung der Quellen, werden in den sogenannten Containerfiles gespeichert. Der Client braucht anschließend nur die Kontrolle diese Files zu übernehmen und spart sich dadurch die separate Steuerung der einzelnen Medien. Damit vereinfacht RTSP das Handling von zusammengesetzten Präsentationen, steigert die Effizienz und verringert den Nachrichtenfluß.
- Endkopplung von Client und Server
  RTSP erlaubt es den Kommunikationspartnern innerhalb einer Sitzung unabhängig voneinander zu agieren. Damit ist es nicht mehr nötig jeden Schritt des Partners zu verfolgen und eine Rückmeldung abzuwarten. Um das zu ermöglichen, bedienen sich Client und Server einer Zustandsmaschine. So kennen die Kommunizierenden den aktuellen Arbeitszustand des anderen. Diesen Mechanismus beschreibe ich genauer im Kapitel 6.

3 Modi

RTSP unterstützt nicht nur das Abspielen oder Aufnehmen von einzelnen Medien, sondern auch Präsentationen und Livesendungen. Die letzteren machen nur Sinn, wenn auch mehrere Benutzer sie gleichzeitig verfolgen können. Demnach kann RTSP nicht nur für Unicast- sonder auch für Multicastsitzungen benutzt werden.

3.1 Unicast

Dies ist der einfachster und meist benutzter Modus. Ein Anwender möchte nur für sich eine aufgezeichnete Präsentationen, Audiodatei etc. wiedergeben oder sein Material auf einem Server aufnehmen. Hier bestimmt der Benutzer die jeweiligen Zieladressen an denen nur er der Empfänger oder Sender (für Aufnahme) ist. Diese Zieladressen sind nicht multicastfähig und können von anderen Anwendern bzgl. des gleichen Mediums nicht benutzt werden. Auch die Steuerung ist nur diesen einen Partner erlaubt.

3.2 Multicast

Multicast eignet sich für Livesendungen z.B. für Radio oder TV aber auch für Live-Präsentationen die Multimediaquellen enthalten. Je nach Charakter der Sitzung werden zwei Multicastmodi unterstützt:

- serverseitig
  Diese Art ist ausschließlich für Liveübertragung vom Interesse. Der Mediaserver bestimmt eine geeignete Multicastadresse und sendet an sie die Daten. Jeder Client der dazu stoßen will, erhält diese Adresse vom Server und übernimmt die Daten von dort. Der Benutzer hat dabei eine passive Rolle, denn er erhält nur die Informationen, die zur Laufzeit gesendet werden. Die Möglichkeit vorhergehende Ereignisse zu betrachten ist hier nicht gegeben.
- clientseitig
  Bei einer Livekonferenz wird oft auf schon vorhandene bzw. vorbereitete Dateien zurückgegriffen. Es soll jedoch vermieden werden, daß alle an der Konferenz beteiligten Partner, die neuen Informationen wieder von einer anderen Multicastadresse beziehen. Es kann nämlich passieren, das nicht alle der Beteiligten an diese Daten gleichzeitig zugreifen und somit einige von ihnen ausgeschlossen werden. Dieses Problem löst man indem der Konferenzführer, der auch ein Client ist, dem Server die schon bestehende Multicastadresse vorgibt. Dadurch wird erreicht, daß alle die neuen Informationen an der „alten“ Adresse empfangenden können.

4 RTSP-Nachricht

Während einer Sitzung werden zwischen den Server und Client Nachrichten ausgetauscht. Diese Nachrichten dienen dazu, die gewünschten Aktionen mitzuteilen, die von dem Partner auszuführenden sind. Auch Statusmeldungen z.B. des Servers, werden über RTSP-Nachrichten getragen. Wie eine Sitzung aufgebaut ist, werde ich im nachfolgendem Kapitel erläutern.

4.1 Format

RTSP basiert auf einem Textformat, der ISO 10646 Zeichen nutzt. D.h. nichts anderes, als daß die Zeichen 8-Bit kodiert sind. Zeilenumbrüche werden mit CRLF eingeleitet. CR oder LF können aber vom Empfänger auch einzeln als Zeilenumbruch interpretiert werden. Eine RTSP-Nachricht endet mit einer leeren Zeile.

Die Tatsache, daß dieser Protokoll texbasierend ist, bringt einige Vorteile mit sich:

- Übertragung
  Alle niederen Protokolle, die 8-Bit persistent sind, können RTSP-Nachrichten tragen.
- Parsen
  Standart MIME- oder HTTP-Parser können eingesetzt werden.
- Handling
  Optionale Parameter können einfach hinzugefügt werden.
- Nachrichtenerstellung
  Skriptsprachen wie z.B. Tcl oder Perl können RTSP-Nachrichten erzeugen.

4.2 Aufbau

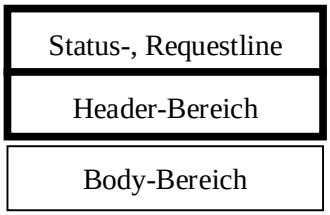


Abbildung 4-1: Aufbau einer RTSP-Nachricht

Jede RTSP-Nachricht wird nach dem in der Abbildung 4-1 angegebenen Schema aufgebaut. So enthält sie eine Status- bzw. Requestline und einen Header-Bereich. Diese zwei Bestandteile sind essentiell und kommen in jeder RTSP-Nachricht vor. Der Body-Bereich, hier auch separat dargestellt, ist optional und muß extra im Haeder-Bereich definiert werden. Als Besonderheit, muß der Body-Bereich von der eigentlichen RTSP-Nachricht mit einer leeren Zeile getrennt werden.

- Status-, Requestline
  Diese Zeile enthält je nach Nachrichtentyp, siehe Unterkapitel 4.3, Methoden bzw. Statusmeldungen, die eine Aktion beschreiben oder deren Ausführung mit einem Status quittieren.
- Header-Bereich
  In diesem Bereich werden Parameter aufgeführt, die z.B. bei der Ausführung der Methoden zusätzlich beachtet werden müssen. Einzelne Parameter werden jeweils in einer Zeile untergebracht, was eine Trennung einzelner Parameter mit sich bringt.
- Body-Bereich
  Enthält, sobald er im Haeder-Bereich definiert wird, protokollfremde Informationen. Es kann sich hierbei um Nachrichten anderer Protokolle aber auch um eigentliche Daten (z.B. Bruchstücke einer Audiodatei) handeln. Die Verarbeitung der Daten hängt natürlich von der client- bzw. serverseitigen Unterstützung des Dateityps ab.

4.3 Typen

In RTSP werden zwei Hauptnachrichtentypen unterschieden, nämlich Request- und Responsenachricht. Der dritte hier aufgeführte Typ ist nicht selbständig und bezieht sich auf Nachrichten, die einen Body einschließen.

4.3.1 Request

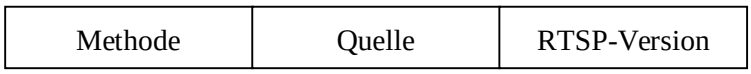


Abbildung 4-2: Aufbau einer Requestline

Eine Request-Nachricht dient dazu eine Sitzung zu initialisieren oder dem Partner mitzuteilen, welche Aktionen er anstoßen, ausführen soll. Sie besteht hauptsächlich aus eine Requestline und dem Header-Bereich.

Requestline wird in drei Teile gegliedert, die mit einem Leerzeichen voneinander getrennt werden, siehe Abbildung 4-2. Der erste Teil enthält eine Methode, siehe Unterkapitel 4.4, die sich auf die angegebene Quelle bezieht. Die Quellenangaben werden ihrerseits nur in der absoluten URL-Notation akzeptiert (z.B. `rtsp://audio.com/twister/audio.en`). Zusammenfassende Angaben, die

*-Zeichen enthalten (z.B. `rtsp://audio.com/twister/*`) werden verworfen. Zusätzlich zu diesen Informationen kommt die Mitteilung der RTSP-Version, nach der diese Nachricht erstellt wurde. Damit soll erreicht werden, daß die Bedeutungen der Methoden bzw. Parameter auch richtig interpretiert werden. Es kann nämlich passieren, daß in einer neueren Version dieses Protokolls einige Parameter, die Ausführung der Aktion, anders beeinflussen.

Der Header-Bereich beherbergt Paremeter, die für Requests sinnvoll sind, siehe [ScRL 98a]. Damit kann auch gegebenenfalls ein Body definiert werden. Auf die Einzelheiten des Bodys werde ich später eingehen.

4.3.2 Response

RTSP-Version	Status	Statusbeschreibung

Abbildung 4-3: Aufbau einer Statusline

Response-Nachrichten melden den aktuellen Status der Aktion, die nach dem Erhalt eines Requests ausgeführt wurde. Sie liefern auch die benötigten Informationen, die mit Hilfe von Parameter ausgedrückt werden oder im Body eingeschlossen sind.

Anders wie bei Requests, sieht die Statusline aus, siehe Abbildung 4-3. Als erstes wird die RTSP-Version angegeben, anschließen folgen der Status und die Statusbeschreibung. Der Status ist mit einer dreistelligen Zahl kodiert. Die dazugehörige Beschreibung ist eher für den Benutzer gedacht, daher enthält sie den Status im Klartext. Für die Anwendung ist jedoch die kodierte Version wichtig.

Die Statuskodierung ist in fünf Klassen eingeteilt. Jeder Klasse ist eine Zahl zugeordnet, die in der ersten Ziffer des Kodes enthalten ist. Die zwei verbliebenen Zahlen spezifizieren den Status genauer. Der Empfänger muß eigentlich nur die Statusmeldung klassifizieren können, um sich nach Ihr zu richten. Die Klasseneinteilung sieht wie folgt aus:

- **1xx**
  Hat einen informativen Charakter. Damit wird mitgeteilt, daß z.B. eine Aktion zur Zeit noch ausgeführt wird. Der Empfänger weiß dann, daß sein Request angekommen und verarbeitet wird, die Ausführung aber noch länger dauern kann.
- **2xx**
  Kodes dieser Klasse, melden eine erfolgreiche Ausführung der geforderten Aktion.
- **3xx**
  Weitere Requests sind nötig um die Aktion erfolgreich abzuschließen. Ist z.B. die Datei verschoben worden, so muß eine erneute Anfrage, diesmal aber mit richtigen Adresse, gesendet werden.
- **4xx**
  Fehler ist clientseitig aufgetreten.
- **5xx**
  Fehler ist serverseitig aufgetreten.

Header- und Body-Bereich können ebenfalls, wie bei Responses, Informationen tragen. Welche das im einzelnen sind, ist im [ScRL 98a] nachzulesen.

4.3.3 Entity

Wie schon vorhergehend erwähnt, ist dieser Typ keine selbständige Nachrichtenart. Vielmehr handelt es sich hier um die separate Spezifikation des Body- und des Header-Bereiches als Einheit.

Die Aufgabe des Bodys beschränkt sich auf den Transport fremder Informationen bzw. Daten. So können Video bzw. Audiodateien, aber auch Nachrichten anderer Protokolle, transportiert werden.

Diese Eigenschaft ist hilfreich, falls sich der Anwender hinter einer Firewall befindet, die z.B. UDP-Transport nicht erlaubt. Ist jedoch nicht die effizienteste Lösung, da die Daten in ISO 10646 Zeichen hin und zurück umgewandelt werden müssen.

Damit jedoch der Empfänger überhaupt weiß, daß nach der leeren Zeile, die eine RTSP-Nachricht standardmäßig abschließt, weitere Daten folgen, gibt es natürlich spezielle Header-Felder. Genaue Beschreibung findet man im [ScRL 98a]. Es sei aber gesagt, daß man mit Hilfe dieser Felder die Länge der angehängten Daten, ihre Kodierungsart und den Datentyp beschreiben kann.

4.4 Methoden

Methoden sind ausschließlich ein Bestandteil von RTSP-Requests. Außer OPTIONS, beziehen sich alle Methoden auf, die in der Request-Line angegebene Quelle. Sie beschreiben Aktionen, die der Partner auszuführen hat (z.B. sende Daten), dienen der Initialisierung des Transfers oder leiten den Abbruch der Verbindung bzgl. der entsprechenden Datei ein. Einige Methoden können außerdem eine Zustandsänderung initiieren. Auf die Funktionsweise des Zustandsmechanismus gehe ich erst im Kapitel 6 ein.

Anzumerken ist, daß nicht alle hier vorgestellten Methoden implementiert werden müssen. Sobald jedoch eine nicht unterstützte Methode im Request auftaucht, wird sie ignoriert und mit einen Errorcode in der dazugehörigen Antwort quittiert. Daraufhin kann der Empfänger der Antwort, sobald nicht anders möglich, die Sitzung beenden oder eine alternative Vorgehensweise wählen. ANNOUNCE, SETUP, PLAY, RECORD und TEARDOWN gehören zu den Funktionen, die in einer minimalen Implementierung des Servers und des Clients, zwingend vorgeschrieben sind.

4.4.1 OPTIONS

Das mögliche Verhalten des Kommunikationspartners hängt also auch von den unterstützten Methoden ab. Da laut der Konvention nicht alle realisiert werden müssen, ist es sinnvoll einen Überblick über die zu haben, die implementiert wurden. OPTIONS ermöglicht es, eine entsprechende Liste anzufordern. Client und auch Server sind berechtigt diese Methode in den Requests zu nutzen.

Als Besonderheit dieser Methode ist die Tatsache, daß sie als einzige, keinen Bezug zur einer Datei hat. Deshalb kann die Quellenangabe aus einem einzigen Zeichen und zwar dem Asterisk „*“ bestehen. (OPTIONS * RTSP/1.0) ist ein Beispiel für die entsprechende Requestline.

4.4.2 DESCRIBE

Jede RTSP-Sitzung wird nach einer Präsentationsbeschreibung aufgebaut, siehe Kapitel 5.1. Über die DESCRIBE-Methode kann diese Beschreibung angefordert werden. Anhand eines optionalen Parameters „Accept“, kann der Sender, ihm bekannte Beschreibungsarten, an dem Empfänger weiterleiten. Dieser hat sich nun daran zu halten, denn ansonsten kann die Information nicht richtig interpretiert werden. Ist „Accept“ nicht in der Anfrage enthalten steht dem Partner frei, die entsprechende Art zu wählen. Verständlicherweise wird diese Methode nur von Client angewandt.

4.4.3 ANNOUNCE

ANNOUNCE-Anfragen können vom Server und Client gleichermaßen verschickt werden, haben dabei aber verschiedene Bedeutungen.

Versendet der Client eine Anfrage mit ANNOUNCE-Methode, will er dem Server Einzelheiten über die von Ihm angestrebte Sitzung oder das Objekt, das in der Requestline angegeben, mitteilen. Diese Situation kann vorkommen, falls in einer laufenden Multicast-Präsentation, der steuernde Client eine neue Multimediaquelle wiedergeben will anstatt der alten. Dabei soll der Präsentationsserver darauf reagieren und diese übertragen, wobei die vorherige Datei verworfen wird.

Wird diese Nachricht vom Server verschickt, so aktualisiert er damit die Sitzungsbeschreibung bzw. Präsentationsbeschreibung zur Laufzeit. Wie der Client darauf reagiert hängt von der aktuellen Situation ab. Ändern sich dabei beispielsweise Auslieferungsadressen, die der Client nicht erreicht, kann er die Sitzung abbrechen.

4.4.4 SETUP

SETUP spezifiziert die Transporteinzelheiten, die bzgl. der Quelle bei der Übertragung gelten sollen. Diese Methode wird ausschließlich vom Client verwendet. Antwortet der Server mit einem Error, so kann der Client die Sitzung abbrechen lassen oder er sucht nach Alternativen und teilt sie gegebenenfalls mit einer weiteren SETUP-Nachricht dem Server mit. Damit kann eine Verhandlung geführt werden bzgl. der Transporteinstellungen. SETUP kann auch zur Laufzeit der Wiedergabe oder der Aufnahme versendet werden. Dies dient dann, zur laufenden Änderung von früheren Transportparameter.

4.4.5 PLAY

PLAY startet die Übertragung der Daten, entsprechend der mit SETUP vereinbarten Einstellungen. Solange jedoch der Client keine positive Bestätigung seiner SETUP-Nachrichten erhält, darf er kein PLAY senden.

Im allgemeinen bewirkt ein PLAY-Request, daß die Quelle in voller Länge übertragen wird. Die Wiedergabe kann natürlich jeder Zeit unterbrochen werden, was auch die Übertragung stoppt. Enthält ein PLAY-Request den Bereichsparameter „Range“, liefert der Server die Daten entsprechend diesen Bereichsangaben. Beispielsweise bewirkt die folgende Angabe (`Range: smpte=0:10:00-0:15:00`), daß die Quelle nur von der 10-ten Minute bis zu der 15-ten übertragen wird.

Erwähnenswert ist auch, daß mehrere PLAY-Request bzgl. einer Quelle hintereinander gesendet werden dürfen. Der Server führt eine PLAY-Queue, wo alle eingegangenen PLAY-Anforderungen gespeichert werden. Ist eine Anforderung abgearbeitet, so wird die Ausführung der nächsten angestoßen. Bei Livesendungen, können verständlicherweise keine Bereichsangaben gemacht werden, da in diesem Fall ein freies Navigieren nicht möglich ist.

PLAY wird solange ausgeführt bis die Quelle ganz übertragen wurde, ein PAUSE-Request ankommt, oder die PLAY-Queue leer ist. Dies setzt natürlich voraus, daß die Sitzung nicht zwischendurch abgebrochen wurde.

4.4.6 RECORD

RECORD verhält sich ähnlich wie die PLAY-Methode. So können Bereiche festgelegt werden, in denen die Aufnahme stattfindet soll. Läuft eine Sitzung bereits, nimmt der Server die Daten unverzüglich auf, ansonsten wartet er die in SETUP angegebenen Aufnahmepunkte ab. Der Ort der Speicherung wird vom Client bestimmt, kann aber vom Server verändert werden. Anschließend muß jedoch der Client über den neuen Ort in Kenntnis gesetzt werden.

Im Gegensatz zu PLAY-Methode ist hier keine RECORD-Queue vorgesehen. Daher werden aufeinanderfolgende RECORD-Requests ignoriert.

4.4.7 PAUSE

PAUSE veranlaßt den Server die Medienübertragung zu unterbrechen. Die Ressourcen werden aber nicht gleich freigegeben, so daß nach einer gewissen Zeit die Wiedergabe oder Aufnahme erneut gestartet werden, ohne daß eine Initialisierung der Quelle mit SETUP nötig wäre. Zusätzlich annulliert PAUSE alle in der PLAY-Queue eingetragenen Anfragen.

Der Bereichsparameter „Range“ erlaubt es den Zeitpunkt der Pause exakt festzulegen. Damit kann die PAUSE-Methode schon vor der eigentlichen Unterbrechung gesendet werden und wird trotzdem vom Server akzeptiert und zu richtigen Zeit ausgeführt. Liegt dieser Punkt außerhalb der Spiellänge wird ein Servererror gemeldet und wird kein „Range“ angegeben, unterbricht der Server die Übertragung auf der Stelle.

Hat der Server doch noch Daten geschickt, die nach dem Pausenzeitpunkt angeordnet sind, muß der Client selber diese ausfiltern um sie nicht mehr wiederzugeben.

4.4.8 TEARDOWN

Mit Hilfe dieser Methode werden alle Aktivitäten beendet, die sich auf die angegebene Quelle beziehen. D.h. keine nachfolgenden Anfragen wie PLAY, RECORD werden bearbeitet. Nur die erneute Initialisierung z.B. über SETUP ermöglicht wieder den Zugriff. Wird ein Containerfile in der Requestline angegeben, beendet der Server alle laufenden Vorgänge, die er für die enthaltenen Dateien gestartet hat.

4.4.9 GET_PARAMETER

GET_PARAMETER erlaubt es den Sender die Werte der aufgeführten Parameter anzufordern, deren Anzahl unbegrenzt ist. Sowohl der Server wie auch der Client dürfen diese Methode nutzen. Die gewünschten Parameter werden im Body der Nachricht angegeben. Der Empfänger bringt in seiner Antwort die Parameter mit ihrem Werten ebenfalls im Body unter.

4.4.10 SET_PARAMETER

Wie auch GET_PARAMETER, können beide Partner diese Methode in ihren Requests versenden. Sie bewirkt, daß der Empfänger diesen Parameterwert in dem weiteren Verlauf der Sitzung berücksichtigt, sobald er von ihm unterstützt wird. Anders sieht es mit der Anzahl der Parameter aus. Hier darf jeweils nur ein Parameter mit dem angegebenen Wert versendet werden. Dies gibt dem Empfänger die Möglichkeit den Parameter oder seinen Wert in einer Antwort abzulehnen. Wichtig ist, daß die Transportparameter nicht über SET_PARAMETER gesetzt werden dürfen und nur der SETUP-Methode vorbehalten sind.

4.4.11 REDIRECT

Diese ist die einzige der Methoden, die ausschließlich vom Server geschickt wird. Damit kann er dem Client mitteilen, daß weitere Informationen von andere Stelle bezogen werden sollen, z.B. von anderen Server. Kann oder will der Client diese Informationen beziehen, so muß er die bestehende Verbindung trennen und eine neue mit dem angegebenen Partner initiieren. Zusätzlich hat der Server die Möglichkeit mit Hilfe des optionalen Parameters „Range“ den Zeitpunkt zu nennen, ab wann diese Umleitung in der noch bestehenden Sitzung gilt.

4.5 Parameter

Schon einige male habe ich in dem vorhergehenden Kapitel, Parameter im Bezug auf Methoden erwähnt. Ihre primäre Aufgabe ist es, die Angeben in der Request bzw. Statusline zu ergänzen. Mit diesen zusätzlichen Informationen präzisieren sie das Verhalten des Empfängers. Dieser Mechanismus stützt sich auf die Werte, die Parameter annehmen können. Solche Werte können in verschiedenen Formaten auftreten. So z.B. enthält der „Session“-Parameter einen ganzzahligen Wert der Sitzung-ID, der „Transport“-Parameter dagegen beschreibt seinen Wert mit Hilfe von Zeichenketten und Zahlenwerten. Als ein gutes Beispiel dafür, daß benutzte Parameter das Verhalten des Empfängers ändern können, dient der „Range“-Parameter, siehe Kapitel 4.4.5. Ein PLAY-Request ohne diesen Parameter

und seinen Wert initiiert die Übertragung der Quelle in voller Länge. Gibt man jedoch den gewünschten Bereich an, so wird nur der verlangte Ausschnitt gesendet.

Parameter selber werden im Header-Bereich untergebracht. Dort werden sie mit Zeilenumbruch voneinander getrennt. Es gibt jedoch Parameter, die mehrere Informationen tragen können. Dann nämlich, wird jeder einzelne von ihnen mit (;)-Zeichen getrennt. Auch Body-Bereich kann Parameter enthalten, siehe Kapitel 4.4.8 und 4.4.9.

Es gibt eine große Anzahl an Parametern, die noch ständig erweitert werden können. Da diese Erweiterung innerhalb einer RTSP-Version erlaubt sind bzw. nicht alle Parameter zwingend unterstützt werden müssen, können auch sie von Empfänger verworfen werden. Nichts desto trotz möchte ich die Bedeutung einiger wichtiger Parameter erläutern, die in fast jeder Nachricht enthalten sind.

- Cseq
  Ist ein Sequenzparameter und muß in jeder Nachricht enthalten sein. Er gibt die Sequenznummer der Nachricht an, anhand welcher ihre Reihenfolge kontrolliert wird. So werden Nachrichten mit höheren Nummer erst zu entsprechenden Zeitpunkt abgearbeitet. Dies ist nötig, da eine Steuerungsverbindung, über die Nachrichten übertragen werden, dynamisch aufgebaut ist und sich ständig ändern kann. Sie Sequenznummer wird in der ersten RTSP-Nachricht der Sitzung mit dem Wert 1 belegt und dann mit jedem weiteren Request erhöht. Die zu einem Request gehörende Response-Nachricht enthält die selber Nummer. Dadurch kann deren Zuordnung durchgeführt werden.
- Session
  Gibt die Sitzung-ID der aktuellen RTSP-Sitzung an. Der Server kann denn anhand dieser Nummer die Zugehörigkeit der Nachricht zu einer bestimmten Sitzung bestimmen.
- Transport
  Mit diesem Parameter werden die Einzelheiten des Transports festgelegt, so z.B. Übertargungsprotokoll (RTP oder UDP), Modi (Multicast oder Unicast).
- Content-..
  Informiert den Empfänger, daß ein Body nach der abschließenden Zeile kommt. Darüber hinaus kann z.B. die Länge, Codierungsart oder Typ des Body angegeben werden

5 RTSP- Sitzung

In diesem Kapitel möchte ich den Verlauf einer RTSP-Sitzung in ihren einzelnen Phasen vorstellen. Die Reihenfolge der Phasen entspricht der unten aufgeführten. Zusätzlich werde ich auch die Protokolle ansprechen, die mit RTSP kooperieren oder von RTSP benutzt werden.

5.1 Beschaffungsphase

Eine Multimediasitzung enthält Quellen, die den Inhalt tragen. Diese können ihrerseits, siehe Kapitel 1, über mehrere Server verteilt werden. Damit sie, beispielsweise wiedergegeben werden können, muß mindestens die Quelle selber und der Server wo sie abgelegt wurde, bekannt sein. Diese Phase dient dazu solche Informationen zu beschaffen.

Eine Präsentation kann mit Hilfe von sogenannter **Präsentationsbeschreibung** definiert werden. Wie im einzelnen solche Beschreibung auszusehen hat, ist vom Server abhängig und fällt nicht unter RTSP. Anzumerken ist daß solche Beschreibungen für den Server eindeutige Informationen enthalten

sollen, anhand welcher er alle für die Präsentation relevanten Informationen an dem Client weiterleiten kann. D.h. eine Präsentationsbeschreibung muß, wie schon oben angesprochen, die Quellen und deren Ort enthalten.

Die Kommunikation in dieser Phase kann mit RTSP-Nachrichten abgewickelt werden, muß aber nicht. Auch mit Hilfe von HTTP kann eine Präsentationsbeschreibung eingeholt werden, siehe [Down 99, ScRL 98a]. Womit auch immer die Informationen beschaffen werden, die Vorgehensweise ist gleich. Als erstes muß der Client wissen, wo sich eine Präsentationsbeschreibung befindet. Dies kann er z.B. aus einem HTTP-Tag , wie in der Abbilgung 5-1 dargestellt, entnehmen.

```
<session>
<group>
<track src="rtsp://audio.mtv.com/movie">
<track src="rtsp://video.mtv.com/movie">
</group>
</session>
```

Abbildung 5-1: HTTP-Tag

Daraufhin wird eine Nachricht an dem Server geschickt, die ihm auffordert diese Information zu senden. Bei RTSP-Response werden solche Präsentationsbeschreibungen meist im Body plaziert und können dem SDP (Session Description Protokoll) entsprechend aufgebaut werden. Wie SDP im einzelnen aussieht, ist im [HaJa 98a] nachzulesen.

Die Abbildung 5-2 zeigt einen möglichen Nachrichtenaustausch in dieser Phase. C bezeichnet den Client und M steht für den Multimediaserver. Die Pfeile zeigen die Übertragungsrichtung der Nachricht.

```
C->M: DESCRIBE rtsp://foo/twister RTSP/1.0
CSeq: 1

M->C: RTSP/1.0 200 OK
CSeq: 1
Content-Type: application/sdp
Content-Length: 164

v=0
o=- 2890844256 2890842807 IN IP4 172.16.2.93
s=RTSP Session
a=control:rtsp://foo/twister
m=audio 0 RTP/AVP 0
```

Abbildung 5-2: Nachrichtenaustausch in der Beschaffungsphase

5.2 Verhandlungs- / Transportinitialisierungssphase

Sind die Einzelheiten der Sitzung nach der Beschaffungsphase dem Client bekannt, kann er die Vorbereitung des Transports angehen. Dabei müssen je nach Modus, die Client-, Server-Adressen ausgetauscht werden, Transportprotokolle festgelegt werden, Sitzung-IDs, Abbruchbedingungen usw. bestimmt werden. All diese Informationen sind essentiell und werden über die SETUP-Nachrichten übermittelt. In dieser Phase haben beide Partner die Möglichkeit, sich abzustimmen. Der Client teilt z.B. als erster, die von ihm unterstützten Protokolle mit. Der Server kann die Anforderungen des Cli-

ents, sobald er sie erfüllen kann, bestätigen. Ist das nicht der Fall, kann er sie mit einer Errormeldung verwerfen. Spätestens hier kann der Client erkennen, ob eine Sitzung überhaupt möglich ist oder nicht.

Wird diese Phase erfolgreich abgeschlossen, kann die Wiedergabe bzw. die Aufnahme gestartet werden.

5.3 Wiedergabe- / Aufnahmephase

Alle Vorgänge dieser Phase stützen sich auf die vorhergehend festgelegten Bedingungen. Die Anwendung der PLAY- oder RECORD-Methoden ist hier nicht vom großen Interesse, da sie auch schon im Kapitel 4 erwähnt wurden. Viel mehr sind der eigentliche Verlauf der Sitzung und die in dieser Phase aufgebauten Verbindungen wichtiger.

Erwähnt habe ich schon, daß RTSP der Steuerung der Streams dient und selber keine Daten überträgt. Daher bedienen sich der Server und Client anderer geeigneter Protokolle wie z.B. RTP oder UDP, mit welchen solche Daten versendet werden können. In der Abbildung 5-3 ist ein möglicher Sitzungsaufbau dargestellt, der typisch für diese Phase ist.

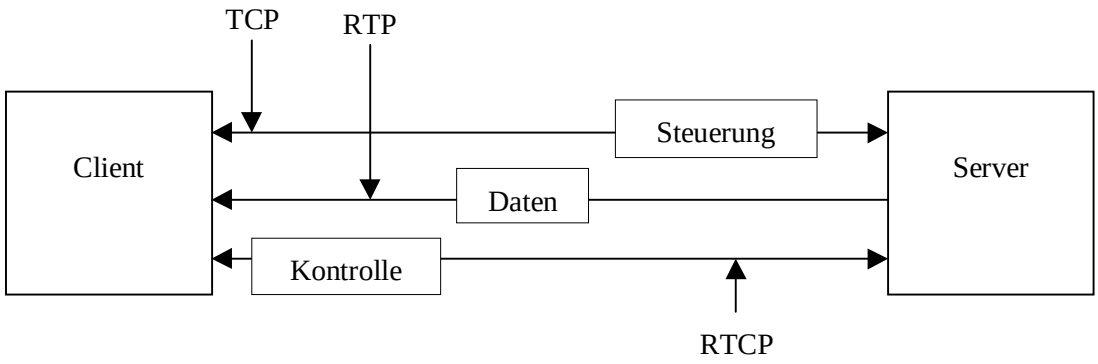


Abbildung 5-3: Sitzungsaufbau in der Wiedergabe- bzw. Aufnahmephase

In diesem Beispiel werden drei Verbindungen unabhängig voneinander aufgebaut.

Über die Steuerungsverbindung werden die RTSP-Nachrichten ausgetauscht. Deswegen zeigen die Pfeile in beide Richtungen. Da sich hier die Partner einer TCP-Pipeline bedienen, kann nicht garantiert werden, daß sie über die Dauer dieser Phase bestehen bleibt. RTSP toleriert jedoch wechselnde Steuerungsverbindung, da die Partner über die Sequenznummer die Nachrichten synchronisieren können. Darüber hinaus unterhalten beide eine Zustandsmaschine, siehe Kapitel 6, die den Stand der Verbindung protokolliert, wodurch sie ständig über die relevanten Aktivitäten des anderen informiert sind.

Der Datenkanal wird hier über RTP abgewickelt. Die Nutzung von RTP ist hier nicht zwingend vorgeschrieben, bittet sich aber an, da dieses Protokoll speziell für die Übertragung von Streamingmedien entwickelt wurde, siehe [ScCF 96a]. So ist hier auch die Reihenfolgesicherheit integriert, die für Multimedia-Quellen von großer Bedeutung ist. In Verbindung zu dieser Datenübertragung ist der RTCP Kanal zu erwähnen. Über den wird der Datenfluß an sich kontrolliert, um z.B. bei Paketverlust ein erneutes senden zu initialisieren. Im [ScCF 96a] werden dies Vorgänge detailliert beschrieben.

5.4 Abbruchphase

Wurde die Wiedergabe aller Quellen erfolgreich durchgeführt, ist die Sitzung noch nicht abgeschlossen. Der Server ist im Wartezustand und erwartet weitere Instruktionen. Ist jedoch die eigentliche Präsentation für den Client beendet, möchte er keine weiteren Dienste des Server bzgl. diese Präsentation

in Anspruch nehmen. Die Abbruchphase wird seitens des Clients eingeleitet. Mit Hilfe der Methode TEARDOWN kann er die bestehende Sitzung beenden.

Natürlich wartet der Server nicht ewig auf den Abbruchbefehl und kann nach einer gewissen Zeit, die auch während der Verhandlungsphase bestimmt werden kann, die Sitzung selbständig beenden.

Abgesehen von diesen zwei Fällen, kann auch ein Abbruch vom Client während der Präsentation angefordert werden. Der Server verhält sich dann wie beim Beenden nach der vollständigen Übertragung und unterbricht alle Verbindungskanäle.

6 Zustandsmodell

Zustandsmodell gehört zu den wichtigsten Merkmalen des RTSP. Hier unterscheidet er sich von den meisten anderen Protokollen. Im Kapitel 5.3 habe ich diesen Mechanismus bereits kurz angesprochen.

Die Motivation der Entwickler die Protokollzustände einzuführen, war es den Server und den Client möglichst unabhängig voneinander arbeiten zu lassen. Dadurch sind die Kommunikationspartner nicht mehr auf dauerhafte Steuerungsverbindung während einer Sitzung angewiesen und können sich auf eventuelle Störungen im Netz besser einstellen.

Im allgemeinen beschreibt die Client- bzw. Server-Zustandsmaschine das Verhalten des Protokolls von der Initialisierung bis zur Terminierung einer Sitzung. Jeder Zustand bezieht sich hier auf jeweils nur ein Objekt. Als Objekte werden die in den Requests angegebenen Quellen in Verbindung mit den RTSP Sitzung-IDs definiert, wodurch eine eindeutige Zuordnung gewährleistet ist. Werden in einer Sitzung mehrere Mediaquellen gleichzeitig genutzt, so wird für jeder Quelle dieser Sitzung ein Objekt mit seinen Zuständen, vom Server sowie vom Client, angelegt und geführt. Die Containerfiles, die mehrere solche Quellen enthalten können, werden als ein Objekt gesehen, so daß keine Zustände mehr für die einzelnen Medien mitzuführen sind.

Anhand des aktuellen Zustandes, wissen die Beteiligten in welcher Phase sie sich und die anderen Teilnehmer gerade befinden und wie sie auf die ankommenden Informationen reagieren sollen. Ist beispielsweise der Server und der Client in der Wiedergabephase, so kann sich der Server sicher sein, daß alle von ihm über den Datenkanal geschickten Daten, Paketverlust ausgeschlossen, vom Client richtig interpretiert und wiedergegeben werden. Dabei ist der Server nicht darauf angewiesen, daß der Client ständig für jedes Paket eine Empfangsbestätigung sendet, was den Nachrichten Verkehr deutlich reduziert.

6.1 Funktionsweise

Die Funktionsweise der Zustandsmaschinen, hängt davon ab, ob sie vom Server oder Client geführt werden und im welchen Modus die Sitzung stattfindet. Im allgemeinen werden die Zustandsänderungen aufgrund, von den im Requests angegebenen Methoden und den daraufhin ausgeführten Aktionen, initiiert. Nicht jede Methode kann eine Zustandsänderung herbeiführen. So sind bzgl. des Zustandsmechanismus die Methoden SETUP, PLAY, RECORD, TEARDOWN und PAUSE vom Interesse.

6.1.1 Client

In der Abbildung 6-1 sind alle möglichen Zustände dargestellt, die ein Objekt aus der Sicht des Clients annehmen kann. Bei einer Multicastsitzung, wo kein explizites SETUP nötig ist (z.B. bei einer Liveübertragung), ist Ready der erste Zustand. In diesem Fall gibt es nur zwei gültige Zustände, nämlich Ready und Playing bzw. Recording. Der Client verweilt dann in Playing, solange bis die Übertragung andauert, oder er bricht mit TEARDOWN die Verbindung völlig ab.

Die Zustandsänderung eines Objekts tritt bei Client nur dann ein, falls der Server eine positive Antwort auf die vorhergehende Anfrage gesendet hat. D.h. daß in der Statusline der Antwort die Kodierung des Status aus der Klasse 2xx stammen muß, siehe Kapitel 4.3.3. Meldet der Server mit 4xx einen Fehler, bleibt der alte Zustand bestehen und falls der Statuscode 3xx enthält wird das Objekt in den Init-Zustand versetzt.

Zustand	Bedeutung
Init	SETUP-Request wurde gesendet, warten auf Antwort des Servers.
Ready	PAUSE / SETUP-Response wurde empfangen und ist positiv markiert (2xx).
Playing	PLAY-Response wurde empfangen und ist positiv markiert (2xx)
Recording	RECORD-Response wurde empfangen und ist positiv markiert (2xx)

Abbildung 6-1: Zustände des Clients

Die PAUSE-Methode hat die Eigenschaft, daß sie zwar „jetzt“ versendet wird, die entsprechende Aktion aber erst zum anderen Zeitpunkt durchgeführt werden kann. Um jedoch eine Empfangsbestätigung zu haben, sendet der Server ein Response mit dem Statuscode 1xx. Nach Erhalt dieser Nachricht ändert der Client seinen Zustand noch nicht und arbeitet wie gehabt weiter. Erst wenn der Server den Befehl zu der richtigen Zeit ausgeführt hat, sendet er eine mit Status 2xx quittierte Antwort, die den Client dazu veranlaßt seine aktuellen Zustand in den Zustand Ready zu überführen.

Ein Zustandsdiagramm in der Abbildung 6-2 zeigt die Zustandsanordnung und die einzelnen Übergänge bzgl. der Methoden. TEARDOWN führt in jedem Fall zur Rückstufung in den Init-Zustand, daher habe ich diese Übergänge weggelassen.

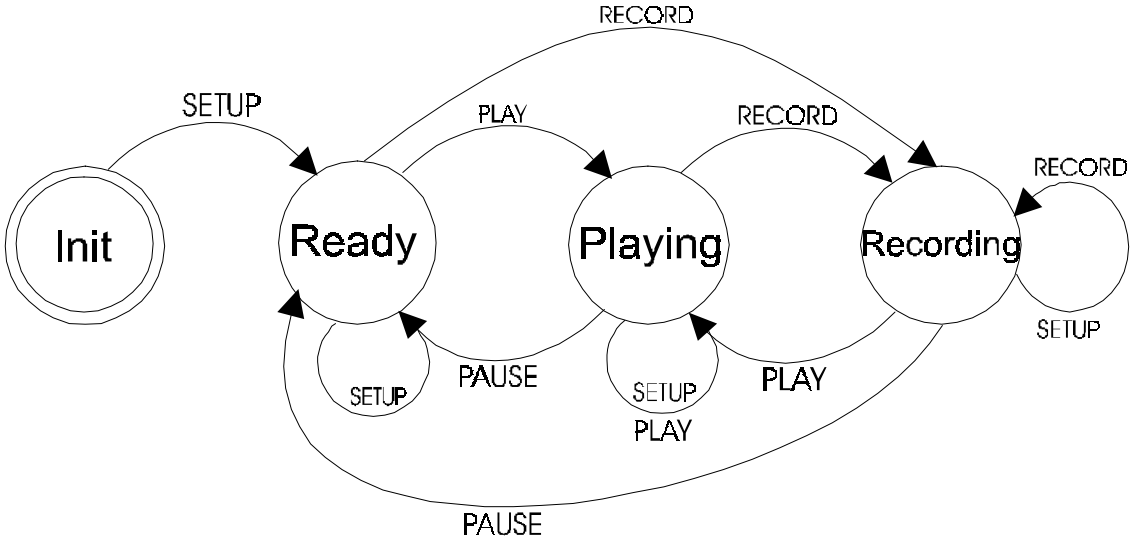


Abbildung 6-2: RTSP Zustandsübergangsdiagramm

6.1.2 Server

Wie auch schon bei Client, können die Serverobjekte vier verschiedene Zustände annehmen, siehe Abbildung 6-3. Bei Multicastsitzungen muß der Server aufgrund seiner Aufgabe, als „Datenlieferant“

zusätzlich zu den Ready- und Playing/Recording-Zuständen, noch den Init-Zustand mitführen. Dieser ist nötig, da sich jedes Objekt, das der Server verwaltet, standardmäßig im Init-Zustand befindet, ohne daß ein Request empfangen wurde.

Nach einer erfolgreichen Ausführung der Aktion, die von dem Client im Request verlangt wurde, ändert der Server dem Zustand des betreffenden Objekts, entsprechend dem Zustandsdiagramm in der Abbildung 6-2, und sendet eine positive Antwort an den Client. Muß der Server einen Fehler melden, so bleibt er in dem aktuellen Zustand und verschickt eine Antwort mit dem Statuscode 4xx. Sendet er in seiner Antwort den Code 3xx, setzt er den Zustand, wie auch der Client, in den Init-Zustand.

Zustand	Bedeutung
Init	Wartezustand, kein Request empfangen.
Ready	SETUP/PAUSE-Request Empfangen, Aktion erfolgreich durchgeführt, Response mit 2xx-Code gesendet.
Playing	PLAY-Request empfangen und bestätigt, die Daten werden gesendet.
Recording	RECORD-Request empfangen und bestätigt, die Daten werden empfangen und aufgenommen.

Abbildung 6-3: Zustände des Srvers

Da der Server nicht auf die Empfangsbestätigung des Clients bzgl. der gesendeten Pakete wartet, kann es vorkommen, daß während der Sitzung der Partner nicht mehr präsent ist. Der Server kann das natürlich nicht feststellen und sendet seine Daten weiter. Solches Verhalten könnte dann zu unnötigen Belastung des Server führen, falls er mehrere Sitzungen bedienen muß, die eigentlich nicht mehr bestehen. Um das zu vermeiden gibt es eine feste Timeout-Zeit, die den Sendeintervall bestimmt in dem der Server „sorglos“ die Daten überträgt. Kommt innerhalb dieser Zeit kein weiterer Request des Clients wird die Sitzung mit allen Kanälen abgebrochen. Diese Zeit wird standardmäßig auf 60 Sekunden festgesetzt. RTSP gibt den Client jedoch die Möglichkeit diese Zeit selber zu bestimmen oder wenigstens als Wunschzeit dem Server mitzuteilen. Dies geschieht mit dem „timeout“-Wert, der mit dem „Session“-Parameter eingebunden werden kann, siehe [ScRL 98a].

Was die Funktion PAUSE angeht, verhält sich der Server dem Client entsprechend. Sobald es gewünscht wird, daß die Aktion erst zum späteren Zeitpunkt durchgeführt wird, z.B. Pause erst nach 2 Minuten, arbeitet der Server wie bisher und liefert seine Daten aus. Kommt der Zeitpunkt für die Pause, führt er die Aktion durch, sendet ein entsprechendes Response und begibt sich in den Ready-Zustand.

7 Szenario

Nach dieser kurzen Vorstellung des RTSP möchte ich ein kleines Beispielszenario präsentieren, daß zum besseren Verständnis dieses Protokolls dienen soll. Die Abbildungen in diesem Abschnitt enthalten die tatsächliche RTSP-Nachrichten, um den möglichen Nachrichtenverkehr besser darzustellen.

7.1 Beschreibung

Des Szenario beschreibt eine Unicastsitzung. Client **C** möchte einen Videobeitrag ansehen, zu dem es eine extra Audioquelle gibt. Die Beschreibung dieser Sitzung liegt auf dem Präsentationsserver **P.** Die einzelnen Quellen, d.h. die Video- und die dazugehörige Audiodatei befinden sich auf verschiedenen Multimediaservern und zwar auf dem Videoserver **V** und dem Audioserver **A**. Der Hinweis auf diesen Beitrag entnahm der Client aus einer WWW-Seite. Wie ein solcher HTML-Tag aussieht, habe ich bereits in der Abbildung 5-1 angegeben.

Im folgenden werde ich auf die einzelnen Phasen eingehen, den Nachrichtenaufbau beschreiben sowie die entsprechenden Zustandsänderungen erwähnen.

In den Abbildungen bezeichnen die Pfeile die Übertragungsrichtung der Nachricht. Die Einzelnen Partner werden abgekürzt mit Großbuchstaben dargestellt.

7.2 Beschaffungsphase

```
C->P: GET /twister.sdp HTTP/1.1
Host: www.example.com
Accept: application/sdp

P->C: HTTP/1.0 200 OK
Content-Type: application/sdp
v=0
o=- 2890844526 2890842807 IN IP4 192.16.24.202
s=RTSP Session
m=audio 0 RTP/AVP 0
a=control:rtsp://audio.com/twister/audio.en
m=video 0 RTP/AVP 31
a=control:rtsp://video.com/twister/video
```

Abbildung 7-1: Nachrichtenaustausch in der Beschreibungsphase

Wie schon oben erwähnt, entnahm der Client den Hinweis auf den Beitrag aus einer HTML-Seite. Der entsprechender Tag enthielt die Ortsangabe, wo die Präsentationsbeschreibung verlangt werden kann. Sie liegt nämlich auf dem Präsentationsserver **P**. Der Client sendet eine Anfrage im HTTP-Format, wobei er angibt, daß die Beschreibung nach SDP aufgebaut werden soll. Der Server **P** liefert die gewünschte Beschreibung, die er im Nachrichtenbody integriert. Aus dieser Beschreibung entnimmt der Client, daß diese Sitzung über zwei Quellen verfügt, die auf verschiedenen Servern verteilt sind und das es sich hier um eine RTSP-Sitzung handelt.

In der Praxis sind die ausgetauschten Nachrichten im HTTP-Format. Der Client führt außerdem noch keine Zustandsprotokolle für die Quellen, da er erst hier nähere Informationen über sie erhält.

7.3 Transportinitialisierungssphase

Der Client ist nun gezwungen beide Server separat anzufragen, da für diese Präsentation keine Aggregationskontrolle vorgesehen ist. Als erstes kontaktiert er den Audioserver und übergibt im die Transporteinzelheiten, wie Portnummer, Modus (hier Unicast), und den gewünschten Übertragungsprotokoll für die Daten (RTP). Dabei setzt er den Zustand dieses Audioobjektes auf Init. Er wartet die bestätigende Antwort des Audioservers **A** ab und versendet anschließend eine weitere Anfrage an dem Videoserver **V**. Wurde auch diese Anfrage positiv beantwortet, kann er mit der Wiedergabe beginnen.

Sobald die Server **A** und **V** die Befehle ausgeführt haben und keine Fehler melden, setzen sie die Zustände der einzelnen Objekte, die durch die Quellenangabe und der Sitzung-ID eindeutig festgelegt sind, auf Ready. Der Client **C** setzt seinerseits nach dem Empfang der positiven Antworten der einzelnen Server auch die Zustände der entsprechenden Objekte auf Ready.

Die dazugehörigen Nachrichten sind in der Abbildung 7-2 dargestellt. Diese Nachrichten werden schon nach RTSP aufgebaut.

```
C->A: SETUP rtsp://audio.com/twister/audio.en RTSP/1.0
CSeq: 1
Transport: RTP/AVP/UDP;unicast;client_port=3056-3057

A->C: RTSP/1.0 200 OK
CSeq: 1
Session: 12345678
Transport: RTP/AVP/UDP;unicast;client_port=3056-
3057;server_port=5000-5001

C->V: SETUP rtsp://video.com/twister/video RTSP/1.0
CSeq: 1
Transport: RTP/AVP/UDP;unicast;client_port=3058-3059

V->C: RTSP/1.0 200 OK
CSeq: 1
Session: 23456789
Transport: RTP/AVP/UDP;unicast;client_port=3058-
3059;server_port=5002-5003
```

Abbildung 7-2: Nachrichtenaustausch während der Initialisierung der Quellen

7.4 Wiedergabephase

Nach der erfolgreichen Transportinitialisierung kann der Client **C** mit der Wiedergabe beginnen. Die einzelnen Server müssen weiterhin separat angesprochen werden, siehe Abbildung 7-3. So versendet der Client seine Requests an den Video- und Audioserver, wobei er hier mit dem Parameter „Range“ mitteilt, daß der Start der Wiedergabe erst ab der 10-ten Minute erfolgen soll. Die Server **A** und **V** sowie der Client **C** ändern die Objektzustände in den Playing-Zustand.

Die Anfragen des Clients werden sequentiell durchgeführt. Es kann daher der Eindruck entstehen, daß deswegen die Quellen Zeitversetzt wiedergegeben werden. Dies ist jedoch nicht der Fall, da der Client die Ankommenden Pakete synchronisiert und entsprechen abspielt. Die Tatsache, daß die Daten an sich nicht zur gleichen Zeit ankommen, beeinflußt im Prinzip nicht den Betrieb. Sobald aber eine der Server seine Daten mit zu großen Verzögerung sendet, stoppt der Client seine Wiedergabe und wartet bis die fehlenden Pakete angekommen sind und setzt seine Tätigkeit fort.

7.5 Abbruchphase

Der Client möchte nun die Wiedergabe beenden. Daher leitet er die Abbruchphase ein. Wie schon in den vorhergehenden Phasen versendet er entsprechenden Nachrichten an die beteiligten Server. Diese Nachrichten enthalten die TEARDOWN-Methode mit der Angabe der Sitzungs-ID. Anhand dieser ID kann der Server die zu der Sitzung gehörenden Verbindung bestimmen und unterbrechen. Anschließend sendet er eine bestätigende Antwort. Damit setzten die einzelnen Server ihre Objektzustände auf den Init-Zustand. Der Client bricht seinerseits die Wiedergabe ab und gibt die Kanäle wieder frei. Die

```
C->V: PLAY rtsp://video.com/twister/video RTSP/1.0
CSeq: 2
Session: 23456789
Range: smpte=0:10:00-

V->C: RTSP/1.0 200 OK
CSeq: 2
Session: 23456789
Range: smpte=0:10:00-0:20:00
RTP-Info: url=rtsp://video.com/twister/video;seq=12312232;
rtptime= 78712811

C->A: PLAY rtsp://audio.com/twister/audio.en RTSP/1.0
CSeq: 2
Session: 12345678
Range: smpte=0:10:00-

A->C: RTSP/1.0 200 OK
CSeq: 2
Session: 12345678
Range: smpte=0:10:00-0:20:00
RTP-Info: url=rtsp://audio.com/twister/audio.en;seq=876655;
rtptime=1032181
```

Abbildung 7-3: Nachrichten in der Wiedergabephase

Sitzung ist nun abgebrochen. Um erneut auf die Daten zuzugreifen, muß der Client mit der Initialisierungsphase die Sitzung aufbauen.

```
C->A: TEARDOWN rtsp://audio.com/twister/audio.en RTSP/1.0
CSeq: 3
Session: 12345678

A->C: RTSP/1.0 200 OK
CSeq: 3

C->V: TEARDOWN rtsp://video.com/twister/video RTSP/1.0
CSeq: 3
Session: 23456789

V->C: RTSP/1.0 200 OK
CSeq: 3
```

Abbildung 7-4: Nachrichtenaustausch in der Abbruchphase

8 Ausblick

Wie wir gesehen haben stellt RTSP eine mächtige Plattform für Anwendungen, die Multimediaquellen über die Netze nutzen wollen. Damit ist Ressourcen sparender Umgang mit den kontinuerlichen Medien möglich, der weit größeren Kreis der Interessierten ansprechen kann als bisher der Fall war. Wichtige Präsentationen, Vorträge bzw. Vorlesungen können so einen Publikum präsentiert werden,

das sonst gar nicht in deren Genuß kommen würde. Firmen ist es möglich Schulungen für alle Mitarbeiter zur Verfügung zu stellen usw.

Mittlerweile wurde RTSP von IETF standardisiert und viele Firmen bauen ihre Produkte auf diesen Standard auf. Einer der bekanntesten sind z.B. Netscape, 3Com Real_Media und Apple. Auch die meisten Browser unterstützen RTSP, was seiner weiten Verbreitung bestimmt dienlich sein sollte.

Literaturverzeichnis

[Down 99] Downey, Richard
http://www.ug.ecs.soton.ac.uk/~rtd195/rtsp/index.html [Stand: 18.09.1999]

[HaJa 98a] Handley, M. - Jacobson, V. (Eds.):
SDP: Session Description Protocol
RFC 2327, Network Working Group 1998

[Real 99] Real Networks
http://www.real.com/devzone/library/fireprot/rtsp/ [Stand: 23.10.1999]

[Reib 99] Reibold, Holger
http://www.pcspiele.de/internet/artikel/tech/199910/rtsp_00-wc.html [Stand: 16.10.1999]

[ScCF 96a] Schulzrinne, H. - Casner, S. - Frederick, R. (Eds.):
RTP: A Transport Protocol for Real-Time Applications
RFC 1889, Network Working Group 1996

[Schi 98] Schichl, Gunter
http://www.forwiss.uni-passau.de/~schichl/seminar/inhalt.html [Stand: 23.01.1998]

[ScRL 98a] Schulzrinne, H. - Rao, A. - Lanphier, R. (Eds.):
Real Time Streaming Protocol
RFC 2326, Network Working Group 1998

[Schw 98] Schwing, Peter
http://www.ito.tu-darmstadt.de/edu/sem-iuk-w97/aus6/inhalt.html [Stand: 16.04.1998]

Das Real-Time Transport Protokoll

Daniel Herche

FB Informatik, Uni Dortmund
Dherche@nef.wh.uni-dortmund.de

Zusammenfassung

Diese Ausarbeitung beschäftigt sich mit dem Real-Time Transport Protokoll (RTP) und dem dazugehörenden Kontrollprotokoll (RTCP). Zu Beginn werden Einsatzgebiete und daraus resultierende Probleme von Transportprotokollen diskutiert, sowie ein Szenario multimedialer Sitzungen vorgestellt. Der Aufbau der Pakete von RTP und RTCP inklusive der verschiedenen Untertypen von Paketen werden beschrieben.
Das Hauptaugenmerk richtet sich jedoch auf das Zusammenspiel der Protokolle bezüglich der Einhaltung von *Quality of Service* (QoS) Parametern und welche Rückschlüsse der Anwender aus der Vielfalt der Daten ziehen kann, die von Paketen des Kontrollprotokolls zu Verfügung gestellt werden, um die Datenströme individuell an sich ändernde oder vorgegebene Gegebenheiten anzupassen.
Diese Ausarbeitung stellt keinen Anspruch auf Vollständigkeit, vielmehr wurde der Schwerpunkt auf das Verständnis der grundlegenden Eigenschaften und Mechanismen von RTP gelegt.
Der Inhalt dieser Ausarbeitung stützt sich im wesentlichen auf die Protokollbeschreibung von [Scho96], es wurden jedoch auch andere Dokumente mit einbezogen.

1 Einleitung

Im Zeitalter multimedialer Anwendungen wächst die Anforderung an das Internet, Datenströme unterschiedlichster Art in Echtzeit übertragen zu können. Dies setzt zum einen eine adäquate Netzinfrastruktur und damit ausreichende Bandbreiten voraus, zum anderen ist die Bereitstellung und der Einsatz leistungsfähiger Transportprotokolle zwingend erforderlich.
Aus diesem Grund wurde von der *Audio-Video Transport Working Group* (AVT WG) im Auftrag der *Internet Engineering Task Force* (IETF) das Real-Time Transport Protokoll für die Übertragung von Video und Audioechtzeitdaten entwickelt.
RTP ist unabhängig von Netzwerkprotokollen unterliegender Transportschichten. So setzt es zum Beispiel auf dem *User Data Protocol (UDP)* auf, was zur Folge hat, daß keine Mechanismen für die Fehlerkorrektur und die Flußkontrolle zur Verfügung stehen. Dies erfordert den Einsatz einer Kontrollinstanz, eines Kontrollprotokolls, das Mechanismen für die Überwachung der Datenströme bereitstellt (*Monitoring*) und die Einhaltung des QoS sicherstellen hilft.

1.1 Problemstellungen

Wir wollen uns nun der Frage zuwenden, welche Probleme bei Echtzeitübertragungen auftreten können, und damit auch gelöst werden müssen. Dies verdeutlichen wir uns anhand eines klassischen Beispiels, der Videokonferenz bestehend aus mehreren Personen an verschiedenen Orten, die über eine Videokamera und ein Mikrophon verfügen (siehe Abbildung 1-1).
Mehrere Teilnehmer eines Netzwerkes möchten mit den jeweils anderen Teilnehmern in Echtzeit kommunizieren, d.h. Bild- und Tondaten austauschen, wobei Ton und Bild als eigenständige

Datenströme angesehen werden, so daß einzelne Teilnehmer entscheiden können, ob sie beide oder nur ein Medium nutzen wollen.
Audio- und Videoanwendungen setzen voraus, daß die zugehörenden Datenströme in gleichmäßigen Abständen bereitgestellt werden, um den Echtzeitcharakter zu wahren. Ein kontinuierlicher Datenstrom muß deshalb sichergestellt sein.
Desweiteren ist die Einhaltung eines minimalen Datendurchsatzes erforderlich, da mediale Daten eine gewisse Bandbreite voraussetzen, um angemessen eingesetzt werden zu können.
Die Teilnehmerzahl einer Videokonferenz ist nicht festgesetzt und sollte variabel sein, zusätzliche Teilnehmer können sich in die Konferenz einloggen oder vorhandene verabschieden. Um auch zum Beispiel einzelne Teilnehmer ausblenden zu können, muß der Anwender Informationen über die Zusammensetzung der Sitzung erhalten sowie eine Benachrichtigung bekommen, falls Teilnehmer sich verabschieden oder neue hinzukommen.

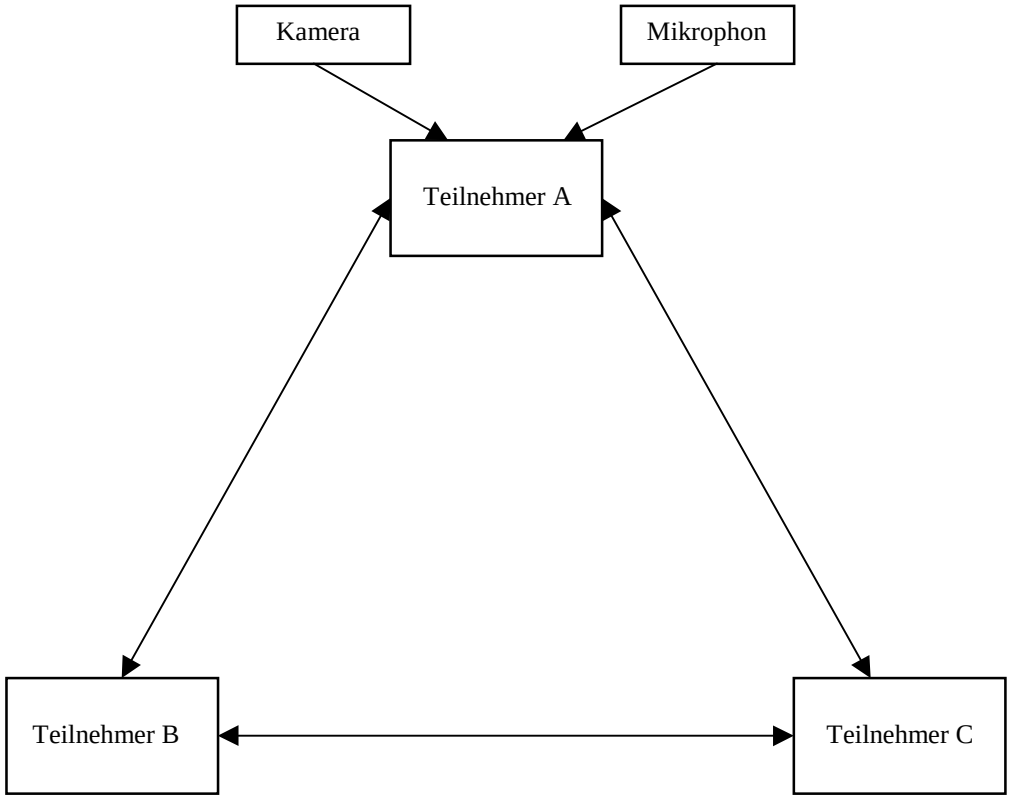


Abbildung 1-1 : Videokonferenz

1.2 RTP und andere Protokolle

RTP wurde so konzipiert, daß es von unterliegenden Protokollen unabhängig ist. In der Praxis setzt es häufig oberhalb von UDP auf, um dessen Multiplex- und Schecksummenfunktionalitäten nutzen zu können. Es läßt darüberliegenden Anwendungsprogrammen freie Hand, gibt keine starren Verhaltensvorgaben, sondern bietet durch sein Kontrollprotokoll eine Fülle von Informationen an, die von der Anwendung ausgewertet werden müssen. Zusätzlich bietet es Platz für anwendungsspezifische Erweiterungen. Aus diesem Grund ist RTP meistens auf Anwendungsebene direkt implementiert.
RTP / RTCP ist nur zuständig für das Übertragen von Daten, nicht für den Verbindungsaufbau, noch für das Bereitstellen der Information, wo bestimmte Daten zu finden sind. Hier können RTP und RTSP eine Einheit bilden. Eine Anwendung holt sich mittels einer RTSP Anfrage von einer Webseite auf einem Server die genaue Adresse und den Port, wo sie sich in eine Videolivekonferenz einklinken kann. Die Anwendung kontaktiert die Adresse unter dem genannten Port, das Übertragen der Daten kann beginnen. Unter der gleichen Adresse, jedoch meist eine Portnummer höher, wird einer weitere Verbindung für das Kontrollprotokoll aufgebaut. Abbildung 1-2 soll dies verdeutlichen.

Nun könnte sich die Frage stellen, warum ein neues Protokoll namens RTP ? Hätte TCP nicht als Basis für Datenübertragungen mit Echtzeitcharakter ausgereicht ? Der große Nachteil von TCP ist, daß es auf dem Prinzip „Warten oder Verwerfen“ beruht. Die Verbindung stockt, falls Pakete verloren gehen. Verloren gegangene Pakete werden neu angefordert und die Übertragungsgeschwindigkeit herabgesetzt, was für Echtzeitdaten inakzeptabel ist, da sie auf einen gleichmäßigen und kontinuierlichen Datenstrom angewiesen sind. Das Wiederholen von Datenpaketen kann zusätzlich zu einer Verzögerung seitens der Wiedergabe führen. TCP ist deshalb als Übertragungsprotokoll nicht geeignet.

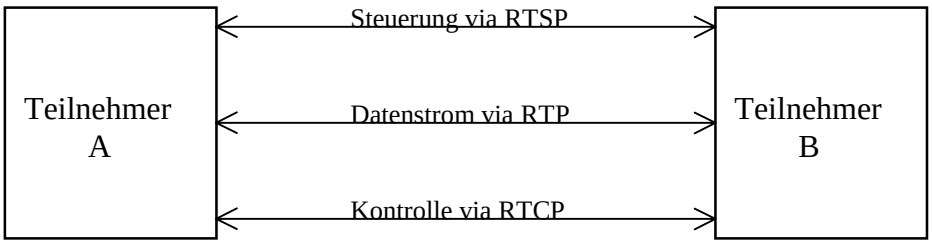


Abbildung 1-2 : RTP und RTSP

2 RTP

Wir beginnen mit der Vorstellung einiger wichtiger Grundbegriffe, die uns im weiteren des öfteren begegnen werden :

- Payload : Die eigentlichen Nutzdaten, die übertragen werden, z.B. komprimierte Videobilder oder Audiosamples
- Header : Kopf eines Pakets, liefert Informationen über ein Paket
- Pakete : Einheit bestehend aus dem Header und Nutzdaten
- Transportadresse : Kombination aus Netzwerkadresse, dem Domainnamen, und der Portnummer. Markiert den Anfangs- bzw. Endpunkt einer Übertragung
- Synchronization source (SSRC) : Bezeichner der Quelle eines Datenstreams

Das Real-Time Transport Protokoll ist zuständig für das Übertragen der Daten, die Echtzeitanforderungen haben. Ein RTP Paket, dessen Größe von den unterliegenden Protokollen abhängt, besteht aus einem festen Header und den eigentlichen Daten. Bevor wir uns mit dem Aufbau des Headers beschäftigen wollen, stellen wir zwei grundlegende Mechanismen von RTP vor.

2.1 Mixer

Stellen wir uns vor, daß zwei Teilnehmer einer Sitzung Audioechtzeitdaten austauschen wollen. Teilnehmer A verfügt über einen Netzzugang mit hoher Bandbreite, Teilnehmer B befindet sich in einem Subnetz mit wesentlich kleinerer Bandbreite. Hier kommen Mixer ins Spiel, sie kombinieren eingehende Datenströme, resynchronisieren die Pakete bzw. reduzieren die Komprimierungsrate der Daten zugunsten der Teilnehmer mit geringerer Bandbreite. Die Hauptaufgabe eines Mixers besteht jedoch darin, mehrere eingehende Ströme in einen ausgehenden umzuwandeln.
Jeder Quelle, und damit dem jeweiligen Datenstrom, sei es ein Mikrophon oder eine Videokamera, ist eine eindeutige Kennung zugeordnet, die *Synchronization Source* SSRC, um verschiedene Pakete einem Datenstrom zuzuordnen. Synchronisiert nun ein Mixer mehrere Datenströme zu einem neuen oder ändert er die Beschaffenheit eines bestehenden Stroms, so definiert er eine neue Quelle. Vom Mixer ausgehende Pakete sind nun mit der SSRC des Mixers markiert. Die SSRCs der eingehenden Ströme werden in eine Liste, der *Contributing Source List* (CSRC Liste) eingetragen und in die ausgehenden Pakete eingefügt.

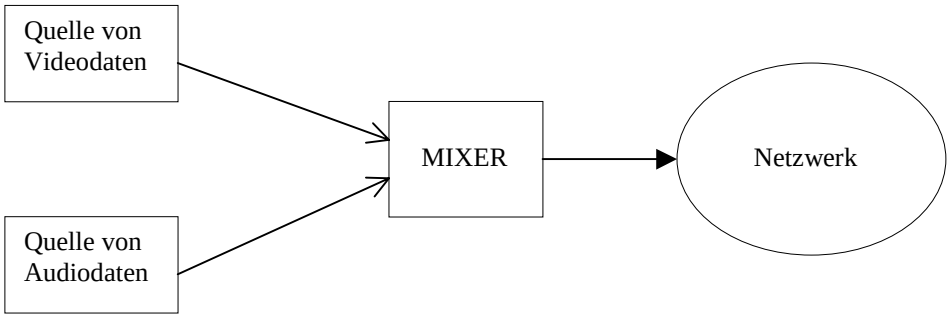


Abbildung 2-1 : Mixer

Abbildung 2-1 zeigt ein Einsatzgebiet von Mixern, das Zusammenführen und Synchronisieren zweier getrennter Datenströme von zwei verschiedenen Quellen, und damit auch verschiedenen SSRCs, zu einem neuen Datenstrom, der an ein Netzwerk weitergegeben wird.

2.2 Übersetzer

Übersetzer sind dafür da, eingehende Daten weiterzuleiten. Sie erzeugen keine neuen Ströme, lassen also die SSRC unangetastet. Ein Einsatzgebiet wäre zum Beispiel das Tunneling von Paketen durch eine Firewall, was den Einsatz von zwei Übersetzern auf jeder Seite der Firewall voraussetzt. Abbildung 2-2 soll dies verdeutlichen. In diesem Fall schickt Teilnehmer A einen Strom zu Teilnehmer B, der hinter einer Firewall sitzt.

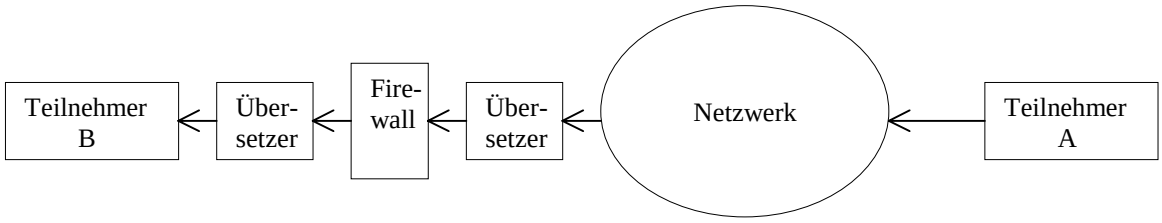


Abbildung 2-2 : Übersetzer

2.3 RTP Header

Betrachten wir nun den Aufbau des Headers eines RTP-Pakets, siehe Abbildung 2-3. Auf die einzelnen Felder gehen wir der Reihe nach ein.

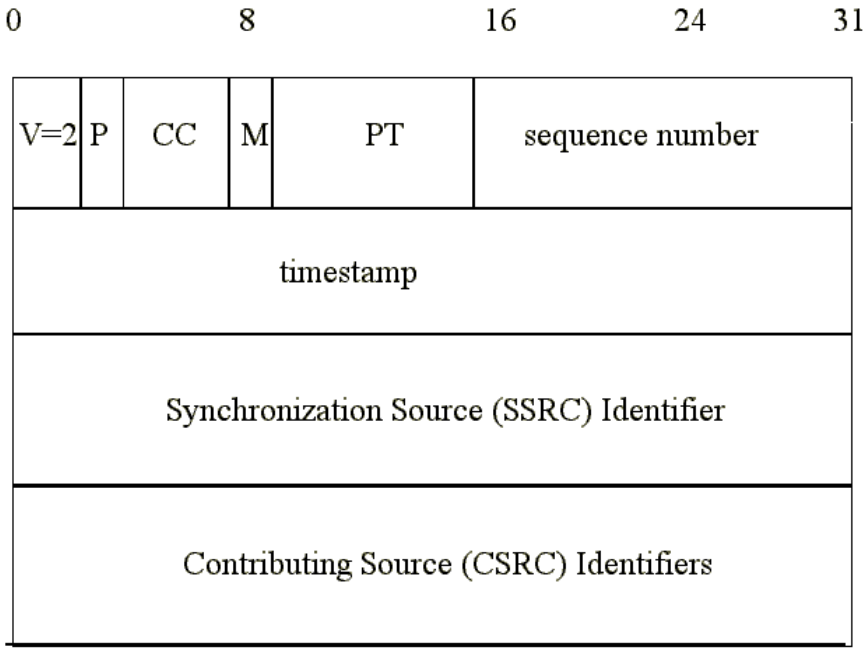


Abbildung 2-3 : RTP Header

- V : Gibt die Version von RTP an, momentan wird Version 2 verwendet

- P : Padding Bit. Wenn gesetzt, so wurden an das Ende des Pakets ein oder mehrere Padding-Oktetts angehängt, wobei das letzte Oktett die Anzahl der zu ignorierenden Oktetts angibt. Manche Verschlüsselungsverfahren benötigen eine feste Blockgröße

- CC : CSRC Zähler. Gibt an, wieviele Elemente in der CSRC-Liste eingetragen sind. Das können null bis fünfzehn sein

- M : Marker, kündigt Ereignisse an, die in einem gesonderten Profil beschrieben sind. Profile sind Zusatzvereinbarungen der Sitzungsteilnehmer, die unabhängig vor der Übertragung der Daten getroffen werden

- PT : Payloadtype. Gibt den Nutzlasttyp der Daten an und wird von der konkreten Anwendung interpretiert

- Sequence number : Eine 16bit Zahl, die von Paket zu Paket um eins erhöht wird. Der Anfangswert wird zufällig gewählt. Die Nummer kann benutzt werden, um Paketverluste zu erkennen bzw. um Paketsequenzen wieder herzustellen

- Timestamp : 32bit Zahl, die den Zeitpunkt wiederspiegelt, an dem das Paket erstellt wurde. Abhängig vom jeweiligen Anwendungsfall, so können zum Beispiel zwei Pakete den selben Zeitstempel enthalten wenn die Daten zum selben z.B. Bild gehören. Der Initialwert wird zufällig bestimmt

- SSRC Identifier : (meist) eindeutige Nummer, kennzeichnet die Quelle des Pakets und wird zufällig ermittelt, was zur Folge hat, daß zwei Quellen die selbe Nummer erhalten können

- CSRC Identifier : Liste von bis zu 15 SSRCs, die von Mixern eingefügt wurden

Dem Header kann eine Erweiterung variabler Länge folgen, die Zusatzinformationen enthält. Sie wird vor der CSRC Liste eingefügt und beinhaltet, neben ihrer Länge, profilspezifische Angaben, die von Anwendung zu Anwendung verschieden sein können und vorher von den Teilnehmenden ausgehandelt festgelegt wurden. Auf diesen Punkt wird an dieser Stelle aber nicht weiter eingegangen.

3 RTP Control Protokoll

RTP und RTCP bilden eine Einheit, wobei RTP sich um das Transportieren der Nutzdaten kümmert. Mit RTP alleine hat der Anwender jedoch nur eine begrenzte Möglichkeit, Informationen bezüglich der Qualität seiner Datensendungen zu erhalten, geschweige Informationen über andere Sitzungsteilnehmer zu bekommen, um darauf zu reagieren. Diese Aufgaben übernimmt das Kontrollprotokoll RTCP, dessen Prinzip daraus besteht, daß in periodischen Zeitabständen Kontrolldaten in Kontrollpaketen an alle Sitzungsteilnehmer verschickt werden.

Die Aufgaben und Funktionalitäten von RTCP lassen sich in vier Bereiche aufteilen :

i.) Rückmeldungen bezüglich der Qualität der Datenverteilung, dies wird dadurch erreicht, daß sowohl Sender als auch Empfänger von Paketen Informationen bereitstellen

ii.) Zuweisung eines eindeutigen Namens (CNAME, *canonical Name*) zu einer RTP-Quelle. Dadurch wird gewährleistet, daß ein Datenstrom einer Quelle eindeutig zuzuweisen ist, da die SSRC einer Quelle, aufgrund von möglichen Kollisionen, nicht eindeutig sein muß

iii.) Überwachung und Regulierung der Bandbreite, die von den Kontrollpaketen eingenommen werden

iv.) Bereitstellung von Informationen über die Sitzungsteilnehmer und Sitzungskontrolle

3.1 RTCP Pakete

Wir wollen nun die Funktionalitäten eingehender durchleuchten und beginnen mit der Vorstellung der Pakettypen von RTCP. Es existieren 5 Typen, die aus einem festen Teil gefolgt von einem variablen Teil bestehen und denen verschiedene Aufgaben zufallen.

3.1.1 Senderberichte

Senderberichte (SR) werden von Quellen verschickt, nachdem sie Datenpakete gesendet haben, und lassen sich in drei Bereiche, siehe auch Abbildung 3-1, einteilen :

- Der Header. Wie bei RTP Paketen enthält er Informationen über die Version und angehängte Padding Oktetts, der Payloadtyp ist auf 200 gesetzt, was das Paket als Senderbericht klassifiziert. Neben der Länge des Pakets in 32bit Wörtern minus eins, findet sich die SSRC des Senders

- Eine 20 Oktetts lange Senderinformation bietet Informationen über die Übertragungsaktivitäten des Senders. Der NTP Zeitstempel gibt den Zeitpunkt an, als der Bericht erzeugt wurde. Er richtet sich nach der Zeiteinteilung des *Network Time Protocols*, also relatiav nach der Anzahl der verstrichenden Sekunden seit dem 01.01.1900. Anhand dieses Stempels und eines zusätzlichen, dem RTP Zeitstempel, der die selbe Zeit wie der NTP Stempel, jedoch im Format der RTP Pakete, wiederspiegelt, lassen sich Zeitdifferenzen verschiedener Pakete berechnen. Wir erinnern uns an dieser Stelle, daß der Zeitstempel der RTP Pakete eine Zufallskomponente hat. Desweiteren gehören zu den Senderinformationen die Anzahl der Datenpakete, die bis zum Zeitpunkt des Erstellens dieses Berichts vom Sender verschickt wurden, sowie die Anzahl aller bis dahin gesendeten Nutzdatenoktetts. Diese beiden letzten Informationen beziehen sich auf die aktuelle Quelle, ändert der Sender seine SSRC, werden beide Zähler auf 0 gesetzt. In RTP Paketen wird ein Zähler mitgeliefert, der die Anzahl der Elemente der CSRC Liste angibt. In RTCP Paketen weicht dieser einem anderen Zähler, dem Empfangsberichtzähler

- Der dritte Teil enthält null oder mehrere Empfangsberichte, je nachdem von wie vielen Quellen der Sender Datenströme empfängt. Jeder Empfangsbericht besteht aus der SSRC, von der er Pakete erhält, sowie Statistiken bezüglich der Übertragung der Daten. Diese Statistiken bestehen aus der Anzahl der verlorengegangenen RTP Pakete seit Sendung des letzten Sender- bzw.

Empfängerberichtes, der Anzahl aller verlorengegangenen Pakete, der höchsten Sequenznummer, die empfangen wurde, der mittleren Abweichung zweier Pakete, die gesendet und empfangen wurden. Letzteres bedeutet die Differenz der Zeitstempel der letzten beiden empfangenden Pakete minus der Differenz der Zeit zu der sie angekommen sind. Der letzte Senderberichtszeitstempel sowie die zeitliche Verzögerung zwischen Erhalt des letzten Senderberichts von der Quelle und Sendung dieses Berichts schließen den Empfangsbericht ab

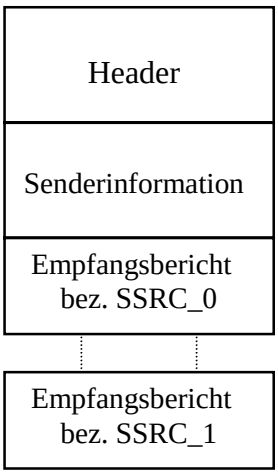


Abbildung 3-1 : SR RTCP Paket

3.1.2 Empfängerberichte

Empfängerberichte (RR) werden dann verschickt, wenn Daten erhalten worden sind und unterscheiden sich von den Senderberichten nur dadurch, daß sie keine Senderinformationen enthalten. Diese Sektion ist bei ihnen nicht vorhanden.

3.1.3 Quellenbeschreibungspakete

Quellenbeschreibungspakete, SDES (Source Description) genannt, stellen zusätzliche Informationen über die Quellen bereit, siehe Abbildung 3-2. Sie bestehen aus einem Header, dessen Payloadtyp auf 202=SDES festgesetzt ist, der wiederum Versionsnummer und vorhandene Padding Oktetts, die Länge des Pakets als auch die Anzahl der beschriebenen Quellen enthält. Angehängt sind null oder mehr Quellenbeschreibungen, die, angeführt von der entsprechenden SSRC und CSRC, eine oder mehrere Einträge enthalten. Die Einträge bestehen aus einem 8bit langen Feld, der den Typ des Eintrags festlegt, einem 8bit langen Zähler, der die Länge der textuellen Beschreibung angibt und der eigentlichen Beschreibung. Wir stellen die möglichen Einträge jetzt vor.

- CNAME, der kanonische Endpunktidentifizierer. Dieser Eintrag ist vorgeschrieben und enthält den Benutzernamen, zum Beispiel das Login, und den Domainname der Quelle, die eindeutig unter allen Sitzungsteilnehmern und während der Sitzung nicht verändert werden sollte
- NAME, der Name des Nutzers
- EMAIL, die Emailadresse des Nutzers
- PHONE, die internationale Telephonnummer des Nutzers
- LOC, der Ort, an dem sich der Nutzer befindet

- TOOL, der Namen des eingesetzten Programms
- NOTE, mögliche Statusmeldungen, die der Nutzer bekannt geben möchte
- PRIV, eröffnet die Möglichkeit neue Quellenbeschreibungen zu versenden. Ein vom Anwender gewählter Name der Beschreibung, die Länge der Beschreibung sowie die Beschreibung selbst werden übertragen

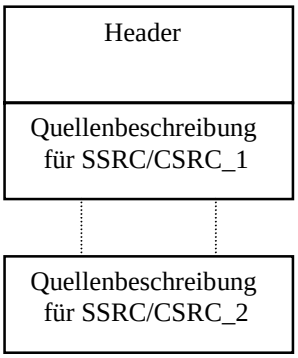


Abbildung 3-2 : Quellenbeschreibungspaket

3.1.4 Goodbye Pakete

Goodbye Pakete (BYE) werden verschickt, wenn einzelne Quellen nicht mehr zu Verfügung stehen, da sich zum Beispiel ein Teilnehmer ausloggen will. Versionsnummer, Anzahl der Padding Oktetts und die Länge des Pakets sind enthalten, der Payloadtyp ist auf 203=BYE festgelegt. Zusätzlich existiert ein Zähler, der angibt, wieviele Quellen sich verabschieden wollen. Die Quellen, gekennzeichnet durch ihre SSRC/CSRC, sind an den Header angehängt und können optional mit einem Text versehen werden, der den Grund des Ausscheidens angibt.

3.1.5 Anwendungsspezifische Pakete

Pakete dieser Art (APP) sind dafür gedacht, neuen Anwendungen die Möglichkeit zugeben, Daten auszutauschen, die durch Pakete bestehender Art nicht abgedeckt werden können. Neben den bekannten Feldern enthält ein solches Paket seinen Namen, einen Subtypeintrag, der eine genauere Einteilung neuer Pakete gleichen Namens ermöglicht, seine SSRC/CSRC und die anwendungsspezifischen Daten. Teilnehmern, denen ein solches Paket zugeschickt wurde, sollte der Name des Paketes bekannt sein und dessen Inhalt Verwendung finden, andernfalls sollte das Paket ignoriert werden.

3.2 Auswertung von Sender- und Empfängerberichten

Die in den Berichtpaketen enthaltenden Informationen sind sowohl für den Sender als auch für den Empfänger von Datenpakete interessant und machen nur Sinn, wenn sie vom Anwender bzw. der Anwendung interpretiert werden und entsprechende regulierende Maßnahmen nach sich ziehen.
So können Empfänger erfahren, ob aufgetretene Probleme lokaler oder globaler Natur sind. Quellen, die Daten verschicken, können, nach Auswertung der von den Empfängern verschickten Rückmeldungen, ihre Datenströme verändern und anpassen. Wir kommen im nächsten Abschnitt näher darauf zu sprechen.

Sowohl in den Senderinformationen als auch in den Empfängerberichten sind diverse Zähler und Zeitstempel enthalten, deren Bedeutung wir uns nun näher anschauen.
Zähler unterscheiden sich dadurch, daß sie sich entweder auf absolute Daten beziehen, also zum Beispiel seit Beginn der Sitzung die übertragene Datenmenge mitzählen. Auf der anderen Seite können sich Zähler auf vorhergegangene Pakete beziehen. Zeitstempel geben den Zeitpunkt an, absolut oder in Bezug zur sendenden Quelle, zum dem ein Paket erstellt bzw. verschickt wurde.
Wir wollen uns nun ansehen, welche konkreten Rückschlüsse zu ziehen sind und welche Informationen berechnet werden können.

Aus der Anzahl der verlorengegangenen Pakete zweier aufeinanderfolgenden Berichte ergibt sich die genaue Anzahl der nicht erhaltenen Pakete. Diese können durch Differenzbildung der zuletzt gelieferten Sequenznummern berechnet werden. Aus dieser Anzahl, dividiert durch die Zeitdifferenz der zugehörenden NTP-Zeitstempel, läßt sich die Anzahl der verlorengegangenen Paket pro Sekunde ausrechnen.
Aus der Anzahl der gelieferten oder nicht gelieferten Pakete läßt sich in Relation zur verstrichenen Zeit, bekannt durch die Zeitstempel, die Übertragungsrate ermitteln. Mit Hilfe der Differenz zwischen der Anzahl der erwarteten Pakete minus der Anzahl der verlorengegangenen Pakete läßt sich die genaue Anzahl der tatsächlich erhaltenen Pakete errechnen.
Die mittlere Abweichung der Zeit, in der zwei Datenpakete gesendet und dann empfangen wurden kann ein Anzeichen dafür sein, daß ein Paketverlust bevorsteht, so daß der Anwender Zeit hat, z.B. die Kodierung seines Datenstroms anzupassen, um die Bandbreite zu schonen. Diese Einschätzung ist mit Vorsicht zu genießen, da sie die Netzauslastung nur für einen bestimmtem Zeitpunkt wiedergibt.
Für zuverlässigere Aussagen ist es erforderlich, daß entweder mehrere Pakete des selben Teilnehmers über einen längeren Zeitraum, oder Pakete verschiedener Teilnehmer im Zusammenhang analysiert werden.

3.3 Konsequenzen der Auswertung

Es stehen also jetzt eine Menge von Informationen, die ausgewertet wollen. Wir beschäftigen uns nun mit der Frage, wer Nutzen aus den Übertragungsstatistiken ziehen kann, sowie welche Konsequenzen gezogen werden können. Diese möglichen Konsequenzen müssen natürlich zur Laufzeit vorgenommen werden.

Folgende Personen können Nutzen aus den Statistiken ziehen ?, wir teilen sie in 3 Klassen ein :

- Sender von Datenströmen
- Empfänger von Datenströmen
- Unbeteiligte Dritte

An dieser Stelle sei daran erinnert, daß mögliche notwendige Anpassungen auf Ebene der laufenden Anwendung vollzogen werden müssen. RTP/RTCP zwingt niemanden, die Übertragungsstatistiken zu berücksichtigen, was jedoch wenig Sinn macht.

Sender, Quellen von Datenströmen, senden ihre Daten mit der Intention, daß ihre Daten beim Empfänger in akzeptablen Zeitgrenzen ankommen, so daß sie verwertet werden können. Aus den Empfangsberichten der Empfänger erfährt der Sender nun die Qualität seiner Dienstleistung. Fällt die Beurteilung seiner Dienste negativ aus, liegt es am Sender, die Übertragung der Daten zu verbessern. Bekommt also eine Quelle die Rückmeldung, daß die Hälfte aller Datenpakete bei einem bestimmten Empfänger verloren gegangen sind, so kann sie die Kodierung der z.B. Videodaten herabsetzen. Dies erreicht sie dadurch, daß sie entweder die Daten umkodiert, oder auf Datensätze, mit geringerer Größe, zurückgreift, die ihr vorher zu Verfügung gestellt wurden. Aus den Informationen nachfolgender RTCP-Kontrollpakete des Empfängers kann sie ablesen, ob ihr Strategie richtig war.

Empfänger von Datenströmen wollen die Daten umsetzen, also in Echtzeit der Anwendung bzw. dem Anwender zu Verfügung stellen, um angezeigt oder abgespielt zu werden. Aus den von den Quellen gesendeten Senderberichten extrahiert der Empfänger die Informationen, um diesen Vorgang den Gegebenheiten anzupassen. So kann eine Anwendung zum Beispiel entscheiden, ob sie zu spät eingetroffene Pakete verwirft.

Unbeteiligte Dritte können zum Beispiel Netzwerkadministratoren sein, die keine Teilnehmer einer Sitzung sind. Mit Hilfe der in den RTCP Paketen enthaltenen Statistiken können sie den Datenverkehr innerhalb ihres Netzwerkes überwachen, wie unter anderem das Ermitteln der Übertragungsgeschwindigkeit und der Übertragungskapazität. Es lassen sich daraus Rückschlüsse auf die Qualität des Netzwerks ziehen, zusätzlich können sie helfen, aufgetretene Probleme und Fehler zu suchen und zu beseitigen.

3.3 Übertragungsintervalle und Bandbreitenregulierung

Die Datenmenge der RTP Pakete reguliert sich scheinbar selbst und nimmt selbst bei steigender Teilnehmerzahl nicht rapide zu, da zum Beispiel bei einer Audiokonferenz nicht alle Beteiligten gleichzeitig reden.

Bei den RTCP Pakten ist das anders, alle Teilnehmer senden in regelmäßigen Abständen z.B. Empfängerberichte und würden dadurch, bei steigender Teilnehmerzahl, die Bandbreite negativ beeinträchtigen.

Aus diesem Grund müssen die Abstände zwischen zwei Kontrollpaketen regulierbar sein und gegebenenfalls herabgesetzt werden, RTCP kontrolliert sich also selbst, um die Übertragung der sensiblen Nutzdaten nicht zu gefährden.

Als Basis dient die sogenannte *Session bandwidth*, die zu Verfügung stehende Gesamtbandbreite geteilt durch die Anzahl der Teilnehmenden. Es wird empfohlen, daß der Anteil der Kontrollpakete 5% dieser Bandbreite nicht übersteigen sollte und sich alle Teilnehmer an diese Marke halten.

25% dieser RTCP Bandbreite sind Sendern vorbehalten, so daß Neuankömmlinge schnellen Zugang zu den Quellenbeschreibungspaketen erhalten. Außerdem sind die in den Senderberichten enthaltenen Informationen die Grundlage dafür, daß der Empfänger Rückschlüsse auf die Empfangsqualität der erhaltenen Daten ziehen kann.

Der zeitliche Mindestabstand zweier RTCP Pakete wurde auf eine halbe Sekunde plus einem Zufallswert zwischen einer halben und eineinhalb Sekunden festgelegt, damit nicht mehrere Sender gleichzeitig Daten verschicken. Er richtet sich auch nach der Anzahl der Teilnehmer und damit auch nach der *Session bandwidth*.

4 Weitere Protokollmechanismen

RTP und RTCP stellen noch eine Reihe von anderen Mechanismen zu Verfügung bzw. es können beim Einsatz dieser Protokolle etwaige Probleme auftauchen, auf die wir jetzt eingehen wollen bzw. die wir nun vorstellen.

4.1 Kollisionen und deren Behandlung

Wir erinnern uns, daß die SSRC einer Senderquelle am Anfang zufällig bestimmt wird, mit dem Ziel, daß alle an der Sitzung beteiligten Quellen eine unterschiedliche Kennung erhalten.
Es besteht nun die, wenn auch sehr geringe Wahrscheinlichkeit, daß zwei verschiedene Quellen die Selbe SSRC zugewiesen bekommen. In solch einem Fall spricht man von Kollisionen, deren Auftreten erkannt und deren Existenz beseitigt werden muß. Die Wahrscheinlichkeit, daß eine Kollision auftritt, ist dann am größten, wenn alle Teilnehmer zur gleichen Zeit, etwa auf Anweisung des Sitzungsleiters, ihre SSRC berechnen.
Im anderen Fall, wenn also ein Neuhinzukömmling einer bestehenden Sitzung beitritt, ist die Wahrscheinlichkeit einer Kollision geringer, da die SSRCs der Quellen sowohl in den Headern der Datenpakete, als auch in verschiedenen Feldern der Kontrollpakete enthalten sind. Der neue Teilnehmer erhält daher relativ schnell Kenntnis über die Bezeichner der Sender, kann diese mit seiner gewählten Kennung vergleichen und gegebenenfalls seine SSRC neu setzen, falls diese mit einer anderen SSRC übereinstimmt. Dies geschieht dadurch, daß er ein Goodbye Paket verschickt, sich also abmeldet, und dann einen neuen Bezeichner generiert. Ein anderer Fall wäre, wenn jemand von zwei verschiedenen Quellen, ein und dieselbe SSRC bekommt. Ihm bleibt nichts anderes übrig, als die Pakete des einen zu ignorieren und zu hoffen, daß einer der beiden Kollisionsverursacher den Konflikt bemerkt und beseitigt.

4.2 Schleifen

Als Schleifen bezeichnet man das Duplizieren von Kontroll- und Datenpaketen, also das mehrfache Versenden der selben Daten. Man darf voraussetzen, daß Anwendungen keine Pakete doppelt verschicken.

Also verbleiben als Ursache für Schleifen mögliche Mixer und Übersetzer, die neue gebündelte Datenströme erzeugen oder weiterleiten. Angemerkt sei an dieser Stelle, daß der Ursprung eines Pakets anhand seiner Transportadresse, bestehend aus der eindeutigen Netzadresse plus der Portnummer, bestimmt werden kann. So kann es zum Beispiel sein, daß ein Übersetzer fälschlicherweise Pakete in die selbe Richtung weiterleitet, von der er sie erhalten hat.

Wie können nun Schleifen, und auch Kollisionen, erkannt werden ?
Jeder Teilnehmer verwaltet eine eigene Liste, in der SSRC und CSRC Bezeichner, der CNAME und die Transportadresse je Eintrag verwaltet werden. Die Informationen, enthalten in dem ersten Paket, das man von einer der Quellen erhält, werden als korrekt angenommen und in die Liste eingetragen. Alle danach ankommenden Pakete werden darauf hin mit den Informationen in der Liste abgeglichen, gegebenenfalls werden neue Einträge aufgenommen. Treffen so zum Beispiel zwei Pakete mit der gleichen SSRC aber unterschiedlichen Transportadressen oder unterschiedlichen CNAMES ein, so liegt ein Konflikt vor, der oben beschriebene Maßnahmen nach sich ziehen sollte.

4.3 Algorithmus zur Erkennung von Kollisionen und Schleifen

An dieser Stelle stellen wir einen einfachen Algorithmus vor, mit dessen Hilfe Kollisionen, sowohl eigen- als auch fremd verschuldet, also auch Schleifen erkannt werden können.

unbekannte Quelle

IF SSRC oder CSRC ist in der Liste nicht vorhanden

THEN einen neuen Eintrag erzeugen, der SSRC/CSRC, Transportadresse und CNAME enthält

CONTINUE

Quelle ist bekannt

IF Transportadresse und SSRC stimmt mit den Einträgen in der Tabelle überein

THEN CONTINUE

mögliche Kollision

IF SSRC wird als nicht die eigene erkannt

THEN IF zugehörender CNAME, laut Tabelle, stimmt nicht mit dem CNAME des Pakets überein

THEN ABORT

selbstverschuldeter Fehler

IF Transportadresse wird als die eigene erkannt

THEN IF CNAME ist die eigene

THEN ABORT

BYE-Paket senden

Neue SSRC erzeugen

Tabelle neu anlegen

CONTINUE

4.3 Sicherheit und Verschlüsselung

Eine sichere Übertragung von multimedialen Daten gewinnt mehr und mehr an Bedeutung. Einige oder alle Sitzungsteilnehmer können sich vor Beginn der Sitzung auf ein anwendungsspezifisches Verschlüsselungsverfahren einigen oder einen speziellen Payloadtyp vereinbaren, der Verschlüsselung voraussetzt, so daß Daten nur für denjenigen eine Bedeutung haben, für den sie gedacht waren.
Mehrere RTCP Pakete können zu einem sogenannten *Compound RTCP packet* zusammengefaßt werden, wovon dann sensible Teile, zum Beispiel die Quellenbeschreibungen, verschlüsselt werden. Die Übertragungsstatistiken wären weiterhin Dritten zugänglich.

5 Ausblick und Abschluß

RTP wurde 1995 als Standart anerkannt und wurde seitdem weiterentwickelt und in Anwendungen eingesetzt (z.B. Netscape, Quicktime). Es bietet den aufsetzenden Applikationen genug Freiheiten, um spezifische Erweiterungen, basierend auf bestehenden Strukturen, zu implementieren. Gelieferte Statistiken bezüglich der Qualität der Übertragung bieten ausreichende Informationen, um eine störungsfreie Übertragung multimedialer Datenströme zu gewährleisten, sowie aufgetretene Fehler und Störungen zu erkennen und zu beseitigen.

6 Literaturverzeichnis

[Schu 96] Schulzrinne, H. :
RTP: A Transport Protocol for Real-Time Applications
RFC 1889,
ftp://ftp.isi.edu/in-notes/rfc1889.txt, 1996

[Herm 96] Hermann, Marc :
Vortragsreihe Moderne Kommunikationssysteme, Universität Ulm,
Real-Time Transfer Protocol und RTP Control Protocol,
http://www.uni-ulm.de/~s_mherma/text/rtp.html, 1996

[Deff 95] Deffner, Bernd :
The Real-Time Transport Protocol RTP,
http://www.fokus.gmd.de/step/acontrol/node2.html, GMD Fokus 1995

[Joan 98] Joanid, Andre :
RTP: Real Time Trasport Protokoll,
http://www.hawo.stw.uni-erlangen.de/~asioanid/seminar/node4.html, 1998

Streamingmöglichkeiten von QuickTime 4

Björn Somberg und Dirk Vleugels

FB Informatik, Universität Dortmund
Bjoern@Somberg-online.de, Dirk@icmp.ping.de

Zusammenfassung

QuickTime ist ein Lösung von Apple Computer zur Speicherung und Darstellung von digitalen Multimedia Daten. Mit der neusten Programmversion 4 ist QuickTime auch in der Lage Videodaten zu speichern, die später über das Internet mittels der Streaming-Technologie verbreitet werden können.

1 QuickTime

Einfach ausgedrückt ist QuickTime eine Software, die es erlaubt digitales Video, genauso wie andere Arten von Multimedia Daten mit dem Computer abzuspielen. Apple Computer selbst sieht QuickTime als „standard, cross-platform, multimedia framework for authoring, publishing, and streaming“ der meist verbreitetsten Formate von Video und Audio, also als eine standardisierte Plattform übergreifende Umgebung, mit der die meisten Multimedia Daten erzeugt, veröffentlicht und über ein Netzwerk per „streaming“ angeboten werden können. Hiermit möchte Apple Computer eine breite Benutzerschicht erreichen, vom einfachen Benutzer, der QuickTime nur zum Empfangen von Multimedia-Inhalten nutzt, bis zum Programmierer, der direkt auf die API zugreift.

1.1 QuickTime 4

QuickTime 4 bietet neben den schon in QuickTime 3 vorhandenen Features die Möglichkeit, Video und Audiodaten live bzw. für Streaming aufbereitet über das Internet anzubieten.

1.2 Installation

Während die Vorgängerversion noch als monolithischer Softwareblock erhältlich war, läßt sich QuickTime 4 völlig modular installieren. Voraussetzung dafür ist allerdings ein Internet Anschluß. Ein über

http://www.apple.com/quicktime

zu beziehender „Aktualisierer“ (Größe nur etwas 450 KBytes) assistiert dem Benutzer bei der Installation und sorgt dafür, daß nur ausgewählte Komponenten heruntergeladen werden. Für diesen Vorgang ist noch nicht einmal ein Internet-Browser nötig. Allerdings sollte man darauf achten, daß die Dateien „QuickTime Updater“ und „QuickTime Install Cache“ im QuickTime-Ordner belassen werden, da diese über 6 MB großen Dateien sonst beim nächsten Update erneut heruntergeladen werden müssen.

Durch die Modularisierung des Softwarepakets können QuickTime Nutzer stets mit der aktuellen Version versorgt werden und auch Verbesserungen der „Open Source Community“ können schnell und kostengünstig verbreitet werden.

Ein für den Hersteller angenehmer Nebeneffekt ist, daß so regelmäßig große Besucherströme auf die eigenen Webseiten gezogen werden, auf denen natürlich nicht nur Informationen zu QuickTime 4 publiziert werden.

Beispielsweise hat der Starwars-Trailer im QuickTime 4 (Beta) Format Plattform übergreifend 8 Millionen Downloads des „Phantom Meance“-Videos und für mehr als 5 Millionen Downloads der Betaversion von QuickTime 4 gesorgt.

Abbildung 1-1: Der QuickTime Client

1.3 Protokolle

Statt eigene Streaming-Protokolle zu propagieren setzt QuickTime 4 auf die nicht proprietären Protokolle RTP (Real Time Transport Protocol) und RTSP (Real Time Streaming Protocol) der IETF, die sich unter Unix als Standard etabliert haben. Anders als IBM, Netscape, SGI, Sun, Vxtreme oder Apple unterstützt Microsoft bisher RTP/RTSP leider nicht, sondern entwickelt eigene Protokolle.

1.4 Multilinguale Versionen

QuickTime 4 hat eine gewisse Intelligenz und nimmt somit dem Benutzer Entscheidungen von vornherein ab. Er wird z.B. das Erstellen von multilingualen Sprachversionen eines Audio- oder Videodokuments unterstützt. Der Client fragt die notwendigen Informationen in den Einstellungen des Betriebssystems ab, ohne beim Benutzer rückzufragen.

1.5 Alternate Data Rates

Vollautomatisch erfolgt auch die Wahl der optimalen Datenqualität für die vorhandene Verbindungsgeschwindigkeit (Alternate Data Rates). Basierend auf der vom Nutzer im QuickTime Setup eingestellten Verbindungsgeschwindigkeit zum Internet wählt der QuickTime Client das angemessen codierte Movie aus. Der Autor eines Movies hat die Möglichkeit mit verschiedenen Übertragungsraten codierte Movies zu erzeugen, welche über eine gemeinsame Referenz angesprochen werden können. Über diese Referenz wird dann der Client das für den Benutzer geeignetste Movie auswählen. Auf

diese Weise erhält der Benutzer immer die bestmögliche Qualität, ohne großen Aufwand für die Verwaltung von verschiedenen Links für verschiedene Qualitäten treiben zu müssen. Da dieses Feature mindestens die Version 3 von QuickTime erfordert, ist für ältere Clients ein „fallback"-Mechanismus vorgesehen, der diesen einen Link auf einen für sie verständliches Format liefert.

1.6 QuickTime 4 Pro

Die erweiterte QuickTime 4 Pro Version bietet darüber hinaus die Funktionen eines Editors, so daß Filme, Bilder und Audio Dateien kreiert, editiert und abgespeichert werden können. Wohingegen QuickTime 4 frei verfügbar ist und nur zur Wiedergabe genutzt werden kann, kostet die QuickTime 4 Pro Lizenz $29.99. Nach einer Registrierung können die zusätzlichen Funktionen per Paßwort in der einfachen QuickTime Version freigegeben werden. Darüber hinaus ist auch noch eine kosteneffektivere Mutliuser-Version erhältlich.

1.7 QuickTime 4.1

Anfang nächsten Jahres, vielleicht sogar schon zur MacWorld Expo, die Anfang Januar 2000 in San Francisco stattfinden wird, will Apple QuickTime 4.1 fertig haben. QuickTime 4.1 soll eine nahtlose Integration von Anzeigen erlauben, was von Website-Betreibern immer wieder gefordert wurde. Das Zusammenspiel mit Firewalls, was unter QuickTime 4 einiger Anpassungen bedurfte, soll ebenfalls verbessert werden. Vollkommen neu ist die Unterstützung von SMIL (Synchronized Multimedia Integration Language), womit sich Multimedia-Elemente wie Filme, Töne oder Bilder zeitlich kontrolliert auf einer Webseite darstellen lassen. Der QuickTime Streaming Server 2, der zusammen mit der neuen QuickTime Version veröffentlicht wird, soll nun auch einen Paßwort gesicherten Zugang zu Multimedia-Inhalten bieten.

2 Streaming

Unter „Streaming" versteht man das kontinuierliche Senden von Daten von einem Server zu einem Client über ein Netzwerk wie z.B. dem Internet/Intranet.

Der Server teilt die zu streamenden Daten in Pakete auf, welche über das Netzwerk versendet werden können. Der Empfänger fügt die Pakete wieder zusammen und die Daten können sofort wiedergegeben werden bzw. nach kurzer Zeit, nämlich sobald genügend Pakete empfangen wurden.

Eine Serie von in Beziehung stehenden Paketen nennt man einen Stream.

Streaming unterscheidet sich in dem Punkt von einer simplen Daten Übertragung, als daß die Daten sofort wiedergegeben werden können, anstatt der Wiedergabe nach einem kompletten Download.

Somit wird von einem Streaming Client niemals ein „streaming" Video heruntergeladen, sondern die einzelnen Pakete werden vielmehr wiedergegeben, sobald sie empfangen werden.

QuickTime 4 bietet die Möglichkeit eine Vielzahl von Multimedia Daten in ein streamingfähiges Format zu überführen. Um nicht ständig alle möglichen Format aufzählen zu müssen, werde ich mich im folgenden auf die Übertragung von „QuickTime Movies" beschränken.

QuickTime Movies können über mehrere Protokolle gestreamed werden:

- HTTP 1.1 (Hyper Text Transport Protocol)
- FTP (File Tranfer Protocol)
- RTP (Real Time Tranfer Protocol)

HTTP und FTP sind eigentlich dafür gedacht, Hypertext bzw. Dateien allgemein zu übertragen. Der QuickTime Client ist imstande Dateien, die auf diese Art übertragen werden, wiederzugeben bevor sie vollständig empfangen wurden. Bei dem HTTP Protokoll ist hierfür allerdings die Version 1.1 nötig.

Für Streaming in Echtzeit wird RTP benutzt. Die Pakete eines Movies werden in Echtzeit gesendet, so daß ein ein-minütiger Film auch in einer Minute über das Netzwerk übertragen wird bzw. wiedergegeben wird. Die einzelnen Pakete haben einen Zeitstempel, so daß sie auch zeitlich synchron wiedergegeben werden können.

Da die Pakete in Echtzeit übertragen werden, ist es auch möglich „Live Inhalte“ über das Netz zu übertragen.

„Real time streams“ können im Unicast-, Multicast-, oder „reflected“-Multicast Verfahren übertragen werden.

2.1 Unicast Streaming

Unicast ist ein „Eins-zu-Eins“-Verfahren vergleichbar mit einer Telefonverbindung. Jeder Media-Stream wird durch eine einzelne Verbindung zwischen Server und Client initialisiert und kontrolliert. Der Client schickt dem Server einen „Request“ bzgl. eines Movies über RTSP (Real Time Streaming Protocol). Der Server antwortet dann dem Client über RTSP mit Informationen, die das Movie als „streaming session“ weiter charakterisieren. RTSP benutzt hierzu TCP/IP.

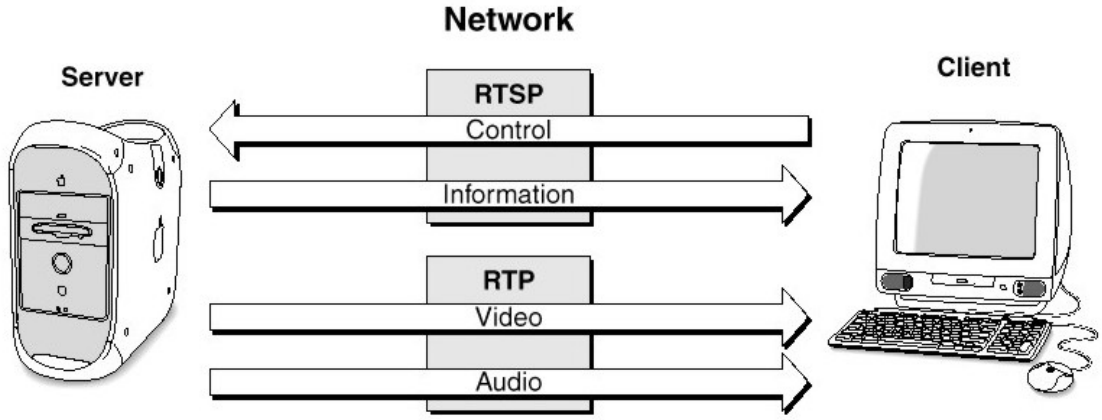


Abbildung 2-1: Unicast Streaming über RTP/RTSP

Eine „streaming session“ kann einen oder mehrere Streams umfassen, z.B. einen „Video Stream“ und einen „Audio Stream“.

Der Server teilt dem Client mit, wieviel Streams zu erwarten sind und übermittelt detaillierte Informationen zu jedem Stream. Hierbei werden Informationen über den Media Typ und den verwendeten Codec (Verfahren zum Codieren und Encodieren) mitgeteilt.

Die Streams werden über RTP gesendet, welches UDP/IP benutzt. Wenn ein QuickTime Movie über RTP gestreamed wird, wird jeder Track des Movies in einem eigenen Stream gesendet.

Ein Stream kann „Live Inhalte“, Börsen Ticker, Radio Sendungen oder aber auch gespeicherte Daten, wie z.B. den Video Track eines QuickTime Movies, beinhalten.

Wenn der Client einen Unicast Stream von gespeicherten Daten empfängt, gibt es die Möglichkeit über den „client movie controller“ zu beliebigen Stellen der Daten zu springen. Dazu ist es nicht notwendig, die gesamten Daten herunterzuladen. Der Client fordert den Server nur auf, mit dem Streaming an einer neuer Stelle zu beginnen.

2.2 Multicast Streaming

Ein Multicast-Stream wird zu einer Gruppen Adresse gesendet, unabhängig davon, ob sich jemand an den Multicast anhängt oder nicht. Ein Multicast-Stream ist somit vergleichbar mit einer Radio Ausstrahlung.

Es wird eine Kopie von jedem Stream in jeden Zweig des Netzwerks geschickt, was den Netzwerk Traffic reduziert, wenn eine große Anzahl von Clients, welche die gleichen Daten anfordern, vorhanden ist.

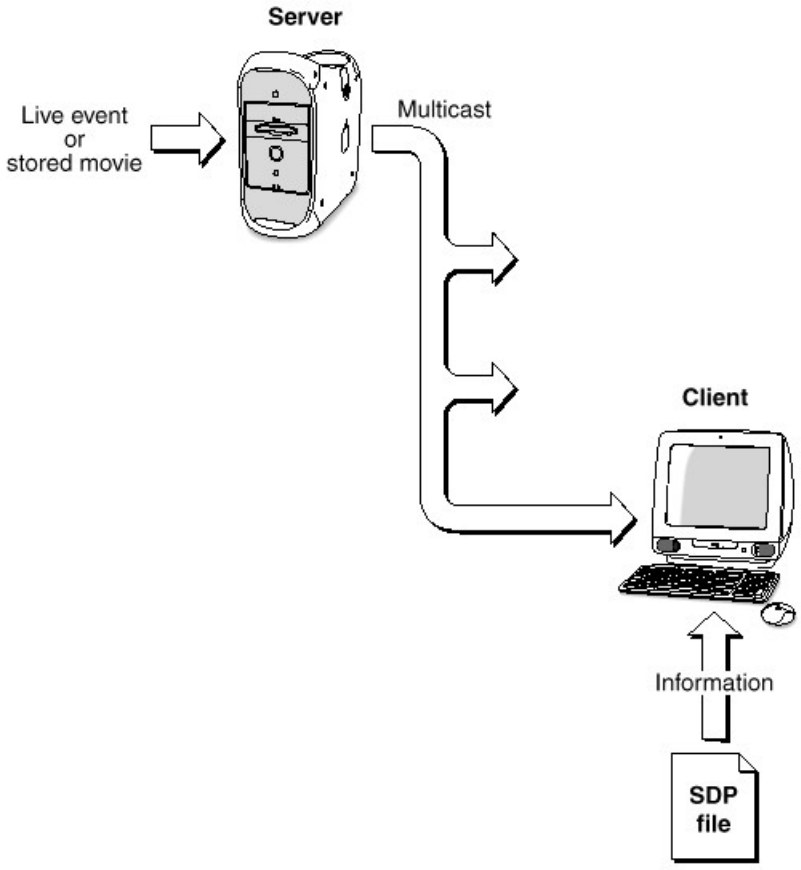


Abbildung 2-2: Multicast Streaming über RTP/RTSP

Ein Client empfängt einen Stream, indem er sich an den Multicast anhängt. Wie dies zu bewerkstelligen ist erfährt der Client indem er die SDP (Session Description Protocol) Datei ließt. Die SDP Datei enthält alle Informationen, die der Client benötigt, um sich an den Multicast anzuhängen. Auf diese Weise werden dem Client „group address“, „port number“, sowie „stream description“ Informationen mitgeteilt.

SDP Dateien werden gemeinhin auf dem Webserver abgelegt, um das Hinzukommen von weiteren Clients zu ermöglichen.

Bei dem Multicast-Verfahren hat der Benutzer nur die Möglichkeit, die Übertragung des/der Streams zu stoppen bzw. wiederaufzunehmen. Er kann nicht zu einer beliebigen Stelle des Movies springen oder die Übertragung von Vorne beginnen.

2.3 Reflected Multicast

Da nicht alle Router Multicast unterstützen, gibt es die Möglichkeit für Clients hinter „nicht-Multicast-fähigen“ Routern, Multicast über einen Reflektor zu empfangen.

Ein Reflektor ist ein RTSP Server, der sich an einen Multicast anhängt und diesen in eine Reihe von Unicasts konvertiert. Diese leitet er dann an Clients weiter, die diesen Stream angefordert haben. Der Reflektor sendet daher den Inhalt „live“ und in Echtzeit, den der ursprüngliche Server über Multicast verbreitet. Mit den oben benutzen Analogien würde dies heißen, daß eine Radio Ausstrahlung empfangen und an jeden Zuhörer per Telefon weitergeleitet wird.

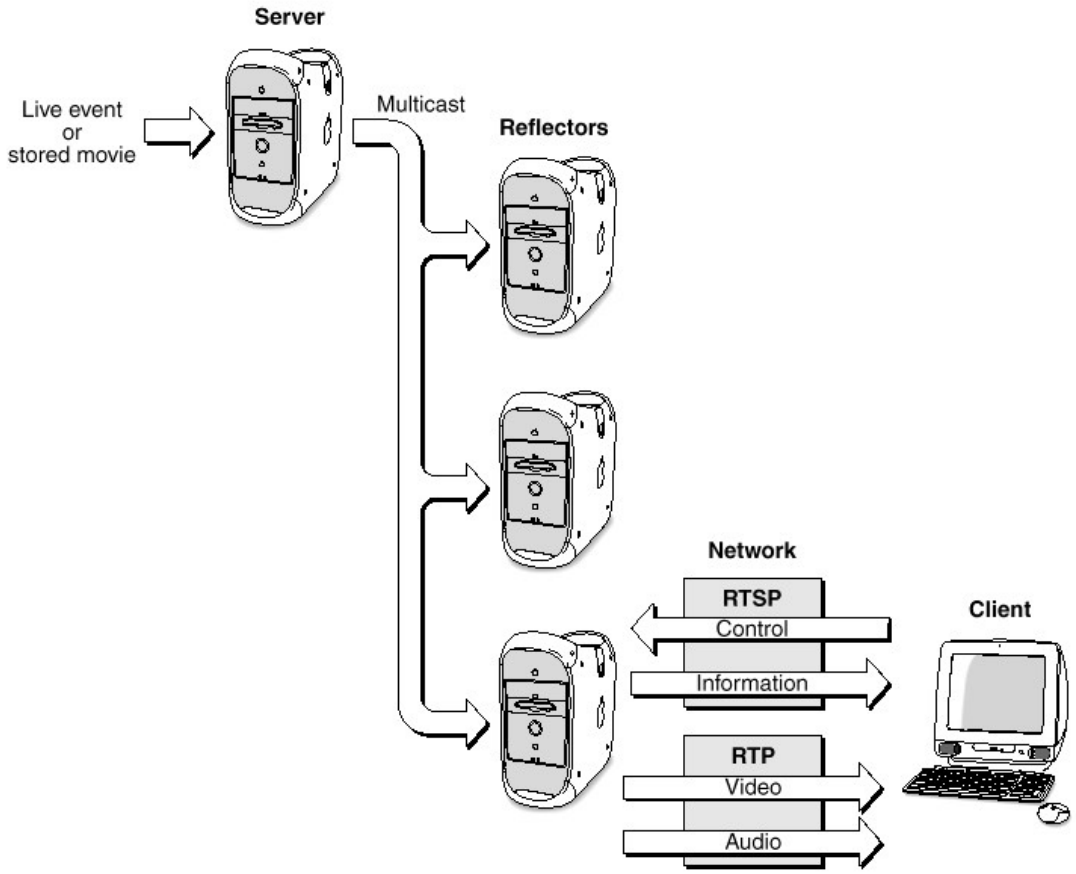


Abbildung 2-3: Multicast Streaming über einen Reflektor

2.4 Streaming über HTTP

HTTP benutzt das TCP/IP Protokoll und stellt damit sicher, daß alle Pakete des zu übertragenden Movies empfangen werden. Notfalls werden verlorengegangene Pakete nochmals gesendet.

HTTP versucht, nicht streaming-fähiges Material in Echtzeit zur Verfügung zu stellen. Um in Echtzeit streamen zu können, muß die Bandbreite des Netzwerkes größer sein als die Datenrate des Movies. Falls die Bandbreite nicht groß genug ist, um in Echtzeit zu streamen, speichert der Client Daten lokal und gibt das Movie wieder, wenn genügend Daten empfangen wurden.

Ein großer Vorteil der Übertragung über HTTP ist, daß die meisten Firewalls und Proxys HTTP ohne es zu filtern durchlassen.

Jedes QuickTime Movie kann über HTTP gestreamed werden. QuickTime 4 unterstützt darüber hinaus Streaming über RTP.

2.5 Streaming über RTP/RTSP

Das Streaming über RTP bietet einige Vorteile gegenüber dem Streaming mittels HTTP/FTP.

RTP kann für Live-Übertragungen und Multicast benutzt werden.

Das Echtzeit-Streaming erlaubt dem Benutzer, lange Filme kontinuierlich zu gucken oder Übertragungen zu verfolgen, ohne daß mehr als ein paar Sekunden lokal gespeichert werden müssen.

Wenn für die RTP Übertragungen RTSP zur Kontrolle genutzt wird, kann der Benutzer zu jedem beliebigen Punkt in einem Film springen, ohne die dazwischen liegenden Daten herunterladen zu müssen.

Ein einzelner Track kann über RTP gestreamed werden, während über HTTP nur das Streaming von ganzen Filme möglich ist. RTP Streams können in einem Movie zusammengefaßt werden, indem „Streaming tracks“ genutzt werden.

Ein „Streaming track“ ist ein track in einem QuickTime Movie, der die URL von streamingfähigem Inhalt enthält.

Ein QuickTime Movie, welches „streaming tracks“ enthält kann ebenfalls „non-streaming tracks“ enthalten, dessen Inhalt lokal auf dem Client Computer vorliegt. Somit ist ein Zusammenspiel zwischen Live-Übertragungen und lokal vorhanden Daten möglich. Z. B. kann in einer Live-Übertragung auf Daten verwiesen werden, die auf einer CD-ROM lokal vorliegen.

RTP benutzt das UDP/IP Protokoll, welches nicht versucht verlorene Pakete nochmals zu übertragen. Dies wäre auch bei einem Multicast bzw. bei einer Live-Übertragung nicht wünschenswert.

2.6 QuickTime Video Codecs

QuickTime bringt schon von Haus aus einige nützliche Video Codecs mit sich:

- Sorenson Video
  bietet schon bei einer niedrigen Datenrate eine sehr gute Bildqualität. Allerdings ist die Komprimierung sehr langsam und die Anforderungen an die Hardware sind höher als z.B. bei Cinepak. Dafür sind die Ergebnisse bei „halber Cinepak Datenrate“ immer noch erstaunlich gut. Der Codec eignet sich vor allem für Veröffentlichungen im WWW und auf CD-ROM.
- Cinepak
  gibt sich mit weit weniger Hardware zufrieden. Schon ein `486 reicht aus, um mit Cinepak kodierte Videos wiederzugeben. Allerdings sind die Ergebnisse bei einer Datenrate von unter 30 KB/s nicht mehr so schön und die Komprimierung ist ebenfalls sehr langsam. Cinepak eignet sich vor allem für die Wiedergabe von CD-ROM, wobei die Bildqualität eher mittelmäßig ist.
- Indeo 3.2
  ist etwas besser als Cinepak, wenn „talking heads“ komprimiert werden. Die Komprimierung ist auch etwas schneller. Indeo 3.2 eignet sich aber auch nur für CD-ROM Publikationen.
- Indeo Video Interactive 4
  bietet eine exzellente Bildqualität ähnlich wie MPEG. Allerdings wird hierfür auch mindestens ein schneller Pentium gebraucht. Hauptanwendungsgebiet sind wieder CD-ROM Publikationen.
- Video
  eignet sich gut zum Testen, aber nicht für das Endprodukt.
- Photo-JPEG
  hat eine ziemlich gute Bildqualität und eignet sich gut für Slideshows oder Movies mit einer geringen Frame-Rate.
- Component Video
  eignet sich zum „capturen“ von Video und als „Zwischenformat“, aber nicht für das Endprodukt.
- MJPEG
  biete bei 100% Qualität einen nur minimalen Bildverlust, allerdings werden hierfür auch viel CPU Power benötigt. Bei großen Bildern oder hohen Frame-Rate sind keine weichen Übergänge ohne Zusatzhardware möglich.
- MPEG-1
  bietet eine sehr gute Bildqualität. Allerdings wird hierfür entweder spezielle Hardware oder ein schneller Computer gebraucht. MPEG-1 ist nur in der MAC Version vorhanden.

3 Design und Implementierung

Mit QuickTime 4 sollten eine Reihe von streamingtechnischen Besonderheiten implementiert werden. Unter anderem der Verweis auf externe (nicht im Movie mitgespeicherte) Tracks sollte möglich gemacht werden. Hiermit ist es möglich einen Film aus den unterschiedlichen Ressourcen für den Nutzer nicht sichtbar zusammen zu mischen. Ein Benutzer kann sich einen Film per HTTP Protokoll herunterladen, dieser Film verweist u.U. auf aktuellere RTP Streams im Internet sowie auf Standard Musikuntermalung und Video Prologe auf der lokalen Festplatte. Dieses mischen von beliebigen Tracks ermöglicht eine Ressourcen schonende Präsentation von Video Daten auch über Netzanbindungen mit geringer Bandbreite. Auf Server Seite wurde besonderer Wert auf Portabilität und Effizienz gelegt. Der Darwin-Streaming-Server wurde komplett in C++ implementiert. Durch die Nutzung von Objektorientierten Konzepten wie Vererbung und Polymorphie konnten die wichtigsten Komponenten des Servers sehr gut gekapselt werden, was ihre Wiederverwendbarkeit in Bibliotheken stark vereinfacht.

Der Server besteht aus drei Hauptkomponenten:

- RTP Server
- RTSP Server
- „Common Utilities“.

Des weiteren wird die QT4 Fileformat Bibliothek genutzt, die allerdings vom Quelltext des Servers getrennt ist.

3.1 Der RTP/RTSP Server

Der RTP sowie der RTSP Server besitzen eine nahezu identische Struktur. Beide sind modularisiert und „thread safe“. Jede Klienten Instanz benötigt eine zugehörige Server Instanz, diese läuft als eigenständiger Thread ab. Der RTSP Server bietet folgende Funktionalität:

- RTSPWebStatsModule: Dies ermöglicht das Lesen von Serverstatistiken über das WWW.
- RTSPWebDebugModule: Debugging Fähigkeit via WWW
- RTSPLoggingModule: Schreibt Fehler und Status Meldungen in zugehörige Logfiles.
- RTSPSvrControlModule: (nur MacOS-X) Erlaubt die Administration mit Hilfe der Streaming Server Admin Konsole.
- RTPInterfaceModule: Steuert die benötigten RTP Instanzen die zu einer RTSP Session gehören.

Der RTSP Server ist in folgende Funktionseinheiten aufgeteilt:

- RTPAccessLogModule: Schreibt Zugriffsinformationen in ein Logfile.
- RTPFileModule: Kapselt den Zugriff auf QuickTime 4 Film Dateien mit Hilfe der QTFile Bibliothek.
- RTPReflectorModule: Zum „reflektieren“ von „live Broadcasts“ über das Mbone.
- RTPFlowControlModule: Analysiert RTCP Informationen und veranlaßt die Änderung von Datenraten oder das Schließen von RTP Verbindungen.

Weder der RTP noch der RTSP Server greifen direkt auf Betriebssystemmittel zurück. Alle Zugriffe auf das Filesystem, Thread oder Socket Implementierung greifen auf diese abstrakten Schnittstellen zu:

3.2 Die „Common Utilities“

Die „Common Utilities“ Bibliothek ist ein Werkzeugkasten, welcher folgende Betriebssystem Ressourcen zur Verfügung stellen soll:

- Thread Management: Die pthread (Posix Threads) Implementierung variiert zwischen den einzelnen Betriebssystemen. Die von den QuickTime 4 Servern benutzte API repräsentiert ein auf allen Plattformen unterstütztes „subset“.
- Daten Strukturen: Zur Verwaltung von RTP/RTSP Klienten werden einige Datenstrukturen benötigt. Diese werden reintrant („thread sicher“) von dieser Bibliothek bereitgestellt.
- Sockets: Unter MacOS-X existiert eine spezielle Schnittstelle über der BSD Sockets Ebene. Werden Daten auf einem Socket gelesen (alle benutzten Sockets innerhalb des QT4 Servers sind asynchron (non-blocking), wird ein Event Objekt an das zugehörige Task Objekt geschickt (welches vorher registriert werden muß). Nachdem das Objekt abgearbeitet wurde, wird dem Socket signalisiert, daß neue Events entgegen genommen werden können. Diese Schnittstelle wurde mit Hilfe des „select“ System Aufrufs auf andere Betriebsysteme portiert.
- Parser Utilities: Eine eigene Stringbibliothek Bibliothek. Analysieren und Formatieren von Zeichenketten werden unterstützt. Auch die Übersetzung in andere Sprachen kann hiermit gekapselt werden.

QTFile sowie die Common Utilities ermöglichen eine vereinfachte Portierbarkeit. Jedes POSIX konforme und Unix basierte Betriebssystem sollte ohne größeren Aufwand unterstützt werden können.

3.3 Fileformat

Ein Video Film im herkömmlichen Sinne ist ein kontinuierlicher Strom von Daten. Ein QuickTime 4 Film kann ebenso einfach aufgebaut sein, muß es aber nicht: Der Film ist nicht das Medium, sondern eine Organisationsstruktur für beliebig viele Datenströme, genannt „Tracks“.

Jeder Track referenziert einen Datenstrom, dies können Video oder Musikdaten, aber auch Untertitel, Werbebanner usw. sein. Wichtig ist die Unterscheidung zwischen tatsächlichen Media-Daten und den Film Metadaten. Beide Informationen können in derselben Datei enthalten sein, fall es nötig ist können die Media-Daten auch über mehrere Dateien verteilt werden (z.B. um Dateisystem Beschränkungen zu umgehen). Alle Informationen die ein Klient über den Film benötigt liefern die Meta Daten: Anzahl der Tracks, Codec Formate, Länge oder auch benötigte Bandbreite.

Das QuickTime Fileformat dient dazu, Dateien, Media- und Metadaten auf einem Datenträger (meist einer Festplatte) zu speichern. Die zugrundeliegende Datenstruktur ist ein Baum mit beliebigem Ausgrad. Die Knoten werden durch die „Atom“ Datenstruktur der QuickTime API repräsentiert. Bis zur QuickTime Version 3 hießen Atome einfach „Atom“, ab Version 4 werden diese einfachen Datenstrukturen durch **QTAtom** Strukturen ersetzt. Wir gehen im folgenden nur noch auf die neue Spezifikation ein, da Apple verlangt, daß neue Applikationen nur „QTAtom“ nutzen. Im weiteren Dokument werden wir allerdings weiterhin nur von „Atomen“ sprechen.

Die gesamte hierfür notwendige Funktionalität ist in einer Bibliothek zusammengefaßt: QTFile (libQTRTPFile.a). Sie enthält den Code zum lesen, parsen und schreiben von QuickTime Film Dateien.

Atome existieren in zwei Varianten:

- Container Atome: Diese enthalten weitere Atome beliebiger Art und Menge.
- Blatt Atome: In diesen werden die tatsächlichen Daten verwaltet.

Alle Informationen, die man zur Speicherung eines QuickTime Films benötigt werden mit Hilfe dieser Datenstrukturen modelliert. Alle Daten werden im „Big Endian“ Byteformat gespeichert. Bei Container Atomen wird die Größe der Kind-Atome zu der des Vater-Atoms hinzugerechnet.

Da jede Information in einem Atom gespeichert ist, erlaubt dies allen Applikation jede Film Datei zu lesen und Atomtypen, die sie nicht versteht können, zu ignorieren. Im folgenden Kapitel werden die wichtigsten Atom Typen erklärt.

3.3.1 Das Movie Atom

Das Movie Atom ist der äußerste Atom Typ. Es fungiert als Container für alle in einem Film benötigten Informationen. Diese Informationen werden wiederum mit anderen Atomen modelliert. Die Struktur eines QT4 Films mit Video und Musik Track sowie den zugehörigen „Hint Tracks“ sieht auf der Atom Ebene so aus:

Abbildung 3-1: Das Movie Atom

Auf der obersten Ebene kann man noch nicht entscheiden, welche Medien Typen in den Tracks verwaltet werden. Dazu müßte man die Track Container Atom öffnen und sich ihre Header Informationen sowie die „Media Data“ Atome in den einzelnen Tracks anschauen. Das „User Data“ Atom bietet die Möglichkeit Benutzer Informationen zu speichern, wie z.B. Copyright Informationen, Regisseur, Produzent oder spezielle Informationen, wie der Film abgespielt werden soll, wie die gewünschte Bildgröße auszusehen hat oder ob einzelne Bildframes übersprungen werden dürfen.

In den folgenden Kapiteln werden einige Atom Typen und ihr Nutzen näher beschrieben.

3.3.2 Das Movie Header Atom

In diesem Atom (welches in einem Movie Atom existieren muß) wird die Charakteristik des gesamten Films festgelegt. Es existieren u.a. folgende Felder:

- Creation time [32 bit int]: Sekunden seit Mitternacht am 01.01.1904, legt den Zeitpunkt der Atom Generierung fest
- Modification time [32 bit int]: Zeitpunkt der letzten Modifikation des Atoms.
- Time scale [32 bit int]: Anzahl der Zeit Einheiten, die in einer Sekunde vergehen
- Duration [32 bit int]: Gesamtdauer des Film in Zeit Einheiten. Hier wird automatisch die Länge des längsten Tracks eingefügt.
- Matrix [36 Byte]: Die Abbildungsmatrix, die benötigt wird, um einen Bildpunkt auf dem Klienten Frame anzuzeigen. Hiermit werden bereits Translation, Stauchung und Drehungen unterstützt.
- CurrentTime [32 bit int]: Der Zeit Wert des aktuellen dargestellten bzw. gesendeten Frames.

QuickTime Filme besitzen also ihr eigenes Zeit und Bild Koordinatensystem.

3.3.3 Das „Track“ Atom

Track Atome repräsentieren die einzelnen „Spuren“ eines Films. Ein Film kann aus einem oder mehreren Spuren bestehen. Alle Spuren sind voneinander unabhängig, sie können ihr eigenes Zeitliches und Räumliches Koordinatensystem verwalten. Zu jedem Track gehört exakt ein Medien Atom und ein „Track Header Atom“. Alle anderen möglichen Kind Atome sind optional.

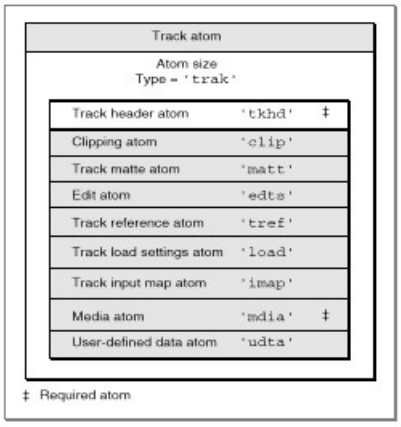


Abbildung 3-2: Das Track Atom

Das mit „Edits“ bezeichnete Atom enthält „Schnittlist“ Atome. Ein Track kann so mit externen Programmen) editiert werden, daß bestimmte Teile eines Tracks übersprungen, wiederholt oder gar nicht abgespielt wird. Fehlt das Edits Atom wird angenommen das der gesamte Track gespielt werden soll.

Track Header Atom

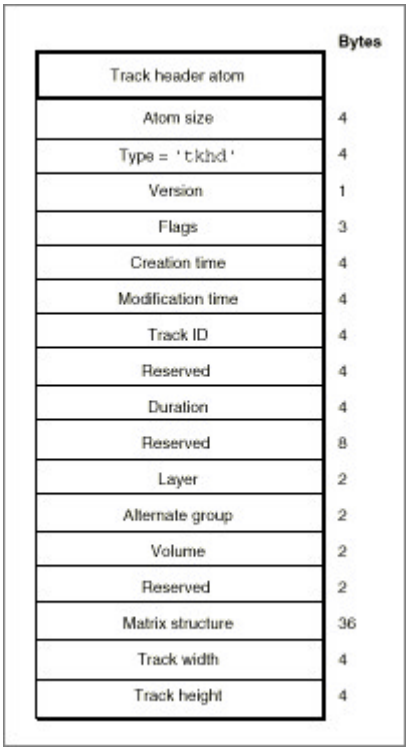


Abbildung 3-3: Das Track Header Atom

Die interessantesten Felder innerhalb eines Track Header Atoms sind:

- Header Flags: Hier wird festgelegt, ob ein Track aktiv oder inaktiv ist, ob er innerhalb des Film überhaupt verwendet wird, oder ob er Teil des Previews bzw. des Posters ist.
- Layer: Die „stacking“ Order der Tracks gibt an welcher Track *über* einem anderen gespielt werden soll. So kann z.B. ein Text Track über einen Video Track gelegt werden. Der höher liegende Track sollte natürlich einen transparenten Hintergrund besitzen.
- Alternate Group: Hier wird eine Menge von Tracks referenziert, die die gleichen Daten in unterschiedlicher Auflösung bieten. Dies wird für Alternate Data Rate Filme benötigt.

- Matrix: Jeder Track kann die im Movie Header Atom spezifizierte Translationsmatrize ergänzen. Die beiden Berechnungen werden hintereinander ausgeführt.

Ansonsten sind die Felder zum Movie Header Atom identisch.

3.3.4 Das Media Atom

Media Atome definieren die Film Daten des Tracks. Sie kapseln die tatsächlichen Media Daten in weiter spezialisierten Atome. In ihnen werden auch die Referenzen auf die Mediahandler verwaltet, diese wissen, wie sie mit den gespeicherten Mediadaten umzugehen haben.

- DataReference Atom: enthalten Daten in Tabellen Form, die auf die Sample Atome verweisen.
- Sample Atom: die tatsächlichen Codec abhängigen Medien

3.3.5 Free Space Atome

Atome des Typs „free“ und „skip“ kennzeichnen nicht genutzten Platz in einem QuickTime File. Wenn ein Film oder ein Audio Track editiert wird, können diese Atome genutzt werden um zusätzliche Informationen aufzunehmen. Werden Teile eines Tracks gelöscht, wird der frei gewordene Platz nicht sofort an das Filesystem zurückgegeben, sondern die Sample Atome als frei deklariert. Diese Atome bestehen ausschließlich aus der Typinformation, der Größenangabe und einer bestimmten Anzahl von leeren Bytes.

3.4 Hint Tracks

Hint Tracks sind beim Konvertieren erzeugte "Schatten" Tracks. Zu jedem streamfähigen Track existiert ein Hint Track. Diese Hint Tracks sind im Vergleich zu den QuickTime Medien Daten relativ klein. Ihre Existenz verhindert nicht das Betrachten des Films via HTTP/FTP oder von einem lokalen Datenträger, sie werden nur beim Streamen der Daten genutzt. Auch sie werden mit Hilfe des QuickTime Fileformats gespeichert. Sie liefern dem Streaming Server beim Export berechnete Informationen zur Umwandlung der Media Daten in RTP Pakete. Umwandlung von Video oder Audio Sample Atomen in RTP fähige Pakete erfordert CPU Zeit. Die "Hint Tracks" nehmen der Serversoftware die meiste Arbeit ab, da die RTP Paketköpfe dann schon existieren. Erst dadurch sind QuickTime Server in der Lage eine große Anzahl von Clienten mit Daten zu versorgen.

Ein weiterer wichtiger Faktor ist, daß die „Hint Tracks“ dafür Sorgen, daß der Server nichts über das spezielle Encoding, das benutzte Kompressionverfahren oder auch den Medientyp der Daten, wissen muß. Die während der Speicherung des Filmes generierten Hint Tracks enthalten alle Informationen, die nötig sind, effizient ein RTP Paket zu generieren:

- Position und Länge der tatsächlichen Streaming Daten
- Sowie Paket Header Informationen zum verwandten Codec

Ein QuickTime RTP Server muß also nur in der Lage sein, „Hint Tracks“ in einem QuickTime 4 Film zu identifizieren und zu lesen.

3.4.1 Beispiel API Aufrufe

Folgender C++ Quelltextauszug nutzt die QT4 File Bibliothek, um Informationen über die „Hint Tracks“ eines QuickTime Films auszugeben. C++ Preprozessor Direktiven, Argument-Analyse wurden weggelassen. Das Programm ist Teil des Streaming Server Quelltextes und im QTFile Unterverzeichnis mit Namen „QTFileInfo“ zu finden.

```
int main(int argc, char *argv[]) {
      // Temporary vars
      char               ch;

      // General vars
```

```
	QTTrack			*Track;
	// Open the file
	file.Open(*argv);

	Track = NULL;
	// Iterate over Tracks
	while( file.NextTrack(&Track, Track) ) {
	QTHintTrack  *HintTrack;
	time_t	unixCreationTime = Track>GetCreationTime()  +
QT_TIME_TO_LOCAL_TIME;
time_t unixModificationTime = Track->GetModificationTime()+ QT_TIME_TO_LOCAL_TIME;

	// Initialize the track and dump it.
	if( Track->Initialize() != QTTrack::errNoError ) {
		printf("!!! Failed to initialize track !!!\n");
		continue;
	}
	if(DeepDebug) Track->DumpTrack();

	// Dump some info.
	printf("-- Track #%02ld ----------------------\n",
			Track->GetTrackID());
		printf("   Name			: %s\n",
			Track->GetTrackName());
		printf("   Created on		: %s",
		asctime(gmtime(&unixCreationTime)));
		printf("   Modified on		: %s",
		asctime(gmtime(&unixModificationTime)));

		//
		// Dump hint information is possible.
	if( file.IsHintTrack(Track) ) {
		HintTrack = (QTHintTrack *)Track;
		printf("   Total RTP bytes    : %ld\n",
				HintTrack->GetTotalRTPBytes());
			printf("   Total RTP packets  : %ld\n",
				HintTrack->GetTotalRTPPackets());
			printf("   Average bitrate    : %.2f Kbps\n",
			((HintTrack->GetTotalRTPBytes() << 3) /
				file.GetDurationInSeconds()) / 1024);
		printf("   Average packet size: %ld\n",
			HintTrack->GetTotalRTPBytes() /
			HintTrack->GetTotalRTPPackets());

			UInt32 UDPIPHeaderSize =
				(56 * HintTrack->GetTotalRTPPackets());
			UInt32 RTPUDPIPHeaderSize =
				((56+12) * HintTrack->GetTotalRTPPackets());
					printf("   Percentage of stream wasted on UDP/IP
				headers    : %.2f\n", (float)UDPIPHeaderSize /
					(float)(HintTrack->GetTotalRTPBytes() +
						UDPIPHeaderSize) * 100);
		printf("   Percentage of stream wasted on RTP/UDP/IP headers:
			%.2f\n", (float)RTPUDPIPHeaderSize / (float)(
		HintTrack->GetTotalRTPBytes() + RTPUDPIPHeaderSize) * 100);
		}
	}

	return 0;
}
```

Diese Programm wurde auf einem Linux Server ausgeführt. Als Argument erhielt es den Dateinamen eines Beispielfilms:

```
[dirk@icmp:StreamingServer-3/QTFile]{597} ./QTFileInfo ugachaka.mov
-- Track #01 --------------------------
   Name               :
   Created on         : Tue Jan  4 14:31:36 2000
```

```
   Modified on        : Tue Jan  4 14:31:41 2000
-- Track #02 --------------------------
   Name               :
   Created on         : Tue Jan  4 14:31:36 2000
   Modified on        : Tue Jan  4 14:31:41 2000
-- Track #03 --------------------------
   Name               : Hinted Video Track
   Created on         : Tue Jan  4 14:31:37 2000
   Modified on        : Tue Jan  4 14:31:41 2000
   Total RTP bytes    : 519948
   Total RTP packets  : 487
   Average bitrate    : 167.39 Kbps
   Average packet size: 1067
   Percentage of stream wasted on UDP/IP headers    : 4.98
   Percentage of stream wasted on RTP/UDP/IP headers: 5.99
-- Track #04 --------------------------
   Name               : Hinted Sound Track
   Created on         : Tue Jan  4 14:31:38 2000
   Modified on        : Tue Jan  4 14:31:41 2000
   Total RTP bytes    : 1150920
   Total RTP packets  : 798
   Average bitrate    : 370.53 Kbps
   Average packet size: 1442
   Percentage of stream wasted on UDP/IP headers    : 3.74
   Percentage of stream wasted on RTP/UDP/IP headers: 4.50
```

Es werden einige Daten zur Größe der „Hinted Tracks“ im Verhältnis zur Größe der Media Daten ausgegeben. Wie man sieht, ist der prozentuale Anteil der in den „Hint Tracks“ gespeicherten RTP Paketköpfen im Vergleich zur Gesamtgröße des Film eher gering. Man kann während des Exportes mit QuickTime Pro eine nur auf den RTP Server optimierte Filmvariante anfordern. In dieser werden die zu den RTP Paketköpfen gehörigen Media Daten direkt in das RTP Paket kopiert, was ein fast lineares lesen der zu streamenden Pakete ermöglicht. Solche Filme sind besonders einfach zu streamen. Sie bieten sich also gerade für stark frequentierte Server an. Der Platzbedarf eines solchen Films steigt jedoch um das Doppelte.

3.4.2 „Packetizer“ und „Reassembler“

Ein Packetizer ist eine Komponente, die dazu dient, QT Media in Pakete zu unterteilen, die sich zum Transport über RTP eignen. Da RTP auf UDP Übertragungen basiert, muß mit Verlust, Verzögerung und „Out of order“ Zustellung gerechnet werden. Das verwendete Codec muß bestimmen, wie auf solche Ausnahmen reagiert wird, ohne die Film bzw. Audio Qualität zu stark einzuschränken. Aus diesem Grund sind für die einzelnen Codecs meist spezielle Packetizer und Reassembler vorgesehen, die benutzt werden, wenn ein Film beim Export mit „Hint Tracks“ versehen wird. Das Standard Paketizer/Reassembler Paar kann alle Medien und Codectypen unterstützen. Um die beste Übertragungsqualität zu erreichen, ist es jedoch notwendig, spezielle Packetizer und Reassembler anzubieten, die die speziellen Eigenschaften des benutzten Codec optimal unterstützen. So können unterschiedliche Strategien implementiert werden, um auf die oben beschriebenen Übertragungsmißstände zu reagieren. Zum Beispiel kann bei einem Stream mit hoher Datenrate „überflüssige“ Bandbreite dazu genutzt werden öfter ein „Key Frame“ (ein vollständiges Bild) zu senden. So wären die Paketverluste weniger tragisch.

4 Streaming Movies Tutorial

In diesem Tutorial wird beschrieben wie man:

- Den von Apple im Quelltext freigegebenen Server übersetzt und installiert.
- QuickTime 4 Filme mit „Hint Track“ Informationen generiert.
- Diese Filme im Netzwerk via RTP/RTSP zur Verfügung stellt.

4.1 Darwin Streaming Server

Der „Darwin Streaming Server“ von Apple wird für die freien Unix Derivate Linux und FreeBSD angeboten. Implementiert wurde er vollständig in C++ unter Verwendung von Posix Threads.

Der Server Quelltext kann unter

http://www.publicsource.apple.com/projects/streaming/

heruntergeladen werden. Dazu muß man sich allerdings als Entwickler erfassen lassen.

4.1.1 Bau und Installation der Software

Der Streaming Server wird als einzelnes File im TAR Format ausgeliefert. Nach dem Auspacken findet der Nutzer drei Unterverzeichnisse vor:

- QTFile: Die Bibliothek zum Öffnen, Lesen und Parsen von QuickTime Filmdateien.
- QTProxy: Ein RTP/RTSP unterstützender Firewall Proxy
- RhapServer: Der RTP/RTSP Server
- QTPlayListBroadcaster: Zu diesem Programm existiert jedoch keine Dokumentation und seine Funktionalität läßt sich nicht aus den Quelltexten bestimmen.

Ein beigelegtes Bourneshell Skript mit Namen *doit* dient als Makefile Ersatz. Hier muß für eine Installation unter Linux der Compiler Name angepaßt werden:

- Ist der EGCS Compiler (mindestens Version 1.1.2) installiert, so muß keine Änderung durchgeführt werden.
- Ist ein GCC Compiler (mindestens Version 2.7.2.3) installiert, müssen die EGCS Aufrufe in den Zeilen 15-17 durch den String „gcc“ ersetzt werden, als Linker kommt in jedem Fall der EGCS bzw. der GCC zum Einsatz.

Die vorhandenen „Makefile“ Dateien sind für MacOS-X bestimmt. Sie sind durch die „Makefile.POSIX“ Varianten zu ersetzen. Ansonsten muß man darauf achten, daß folgende Bibliotheken mit mindestens der angegebenen Versionsnummer installiert sind:

- Posix Threads: /usr/lib/libpthread in der Version 0.8.
- Die C++ Standard Template Bibliothek /usr/lib/libstdc++.so.2.X in der Version 2.9.
- Die Libc sollte mindestens die Version 2.0 haben.

Im Unterverzeichnis QTFile wird eine Bibliothek mit Namen libQTRTPFile.a generiert, diese muß per Hand in das RhapServer und QTProxy Verzeichnis kopiert werden, ansonsten bricht der Linker beim Linken der Datei „QuickTimeStreaminServer“ mit einer Fehlermeldung ab.

Es wird ein einziges ausführbares Programm generiert:

RhapServer/QuickTimeStreamingServer

4.1.2 Konfiguration des Proxys

Der RTP/RTSP Proxy für Linux/BSD muß im Verzeichnis QTProxy/proxy_unix kompiliert werden. Er sucht seine Konfigurationsdatei unter /etc/qts_proxy.conf. Es existieren nur vier Variablen:

Variablename	Voreingestellt	Erklärung
allow	17.0.0.0/8	Netzmaske aller erlaubten Klienten. 0.0.0.0/0 würde die Benutzung des Proxys für das gesamte Internet freigeben.

users	150	Maximale Anzahl gleichzeitig erlaubter Benutzer. Hier sind RTSP Verbindungen gemeint.
listen	554 / 7070	Auf diesen Ports erwartet der Proxy die Klienten RTSP Verbindungen.
port-range	10000-15000	Der Proxy wird nur UDP Ports anfordern, die in diesem Bereich liegen.

Bis auf die „allow“ Variable sind alle voreingestellten Werte sinnvoll. Allerdings muß der Proxy mit „root“ Berechtigung gestartet werden, da er sonst nicht auf den Port 554 zum Lesen und Akzeptieren von TCP Verbindungen zugreifen kann.

4.1.3 Konfiguration des Servers

Der Streaming Server benutzt nur eine einzige Konfigurationsdatei, *QTSS.conf,* die unter */etc* installiert werden muß. Im folgenden werden die wichtigsten Variablen und ihre Bedeutung beschrieben. Alle Variablen, die die Leistung des Servers stark beeinflussen, sind **fett** geschrieben, Variablen, die den Klienten beeinflussen, *kursiv* :

Variabelname	Voreingestellt	Erklärung
rtsp_port	554 7070	Die TCP Ports auf die der Streaming Server horcht. Beide können in RTSP URL’s verwendet werden.
rtsp_timeout	60 sec	Der Idle Timeout des Servers für RTSP Verbindungen.
maximum_connections	1000	Größte Anzahl erlaubter RTSP Verbindungen.
movie_folder	/Pfad	Absoluter Verzeichnisname in dem die Film Dateien liegen.
user_movie_folder	/public_movies	User Verzeichnis für Streaming Daten. Vergleichbar mit dem ~/public_html Verzeichnis für WWW Daten.
request_loggin	Disabled	Muß auf „enabled“ gestellt werden damit Verbindungen mitgeloggt werden.
bind_ip_adress	0 (alle Interfaces)	Auf welchen Netzwerkadressen soll der Server horchen? Nur interessant bei Maschinen mit mehr als einer IP.
maximum_bandwidth	1000000 (10Mbit)	Maximum der Bandbreite die für **alle** RTP Verbindungen genutzt wird.
buffer_seconds	8	Wie viele Sekunden werden für jeden gesendeten Track im Hauptspeicher gebuffert.

rtp_timeout	60 sec	Der Idle Timeout des Servers für RTP Verbindungen.
average_bandwidth_update	60 sec	Zeitraum zwischen den gesamt Bandbreite neu Berechnungen.
save_play_duration	600 sec	Falls mehr Bandbreite als erlaubt genutzt wird, werden die jüngsten Klienten getrennt, allerdings nur, wenn ihre Verbindung nicht länger als save_play_duration Sekunden alt ist!
sdp_url	-	Dieses URL wird in die dynamisch generierten SDP Daten eingefügt. Sollte auf die Homepage der Organisation verweisen.
admin_email	-	Administrativer Kontakt
web_stats_url	-	Die Statistik Seite des Servers. Falls gesetzt, zu erreichen unter: http://server:554/stats_page.
loss_thin_tolerance *num_losses_to_thin*	30/3	Wenn der Paket Verlust über num_losses_to_thin RTCP Paketen loss_thin_tolerance (in %) übersteigt, wird die Bitrate des Streams gesenkt.
loss_thick_tolerance *num_losses_to_thick*	5/6	Wenn der Paket Verlust unter *loss_thick_tolerance Prozent* für *num_losses_to_thick* RTCP Pakete liegt, wird die Bitrate des Streams erhöht.

4.1.4 Performance Tips

Alle diese Informationen stammen aus der QuickTime 4 Entwickler Email Diskussionliste, und sind prinzipiell nur für einen QuickTime 4 Server mit hoher Belastung relevant. Weniger als 50 Klienten sollten sich auch bei einem nicht optimierten Server kaum auswirken.

Timeouts

Eine wichtige vom Betriebssystem vorgegebene Systembeschränkung ist die Anzahl der möglichen offenen Filedeskriptoren. Im Falle von Linux sind dies 256 (beschränkt durch den „select“ System Aufruf, „poll“ unterstützt bis zu 1024, wird aber noch nicht durch den Darwin Server unterstützt). Alle offenen RTSP Verbindungen nutzen einen Filedeskriptor, es sind deshalb *nur* 256 offene RTSP Verbindungen möglich. Der **rtsp_timeout** sollte also möglichst klein gehalten werden. Für den **rtp_timeout** gelten solche Einschränkungen nicht, es können beliebig viele UDP Adressaten existieren (eine implizite Beschränkung ist hier jedoch die maximale Anzahl von Threads, für jeden RTSP Klienten benötigt der QuickTime 4 Server einen Thread. Bei Linux sind z. Z. 1024 Threads per Prozeß möglich).

Bandbreite und Anzahl der Klienten

Die maximal zu nutzende Bandbreite wird durch **maximum_bandwidth** bestimmt. Der Default von 1000000 (10 Mbit) ist bei voller Auslastung zuviel für eine normale 10Mbit Netzwerkkate (RTSP

Verbindungen werden nicht mitgerechnet). Auch muß berücksichtigt werden ob der Rechner noch andere Aufgaben im Netzwerk zu leisten hat. Für einen dedizierten QuickTime 4 Server sollten 9Mbit für eine 10Mbit Netzwerkkarte sowie 95Mbit für eine 100Mbit Verbindung gewählt werden. Die Variablen zur Neuberechnung der Datenrate zu den einzelnen RTSP Klienten:

- *loss_thin_tolerance, num_losses_to_thin:* um zu entscheiden, wann die Datenrate vermindert wird
- *loss_thick_tolerance, num_losses_to_thick:* um zu entscheiden, wann die Datenrate erhöht wird

sollten nur geändert werden, wenn genügend Daten vorliegen (stark schwankende Bandbreite, große Anzahl Modem Klienten) die ein Experimentieren mit geänderten Parametern lohnenswert erscheinen läßt. Auf QuickTime 4 Servern mit sehr großer Belastung sollte **average_bandwidth_update** groß gewählt werden, um eine zu häufige Neuberechnung der Variablen, sowie ein Prüfen der Bandbreite abhängigen Aktionen zu unterdrücken. Laut Aussage eines Entwicklers ist dies eine teure Operation, da viele durch ein Semaphore geschützte Variablen gelesen und geschrieben werden müssen.

Filesystem

Das QT4 Fileformat wurde so entworfen, daß ein effizienter Zugriff auf Filmdateien möglich ist. Auch sollte die CPU weitgehend vom Kodieren/Dekodieren verschont bleibt (durch Hint Tracks). Nichtsdestotrotz müssen die Daten von einer Festplatte gelesen werden. Existieren eine größere Anzahl von Filmdateien und mehrere Klienten, die auf diese zugreifen, so kann es passieren, daß die Festplatten zuviel Zeit damit verschwenden, die benötigten Datenblöcken anzusteuern und weniger Zeit finden sie zu lesen. Die RTSP Threads lesen meist nur kleinere Blöcke von Daten (Informationen für 8 Sekunden per Default), und dies würde die gesamt Leistung des Servers vermindern. Um dies zu verhindern, sollten die Filmdateien möglichst auf verschiedenen physikalischen Medien gespeichert werden. Da der QuickTime 4 Server jedoch nur aus einem Verzeichnis lesen kann (**movie_folder**), müssen sie mit dem Link („ln“) Befehl im Server Verzeichnis erreichbar gemacht werden. Es ist auch in jedem Fall ratsam ein SCSI bzw. Firewire Plattensystem zu nutzen. IDE Plattensysteme eignen sich weniger für IO lastige Serveraufgaben, da die Kontroller weniger intelligent sind und Fähigkeiten wie command_queueing bzw. reordering nicht unterstützt werden. Diese sind aber bei stark konkurrierendem Zugriff notwendig.

4.2 Erzeugung von QuickTime Filmen

Die einzige Möglichkeit QuickTime 4 Medien zu erzeugen, die wir evaluieren konnten, ist die Verwendung von QuickTime 4 Pro. Mit diesem von Apple zu Verfügung gestellten Programm ist es möglich, Filme in den unterschiedlichsten Formaten zu betrachten, zu editieren und als QuickTime 4 Film zu exportieren. QuickTime Filme existieren in zwei Varianten:

- Als Client Film: Dieser enthält entweder verschiedene streaming fähige Tracks mit den entsprechenden Medien Daten, oder eine URL die auf einen Film verweist, der von einem RTP bzw. einem HTTP/FTP Server angeboten wird. Ein durch RTP übertragener Film kann dann wiederum aus mehr als einem Track bestehen.
- Server Film: Dieser beinhaltet ebenfalls die Medien Daten, zusätzlich wird jedoch die Existenz von „Hint Track“ Informationen verlangt. Für jeden Stream fähigen Track wird ein Hint Track benötigt, damit aus ihnen effizient RTP Pakete generiert werden können.

QuickTime 4 Pro bietet die Möglichkeit nur Audio (optimiert für Musik oder Sprache), oder Video und Audio (Sorenson Encoding) für verschiedenste Bandbreitenbedürfnisse zu generieren. Bei den Sorenson Encodings wird vereinfachend „CD ROM“ Single oder Double Speed angeboten (90 bzw. 150 kbit benötigte Übertragungsgeschwindigkeit pro Sekunde). Die Audio Exporte können feiner aufgeschlüsselt werden, hierbei kann darauf geachtet werden, ob gesprochene Dialog, Musik oder Midi Tonspuren gesendet werden sollen.

4.2.1 Alternate Data Rate Filme

Dieser Filmtyp ist mehr eine Referenz auf eine Menge von Filmen, die mit demselben Inhalt aber unterschiedlichen Bandbreitenbedürfnissen aufgezeichnet wurden. Mit dem von Apple für Windows und MacOS zur Verfügung gestellten Programm **MakeRefMovie** kann eine solche Referenz Filmdatei angelegt werden.

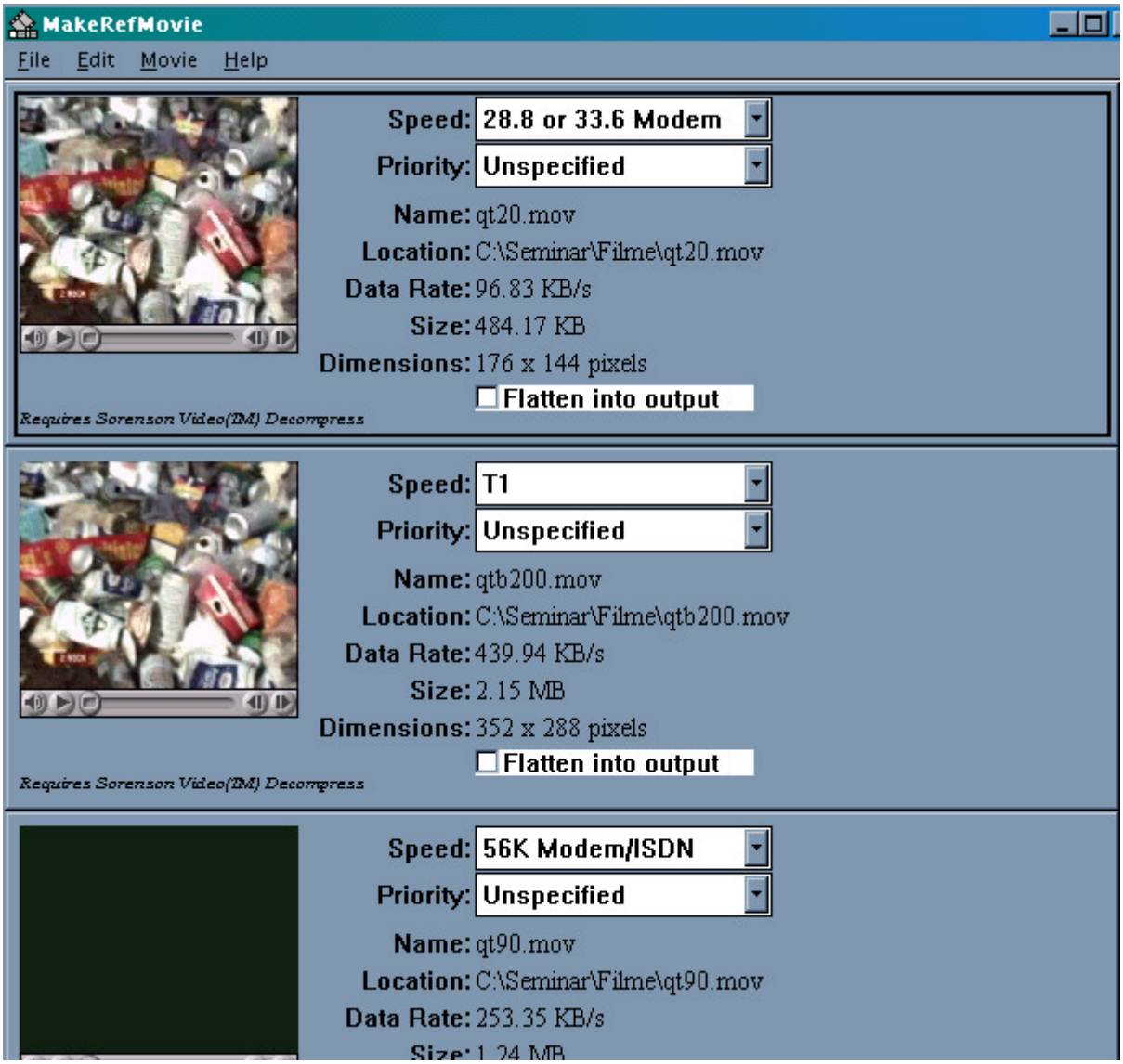


Abbildung 4-1: MakeRefMovie

In diesem Screenshot wird ein Referenzfilm mit drei Filminstanzen für die Datenraten 20Kbit (geeignet für ein 28.8 Modem), 90Kbit (Single Speed CDROM) und 150 Kbit (Double Speed CDROM) erschaffen (die Angaben im Screenshot sind falsch). Dieser Referenzfilm kann dann direkt von der Platte oder via RTSP/RTP betrachtet werden. Der QuickTime Player informiert den Streaming Server über die bevorzugte Geschwindigkeit (wählbar über den Präferenzen Menüpunkt) und erhält den Film mit der angemessenen Datenrate als RTP Stream gesendet.

4.3 Empfang

Zum RTP Empfang und Abspielen der Filme existiert der Apple QuickTime Klient. Bei seiner Installation wird auch ein WWW Browser Plugin installiert (für alle installierten Web Browser), mit dem Filme innerhalb einer Webseite abgespielt werden können.

Der QT4 Spieler erlaubt das Abspielen von bis zu 40 unterschiedlichen Video und Sound Dateiformaten. Er unterstützt das Verändern der Lautstärke, der Balance, der Mixer Einstellungen und der Abbildungsgröße. Wenn nur Sound Tracks abgespielt werden; verschwindet das Bild Anzeigefenster.

Abbildung 4-2: Audioeinstellungen im Client

5 Literatur

[Adb 99] (adb):
QuickTime 4.1 mit Werbeeinblendungen,
c‘t 24/1999, S. 33, Heise Verlag

[Adle 99] Adler, Douglas:
QuickTime Live Conference
L.A. Convention Center, November 7-11,1999
http://www.hbase.net/quickTimeLive/QTLiveConfRep.html [Stand: 08.01.2000]

[Appl 99a] Apple Computer, Inc.:
QuickTime Streaming,
http://developer.apple.com/techpubs/quicktime/qtdevdocs/PDF/QTStreaming.pdf [Stand: 08.01.2000]

[Appl 99b] Apple Computer, Inc.:
QuickTime File Format, Specification, May 1996,
http://developer.apple.com/techpubs/quicktime/qtdevdocs/PDF/QTFileFormat.pdf [Stand: 08.01.2000]

[Appl 99c] Apple Computer, Inc.:
QuickTime - Streaming
http://www.apple.com/quicktime/authoring/hinttrack.html [Stand: 08.01.2000]

[Appl 99d] Apple Computer, Inc.:
QuickTime – Alternate Data Rate,
http://www.apple.com/quicktime/authoring/datarate.html [Stand: 08.01.2000]

[Appl 99e] Apple Computer, Inc.:
QuickTime – Version 4.0,
http://www.apple.com/quicktime/upgrade/ [Stand: 08.01.2000]

[Appl 99f] Apple Computer, Inc.:
Open Source Projects at Apple – Darwin Streaming Server,
http://www.publicsource.apple.com/projects/streaming/ [Stand: 08.01.2000]

[KoMa 99] Kobylinska, Anna – Martins, Filipe:
Alles fließt
Live-Video im Internet mit QuickTime 4.0,
iX 8/1999, S. 73-75, Heise Verlag

[Terr 99a] Terran Interactive, Inc:
Codec Central – Quick Time
http://www.terran.com/Codec/Central/Architectures/QuickTime.html [Stand: 08.01.2000]

[Terr 99b] Terran Interactive, Inc:
How to Produce High-Quality QuickTime
http://207.220.219.69/TLA/HowToProduceGreatQT.pdf [Stand: 08.01.2000]

RealNetworks

Stefan Michaelis **Oliver Pauls**

FB Informatik, Uni Dortmund
michae00@marvin.cs.uni-dortmund.de
opauls00@marvin.cs.uni-dortmund.de

Zusammenfassung

Dieser Seminarbeitrag befasst sich mit den Einsatzmöglichkeiten von RealNetworks-Komponenten als Streaming-Plattform. Ziel der Ausarbeitung ist es, einen Überblick über den aktuellen Stand der Entwicklung zu geben. Schwerpunkt liegt dabei auf der Einsetzbarkeit der einzelnen Programmpakete und der Streamingformate. Der Leser soll in die Lage versetzt werden, den Nutzen der Software von RealNetworks für seinen spezifischen Anwendungsfall abschätzen zu können.

1 Die Firmengeschichte von Real

RealNetworks sieht sich als einer der Pioniere und Marktführer in der multimedialen Streamingtechnologie über das Internet. Zielsetzung ist die Transformierung des Internets auf eine Ebene, die Möglichkeiten des Broadcasts von Daten praktikabel und kommerziell interessant macht. Die in Seattle ansässige Firma bietet in zunehmendem Maße Produkte und Services an, die professionellen Einsatz fördern sollen.

Begonnen hat die Firmengeschichte 1995 mit der Freigabe der Version 1.0 des *RealPlayer*. Dieser hat sich von einem Werkzeug zum Abspielen reiner Audio-Ströme weiterentwickelt und beherrscht zur Zeit etliche weitere Formate. Seit der ersten Version ist der RealPlayer von mehr als 80 Millionen unterschiedlichen Nutzern von der Real-Internetseite geladen worden. Die aktuelle Rate von neuen Registrierungen übersteigt 175.000 pro Tag. Innerhalb der letzten zwei Jahre entspricht dies einer Steigerung um über 270%.

RealNetworks gibt an, dass ihre Software auf mehr als 85% aller Web-Seiten benutzt wird, um entsprechende Inhalte anzubieten. So befinden sich beispielsweise mehr als 1.700 Radiostationen mit ihrem Angebot im WWW. Stand dieser Angaben ist Dezember 1999.

2 Systembetrachtung

RealSystem vereinigt eine Palette an Werkzeugen zur Erzeugung, Verbreitung und Anzeige digitaler Daten über das Internet. Für jeden Arbeitsschritt stehen die entsprechenden Methoden zur Verfügung. Zielgruppe von RealNetworks ist dabei der Endanwender mit geringer bis professioneller Erfahrung. Kenntnisse im Umgang mit Computern werden erst im Bereich der Verbreitung von Daten benötigt. Die Kreativität der Designer steht im Vordergrund. Die Schritte zur Präsentation lassen sich in die folgenden Punkte unterteilen:

1. Encoding — Erstellung der streamingfähigen Daten
2. Server — Bereitstellen und übertragen der Daten
3. Player — Abspielen der Präsentation

Für jede Phase des Prozesses stellt RealNetworks die entsprechenden Tools zur Verfügung. In den meisten Fällen existieren zwei Varianten der Programme: Eine freie Version und das kommerzielle Gegenstück. Die frei verfügbaren Programme besitzen alle für die Präsentation nötigen Funktionen, manche allerdings nur eingeschränkt. Sie bieten allerdings genügend Komfort, um sich in der Welt des Streamings zu orientieren, für einfache Anwendungen sind sie häufig sogar ausreichend. Die kommerziellen Produkte besitzen an einigen Stellen verbesserte Funktionalität, wie z.B. Codier-Algorithmen, die eine erhöhte Qualität in Bild, Ton und geringere benötigte Bandbreite bieten.

Um den Anwender bei der Erstellung von streamingfähigen Daten und ganzen Präsentationen zu unterstützen sowie auch dem unerfahrenen Designer einen möglichst einfachen Einstieg zu ermöglichen, beginnt die Dokumentation von Real bereits bei der Gewinnung der Daten.

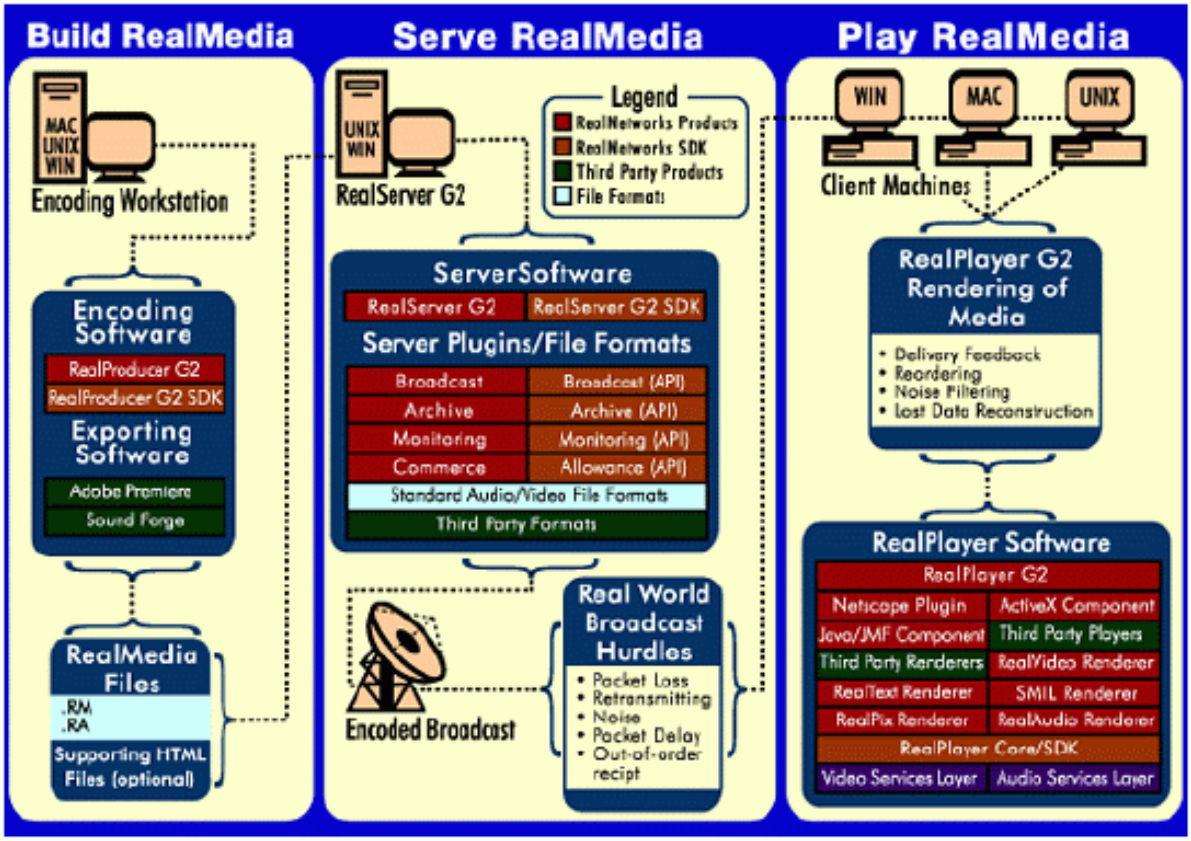


Abbildung 1: Die RealSystem-Ebenen

2.1 Player

Zum Abspielen der erstellten Präsentationen benötigt der Anwender einen Player. Dieser existiert als Stand-Alone Programm oder als Plug-In für diverse Internetbrowser. Unterstützt werden die meisten bekannten Betriebssysteme und Prozessoren, z.B. die einzelnen Versionen von MS Windows, Apple Macintosh, Linux, diverse Unix-Varianten. Einen eindeutigen Schwerpunkt legt RealNetworks allerdings auf die kommerziell interessanteren Systeme Windows und Macintosh. Für diese erscheinen die Player meist erheblich eher als für die übrigen Systeme. Zum Zeitpunkt der Entstehung dieses Dokumentes war die aktuelle Version des RealPlayer G2 nur für diese beiden Systeme verfügbar, unter Unix mußte sich der Nutzer mit der vorhergehenden Version 5 des Players begnügen.

In der neuesten Version G2 sind einige Verbesserungen hinzugekommen, vor allem was die

Breite der abspielbaren Formate angeht. Tabelle 2 gibt einen Überblick über Versionen der Player und deren streamingfähigen Daten.

Streamingformat	G2	5	4	3	2	1
RealAudio	J	J	J	J	J	J
RealVideo	J	J	J	N	N	N
RealFlash	J	J	N	N	N	N
RealPix	J	N	N	N	N	N
RealText	J	N	N	N	N	N
Offene Formate (WAV, AVI...)	J	N	N	N	N	N
SMIL	J	N	N	N	N	N

Abbildung 2: Streamingformat vs. Player-Version

Hinzugekommen sind Formate für das Streaming von einfachen Text- und Bilddateien sowie normalerweise nicht für das Streaming vorgesehene Formate oder Unterstützung für SMIL.

Die neuen Codier-Algorithmen sollen sich im Gegenzug zu den vorhergehenden Versionen noch einmal stark verbessert haben. Neu hinzugekommen sind Abspielmöglichkeiten für Ströme unterschiedlicher Bandbreite innerhalb des gleichen Streams. Diese Möglichkeiten werden in späteren Kapiteln dieses Dokumentes ausführlicher dargestellt.

Der RealPlayer ist kompatibel zu den alten Formaten, d.h. alte Clips können weiterhin abgespielt werden.

Der Aufbau der RealPlayer-Anwendung orientiert sich weitgehend an bekannten Vorbildern, wie z.B. CD-Playern oder Kassettendecks. Die üblichen Kontrollen für Play, Pause, Stop usw. sind vorhanden. Besonderheit ist ein Balken, der den Fortschritt im Verlauf des Clips anzeigt. Über ihn kann jede beliebige Stelle innerhalb der Präsentation angesprungen und in der gleichen Zeit wie bei der Initialsierung und dem Start des Anzeigevorgangs abgespielt werden. Es ist dazu nicht nötig, die gesamten davorliegenden Daten auf den eigenen Rechner zu übertragen.

Kann eine auf einem Rechner installierte Version des RealPlayers den angeforderten Stream nicht abspielen, so wird der Anwender auf eine entsprechende Adresse bei RealNetworks umgeleitet und erhält eine Meldung über den fehlgeschlagenen Versuch. Optional besteht neuerdings die Möglichkeit, direkt fehlende Plug-Ins von der Real-Homepage herunterzuladen. Der Vorgang verläuft dabei für den Nutzer völlig transparent, Fragen wie die Art, Version oder Plattform des Plug-Ins werden automatisch vom Player geklärt und an die Download-Seite weitergeleitet.

Der Designer kann bei der Produktion seines Clips das Erscheinungsbild des Players weitgehend selbst festlegen. Er kann wählen, ob das Abspielfenster eigenständig oder als Bereich der Web-Seite dargestellt werden soll. Zusätzlich können die verfügbaren Funktionen und Buttons eingeschränkt werden. Aus Übersichtsgründen kann z.B. in manchen Fällen auf einen Slider für die Lautstärkeregelung verzichtet werden.

2.2 Encoder

RealNetworks bietet genau wie beim RealPlayer und RealServer eine kommerzielle und eine frei verfügbare Variante des RealEncoders an. Die freie Version ist vom Leistungsumfang allerdings voll ausreichend und bietet genügend Einstellungsmöglichkeiten, um die Präsentationen optimal aufzubereiten. Der Encoder ist für die gängigen Computertypen verfügbar.

Zur Erstellung von Streams sind dem Anwender umfangreiche Hilfsmittel gegeben, welche in Form von Wizards in die Oberfläche integriert sind. Sie fordern durch einfache Dialoge die benötigten Daten vom Benutzer an, ohne dass dieser über besondere Kenntnisse verfügen muss. Ein Beispiel für einen solchen Produktionszyklus ist in der Abbildung 5 gegeben.

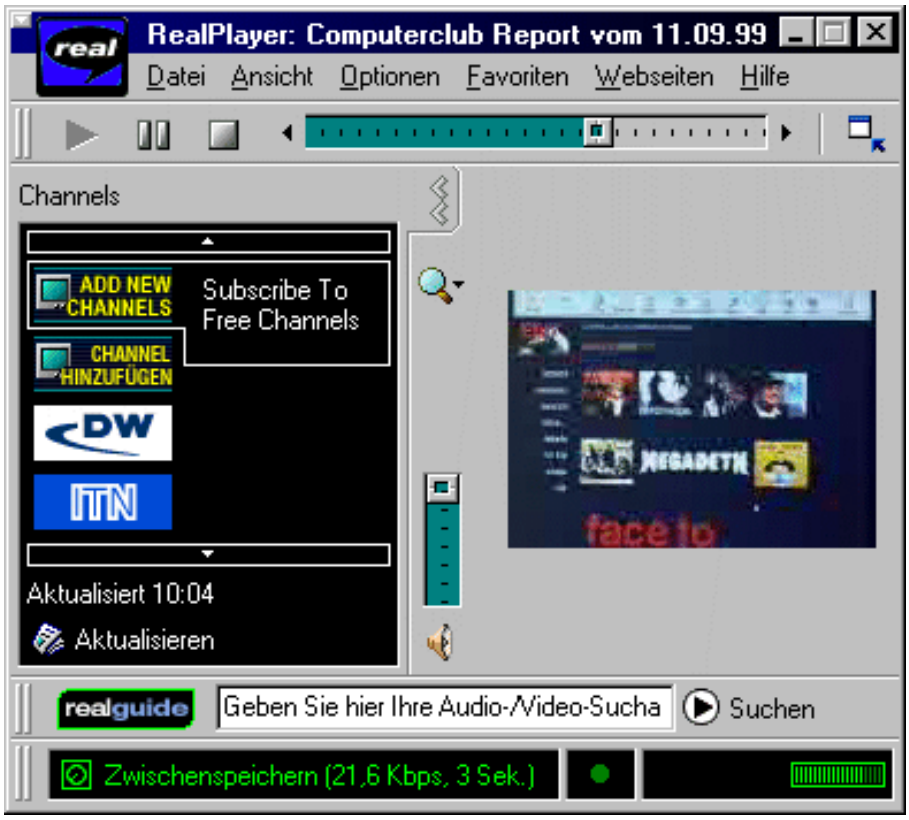


Abbildung 3: RealPlayer

Der Benutzer hat also zunächst einmal die Auswahl, was für eine Art von Stream erstellt werden soll. Die Kodierung kann zum einen als statischer Stream in Form einer festen Datei oder als dynamischer Stream, der zur Laufzeit in Zusammenspiel mit dem RealServer erzeugt und für Live-Übertragungen verwendet wird, erfolgen. Der RealEncoder bietet zudem noch Einstellungsmöglichkeiten zur Datensicherheit. so kann der Erzeuger von Streams von vornherein festlegen, ob seine Streams gespeichert oder während des Abspielens mitgeschnitten werden dürfen. Dazu muss er vor der Erstellung seines Streams entsprechende Preferenzen einstellen. Je nach Art der Daten, können unterschiedliche Codecs zur Komprimierung ausgewählt werden. Die Auswahl richtet sich vor allem nach den Gesichtspunkten Qualität und Zielbandbreite. Es muss ein Gleichgewicht zwischen diesen beiden Punkten gefunden werden, damit die Präsentation genau den Bedürfnissen der Konsumenten entspricht. So können bei der Erstellung eine oder mehrere Zielbandbreiten angegeben werden und der Encoder erzeugt dann entsprechende Streams. Für die frei verfügbare Version gilt hier die Einschränkung, dass solche SureStreams nur für zwei unterschiedliche Bandbreiten erzeugt werden können. Die vom Ersteller zu machenden Angaben sind in Abbildung 6 dargestellt.

Die Erzeugung des Streams kann während der Kodierung beobachtet werden. Zu diesem Zweck ist im RealEncoder (Abb. 7) links das Originalvideo und rechts das kodierte Video zu sehen. Dabei fallen je nach verwendetem Codec Qualitätsunterschiede auf. Nach Abschluss dieses Verfahrens ist es möglich Statistiken (Abb.: 8) über die erzeugeten Streams anzeigen zu lassen, mit Hilfe derer festgestellt werden kann, ob der gewählte Codec das gewünschte Ergebnis generiert hat.

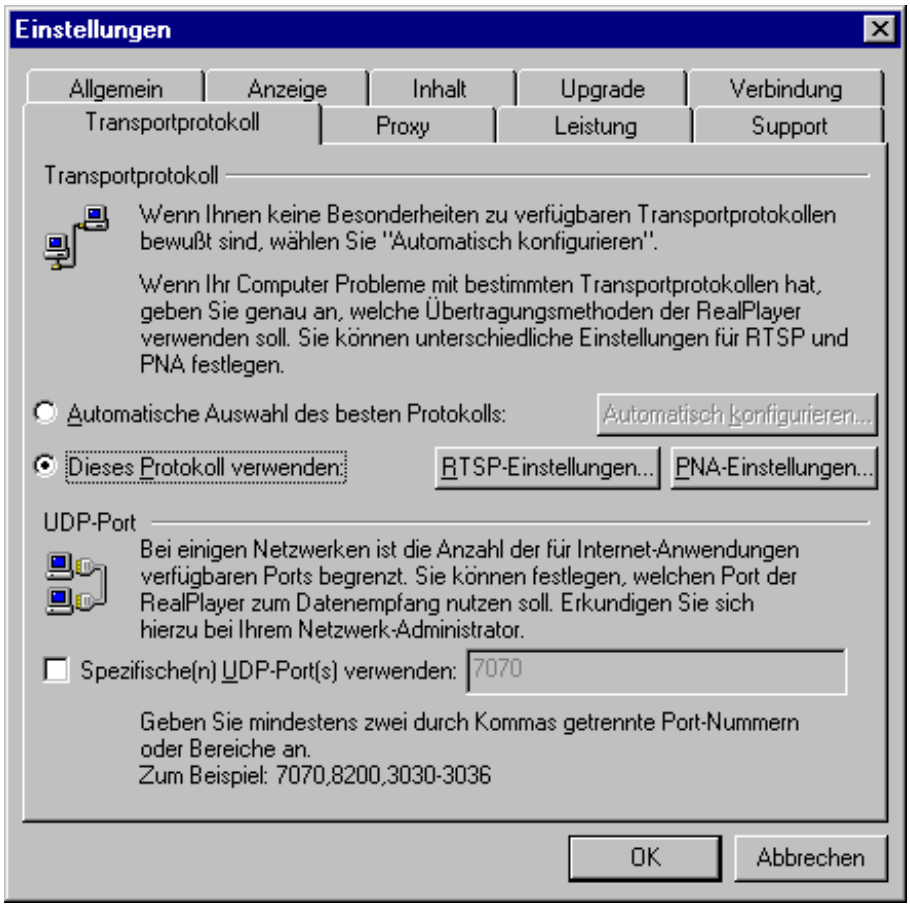


Abbildung 4: Einstellungen des RealPlayers

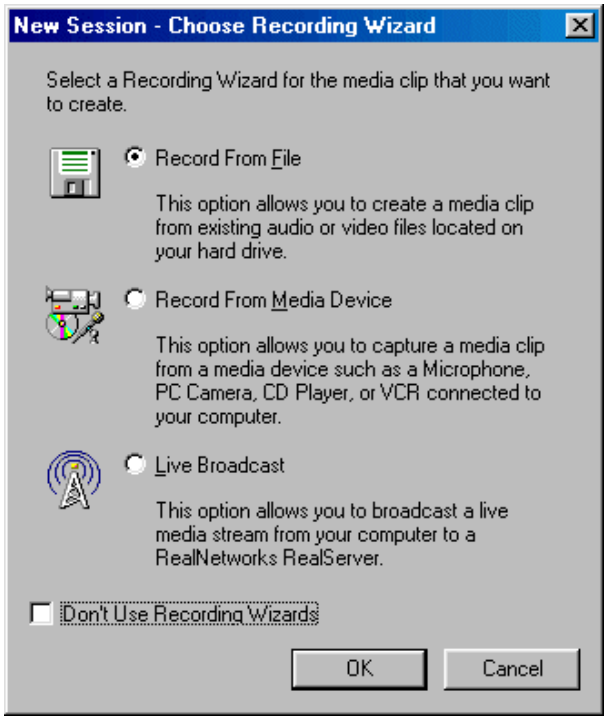


Abbildung 5: Einstellungen des RealEncoders

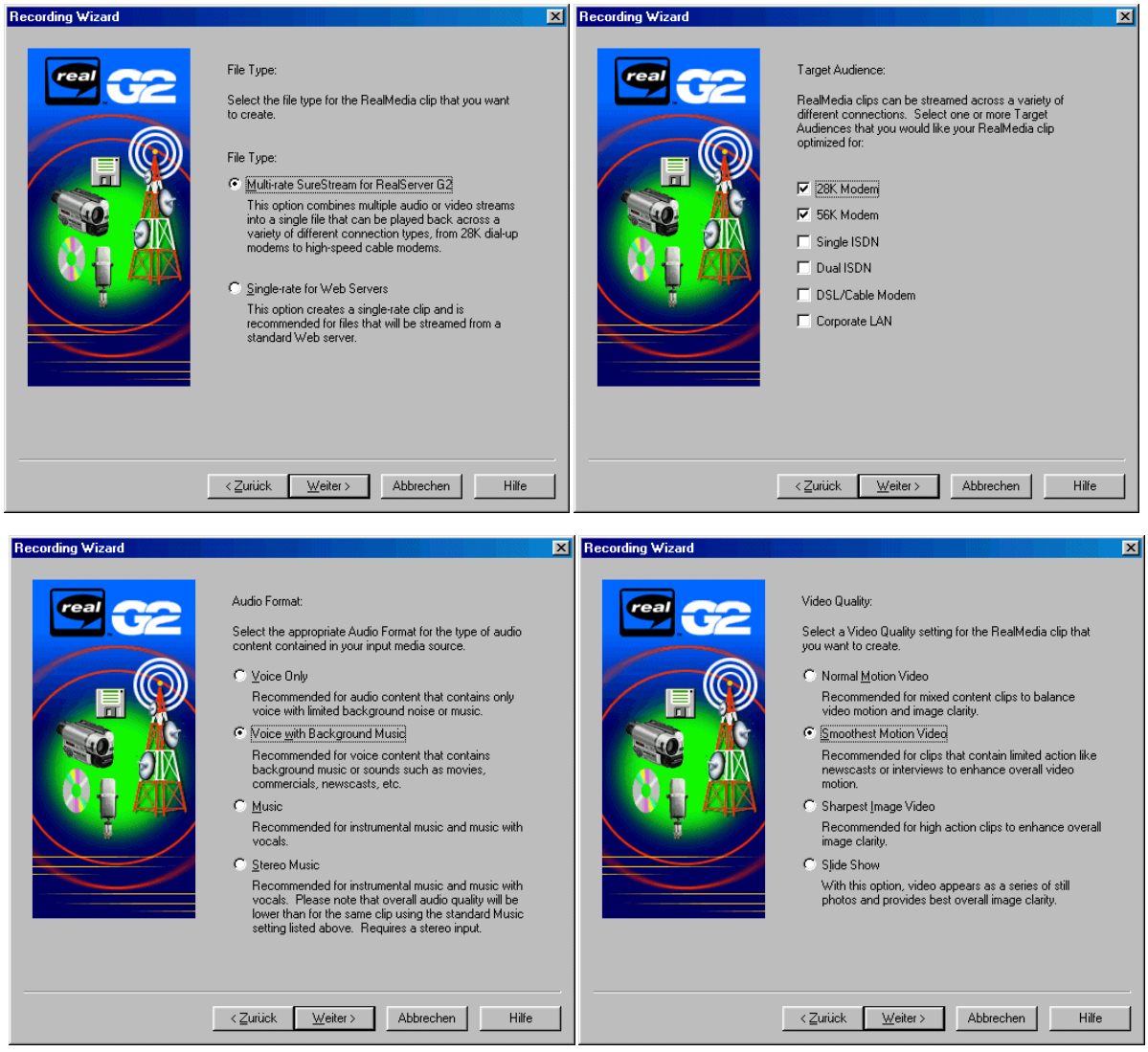


Abbildung 6: Produktionszyklus für einen Videostream

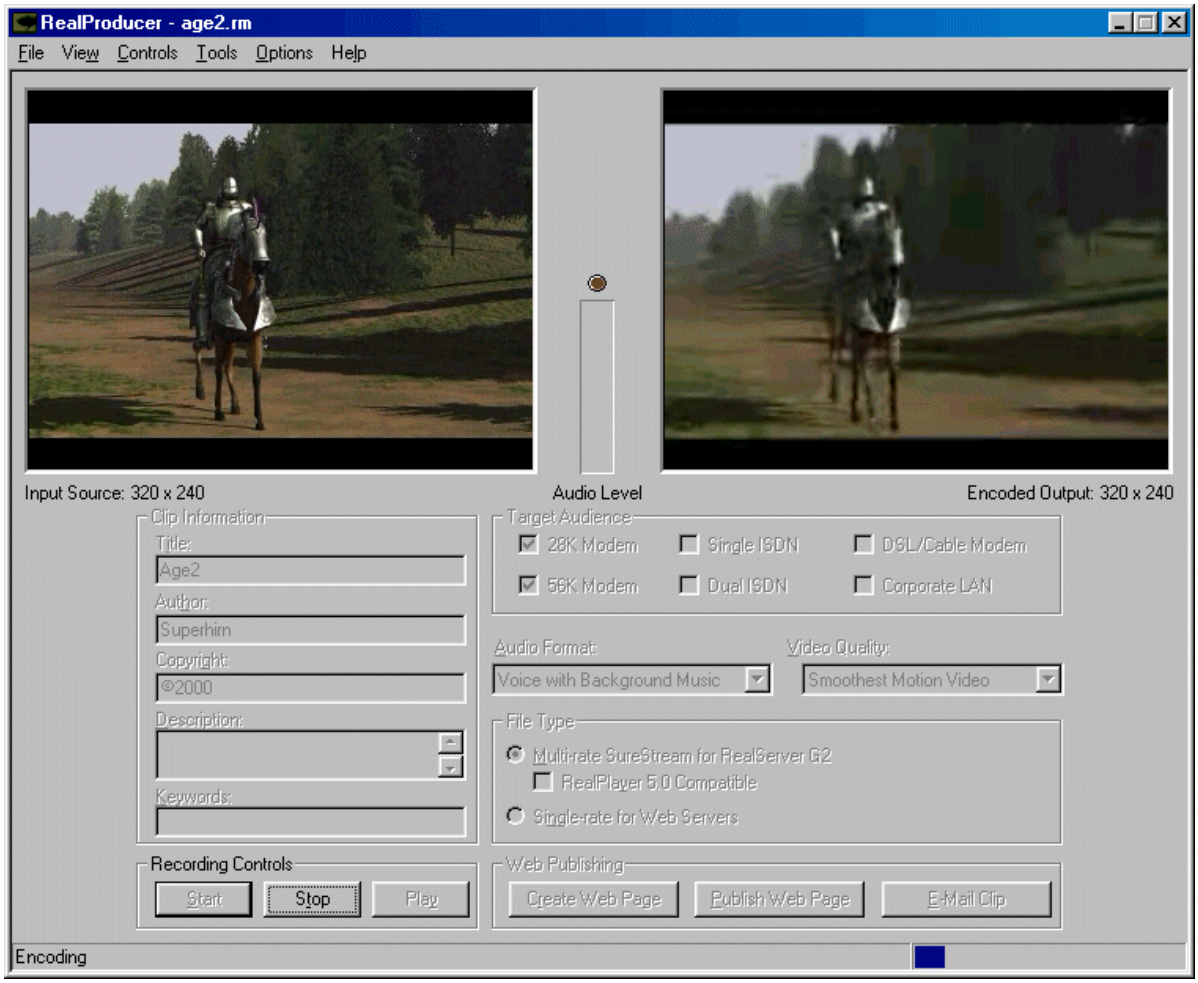


Abbildung 7: RealEncoder

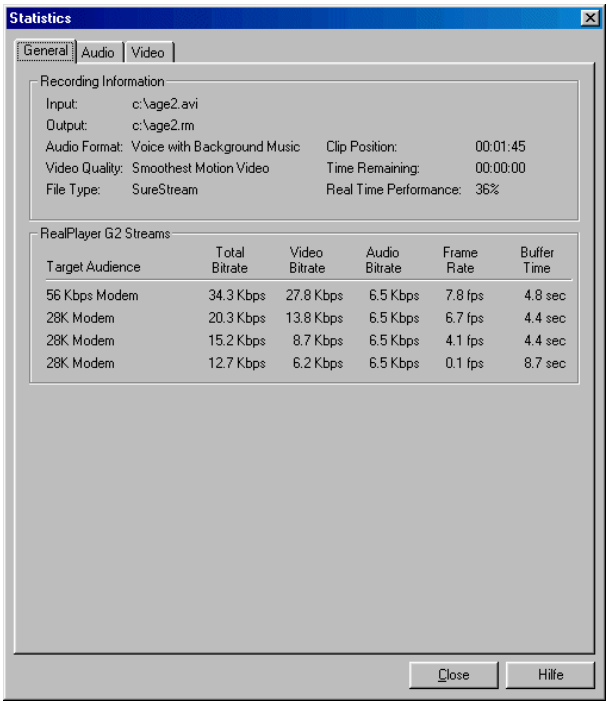


Abbildung 8: Statistiken zu den erzeugten Streams

2.3 Server

Der von RealNetworks angebotene G2 Server ist im Gegensatz zum RealPlayer für alle gängigen Plattformen verfügbar. Er existiert in einer unbeschränkten, kommerziellen und in einer frei verfügbaren Variante, die im Leistungsumfang, was Anzahl von verbundenen Playern und die Menge der angebotenen Streams betrifft, begrenzt ist.

Ist der Server erst einmal installiert, erfolgt die eigentliche Administration über eine passwortgeschützte Webseite, von der aus Informationen abgefragt und Einstellungen getroffen werden können. Diese Webseite ist in Abbildung 9 dargestellt.

Die interessanteste Funktion dieser Webseite ist der Monitor. Hier kann der Administrator bequem überwachen, wie stark das System ausgelastet ist und welche Streams zur Zeit abgerufen werden. Die Informationen können dann dazu benutzt werden, das Angebot und die Leistungsfähigkeit des Servers anzupassen. Anzumerken ist, dass die durch die Streams erzeugte Belastung der CPU sehr gering ist und erst bei hohen Zugriffszahlen ansteigt.

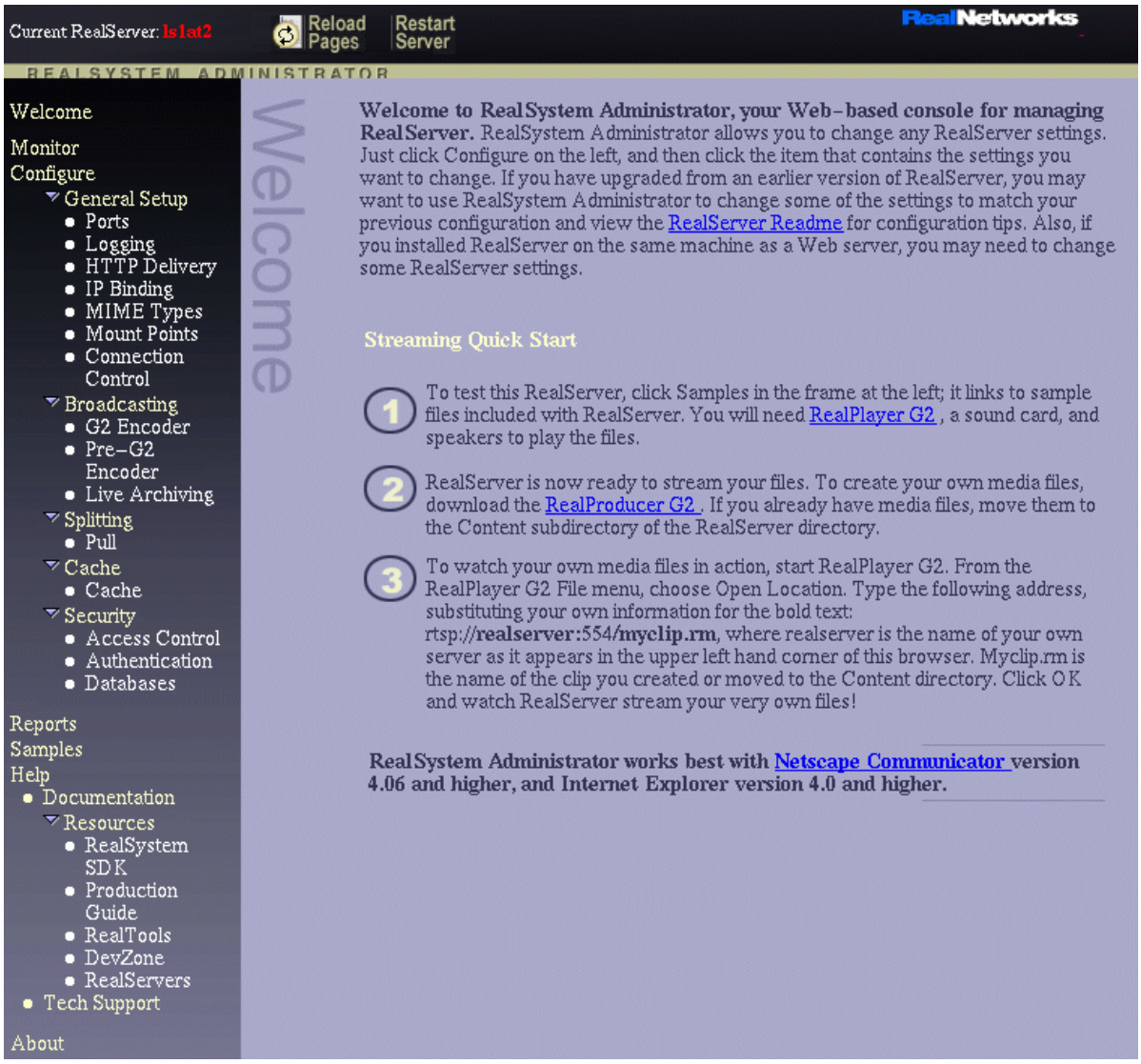


Abbildung 9: Administration des RealServers

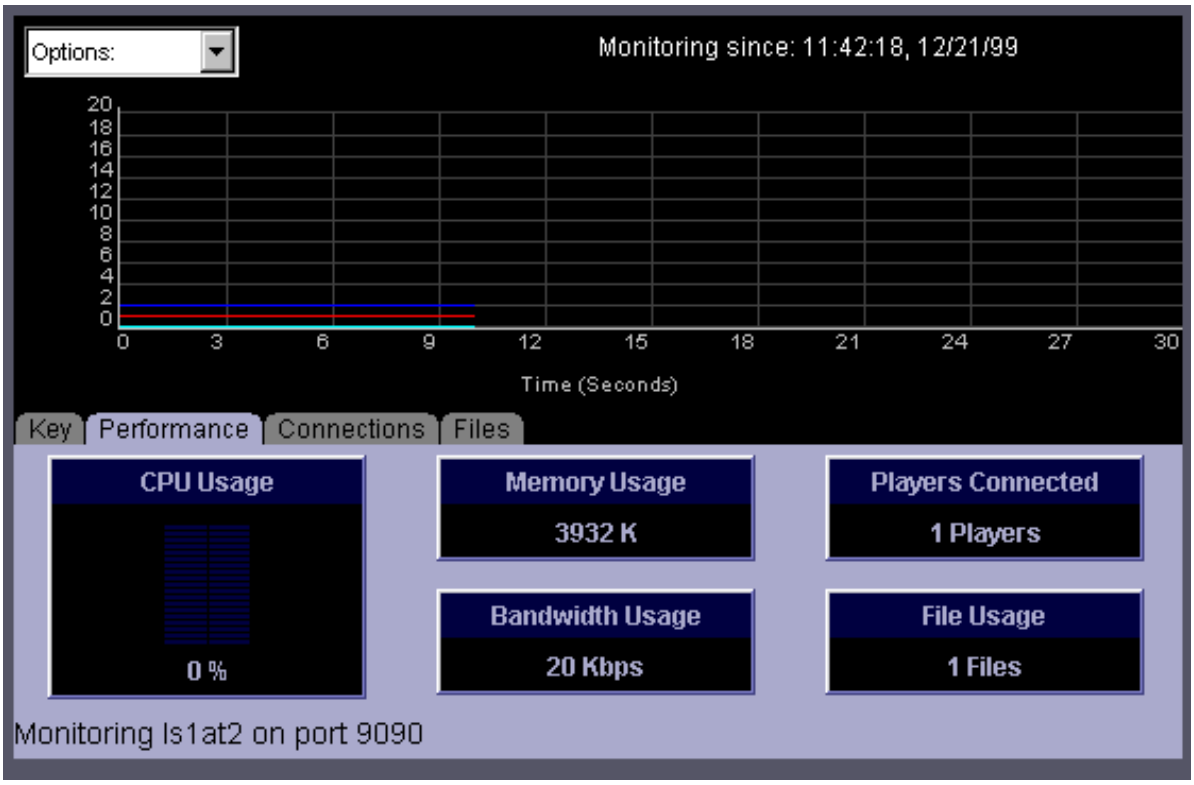


Abbildung 10: RealServer Monitor

3 Erstellung von RealSystem-Präsentationen

In der heutigen Zeit entwickelt sich ein immer größer werden Bedarf an komplexen, multimedial aufbereiteten Informationen, die über Computernetzwerke, vor allem das Internet, angeboten werden. Diese Daten sollen möglichst schnell und in ansprechender Form verfügbar gemacht werden, was durch das *Streamen* dieser Daten erreicht wird. Die Firma RealNetworks bietet zu diesem Zweck verschiedene Software-Werzeuge in Form von Server- und Clientprogrammen an. In diesem Abschnitt soll zunächst geklärt werden, welche Art von Daten gestreamt werden können und wo die Einsatzgebiete für Streaming-Technologie liegen.

Allgemein besteht eine RealSystem-Präsentation aus einem oder mehreren Multimedia-Clips, die sequentiell oder parallel ausgeführt werden. Eine genauere Betrachtung dieser Vorgehensweise erfolgt zu einem späteren Zeitpunkt innerhalb dieses Kapitels. Die zum Streamen ausgewählten Datenformate können in folgende Gruppen aufgeteilt werden:

- Audio: RealAudio, AIFF, AU, WAV
- Video: RealVideo, ASF, AVI, QuickTime, Vivo
- Animationen: RealFlash
- Bilder: JPEG, GIF
- Text

Anwendungsgebiete für die verschiedenen Formate liegen auf der Hand. So ist es heute möglich, diverse Radiosendungen über das Internet mit Hilfe von Audiostreams zu empfangen sowie eine Reihe von „archivierten“ Fernsehsendungen direkt von bereitgestellten Servern abzurufen. Allerdings ist für derart komplexe und umfangreiche Datenmengen die Bandbreitenbeschränkung

heutiger (Tele-)Kommunikationsnetze noch so gravierend, dass eine ansprechende Darstellung von Videos noch nicht möglich ist. Im Gegensatz dazu ist der Einsatz von Text-Streams zur Realisierung von Nachrichten- oder Börsentickern, auf Grund der wenig komplexen Daten relativ einfach.

3.1 Grundsätzliche Vorgehensweise bei der Erstellung eines Streams

Die Erstellung eines Streams erfolgt stets in folgenden Schritten:

- Auswahl der Daten
- Daten in geeignetes Streamingformat konvertieren
- Bereitstellung durch einen Server

Bei der Auswahl der Daten ist zunächst einmal zu beachten, dass die Daten, um schnell übertragen werden zu können, stark komprimiert werden müssen. Da diese Kompression im allgemeinen verlustbehaftet ist und die Daten zum Teil sogar auf ein Zwanzigstel ihrer Ursprungsgröße reduziert werden, muss das Ausgangsmaterial in einer möglichst hohen Qualität vorliegen, denn sonst würden sich negative Eigenschaften wie Rauschen und Unschärfe weiter verstärken.

Nach der Auswahl der Daten folgt die Konvertierung in ein Streamingformat mit Hilfe des RealEncoders. Es ist zwar prinzipiell auch möglich Standardformate zu streamen, was allerdings nur in Ausnahmefällen geschehen sollte, da Streamingformate wegen ihrer besonderen Eigenschaften eine höhere Performance der Präsentation versprechen.

Zum Schluß müssen die Streams noch auf einem Server bereitgestellt werden, was im Idealfall ein RealServer sein sollte. Es ist zwar auch möglich Streams über einen herkömmlichen Webserver anzubieten, was allerdings wegen starker Performanceeinbußen nicht zu empfehlen ist. Näheres hierzu findet sich in Kapitel 6 dieser Ausarbeitung.

4 Selektion von Bandbreiten

Eines der großen Probleme bei der heutigen Form der Datenübertragung ist die unterschiedliche Verfügbarkeit von Bandbreiten. Oft ist die dem Anwender zur Verfügung stehende Bandbreite nicht bekannt oder es können während des Streaming-Vorgangs konstante Datenübertragungsraten nicht garantiert werden. In der möglichen Qualität bestehen gravierende Unterschiede bei einer Rate von 28,8 Kbps bei Benutzung eines älteren Modems und den Ressourcen eines lokalen Netzwerkes. Die Betrachtung der unterstützten Bandbreiten spielt somit eine wesentliche Rolle bereits im Vorfeld des Produktionsprozesses. Der einzige Fall, in dem eine Berücksichtigung maximaler Übertragungsraten nicht nötig ist, besteht im lokalen Abspielen von Datenströmen.

Soll eine möglichst breite Masse von Nutzern erreicht werden, wird als Zielbandbreite 28,8 Kbps empfohlen. Der RealPlayer ermöglicht zwar das Abspielen eines Videos, das für eine Übertragungsrate von 56 Kbps codiert wurde, allerdings wird das Video beim Empfänger solange gepuffert, bis durchgehendes Anzeigen möglich ist. Sonst müßten für jede Sekunde Video zuvor zwei Sekunden lang Daten übertragen werden. Die Dauer für das Puffern der nötigen Daten ist abhängig von der Länge der Präsentation. Bei langen Videos kann dieser Vorgang mehrere Minuten in Anspruch nehmen. Die meisten Anwender werden nicht bereit sein, so lange zu warten. Damit verliert das Data-Streaming seinen größten Vorteil, die sofortige Verfügbarkeit der (ersten) Daten.

4.1 Preroll

Mit *Preroll* werden die initialen Daten bezeichnet, die vom RealServer an den Player gesendet werden, bevor die Wiedergabe beginnt. Nach einer Anforderung stellt der Server als erstes die

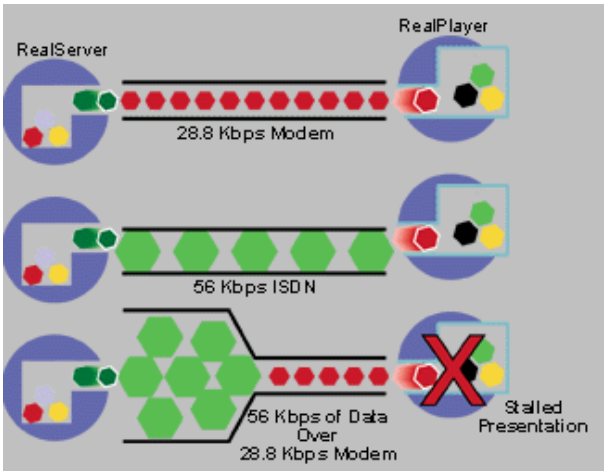


Abbildung 11: Flaschenhals

Größe und die Timeline der Präsentation fest. Anhand der verfügbaren Übertragungsrate gibt der Server dem Player dann vor, wieviele Daten vor Beginn des Abspielvorgangs übertragen werden müssen. Dies soll eine unterbrechungsfreie Präsentation auch bei schlechten Übertragungswegen ermöglichen.

Zeiten für Preroll unter 15 Sekunden gelten als tolerierbar, unter 10 Sekunden ist das Idealmaß für gute Präsentationen.

4.2 Auswahl der Zielbandbreite

Die Zielbandbreite, für die eine Präsentation vorbereitet werden soll, ist die kleinste zur Verfügung stehende Übertragungsrate. Die Gesamtrate des Streams muss unterhalb dieser Schranke liegen. Die verfügbare Bandbreite läßt sich in zwei Bereiche gliedern:

- Maximale Bitrate, die von der Summe aller Einzelströme konsumiert wird. Besteht die Präsentation aus mehreren Strömen, kann die benötigte Rate über den Zeitverlauf variieren. Hier ist der Spitzenwert zu wählen, um zu jedem Zeitpunkt einen flüssigen Verlauf zu garantieren.

- 25% der Zielbandbreite sollten als Toleranz zum Abfangen von Overhead (Störungen, Datenverlust, Packet Overhead) reserviert werden. Das Verhältnis ist ebenfalls abhängig von den allgemeinen Netzwerkvoraussetzungen. Verbindung über Modems sind in der Regel wesentlich schlechter als theoretisch gleichschnelle ISDN-Verbindungen. Tabelle 12 gibt einen beispielhaften Überblick über empfohlene Verhältnisse.

14.4 Kbps Modem	10 Kbps
28.8 Kbps Modem	20 Kbps
56.0 Kbps Modem	32 Kbps
56.0 Kbps ISDN	45 Kbps
112 Kbps ISDN Kanalbündel	80 Kbps

Abbildung 12: Für das Streaming verfügbare Bitraten

4.3 Verhältnisse unterschiedlicher Streams

Gerade bei Multimedia-Strömen kommt der Aufteilung der zur Verfügung stehenden Bandbreite ein besonderes Gewicht zu. Je nach Anforderung bieten sich unterschiedliche Strategien an.

Ein Audio-Stream mit Musik besitzt im Vergleich zu einem Interview einen relativ großen Dynamikbereich. Dieser benötigt deshalb eine größere Bitrate als einfache Sprache um einen akzeptablen Eindruck zu erreichen.

Analog besitzen Videos mit schnellen Bewegungen oder hoher Auflösung einen erhöhten Bedarf. Kann dieser nicht gedeckt werden, wird das Video nicht flüssig oder unscharf abgespielt. Der gewünschte Eindruck ergibt hier die Richtlinie für die Aufteilung der Bandbreite. In den meisten Fällen wird dem Ton Vorrang gegeben und eine schlechtere oder ruckelnde Video-Qualität in Kauf genommen. Die meisten Menschen empfinden schlechte Audio-Qualität als erheblich störender im Vergleich zu langsamen Bildfolgen.

Problematisch sind Streamingformate ohne konstante Übertragungsrate, wie z.B. RealFlash oder RealPix. Orientiert man sich an den Spitzenwerten, wird die übrige Zeit ein großer Teil an verfügbarer Bandbreite verschwendet. Unterschiedliche Strategien bei der Kombination von Multiclip Präsentationen zeigen die Abbildungen 13 und 14. Die zweite Abbildung stellt eine wesentlich bessere Zusammenstellung der Timeline der Streams dar. Wartezeiten zwischen Clips werden ebenfalls reduziert, wenn der Ersteller darauf achtet, nur eine aktive Preroll-Phase zu jedem Zeitpunkt auszuführen. Die Startup-Zeit von großen Clips kann reduziert werden, wenn die Präsentation mit einem low-level Clip beginnt und dem RealPlayer damit die Möglichkeit gibt, den Preroll im Hintergrund durchzuführen.

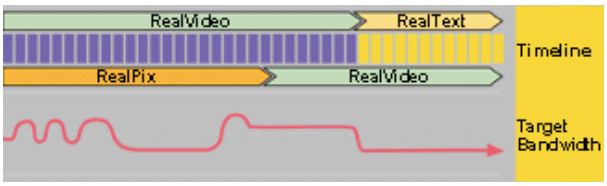


Abbildung 13: Schlechte Zusammenstellung der Bandbreiten

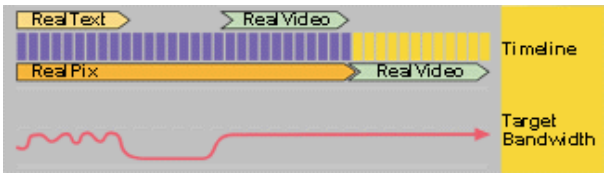


Abbildung 14: Gute Zusammenstellung der Bandbreiten

Um die nötigen Kompromisse bei der Erstellung der Präsentationen möglichst klein zu halten, bietet RealServer die Option, Clips für Verbindungen mit unterschiedlichen Bandbreiten bereit zu halten. Diese *SureStream* genannte Technologie stellt die einzelnen Streams unter dem gleichen Internet-Link zur Verfügung. Sobald ein Nutzer auf einer Web-Page diesen Link aktiviert, versuchen RealPlayer und RealServer die Bandbreite der Verbindung festzustellen und der Server wählt dementsprechend die passenden Clips aus. Die Auswahl wird dynamisch während der gesamten Spieldauer überwacht, so dass es problemlos möglich ist, bei einem Einbruch der Rate auf einen Stream niederer Qualität und bei einer Besserung wieder auf die alte Version umzuschalten.

5 Streamingformate

Vor der Verbreitung von Präsentationen über das Internet müssen die Clips zunächst in spezielle Formate übertragen werden. In der aktuellen Version des RealPlayer G2 ist es zwar möglich, Standardformate wie z.B. AVI direkt als Stream zu übertragen, diese Formate sind allerdings nicht auf die besonderen Bedingungen ausgerichtet, d.h. der Verlust oder die Verzögerung von Paketen lassen sich nur schwer oder gar nicht verbergen. Die von Real speziell entwickelten Formate sollen gerade in diesem Bereich deutliche Vorteile besitzen.

Die nachfolgenden Kapitel gehen sowohl auf die älteren Formate für Audio und Video als auch auf die neu hinzugekommenen zur Übertragung von Bildern und Text ein. Gezeigt werden sollen die Besonderheiten und Anwendungsmöglichkeiten der unterschiedlichen Codecs.

5.1 RealAudio

Die Firma RealNetworks stellte 1995 erstmalig eine Streamingformat vor. Es handelte sich dabei um RealAudio. Zu diesem Zeitpunkt war es das einzige streambare Format, welches für Computer-Netzwerke zur Verfügung stand. Es entwickelte sich somit zu einem Standard-Format, das konsequent weiterentwickelt wird.

Wie bei allen Streamingformaten ist auch die Qualität von RealAudio-Präsentationen stark von der Bandbreite der Übertragung und somit auch von der zu übertragenden Datenmenge abhängig. Um diese möglichst gering zu halten, werden unterschiedliche Codecs eingesetzt, die auf den jeweiligen Verwendungszweck abgestimmt sind. So müssen prinzipiell zwei grundlegende Eigenschaften abgewägt werden und zwar die Art der Daten und die Zielbitrate. Bei den Daten hängt es stark davon ab, ob Sprache, Musik oder eine Mischung von beiden übertragen werden soll. Hiefür existieren an unterschiedliche Bandbreiten angepasste Codecs, die für diese Anwendungsfälle optimale Ergebnisse liefern.

Diese Optimierung ist gerade deswegen notwendig, da die Kompression der Audiodaten verlustbehaftet ist. Das heißt, es werden je nach Codec unterschiedliche Charakteristiken der Daten hervorgehoben bzw. entfernt. Nur so kann eine Kompressionsrate von bis zu 1:20 bei beispielsweise WAV-Dateien errreicht werden. Die Unterscheidung zwischen Sprache und Musik ist deshalb wichtig, da Musikstücke einen viel breiteren Frequenzbereich benötigen, als menschliche Sprache, die sich nur in einem geringen Band bewegt. Der Frequenzbereich kann so unterschiedlich stark reduziert werden, ohne dass der durch die Kompression verursachte Qualitätsreduktion zu starkem Informationsverlust führt.

Die Tabelle 15 zeigt die Auswahl an Codecs für Audiodateien, die über eine Verbindung mit hoher Übertragungskapazität verbreitet werden sollen. Für niedrige und mittlere Bandbreiten existieren ähnliche Codecs, auf deren Aufführung hier allerdings verzichtet werden soll. In diesem Zusammenhang sei auf das Kapitel 4 in [RPG] verwiesen.

5.2 RealVideo

Zweites von RealNetworks eingeführtes Format für das Streaming von Präsentationen ist RealVideo. Neben RealAudio besitzt es derzeit die größte Bedeutung. Dieses Format ist speziell für Streaming über das Internet von Bedeutung und bietet die entsprechenden Möglichkeiten, auf schlechte Netzwerkbedingungen oder Paketverlust zu reagieren. Eingeführt wurde das Format mit der Version 4.0 des RealPlayer. Grundsätzlich ist es inzwischen in der aktuellen Version G2 des RealSystem zwar ebenfalls möglich, andere Video-Formate (AVI, Quicktime...) vom RealServer übertragen zu lassen, diese sind allerdings eher für den lokalen Einsatz designed worden. Deshalb soll hier nicht weiter auf diese Formate eingegangen werden.

RealVideo ist ein Datenformat für eine konstante Bitrate, unabhängig von den gezeigten Inhalten. Deshalb muss bereits im Vorfeld die zu benutzende Bitrate gewählt werden. Dieser Vorgang wird ausführlich im Kapitel 4, „Selektion von Bandbreiten“, beschrieben. RealVideo

Codec	Rate	Comment
32 Kbps Voice G2 mono	22.05 kHz	Use with SureStream clips.
32 Kbps Music G2 mono	44.1 kHz	Use with SureStream clips.
32 Kbps Music G2 stereo	45 kHz	Use with SureStream clips.
32 Kbps Music mono	16 kHz	DolbyNet codec.
32 Kbps Music stereo	11.025 kHz	DolbyNet codec.
40 Kbps Music mono	22.05 kHz	DolbyNet codec.
40 Kbps Music stereo	16 kHz	DolbyNet codec.
44 Kbps Music G2 mono	44.1 kHz	Use with SureStream clips.
44 Kbps Voice G2 stereo	44.1 kHz	Use with SureStream clips.
64 Kbps Voice G2 mono	44.1 kHz	Use with SureStream clips.
64 Kbps Music G2 mono	44.1 kHz	Use with SureStream clips.
64 Kbps Music G2 stereo	44.1 kHz	Use with SureStream clips.
80 Kbps Music mono	44.1 kHz	DolbyNet codec.
80 Kbps Music stereo	32 kHz	DolbyNet codec.
96 Kbps Music G2 stereo	44.1 kHz	Use with SureStream clips.

Abbildung 15: RealAudio-Codecs für hohe Bitraten

baut ähnlich wie die MPEG-Standards auf einer verlustbehafteten Kompression auf. Je stärker der gewählte Grad der Kompression, um so höher der Verlust an Bildqualität. Ebenso wirkt sich der Verlauf des Original-Videos auf die Kompression aus. Da der zugrundliegende Algorithmus die Veränderungen von einem Bild zum nächsten stark in seine Berechnungen mit einbezieht, wirken sich schnelle Szenenwechel negativ auf die Bildqualität aus. Grundsätzlich ließe sich zwar der erhöhte Bedarf an Leistung auf eine größere Bandbreite umlegen, dies widerspräche aber dem Konzept einer konstanten Bitrate. Zusätzlich sind die Kompressions-Algorithmen auf 24 Bit Echtfarbbilder ausgelegt und liefern nur dafür optimale Ergebnisse.

Mit RealVideo besteht die Möglichkeit, Videos mit einer Bildrate von bis zu 30 fps[1] zu codieren. Damit können alle bekannten Video-Formate, wie z.B. Fernsehen mit 25 fps, abgedeckt werden. Empfohlen wird dagegen eine Rate von 15 fps, damit auch ältere Maschinen problemlos die Decodierung bewältigen können. Die Bildrate des codierten Real-Streams muss dagegen nicht die vollen 15 fps ausnutzen, sondern richtet sich nach der zur Verfügung stehenden Bitrate und dem Gewicht auf weiche Bewegungen oder scharfe Bilder. Die effektive Rate kann damit zwischen 1 und 15 fps liegen. 15 fps ist dabei der Bereich, bei dem Einzelbildfolgen vom Auge noch als kontinuierlicher Verlauf wahrgenommen werden. Bei niedrigeren Raten wirkt das Video schnell unruhig.

RealVideo besitzt keine Einschränkungen hinsichtlich der Größe und des X/Y-Verhältnisses. Lediglich die benutzte Bandbreite beschränkt die Bildgröße. Tabelle 16 zeigt einige häufig verwendete Größen, die mit ihrm 4:3-Verhältnis dem alten TV-Standard entsprechen. Dargestellt ist die benötigte Bandbreite und die damit erreichbare Bildqualität.

Breite x Höhe	Bandbreite	Bildqualität
176x132	20–500 Kbps	Gut bis sehr gut
240x180	100–500 Kbps	Sehr gut
320x240	200–500 Kbps	Sehr gut

Abbildung 16: Kriterien für die Bildqualität von Video

RealVideo bietet für die Kompression unterschiedliche Codecs an. Tabelle 17 zeigt die un-

[1] Frames per second

terstützten Versionen.

Codec	G2	V 5	V 4	Kommentar
G2 Standard (Vorgabe)	J	N	N	Schnellste Codierung/Decodierung. Ermöglicht unterschiedliche Bitraten in einem Stream (SureStream)
Standard(alte Vorgabe)	J	J	J	Frühere Version. Einfache Bitrate
Fraktal	J	J	J	In der Version G2 nur noch Abspielen, keine Codierung mehr

Abbildung 17: RealVideo Codecs

Der aktuelle Encoding-Standard schneidet die Ausmaße des Bildes auf Vielfache von 4 Pixeln, bei dem älteren Verfahren auf Vielfache von 16. So würde ein Videobild mit den Dimensionen 176x132 in der älteren Version auf 176x128 Pixel abgeschnitten.

5.3 RealFlash

RealFlash ist das Streamingformat zur Darstellung von Animationen. Es vereint das Shockwave Flash Vektorformat von Macromedia mit dem RealAudio Format.

RealFlash orientiert sich am Stil klassischer Zeichentrickfilmen. Der Produktionsprozess ist daher entsprechend anders. Für die Formate zur Verbreitung von Audio und Video müssen vor dem Encodingvorgang die Daten in der realen Welt vorbereitet werden, die Produktion von Flash-Animationen ist dagegen eher im rein künstlerischen Bereich anzusiedeln.

Da eine RealFlash-Präsentation in den meisten Fällen aus der eigentlichen Animation im Shockwave-Format und einem begleitenden Audiotrack im RealAudio-Format besteht, muss dementsprechend die zur Verfügung stehende Bandbreite auf beide Streams aufgeteilt werden. Im Gegensatz zu den übrigen Formaten gelten hierbei allerdings andere Bedingungen. RealFlash ist ein reines Vektorformat. Im Verlauf der Präsentation werden nur an wenigen Stellen komplette Bildinformationen übertragen, auf diesen bauen Anweisungen zur Transformation der übertragenen Daten auf. Vorteil dieser Art des Streamings ist, dass sich auch für geringe Bitraten die Bildqualität nicht verschlechtert. Hauptkriterium für die Qualität der Darstellung ist nicht die Bandbreite, sondern die Rechenleistung des empfangenden Rechners. „Intelligentes“ Management der Quelldaten ermöglicht gleichwertige Präsentationen bezogen auf Clips mit einem höheren Bedarf an Bandbreite.

RealFlash kann auch von der Version 5.0 des RealPlayers abgespielt werden, dabei sind allerdings einige Punkte zu beachten:

- RealPlayer 5.0 unterstützt lediglich Macromedia Flash in der Version 2.0, RealPlayer G2 auch die Version 3.0 ohne Transparenzeffekte.
- Der zugehörige Audio-Soundtrack muss mit dem speziellen Kompatibilitäts-Flag codiert worden sein.

Tabelle 18 gibt eine kurze Übersicht über Empfehlungen, wie die Bandbreite für eine 20Kbps Präsentation zwischen Animation und Audio aufgeteilt werden soll. Zu beachten sind die Unterschiede, je nach dem ob Kompatibilität zur Versiom 5.0 des Players gewünscht ist. Die neuen Codecs zur Generierung von Audio haben sich in der Beziehung verbessert, so dass bei gleicher Qualität weniger Bandbreite verwendet wird, bzw. die Codecs für Musik und nicht für Sprache verwendet werden können.

Da die Bandbreite eines Flash-Clips über den Zeitverlauf dynamisch ist, sind die Werte aus Tabelle 18 nur als Durchschnittswerte anzusehen. Um eine störungsfreie Präsentation zu

Präsentationstyp	RealAudio	Shockwave Flash
Schwerpunkt auf Animation mit gutem Ton (Sprache) für Version G2 und 5.0	5 Kbps Voice	15 Kbps
Schwerpunkt auf Animation mit bestem Ton (Sprache) für G2 und 5.0	8,5 Kbps Voice	11,5 Kbps
Schwerpunkt auf Animation mit gutem Ton (Musik) für G2 und 5.0	8 Kbps Music	12 Kbps
Schwerpunkt auf Animation mit gutem Ton (Musik) für G2	6 Kbps Music	14 Kbps
Schwerpunkt auf Animation mit bestem Ton (Musik) für G2	11 Kbps Music	9 Kbps

Abbildung 18: Aufteilung der Bandbreite zwischen Flash und Audio

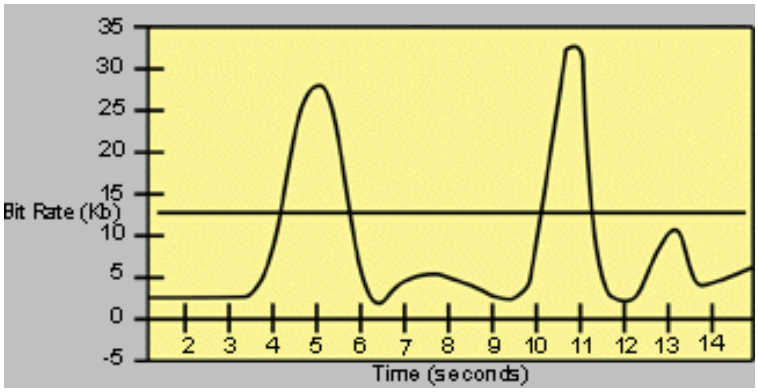


Abbildung 19: Beispielrate für einen Flash-Clip

garantieren, muss der Stream nach der Erstellung noch weiter untersucht werden. Insbesondere die Stellen, an denen die maximale Übertragungsrate überschritten wird, sind kritische Punkte. Beim Streaming des Files müssen diese Stellen bereits vor dem Zeitpunkt ihrer Ausführung komplett übertragen worden sein. Dazu muss von der zur Verfügung stehenden Bandbreite vor den Spitzenwerten noch genug verfügbar. Verglichen mit dem Beispiel in Abbildung 19 hieße das, dass die Fläche unterhalb der Bandbreite vor den Spitzen größer sein muss als der sich über der Bandbreitenlinie befindende Teil.

Um die Bitrate eines Flash-Clips klein zu halten, bzw. zu reduzieren, kann an folgenden Punkten angesetzt werden:

1. Das Verhältnis von Dateigröße zu Spielzeit sollte möglichst klein gehalten werden. Dies verhindert zwar keine Spitzen, hilft aber diese klein zu halten.

2. Kleine Anzahl an Key Frames. Key Frames benötigen immer ein Übertragung aller Objekte, die Frames dazwischen bauen auf diesen dann auf.

3. Shockwave Flash ermöglicht die Wiederverwendung von Objekten über sogenannte Symbole.

4. Die Elemente sollten so einfach wie möglich gehalten werden. Macromedia bietet Funktionen zum Entfernen überflüssiger Vektorkoordinaten.

Neben der Bitrate spielt aber auch die Prozessorbelastung eine entscheidende Rolle. Hier sollte man sich an dem aktuellen Stand der Rechenleistung orientieren und Durchschnittswerte für die Rechenleistung des typischen Anwendersystems kennen. Möglichkeiten zur Reduzierung der benötigten Leistung lassen sich folgendermaßen angeben:

1. Niedrige Framerate (7 fps liefert noch ansprechende Ergebnisse).

2. „Tweening“ optmieren. Tweening bedeutet Interpolation der Animation zwischen den Frames. Je höher das Tweening, umso weichere Animation, aber höhere Rechenleistung.

3. Die Zahl der sich bewegenden Objekte klein halten.

Eine Besonderheit, die RealFlash von den übrigen Formaten unterscheidet, ist die Interaktivität. Spezielle Shockwave-Kommandos können an Teile der Animation angebunden werden. Diese Kommandos werden dann auf die Funktionen des RealPlayers übertragen. Zur Verfügung stehen z.B. *Play*, *Stop*, *Goto* und *Goto and Play*.

5.4 RealText

Ein weiteres von RealNetworks eingeführtes Streamingformat ist RealText. Es kann wie jedes andere Streamingformat als Live-Stream oder als Datei angeboten werden.

Um eine RealText Präsentation auf einem Server bereitstellen zu können, ist zu beachten, dass dazu mindestens ein RealServer G2 benötigt wird, denn nur dieser ist in der Lage RealText mit allen Features zu übertragen. Es ist zwar prinzipiell auch möglich RealText von einem Webserver aus anzubieten, allerdings können in diesem Fall nur statische Streams, d.h. keine Live-Streams angeboten werden.

Eine RealText-Datei kann mit jedem beliebigen Texteditor erstellt werden, der die Daten als reine Textdatei (plaintext) abspeichern kann. Das Ziel einer RealText Präsentation ist es, Text in einer ansprechend aufbereiteten Form anzubieten. Zu diesem Zweck wurde von der Firma RealNetworks die RealText Markup Language entwickelt, die es in einem HTML-ähnlichen Stil ermöglicht, Text für Präsentationen aufzubereiten, was deutlich über das in HTML mögliche Formatieren hinaus geht. Durch die nahe Verwandtschaft zu HTML ist es dem geübten HTML-Benutzer leicht möglich in RealText einzusteigen und ansprechende Präsentationen zu erstellen.

Eine Präsentation mit RealText hat nur sehr geringe Anforderungen an die Bandbreite, die größtenteils unterhalb von 1 Kbps liegen. Aufgrund dieses Vorteils ist RealText als ideale Ergänzung zu aufwendigen Präsentationen mit anderen Streamingformaten, z.B. RealVideo, zu sehen. Ein solches Zusammenspiel verschiedener Streamingformate wird in Kapitel 5.6 genauer vorgestellt.

Wie bereits erwähnt, arbeitet RealText mit HTML-ähnlichen Tags, die hier nicht im einzelnen vorgestellt werden, da dies den Rahmen eines Kurzüberblicks sprengen würde. Für die Präsentation stehen folgende Fensterformate zur Verfügung:

- Generic: Ohne vordefinierte Eigenschaften
- Scrolling News: Text scrollt von unten nach oben
- Ticker Tape: Text läuft von rechts nach links am oberen, bzw. unteren Rand durch das Fenster
- Marquee: ähnlich wie Ticker Tape mit dem Unterschied, dass der Text vertikal zentriert im Fenster abläuft
- Teleprompter: ähnlich zu Scrolling News, mit dem Unterschied, dass der Text zeilenweise nach oben „springt"

Die üblichen Textformatierungen wie Farbe, Font, Schriftgröße etc. sowie die Layoutfunktionen innerhalb des Präsentationsfensters unterscheiden sich nicht von denen in HTML und werden deshalb hier nicht genauer erklärt. Eine erwähnenswerte Besonderheit in RealText ist die Möglichkeit Start- und Endzeiten sowie die Anzeigedauer von Texten zu bestimmen. Diese Zeitangaben orientieren sich an der Präsentations-Timeline, die genau im Abschnitt 5.6 vorgestellt wird. Desweiteren ist bei der Erstellung darauf zu achten, dass der Text standardmäßig im ASCII-Format übertragen wird. Sollen also landesspezifische Sonderzeichen übertragen werden, muss explizit der gewünschte Zeichensatz angegeben werden. Eine detaillierte Liste von unterstützten Zeichensätzen ist in [RPG]zu finden.

Mit RealText ist es auch möglich, eine interaktive Präsentation zu erstellen. Dies geschieht mit Hilfe sogenannter Command-Tags, welche mit Hyperlinks in HTML vergleichbar sind. Sie können dazu benutzt werden, an bestimmte Stellen innerhalb der Präsentation zu springen, oder den Webbrowser mit einer bestimmten Seite aufzurufen. Abschließend soll das Beispiel aus Abbildung 20 einen Eindruck einer einfachen RealText-Datei vermitteln. Die Ausgabe des Programmablaufs ist in Abbildung 21 zu sehen.

```
<window type=¨tickertape''duration=¨1:00" width=¨350¨ loop=¨true¨
  underline_hyperlinks=¨false¨ link=¨white¨>
  <br/><b>
  <tu><a href=¨http://www.dowjones.com/¨>DJIA</a></tu>
  <tl>7168.35 +36.52 </tl>
  <tu>NIKEI 225 Index</tu>
  <tl>20603.71 +271.3</tl>
  </b>
</window>
```

Abbildung 20: Einfaches Beispiel für eine RealText-Datei

5.5 RealPix

RealPix gehört zu den neuesten Formaten von RealSystem. Es dient dazu, Bilder über das Internet zu streamen. Unterstütze Bildformate sind GIF (Version 87 und 89) sowie JPEG.

Abbildung 21: Börsenticker mit Hyperlink

Animated GIFs werden nicht unterstützt. Der Aufbau einer Präsentation erfolgt nicht über ein spezielles Tool, sondern auf Basis der HTML-Sprache.

Da Real G2 keinen Disketten-Cache beinhaltet und den direkten Download von Bildern verbietet, bleibt Material unter Copyright weiterhin geschützt. Ein Cache im Hauptspeicher ist vorhanden und ermöglicht damit effektiv einzelne Bilder einer Präsentation an verschiedenen Stellen wiederholt zu benutzen. Für die eigentliche Übertragung der Bilder bietet sich der RealServer an, da er einen kontrollierten Ablauf hinsichtlich der Timeline ermöglicht. Ein Standard-Webserver startet gleich zu Beginn der Präsentation mit der Übertragung sämtlicher Bilder, was sich in einem hohen Preroll niederschlägt.

Bei der Zusammenstellung einer Timeline ist zu beachten, ob der RealPix-Clip als einzige Anwendung laufen soll oder in Kombination mit anderen Strömen. Im ersten Fall kann der zeitliche Ablauf frei durch die Bildfolge festgelegt werden, Bandbreitenbetrachtungen spielen nur in extremen Fällen eine Rolle. Häufig wird aber die Präsentation aus mehreren Strömen bestehen. Dann ist es sinnvoll, den Ablauf an den anderen Clip, z.B. RealAudio, anzupassen und zu synchronisieren.

Für den Ablauf einer RealPix lassen sich verschiedene Einstellungen vornehmen, in erster Linie selbstverständlich die Reihenfolge und Verweildauer auf dem Bildschirm. Zusätzlich existiert eine Reihe von Übergängen, wie z.B. Einblenden (Fading) oder seitliches Einschieben (Wipe).

Abbildung 22: Crossfade zwischen zwei Bildern

Ein Spezialfall von RealPix ist der Broadcast von Bildern. RealNetworks stellt ein freies Programm zur Verfügung, dass sich an einen RealServer andockt und ein bestimmtes Verzeichnis überwacht. Jede Sekunde wird dieses Verzeichnis nach neuen Bildern durchsucht. Findet das Broadcast-Tool ein Update eines Bildes, wird dieses sofort an den RealServer gesendet, der den Broadcast an alle verbundenen RealPlayer regelt. Der bei den Playern durchgeführte Übergang kann entweder ein Fade-In oder ein Wipe sein.

Zum Start der Broadcast-Applikation müssen die Adresse des RealServers, der Port, an den das Bild gesendet werden soll, der Pfad und die Darstellungsgröße des Bildes bekannt sein. Zu beachten ist, dass das Update nicht häufiger durchgeführt wird, als die Übertragungsrate zu den Empfängern zuläßt, da sonst ein Datenstau am RealServer entstehen kann.

5.6 SMIL

Das Ziel dieses Abschnitts besteht darin, die Einsatzmöglichkeiten von SMIL aufzuzeigen. Zu Beginn wird eine kurze Einordnung von SMIL gegeben, welche die allgemeine Struktur von SMIL vorstellt. Im zweiten Teil wird dann das primäre Einsatzgebiet, die Synchronisation mehrerer Streams erklärt. Aus den vorgestellten Grundeigenschaften von SMIL lassen sich

dann komplexere Anwendungsmöglichkeiten konstruieren, welche Gegenstand des letzten Teils dieses Abschnitts sind.

5.6.1 Allgemeine Einordnung von SMIL

SMIL ist eine mit HTML vergleichbare Markup Sprache, die für die Präsentation mehrerer verschiedener Multimdia-Clips konzipiert wurde. Die Abkürzung SMIL steht dabei für **S**ynchronized **M**ultimedia **I**ntegration **L**anguage, wodurch das primäre Einsatzgebiet, die Synchronisation von mehreren verschiedenen Multimedia-Streams, deutlich hervorgehoben wird. Die Gemeinsamkeiten mit HTML ermöglichen so dem geübten HTML Benutzer einen problemlosen Einstieg in die Programmierung mit SMIL. SMIL-Files können mit jedem beliebigen Texteditor erstellt werden, der die Datei im *plain Text Format* speichern kann. Ein einfaches Beispiel für eine solche SMIL-Datei ist in Abbildung 23 dargestellt.

```
<smil>
  <head>
    <meta name="base" content="rtsp://realserver.comapny.com/"/>
  </head>
  <body>
    <audio src="eins.rm"/>
    <audio src="zwei.rm"/>
    <audio src="drei.rm"/>
  </body>
</smil>
```

Abbildung 23: Einfaches Beispiel für eine SMIL-Datei

Das SMIL-File bewirkt, dass die Multimedia-Clips eins.rm, zwei.rm und drei.rm der Reihe nach abgespielt werden, ohne dass gesonderte Synchronisationsbedingungen eingestzt werden müssen. Dieses Beispiel zeigt deutlich die Parallelen zwischen HTML und SMIL allein durch die verwendeten *Tags* auf. Desweiteren ist es mit SMIL möglich, ähnlich wie in HTML, das Layout einer Präsentation durch spezielle Kommandos zu verändern. Diese Funktionalität und damit verbundene Einsatzmöglichkeiten von SMIL in Bezug auf benutzerspezifische Anforderungen werden im letzten Teil dieses Kapitels vorgestellt.

5.6.2 Synchronisation

Die Synchronisation mehrerer Multimedia-Clips wird durch foldende Methoden erreicht, die im weiteren Verlauf des Kapitels genauer vorgestellt werden:

- Gruppierung in parallele und sequenielle Clips
- Setzen genauer Start- und Endzeiten einzelner Clips
- Individuelle Anpassgung der Spieldauer einzelner Clips

Für die Synchronisation mehrerer Streams ist es notwendig, die unterschliedlichen Multimedia-Clips in Gruppen aufzuteilen. Diese Gruppierung erfolgt dann über die Tags $< par >$ und $< seq >$. Durch diese einfache, vorgegebene Struktur innerhalb von SMIL-Dateien ist es leicht möglich, parallele und sequentielle Wiedergabe sowie Wiederholung von Multimedia-Clips zu realisieren. Die sequentielle Wiedergabe ist prinzipiell auch ohne die Angabe der entsprechenden Tags möglich, jedoch ist bei kombiniertem Abspielen von sequentiellen und parallelen Clips die Angabe zur Unterscheidung der Abschitte notwendig. Beispiele für solche Vorgänge zeigen die Ausschnitte aus SMIL-Dateien in den Abbildungen 24 und 25.

```
<seq>
  clip1
  <par>
    clip2
    clip3
  </par>
  clip4
</seq>
```

Abbildung 24: Beispiel 1 für paralleles und sequentielles Streaming

In diesem Beispiel wird im ersten Schritt Clip1 vollständig abgespielt und im zweiten Schritt dann Clip2 und Clip3 gleichzeitig wiedergegeben. Ist dieser Vorgang abgeschlossen, so wird zum Schluss dann Clip4 wiedergegeben. Es ist offensichtlich, dass eine Vertauschung der Tags ein völlig anderes Verhalten des Streams auslöst.

```
<par>
  clip1
  <seq>
    clip2
    clip3
  </seq>
  clip4
</par>
```

Abbildung 25: Beispiel 2 für paralleles und sequentielles Streaming

In diesem Beispiel beginnen Clip1, Clip2 und Clip4 gleichzeitig mit der Wiedergabe, da sie alle parallel ablaufen. Clip3 startet erst dann, wenn Clip2 fertig ist und läuft dann parallel zu Clip1 und Clip4.

Die Abbildung 26 bietet eine Veranschaulichung der geschilderten Abläufe der Beispiele aus den Abbildungen 24 und 25.

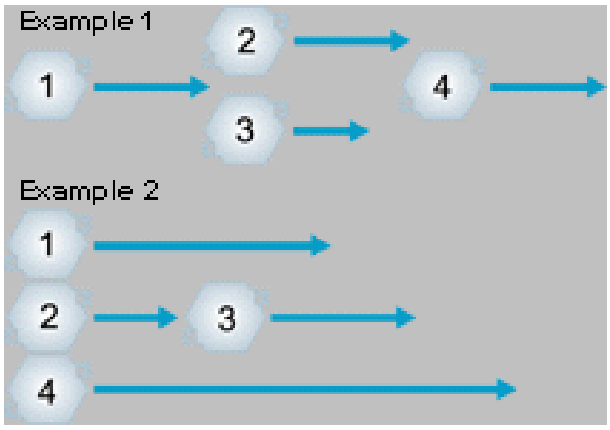


Abbildung 26: Wiedergabeverhalten bei unterschiedlicher Gruppierung

Bei der Erstellung einer Präsentation mit SMIL ist es möglich die Synchronisation einzelner Clips noch zu verfeinern. Es besteht die Möglichkeit genaue Angaben zur Start- und Endzeiten

bestimmter Clips sowie ihrer Spieldauer zu machen. Auf diese Weise gewinnt die Präsentation einen erhöhten Grad an Flexibilität bezüglich der Synchronisation im Vergleich zur einfachen Aufteilung in parallele und sequentielle Gruppen. Die Zeitangaben orientieren sich dabei an der Präsentations-Timeline. Diese startet, sobald die Wiedergabe eines Teils der Präsentation beginnt, wie in Abbildung 27 gezeigt wird.

```
<par>
  <audio src="song1.rm" clip-begin="30.4s" dur="30s"/>
  <audio src="song2.rm" begin=="28s" clip-begin="2.4s" clip-end="13.7s"/>
</par>
```

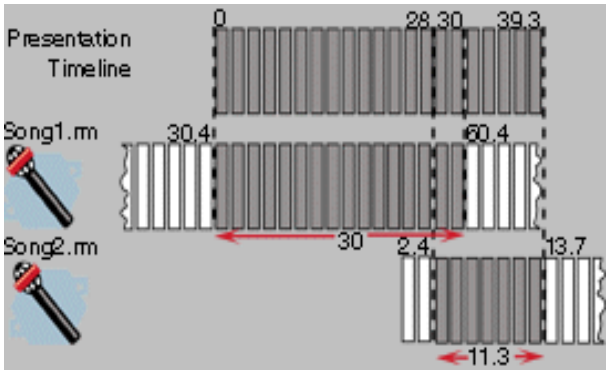


Abbildung 27: Timeline einer Präsentation mit SMIL

Die in diesem Beispiel (Abb. 27) verwendeten Zeitangaben verändern den parallelen Abschnitt dahingehend, dass die Wiedergabe der beiden Clips zu unterschiedlichen Zeitpunkten gestartet wird. So beginnt Clip1 seine Wiedergabe sofort beim Start der Präsentation, da keine Angabe zum Startzeitpunkt gemacht wird. Clip2 wird erst nach einer Zeit von 28 Sekunden abgespielt, wodurch sich eine Überschneidung der beiden Clips von 2 Sekunden ergibt, weil Clip1 für eine Dauer von 30 Sekunden läuft. Es ergibt sich also eine Gesamtspielzeit von 39.3 Sekunden. Es ist offensichtlich, dass ohne die Einführung dieser Zusatzelemente ein solches Wiedergabeverhalten nur sehr schwer durch „Zerstückelung“ der einzelnen Clips zu erreichen wäre.

5.6.3 Dynamische Ablaufanpassungen einer Präsentation

Bei der Erstellung einer RealSystem Präsentation bietet SMIL einen weiteren großen Vorteil gegenüber der Verwendung eines einfachen Streamingformats. Es ist möglich, das Wiedergabeverhalten eines Streams den äußeren Gegebenheiten anzupassen, so dass der Betrachter immer die optimale Übertragung erhält, ohne unterschiedliche Streams auswählen zu müssen. Dies wird durch die Angabe von möglichen Alternativen innerhalb der SMIL-Datei erreicht, die an bestimmte Bedingungen geknüpft sind. Die Abbildung 28 zeigt den prinzipiellen Aufbau einer solchen, durch das $< switch >$ Tag eingeleiteten, Auswahlmöglichkeit.

Ein Anwendungsgebiet für dieses Konstrukt ist das Angebot mehrsprachiger Präsentationen, die sich bei der Wiedergabe auf die jeweilige Landessprache einstellen. So ist es z.B. einfach möglich, einen Vortrag als Videoclip zu erstellen und mit Audioclips in unterschiedlicher Sprache anzubieten. Der Realplayer würde dann, durch das SMIL-File gesteuert, den zur Systemsprache passenden Audiostream auswählen und wiedergeben.

Auf die gleiche Weise ist es möglich, eine Präsentation an unterschiedliche Bandbreiten anzupassen, ohne dabei unterschiedliche Adressen oder Server benutzen zu müssen. Das Beispiel

```
<switch>
  <Auswahl1 Bedingung1="Wert1"/>
  <Auswahl1 Bedingung2="Wert2"/>
</switch>
```

Abbildung 28: Verzweigungen in SMIL

aus Abbildung 29 illustriert dieses Konzept anhand einer mehrsprachigen Videopräsentation, die sich zusätzlich an unterschiedliche Übertragungsbandbreiten anpasst.

```
<switch>
  <par system-bitrate="75000">
    <video src="vortrag75.rm"/>
    <switch>
      <par>
        <audio src ="audio75_d.rm" system-language="de"/>
        <audio src ="audio75_s.rm" system-language="es"/>
        <audio src ="audio75_f.rm" system-language="fr"/>
        <audio src ="auidio75.rm"/>
        <!-- Dies ist die standard Wiedergabe, wenn
         Sprachinformationen  nicht gefunden wurden -->
      </par>
    </switch>
  </par>
  <par system-bitrate="56000">
    <video src="vortrag56.rm"/>
    <switch>
      <par>
        <audio src ="audio56_d.rm" system-language="de"/>
        <audio src ="audio56_s.rm" system-language="es"/>
        <audio src ="audio56_f.rm" system-language="fr"/>
        <audio src ="auidio56.rm"/>
      </par>
    </switch>
  </par>
</switch>
```

Abbildung 29: Mehrsprachige Videopräsentation für unterschiedliche Bandbreiten

Zum Abschluß dieses Kapitels wird noch abrundend auf die Layout-Möglichkeiten innerhalb einer SMIL-Präsentation eingegangen. Hintergrund dabei ist, dass wenn man mehrere Multimedia-Clips gleichzeitig in einem Realplayer Fenster abspielt, natürlich festgelegt werden muss, an welcher Position innerhalb des Fensters welcher Clip gezeigt wird.

Zu diesem Zweck existieren in SMIL Befehle ähnlich wie in HTML, durch welche die Größe und Position der einzelnen Clips bestimmt wird. Auf eine konkrete Vorstellung der einzelnen Kommandos soll hier aber verzichtet werden, da dies den Rahmen eines konzeptionellen Überblicks deutlich überschreitet.

Ein Anwendungsbeispiel wäre etwa ein Videostream, zu dem zusätzliche Informationen in Form von Untertiteln oder Werbung übertragen werden sollen. Dies ist besonders bei Live-

Streams denkbar die etwa in einigen Jahren den Empfang von Fernsehprogrammen ersetzen könnten.

6 Protokollanbindung

Präsentationen sollten grundsätzlich über speziell dafür designte Server angeboten werden, denn nur so kann optimale Performance erreicht werden. In diesem Abschnitt werden zunächst die Möglichkeiten vorgestellt, mit denen man eine RealSystem-Präsentation in einem Netzwerk anbieten kann. Als Server kommen ein herkömmlicher Webserver, der über das *HTTP-Protokoll* angebunden wird und der RealSystem G2, der das *RTSP-Protokoll*[2] einsetzt, in Frage. Es wird nicht auf die konkreten Protokolle eingegangen, da diese Gegenstand anderer Vorträge sind und als bekannt vorausgesetzt werden.

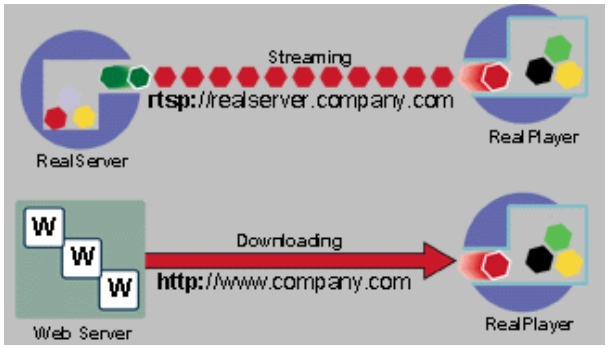


Abbildung 30: Protokolle zur Übertragung von Streams

Die Abbildung 30 stellt die Verbindung zwischen den entsprechenden Servern und der Client Anwendung, dem RealPlayer oder dem Webbrowser dar. Echtes Streaming ist nur von dafür konzipierten Präsentations-Servern möglich und Webserver realisieren das Streaming prinzipiell durch vollständiges Herunterladen der Clips, was bei großen Datenmengen und geringen Bandbreiten zu Problemen führen kann. Genau dieser Sachverhalt wird in den folgenden Beispielabläufen genauer beleuchtet. Eine durch ein SMIL-File definierte Präsentation wird jeweils über einen der beiden Servertypen bereitgestellt.

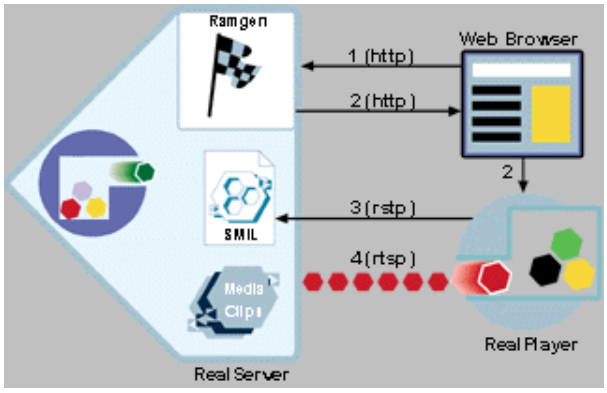


Abbildung 31: Abspielen eines SMIL-Files von einem Real-Server

[2] Real Time Streaming Protocol

Der Aufruf der SMIL-Datei erfolgt bei der Bereitstellung über einen RealServer (siehe Abb.: 31) in folgenden Schritten:

1. Unter Benutzung des HTTP Protokolls fordert der Webbrowser die SMIL-Datei vom RealServer an. Die Anfrage enthält zusätzlich Parameter, die an RAMGEN übergeben werden. Dieser ist dafür verantwortlich, dass die benötigten Streams abgerufen werden können.

2. Der RealServer veranlaßt den Browser, das Realplayer-Plugin zu starten und übergibt ihm die URL der Präsentation.

3. Der RealPlayer fordert die SMIL-Datei vom RealServer unter Benutzung des RTSP-Protokolls an.

4. Der RealPlayer wertet das SMIL-File aus und spielt die entsprechenden Multimedia-Clips unter Verwendung des RTSP Protokolls ab.

Durch die Benutzung des RTSP-Protokolls kann der Server während der Präsentation Einfluß auf die zu übermittelnden Daten nehmen. So können Engpässe in der Bandbreite durch Umstellung von zu übertragenden Paketen erreicht werden. Zu diesem Zweck wertet der Server die Prioritäten der einzelnen Pakete aus und übermittelt während eines Engpasses nur solche, die für den reibungslosen Fortlauf der Präsentation notwendig sind. Ebenso ist es nur von einem RealServer aus möglich SureStreams sowie Live-Übertragungen anzubieten, da benötigte Zusatzinformationen nur über RTSP bzw. RTCP[3] verfügbar sind.

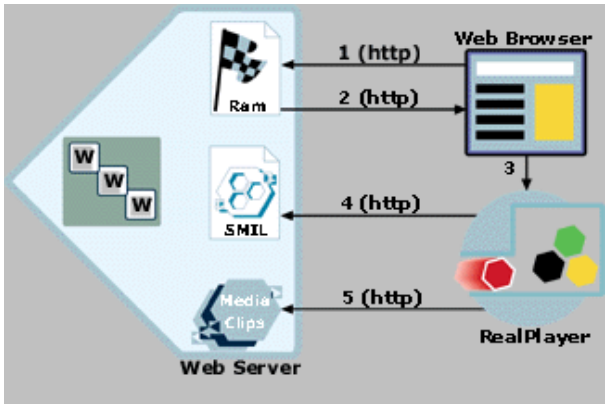


Abbildung 32: Abspielen eines SMIL-Files von einem Webserver

Der Aufruf der SMIL-Datei erfolgt bei der Bereitstellung über einen Webserver (siehe Abb.: 32) in folgenden Schritten:

1. Der Webbrowser fordert das für die Präsentation benötigte RAM-File vom Webserver über HTTP an.

2. Der Server übermittelt das RAM-File über HTTP.

3. Der RealPlayer wird als seperate Applikation vom Webbrowser gestartet und erhält dabei das RAM-File als Parameter

4. Der RealPlayer fordert die SMIL-Datei über HTTP vom Webserver an.

[3]Real Time Control Protocol

5. Die Präsentation wird über HTTP an den RealPlayer übermittelt.

Dadurch, dass die gesamte Kommunikation zwischen Server und Player über HTTP abläuft, können keinerlei Zusatzinformationen über den Präsentationsverlauf ausgetauscht werden. Bei aufwendigen Multimedia-Clips kann dies zu Unterbrechungen im Präsentationsablauf führen. Da Surestreams nicht von Webservern aus angeboten werden können, kann nur über die Verwendung von SMIL-Dateien auf unterschiedliche Bandbreitenanforderungen reagiert werden. Da stets der gesamte Clip heruntergeladen werden muss, ist der Preroll bei aufwendigen Präsentationen unangenehm hoch. Desweiteren sind Vorwärtssprünge innerhalb einer Präsentation nicht möglich, denn es kann nur auf bereits heruntergeladenen Daten zugegriffen werden. Der Anwender muss also warten, bis der kontinuierliche Download an der entsprechenden Stelle angelangt ist. Ein weiterer Nachteil beim Angebot von Streams über einen Webserver ist, dass dieser das Timeline-Konzept nicht unterstützt. So ist bei einer RealPix Präsentation der Preroll auch entsprechend hoch, denn der Download startet für alle Bilder zu Beginn der Präsentation (Preroll). Eben so ist das Live-Encoding, welches für einen Live-Broadcast benötigt wird nicht von herkömmlichen Webservern, sondern nur von RealServern möglich.

Zusammenfassend läßt sich also feststellen, dass für multimediale Präsentationen mit gehobenem Anspruch ein RealServer einem herkömmlichen WebServer klar vorzuziehen ist.

7 Sicherheitsaspekte

Bislang haben sich die meisten Streaming-Anwendungen auf die Sicherheit des benutzten Web-Servers verlassen. Für die meisten aktuellen Anwendungen ist dies ausreichend. Im Zuge der ständigen Verbesserung und Verbreitung von Multimedia-Streaming werden allerdings kommerzielle Angebote interessant, die strengere Anforderungen an die Sicherheit von Servern und Übertragung stellen. Für Pay-per-View Filme muss sichergestellt werden, dass nur autorisierte (zahlende) Nutzer diese sehen können. Dies schließt den Abruf der Filme, die abhörsichere Verbindung und eine nicht speicherbare Präsentation ein. RealSystem bietet verschiedene Möglichkeiten um dieses Sicherheitsniveau zu garantieren und setzt dazu auf den Möglichkeiten von Standard-Webservern auf. Ein funktionsfähiges System mit Firewall, sicheren Kernels und physikalischen Zugriffsbeschränkungen wird vorausgesetzt.

Dieses Kapitel stellt kurz die von RealSystem zur Verfügung gestellten Mechanismen vor, ohne tiefer auf die zugrundeliegenden Verfahren einzugehen. Ziel ist es, eine Zusammenfassung der Möglichkeiten für eine sichere Übertragung zu geben.

Erster Punkt der Sicherheitsbetrachtung ist der Zugriff auf die Daten des Servers. Je nach dem wie kritisch die Inhalte einer Präsentation sind, ergeben sich unterschiedlich aufwendige Stufen des Schutzes. Die einfachste ist sicherlich das Verstecken der Daten (*Security by Obscurity*), so dass nur User, denen die zugehörige URL bekannt ist, darauf zugreifen können. Dieser Standard eignet sich allerdings nur für die wenigsten Ströme und bietet die geringste Sicherheitsstufe.

Eine bessere Version ist die *File System Security*, die auch von dem aktuellen RealSystem G2 unterstützt wird. Hierbei wird bestimmten Rechnern der Zugriff auf die Daten gestattet. Der Administrator hat die Aufgabe, eine Datenbank mit den Berechtigungen zu verwalten. Die Computer identifizieren sich dazu über ihre IP-Nummer. Es muss sichergestellt werden, dass die Rechner nicht von unberechtigten Usern benutzt werden können. Diese Methode eignet sich damit vor allem für kleine Intranets. Ein weiteres Problem ist die einfache Möglichkeit des Abfangens/Mithörens der Übertragung einer IP und dem anschliessenden Maskieren der eigenen IP, was dem Angreifer die gleichen Privilegien verschafft wie dem autorisierten Nutzer.

Die am weitesten verbreitete Möglichkeit zur Gewährleistung der Sicherheit ist die *Application Level Security*. Bei diesem Verfahren wird ein Zugang zu den Daten nur dann gewährt, wenn der Anwender sich vorher zuverlässig identifiziert hat. Die Privilegien der entsprechen-

den User werden dabei in einer Datenbank gehalten, die den Zugriff auf die Streams regelt. Die RealSystem G2 Software unterstützt dabei folgende Verfahren zur Authentisierung:

- Digest: Dieses Verfahren wurde basierend auf dem HTTP/1.0 RFC 2069 Standard entwickelt. Das Passwort wird nicht im Klartext sondern als Hash-String übermittelt. Dieses Verfahren ist damit relativ gut gegen Abhören gesichert.
- HTTP-Basic: Hierbei werden Username und Passwort unverschlüsselt übertragen. Diese Form der Authentisierung eignet sich damit lediglich für nicht kritische Bereiche.
- NTLM: Windows NT Authentisierung. Dieses Verfahren soll die einfache Integration in bestehende NT-Netzwerke ermöglichen.
- Zertifikate: Die Identität des Anwenders wird durch den Austausch von Zertifikaten über eine dritte Stelle verifiziert. Dieser Mechanismus soll sicherstellen, dass die Verbindung zwischen Server und Player direkt stattfindet und kein Zwischenglied in der Verbindung Identitäten oder Aktivitäten verfälscht.

Die Authentisierung des Anwenders läßt sich in zwei Bereiche unterteilen: *Player-Based* und *User-Based*. Das erste Verfahren bleibt völlig transparent für den Nutzer. Nach einer Registrierungsphase wird dem RealPlayer eine eigene ID zugewiesen und der Zugriff auf gesicherte Inhalte erfolgt ohne weitere Abfragen. Dieses Verfahren ist *nicht* mit demjenigen gleichzusetzen, bei dem der RealPlayer eine weltweit eindeutige ID erzeugt und somit die Zugriffe des Users protokolliert werden können. Im Gegensatz dazu verlangt das Verfahren zur Authentisierung des Anwenders zur Registrierung das Einverständnis und die Kommunikation mit dem Nutzer. Bei diesem Verfahren muss entweder sichergestellt werden, dass der Zugriff auf den RealPlayer nur den zulässigen Personen erlaubt wird, oder die Clips von einer ganzen Gruppe ohne feste Computerzuteilung gesehen werden (z.B. Trainingsvideos in Schulungsräumen).

Das zweite Verfahren (User-basierend) verlangt ebenfalls eine Registrierungsphase, in der Username und Passwort auf dem Server festgelegt werden. Im Gegensatz zur ersten Möglichkeit wird hier vom Andwender bei jedem Zugriff auf einen gesicherten Inhalt die Eingabe des Namens und des Passwortes verlangt. Anwendungsgebiete hierfür sind z.B. Pay-per-view Filme, bei denen der Nutzer eindeutig identifiziert werden muss.

Sollen beide Verfahren für dieselben Inhalte und Web-Seiten eingesetzt werden, muss für jedes ein eigener RealServer eingerichtet werden.

Neben den Verfahren zur Sicherstellung der Identität des Users bietet RealSystem folgende Zugriffsmöglichkeiten auf die Daten:

- Directory, File or Event-based access: Vollzugriff auf die Daten.
- Countdown access: Der Anwender bekommt ein „Zeitkonto“. Auf diesem Konto wird eine Zeit festgehalten, für die auf die Daten zugegriffen werden kann. Bereits gesehene Minuten werden entsprechend abgezogen.
- Countup access: Der Anwender darf den Clip über eine beliebige Zeit ansehen, die Zeit wird allerdings überwacht und aufgezeichnet.
- Date-based access: Vollzugriff bis zu einem bestimmten Datum.

Nach der Sicherstellung der Identität des Anwenders ist eine geschützte Übertragung der Daten wichtig. Grundsätzlich besteht die Möglichkeit, die Clips durch den Secure Sockets Layer (SSL) zu schicken, dies wird aber nicht empfohlen. Der durch die Anwendung von SSL produzierte Overhead macht einige Vorteile des Streamings wieder zunichte. Die Zeit für Codierung und Decodierung geht zu Lasten des Clips, was einen Verlust an Qualität mit sich bringt. Zusätzlich ist das für Streaming in der Regel verwendete Protokoll UDP, für SSL-Übertragungen TCP.

RealNetworks sieht keinen Nachteil darin, dass SSL nicht verwendet werden sollte. Nach ihren Angaben ist der Vorgang des Zusammenfügens von während einer Übertragung abgefangenen Paketen nicht in vernünftiger Zeit durchführbar. Die Gefahr, dass eine Präsentation von nicht berechtigten Personen benutzt wird, sei damit sehr gering.

Desweiteren besteht bei der Codierung von Strömen die Möglichkeit, eine Option zu setzen, die das Speichern auf der Host-Seite verbietet. Damit kann der Anwender die Präsentation ansehen, aber nicht lokal sichern oder verarbeiten. Problematisch ist dagegen das sofortige Weiterleiten der Pakete an andere Player. Der Server besitzt darüber keine Kontrollmöglichkeit mehr, d.h die Streaming-Daten müßten im Verlauf der Präsentation sofort nach ihrer Darstellung vernichtet werden.

Schätzt man abschließend die Wirkung der Sicherheitsstandards ab, so offenbaren sich einige Lücken. Vollständige Sicherheit oder ein ähnliches Maß z.B. wie bei heutigen Banktransaktionen[4], wird nicht erreicht. Sollte Multimedia-Streaming in einiger Zeit auch in größerem Umfang kommerziell interessant werden, muss gerade der Sektor Sicherheit weiter ausgebaut werden.

8 Ausblick

Durch die hohe Akzeptanz des Internets gewinnt das Streaming multimedialer Daten zunehmend an Popularität. Der heutige Stand der Technik erlaubt bereits ansprechende Präsentationen. Insbesondere sind im Audio-Bereich Übertragungen in Radioqualität über Modem-Verbindungen zu empfangen. Für die Video-Übertragung sind die Grundsteine für eine weitere Entwicklung gelegt. Ständige Verbesserungen der Codecs führen zu einer weiteren Verringerung der benötigten Bandbreite bei gleichzeitiger Qualitätssteigerung. Der kommerzielle Einsatz wird derzeit noch durch die geringe Leistungsfähigkeit der Hardware im Bezug auf Bandbreite gehemmt. Die technischen Voraussetzungen (z.B. durch entsprechend konfigurierte Server) sind verfügbar, so dass in diesem Sektor aufgrund der Weiterentwicklung der Netzanbindung ein wachsender kommerzieller Markt besteht.

Es ist in einigen Jahren durchaus vorstellbar, dass sich der Schwerpunkt im Konsum auf das Internet verlagert. Dieser Trend könnte bei entsprechender Qualität durchaus eine Konkurrenzposition zum Fernsehen einnehmen. Sollte sich diese Tendenz bestätigen, ist damit zu rechnen, dass weitere Firmen in dieses Marktsegment drängen und die Entwicklung forcieren.

[4] Stichwort: HBCI (HomeBankingComputerInterface)

Literatur

[Real] http://www.real.com: Startpunkt für alle Informationen zu RealNetworks

[RPG] RealNetworks: *RealSystem G2 Production Guide*,
http://www.realnetworks.com/devzone/library: Grundlegende Informationen zum Produktionsprozeß (RealAudio, RealVideo...)

[RAG] RealNetworks: *RealSystem G2 Administrators Guide*,
http://www.realnetworks.com/devzone/library: Serverdokumentation

[G2Kit] http://proforma.real.com/mario/tools/authkit.html, Weiterführende Dokumentation (enthält u.a. den Production Guide und gesonderte Dokumentation zu RealPix und RealText sowie einige Tools)

[Download] http://www.realnetworks.com/products/: Downloadseite der freien und kommerziellen Basiswerkzeuge

Das Advanced Streaming Format (ASF)

Gerd Zellmer

FB Informatik, Uni Dortmund
G.Zellmer@online.de

Zusammenfassung

Dieser Vortrag über das Advanced Streaming Format (ASF) soll einen Einblick in den Aufbau, die Funktionalität und Einsatzgebiete von ASF geben. Der Schwerpunkt liegt dabei auf dem Einsatz der ASF- Streamingtechnologie im Internet.

1 Was ist ASF ?

Advanced Streaming Format (ASF) ist ein Dateiformat zum Speichern von Multimediadaten. Wie mit „advanced“ angedeutet, handelt es sich um ein Dateiformat, welches im Gegensatz zu vielen anderen Dateiformaten viele verschiedene multimediale Elemente vereint. Dazu gehören Text, Grafik, Animationen, MIDI, Video (AVI, MPEG...) und Audio (WAV). Diese Multimediadateien können lokal gespeichert, wie auch über Netzwerke übertragen werden. In diesem Vortrag liegt der Schwerpunkt auf letzterem. Das Advanced Streaming Format (ASF) ist ein Präsentationsformat und im Gegensatz zu einem Editierformat zur effizienten Wiedergabe der multimedialen „Streams“ vom Server zum Client entwickelt worden. ASF-Dateien können durchaus auch editiert werden, doch ersetzen sie nicht die Editierwerkzeuge und Tools zum Editieren von high-end Videos und Animationen.

Ziel war es, mit dem Advanced Streaming Format (ASF) einen Standrad zu schaffen, der alle gängigen Multimediaformate unterstützt, sie in einem Datenstrom vereint und diesen dann über eine Vielzahl von Protokollen (HTTP, UDP, TCP, RTP ...) über das Netzwerk überträgt.

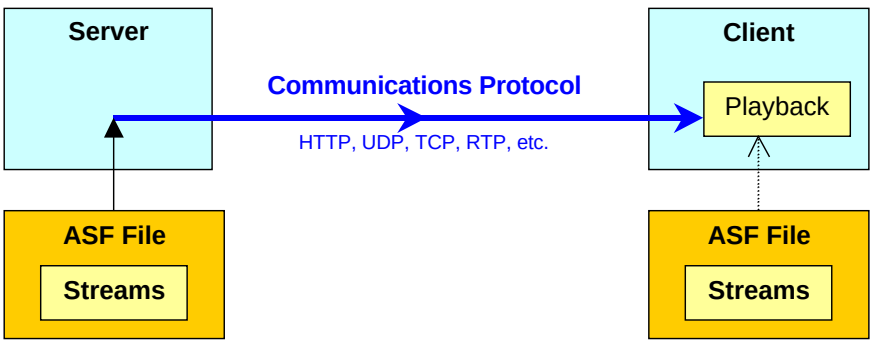


Abbildung 1-1: Streaming Wiedergabe einer ASF-Datei

2 Zeitliche Entwicklung des Advanced Streaming Format

Frühere Multimediaformate wie WAVE, AVI und QuickTime wurden nicht zum Übertragen über ein Netzwerk entwickelt. Sie entstanden in einer Zeit, in der Einzelplatzrechner noch weiter verbreitet waren als Netzwerkrechner und unter der Annahme, daß diese Einzelplatzrechner eine ausreichend große Datenübertragungsrate haben. Da heute Netzwerke, insbesondere das Internet, eine große Verbreitung haben, ist es unerlässlich, neue Möglichkeiten der Übertragung von multimedialen Daten über diese Netze, ohne lange Downloadzeiten (möglichst live), einzusetzen. Dafür gab es viele Ansätze, jeder Hersteller ging jedoch seinen eigenen Weg. Es entstanden eine Menge nicht zueinander kompatibler Dateiformate, wie z.B. ASFv1 von Microsoft, RMFF von Real Networks, VDO von VDONet, VIV von Vivo und VXI von Vxtreme. Das machte den Austausch von Multimediadaten untereinander sehr schwierig. Man einigte sich schließlich darauf, gemeinsam einen neuen Standard aus all diesen Formaten zu entwickeln. Mit dem Advanced Streaming Format (ASF) wurde ein Schritt in diese Richtung gemacht. Ziel war es, einen Standard zu entwickeln, der es erlaubt, Multimediadaten effizient über Netzwerke zu übertragen. Des weiteren ist es das Ziel, der Industrie einen Standard ähnlich des VHS und NTSC zu liefern.

Also schlossen sich die Entwickler von WAVE, AVI, ASFv1, VIV, RMFF, AVI2 und VXI, allen voran Microsoft, Real Networks, Intel, Adobe und Vivo zusammen, und entwickelten das Advenced Streaming Format (ASF). Erweitert und verbessert wurde das Advanced Streaming Format (ASF) von über 40 weiteren Firmen[1]. Bis zur endgültigen Version waren über 100 Firmen und Universitäten[2] an der Entwicklung beteiligt.

---

[1] Apple Computer, Asymetrix, Avid, Cakewalk Music Software, Dictaphone Company, Digidesign, Digital Lava, Digital Media Technologies, Digital Renaissance, Dolby Laboratories, The Duck Corporation, Eloquent, Ephyx Technologies, Frauenhofer Institute, Image Mind, Liquid Audio, Lucent Technologies, Macromedia, MGI Software, Olivr, Oracle, Perceptual Robotics, Precept Software, Silicon Graphics, Sonic Foundry, Starlight Networks, Syntrillium, Telos Systems, VDOnet, Vidsoft, Voxware, Pitango Multimedia, Waves, WebTV, Xing Technology

[2] 3Com, Adobe, NRGPR.com, Apple, Telos Systems, Imaginet Design, Avid, Bell Atlantic, Cakewalk, Intel, Digital Renaissance, Digital Lava, Georgia Tech, Duck Corp, Eloquent, Ephyx, Eurodat, Microsoft, FHG, ImageMind, Starlight Networks, Isone, Liquid Audio, Live Picture, Macromedia, Perceptual Robotics, Precept, Progressive Networks, YOUBET.com, Sonic Foundry, Syntrillium, VDONet, Vivo, Voxware, Xing Technology, Aristech, MOT.com, CODETEL.com, Nextel, Earthlink, SOTON.AC.UK, SaskTel, WebTV, IBM, Icast, Sun, Tandem, Decinc, ALGONET.SE, EIDOS.CO.UK, ONLINE.NO, Siemens, InfoNova, Integra, SBC, Cornell, SmallPlanet, Netscape, TMMNA, AccessPro, OneTouch, Creaf, Hitachi, UniNetT.NO, MACNICA.CO.JP, Cyclovision, Televitesse, CSI-Health, Office of Naval Research, Oracle, Iterated, UMR, CNET, NCSA.UIUC, INRIA, Digev, University of Trieste, AmericaSTTV, Saville.com, Aerojet, Activate, MediaOne, Today1, Virtual Café, SAATEL.IT, Susanville.com, Hal-PC.org, Best.com, Redbox.net, Jenn.com, InternetMCI.com, KC-INC.net, VideoMaker, Adaptive Media, Enabel, Samsung, Learning Co, Flash.net, Motics.com, Stellar.com, Pipex.com, Hollyworlds, USLead, Bayarea.net, Centillion Data, FP3D, Telemedia, Duplexx, Lucent Technologies, Direct.CA, Wide.AD.JP, NEC, University of Wisconsin, Eidos, Fivebats, Perey.com, Vosaic, Soton.AC.UK, OZ.IS, Netdesign.net, University of Ulm, Silicon Graphics Inc, LTH.SE, SEA.CO.IL, Winnov, Mediastra, Portal.CA, Periphere.BE, MC.net, 4CS.com, Uni-Paderborn.de, University of California San Diego, AT&T, Boystown, UMN.edu, DalSemi.com, Livewire Entertainment, Sinolib.ust.hk, Philips, Boeing, Nokia, Planetepc.fr, Disceurope.co.uk, Merging.com, Kolumbus.fi, Hisec.com, Vela.com, Cyberway, Digidesign, Columbia University, ABM.COM.SG, British Telecom, Washington University, Optivision, Array, SQLI, Berkom.de, Cisco, Pitango Multimedia, Dolby, Waves, Videopress Software, MGI Software

2.1 Übersicht über die zeitliche Entwicklung des Advanced Streaming Format (ASF):

i.) 1996 Entwickelt Microsoft die erste Version von ASF und implementiert sie in NetShow™

ii.) 14. August 1997: Erste draft Version von Microsoft, Real Networks, Intel, Adobe und Vivo war fertig.

iii.) 10. September 1997: Zweite draft Version unter Mitwirkung von 40 weiteren Firmen fertiggestellt.

iv.) 30. September 1997: Komplette Spezifikation fertiggestellt.

v.) 1.Quartal 1998: Developertools stehen zur Verfügung .

3 Die Eigenschaften von ASF

Das Advanced Streaming Format (ASF) ist ein erweiterbares Dateiformat, konstruiert um synchronisierte Multimediadateien zu speichern und diese über viele verschiedene Netzwerke und Protokolle übertragen zu können. Doch auch das lokale Abspielen der Dateien ist möglich.

Das Advanced Streaming Format (ASF) unterstützt die verschiedene Medientypen wie Text, Grafik, Audio, Video, URLs und Animationen. Dabei ist es weitgehend egal, in welcher Dateiform diese Daten vorliegen und wie sie zusammengesetzt werden sollen. Mit den entsprechenden Tools (z.B. Microsoft Media Tools) werden diese Dateien einfach zusammengefügt, bearbeitet und erst dann zu einer ASF- Datei konvertiert. Dabei können verschiedene Bandbreiten für die Übertragung festgelegt werden. So z.B. kann ein und die selbe ASF-Datei eine Präsentation in verschiedenen Bandbreiten enthalten (z.B. 28.8k, 56K ...). Der Client wählt dann die Bandbreit, die das Netz ihm bereitgestellt hat, und erhält den Datenstrom in der für ihn am besten geeigneten Qualität.

Nicht nur die Bandbreite kann gewählt werden, auch ist es möglich, die Präsentation in vielen verschiedenen Sprachen zu produzieren. Dabei kann es sich um Audio oder auch Texte handeln. Der Client kann dann die gewünschte Sprache wählen, in der er die Präsentation abspielen möchte.

Des weiteren können innerhalb einer Präsentation Sprungmarken definiert werden, zu denen dann beliebig gesprungen wird. Diese Marken können auch noch nachträglich eingefügt und geändert oder entfernt werden (abgesehen von Liveübertragungen). Auch das Springen an beliebige Stellen in der Datei ohne Sprungmarke ist möglich (schneller Vor- bzw. Rücklauf).

Eine weitere Eigenschaft des Advanced Streaming Format (ASF) ist, daß es unabhängig von dem Betriebssystem, Datenkommunikationsprotokoll und der Beschaffenheit der Multimediaumgebung ist. Das heißt, daß das ASF-Dateiformat ein sehr universelles Dateiformat ist, welches sich an viele Situationen und Umgebungen anpaßt.

4 ASF Dateistruktur

Eine ASF- Datei enthält drei übergeordnete Objekte. Ein „Header Objekt“, ein „Data Objekt“ und ein „Index Objekt“. Das „Header Objekt“ und das „Data Objekt“ müssen unbedingt in der ASF-Datei vorhanden sein und sie dürfen jeweils nur einmal vorkommen. Das „Index Objekt“ ist optional, aber es wird empfohlen, es grundsätzlich mit zu verwenden.

Eine minimale Ausführung einer ASF- Datei besteht erstens aus einem „Header Objekt“, welches ein „File Properties Object“, ein „Stream Properties Object“ und ein „Language List Object“ enthält. Es muß immer am Anfang in einer ASF-Datei stehen. Zweitens aus einem „Data Objekt“, welches die „ASF data units“ enthält und direkt dem „Header Objekt“ folgt.

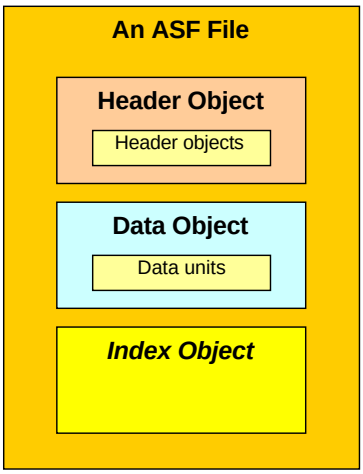


Abbildung 4-1: ASF – Dateiaufbau

4.1 Die ASF Object-definition

Die Basiseinheiten der ASF-Dateien sind die ASF-Objekte. Jedes ASF-Objekt hat den gleichen Aufbau. Die ASF-Objekte sind in drei Bereiche aufgeteilt. Den ersten Teil bildet ein 128-Bit „global unique identifier“, der zweite Teil ist ein 64-Bit großes Objekt, welches die Länge der folgenden Daten angibt. Der dritte und letze Teil ist ein Datenteil mit variabler Länge, welche den eigentlichen Inhalt der ASF- Datei enthält. Also hat das gesamte Objekt eine Länge von 24-Bytes zuzüglich der Länge der Daten-Bytes.

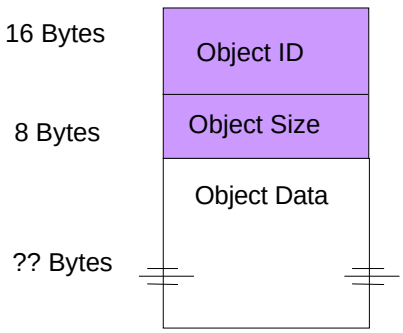


Abbildung 4-2: Das ASF Object

Diese Dateiorganisation wurde in Anlehnung an das „Resource Interchange File Format“ (RIFF) gewählt, da dieses die Basis für das AVI- und WAVE- Dateiformat ist, welches langfristig durch das ASF- Dateiformat ersetzt werden soll.

ASF-Datei Objekte	Optional	Objekt instanzen	Beschreibung
Header Object	Nein	Nur eines	Das “Header Object” beschreibt den ASF-multimedia stream als Ganzes. Es enthält globale, sowie spezifische Informationen über den Inhalt des streams, optionale Indexinformationen, optionale Keyinformationen und media stream Definitionsinformationen. Diese Informationen können separat über ein “sicheres” Protokoll übertragen werden.
Data Object	Nein	Nur eines	Dieses Objekt beinhaltet den Multimedia-Dateninhalt. Es enthält die “Daten units”, welche in Paketen nach Sendezeit geordnet organisiert sind.
Index Object	Ja	mehrere	Dieses Objekt enthält Indexeinträge zu den “Daten units” innerhalb der Daten Objekte. Diese “Index Objekte” sind nicht Teil des Streaming. Sie dienen zum schnellen Nachschlagen, Suchen und Überwachen. Des weiteren können sie wichtige Informationen über den multimedialen Inhalt enthalten, wie z.B die video key frames.
Other object	Ja	mehrere	Die ASF-Definition erlaubt auch die Nutzung anderer Objekte, welche durch ihren eigenen UUID (universal unique identifier) definiert sind.

Abbildung 4-3: ASF Datei-Objekte

4.2 Das ASF Header Object

Das ASF *Header Object* ist eine Bytesequenz am Anfang einer jeden ASF- Datei. Sie beschreibt dem Client, welche Daten er erhalten wird, wenn er die Datenübertragung beginnt. Ohne diese Informationen wäre es dem Client unmöglich zu entscheiden, was er mit den erhaltenen Daten anfangen soll.

Das ASF *Header Object* ist das einzige, welches auch andere ASF- Objekte enthalten kann. Dies sind im Einzelnen folgende:

4.2.1 File Properties Object

Das *File Properties Object* enthält Basisinformationen über die ASF- Datei. So z.B. das Erstellungsdatum oder die Anzahl der verschiedenen Streaming- Typen in dieser ASF- Datei.

4.2.2 Stream Properties Object

Das *Stream Properties Object* enthält die medienspezifischen Angaben zu jedem in der ASF- Datei enthaltenen Stream. So gibt es z.B. für ein Audio-Stream die Samplerate pro Sekunde an, oder bei einem Videostream die Höhe und Breite der Frames. ASF kennt sieben Kernmedien:

Audio, Video, Image, Timecode, Text, MIDI, und Command.

Auch ist es möglich, über das *Data unit Extension Object* eigene Medientypen (andere als die sieben schon vorhandenen) zu definieren.

4.2.3 Content Discription Object

Dieses Objekt dient zum Speichern von individuellen Informationen der Präsentation. Das sind z.B. Name der Präsentation, Name des Autors, Copyright und andere Beschreibungen.

4.2.4 Script Command Object

Dieses Objekt enthält Angaben über URLs, die zwischendurch aufgerufen werden. Oder aber auch Befehle andere Dateien zu laden, welche dann in einem separaten Fenster dargestellt werden sollen.

4.2.5 Marker Object

Mit diesem Objekt erhält der Autor die Möglichkeit, Marken in eine laufende Präsentation zu setzen. Er kann z.B. einem Song einen Titel geben, oder Überschriften für eine laufende Präsentation definieren, die dann automatisch angezeigt werden und im richtigen Augenblick wechseln.

4.2.6 Component Download Object

Hier stehen alle Informationen, welche benötigt werden, um die Datei abzuspielen, also die Version des Tools zum Abspielen und die Adresse, an der die nötige Version zum Download bereit steht.

4.2.7 Stream Groups Object

Gibt dem Stream eine Gruppenzugehörigkeit.

4.2.8 Scalable Object

Der Basisstream ist so gespeichert, daß der Client im Stande ist ihn auch bei schlechten Verbindungen abzuspielen. Der Stream kann aber auch in mehreren Qualitäten gespeichert sein. Das scalable Object gibt an, in welchen Qualitäten der Stream vorhanden ist.

4.2.9 Prioritization Object

Hier kann der Autor Prioritäten für seine Präsentation angeben. Je nach Netzwerkbandbreite ist es sinnvoll, bei einem Video dem Ton eine höhere Priorität zu geben als dem Bild. Damit hätte der Betrachter zwar kein fortlaufendes Bild, aber der Ton wäre durchgehend und nicht abgehackt. Diese Priorität ist nicht immer bindend. Schaltet der Betrachter den Ton mit dem Mute-button aus, so kann die Tonpriorität aufgehoben werden. Dadurch erhält man eine höhere Videoqualität.

4.2.10 Mutual Exclusion Object

Sind in einer ASF- Datei verschiede Sprachen oder Bandbreiten vorhanden, so werden alle außer der mit der *Mutual Exclusion* Eigenschaft ignoriert.

4.2.11 Inter-Media Dependency Object

Macht verschiede Medienstreams voneinander abhängig.

4.2.12 Rating Object

Stellt die W3C Plattform for Internet Content Selection (PICS) – Definition zur Verfügung.

4.2.13 Index Parameters Object

Liefert Informationen, um den originalen Index der Datei wieder herzustellen.

4.2.14 Color Table Object

Stellt eine *ColorTable* für den Media Stream zur Verfügung. Es können auch mehrere dieser *Color Tables* vorhanden sein. Für jeden Media Stream einer.

4.2.15 Language List Object

Dieses Objekt gibt an, in welchen Sprachen die Präsentation vorhanden ist.

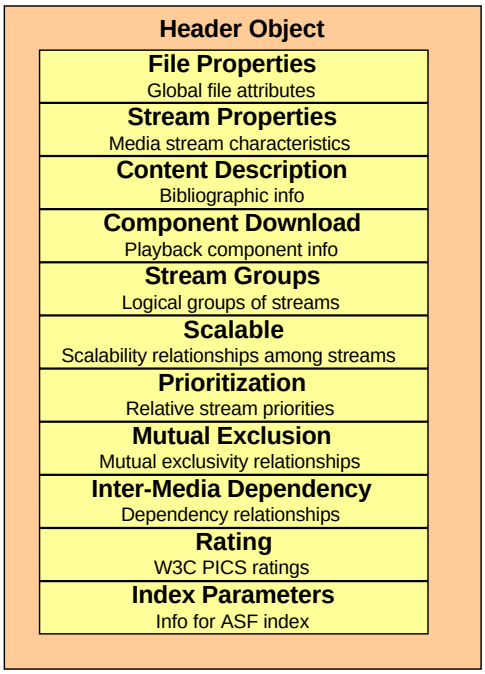


Abbildung 4-4: Aufbau des Header Objects

4.3 Das Data Object

In den *Data Objects* ist der aktuelle Multimediainhalt gespeichert. Zeigt die ASF-Datei gerade ein Video, dann sind in den *Data Objects* die Bilder und der Sound des aktuellen Videos gespeichert. Das Datenfeld des *Data Objects* ist in viele ASF-„data packets“ zerlegt. Alle diese Pakete haben den gleichen Aufbau.

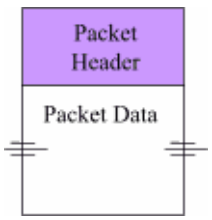


Abbildung 4-5: ASF-„Daten packet“

Der „Packet Header“ beinhaltet die Informationen, um die Datei in der richtigen Reihenfolge zu streamen. Die Packet Data sind die zum Header gespeicherten Daten, die gerade angezeigt werden. Jedes Datenpaket kann nur ein und den selben Datentyp beinhalten. So kann eine Paket mit Videobildern keinen Sound enthalten und umgekehrt. Die Größe der Datenpakete ist nicht vorgegeben. Es können viele Videoframes in einem Paket enthalten sein. Auch ist es erlaubt, nur ein Frame oder nur ein Teil eines Frames in ein Paket zu packen. Häufig wird genau ein Frame in einem Paket übermittelt.

Beim Senden einer Präsentation vom Server zum Client werden die Pakete nacheinander abgeschickt. Dabei werden die Pakete vom Client in der Reihenfolge zusammengesetzt, wie der Server sie abgeschickt hat. Je nach Geschwindigkeit des Netzwerks und gegebener Priorität werden die Pakete auch vom Server geordnet und abgesendet. Dabei wird nicht auf den Typ der zu sendenden Daten geachtet.

4.4 Index Object

Ein *Index Object* ist nicht unbedingt notwendig. Durch ein *Index Object* besteht die Möglichkeit, an bestimmte Stellen in der ASF-Datei zu springen. Es ermöglicht auch den schnellen Vor- und Rücklauf innerhalb einer ASF-Datei.

5 Real Time ASF- Übertragungen über RTP

Wie schon angedeutet, können Präsentationen mit ASF nicht nur effizient über das Netzwerk übertragen und vom Client wiedergegeben werden, sondern auch in Echtzeit abgespielt werden. Dabei werden die Daten beim Aufnehmen (Video und/oder Audio) direkt in das ASF-Dateiformat umgewandelt und über das „Real-Time Transport Protocol“ (RTP) übertragen. ASF-Dateien besitzen zwei Zeitstempel. Zum einen die Sendezeit und zum anderen die Präsentationszeit. Diese Zeiten sind notwendig, um die Präsentation später wieder richtig zusammenzufügen. Durch die Präsentationszeit läuft die Präsentation immer in der richtigen Geschwindigkeit ab. Mit der die Sendezeit können verlorengegangene Pakete festgestellt und auch die verschiedenen Streams richtig zusammengesetzt werden. Bei einer Real-Time-Übertragung (Live-Übertragung) wird nur der Sendezeitstempel verwendet, um die Pakete zu ordnen, und ggf. verlorene Pakete zu bemerken.

Im „Header Object“ einer ASF-Datei sind alle nötigen Informationen gespeichert, die der Client benötigt, um die ASF- Datei abspielen zu können. Deshalb muß das „Header Object“ in jedem Fall vor den „Daten Objekten“ einer ASF-Datei verarbeitet werden. Ohne die Header-Informationen sind die „Data Objects“ wertlos. Es wäre aber auch unsinnig, mit jedem RTP-Paket den Header mitzusenden. Um das Fehlen des „Header Objects“ beim Einstieg eines Client in einen Live Videostrom zu vermeiden, gibt es verschiedene Möglichkeiten. Zum einen könnte das „Header Object“ über ein anderes Protokoll übertragen werden. Eine andere Möglichkeit wäre, alle paar Sekunden das „Header Object“ mitzusenden. Das würde aber wieder dem Netzwerk mehr Bandbreite abverlangen. Eine bessere Möglichkeit besteht darin, die Präsentation über ein Real Time Streaming Protokoll (RTSP) zur Verfügung zu stellen. Ein Mediaserver würde über das RTSP Protokoll eine „Präsentationsbeschreibung“ bereitstellen, die dann über das RTP Protokoll übertragen werden kann. Diese „Präsentationsbeschreibung“ würde dann das ASF „Header Object“ zur Verfügung stellen.

6 Kompatibilität von ASF mit anderen Formaten

Entwickelt wurde ASF nicht, um andere Dateiformate wie MPEG und QuickTime zu ersetzen, sondern um ein Dateiformat zu schaffen, welches ein hohes Maß an Flexibilität und Kompatibilität besitzt.

6.1 ASF und AVI

ASF ersetzt AVI. Daß das nicht von heute auf morgen auf einen Schlag geschieht, ist sicher klar. Aber es ist das Ziel, AVI völlig durch ASF zu ersetzten. AVI wurde vor langer Zeit entwickelt, lange bevor das Internet eine so bedeutende Rolle spielte wie heute. AVI (Audio Video Interleave) ist zum Speichern und Abspielen von digitalen Videos auf Festplatte und CD-Rom entwickelt worden und nicht zum Übertragen der digitalen Videodaten über ein Netzwerk wie das Internet. Der Standard im Internet ist heute die HTML-Seite. Zur informativen Darstellung gehören nicht nur einzelne Bilder, Text und Animationen, sondern auch Seiten, in denen alles vereint ist. ASF liefert diese Möglichkeit. ASF-Dateien und AVI- Dateien bestehen beide aus einem „Header Object“, „Data Object“ und einem optionalen „Index Object“. Bei beiden beginnt der Header mit einem Dateneigenschaftenblock. Aber ASF-Dateien und AVI- Dateien haben nicht nur Gemeinsamkeiten.
Im Gegensatz zu AVI- Dateien sind die ASF- Dateien objektorientiert aufgebaut. Hat eine ASF- Datei viele verschiedene Objekte, so hat eine AVI- Datei an dieser Stelle sogenannte „chunks“. Diese „chunks“ sind nicht so universell einsetzbar wie die Objekte der ASF- Dateien. Als identifier nutzen die ASF- Dateien die sogenannten GUIDs, welche variabel den Dateninhalt der Datei beschreiben. Diese GUIDs sind einmalig und können erweitert werden, um eine neue Medienart zu beschreiben. AVI-Dateien dagegen nutzen fest vorgegebenen FOURCC (four-character code). Dieser ist fest und nicht erweiterbar. Auch bei den Längen-Bytes gibt es große Unterschiede. AVI verwendet eine 4-Byte Längenangabe für die Daten, was dazu führt, daß Videos nicht länger als ca. 2,5 – 3 Stunden seien können. ASF verwendet eine 8-Byte Längenangabe für die Daten. Das ergibt viele Stunden Video in einer Datei. Das weiteren hat eine ASF- Datei viele im „Header Object“ festgelegte Eigenschaften. Diese sind der AVI- Datei völlig fremd. AVI- Dateien sind auf eine feste Framerate ausgelegt. Dies ist sehr hinderlich in der Welt des Internets, da sich dort die zur Verfügung stehende Bandbreite ständig ändert. ASF paßt sich immer der aktuellen Bandbreite an. AVI kennt nur Audio und Video. Alle anderen multimedialen Elemente werden nicht unterstützt. ASF dagegen kennt von Grund auf schon sieben Medienarten (Audio, Video, Image, Timecode, Text, MIDI, und Command) und ist dort immer noch erweiterbar.

6.2 ASF und MPEG-1, MPEG-2 und QuickTime

MPEG-1 und MPEG-2 sind Videoformate von sehr hoher Qualität. Sie benötigen aber eine sehr große Netzwerkbandbreite (bis zu 1 MB) zum Übertragen von Videos. Steht eine solche Netzwerkbandbreite zur Verfügung, ist MPEG-1 und MPEG-2 durchaus mit unter den ersten Formaten, die man wählen würde. In dem Zeitalter des Internets sind Bandbreiten von 1 MB nur für die Wenigsten zu erreichen, für den normalen Anwender jedoch illusorisch. Auch unterstützen MPEG-1 und MPEG-2 nur Audio und Video und sonst keine anderen multimedialen Elemente. Deshalb ist es sinnvoll, für Netzwerke mit kleiner Bandbreite, wie das Internet, ASF-Dateien an Stelle der von MPEG-1- oder MPEG-2-Dateien zu nutzen.

Feature	MPEG-1	MPEG-2	QuickTime	ASF
Local and network playback				
Local playback	YES	YES	YES	YES
HTTP server delivery	YES	YES	YES	YES
Media server delivery	YES	YES	NO	YES
Extensible media types	NO	NO	YES	YES
Component download	NO	NO	NO	YES
Scalable media types	NO	YES	NO	YES
Prioritization of streams	NO	NO	NO	YES
Multiple language support	NO	NO	NO	YES
Environment independence	NO	NO	NO	YES
Rich inter-stream relationships	NO	NO	NO	YES
Expandability	NO	NO	NO	YES

Abbildung 5-1: ASF-Eigenschaften gegenüber anderen Formaten (Stand 1997 Quelle: Microsoft)

QuickTime wurde wie auch MPEG-1 und MPEG-2 nicht im Hinblick auf das Internet entwickelt. Alle diese Formate stammen aus der Zeit, bevor das Internet zum „alltäglichen“ Medium wurde. QuickTime ist durchaus auch streaming-fähig, aber seine Vorzüge und Schwerpunkte liegen anders. QuickTime ist ein multimediales Dateiformat, welches den gesamten Zykus abdeckt wie: aufnehmen, bearbeiten, übertragen und abspielen. Es existieren Tools zum Bearbeiten der Präsentationen. Doch die Stärken von QuickTime liegen nicht im Übertragen der Daten über das Netzwerk. Da ist es sinnvoller, vor dem Übertragen der QuickTime-Dateien diese in ASF abzuspeichern.

Das ASF- Format wird über kurz oder lang das AVI- Format ablösen. Das gilt nicht für die anderen Formate wie MPEG-1, MPEG-2 und QuickTime und ist auch nicht so geplant. ASF soll vielmehr in Zukunft als ein neuer Standrad neben diesen vorhandenen Formaten dienen. Dabei bedient sich ASF auch dieser anderen Formate, wie z.B. MPEG-1 und MPEG-2 als Komprimierungswerkzeug in einer ASF-Datei. Genauso können QuickTime-Dateien als ASF-Dateien abgespeichert werden, und somit mit der ASF-Streamintechnologie übertragen werden. Die Zeit wird zeigen, ob ASF sich als Standard etablieren kann und wird.

6.3 ASF und VXF, RA, RMFF und VIV

Bevor ASF als Multimediaformat entwickelt wurde, gab es nur M-JPEG, MPEG-1, MPEG-2 und QuickTime. Bald darauf kam dann noch AVI hinzu. Alle diese Formate waren nicht streaming-fähig. Als das Internet zum „alltäglichen“ Medium wurde, entstanden viele verschiedene streaming-fähige Dateiformate, wie z.B. VXF, RA, RMFF und VIV, um nur einige zu erwähnen. Diese Formate waren alle inkompatibel zueinander. So schlossen sich die Hersteller dieser Formate und Microsoft zusammen und entwarfen das ASF- Dateiformat.

6.4 ASF und MPEG-4

Auch MPEG-4 soll kompatibel zu ASF werden. Doch lagen mir bis heute noch keine Informationen vor, wie dieses verwirklicht werden soll. In einem Vorschlag an die MPEG-4 Requirements and System groups zum 42nd MPEG-meeting wird für die MPEG-4 Dateistruktur eine ähnliche wie für das ASF-Dateiformat vorgeschlagen. Was folgende Abbildungen verdeutlichen sollen.

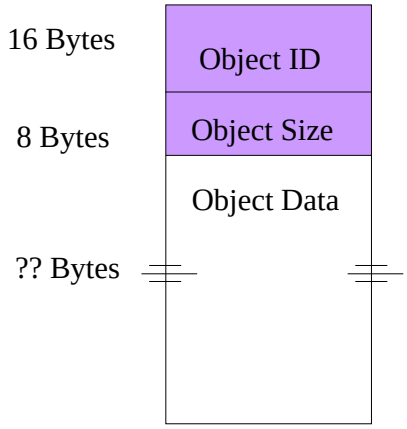


Abbildung 5-2: MPEG-4 Intermedia Format (MIF) Object

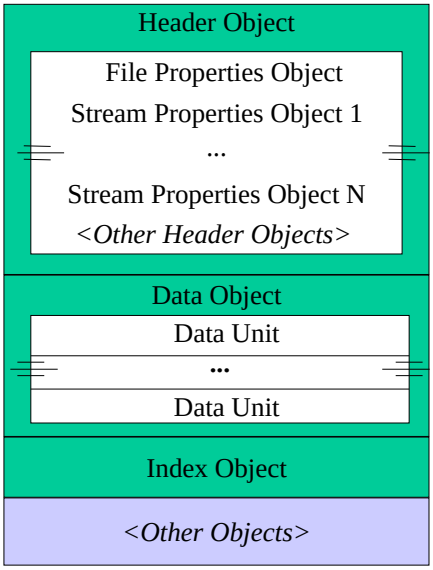


Abbildung 5-3: MPEG-4 Intermedia Format (MIF) Dateistruktur

Dabei haben die „Header Objects“ die selben Funktionen, wie die der ASF- Header. Selbiges gilt auch für die „Data Objects“, sowie die „Index Object“. Im „Header Object“ sind folgende Objekte enthalten:

- File Properties Object – beschreibt die Globalen Attribute der Datei.
- Stream Properties Object – definiert die charakteristischen Eigenschaften der Mediastreams.
- Object Descriptor Object – beinhaltet eine Beschreibung des MPEG-4 Objektes.
- Stream Map Tabel Objekt – enthält den MPEG-4 Stream Map Table

Wie man leicht erkennen kann, wäre diese Dateiform einer MPEG-4-Datei einer ASF- Datei sehr ähnlich.

7 Die ASF-Tools

Um ASF- Dateien abzuspielen und zu bearbeiten gibt es verschiedene Tools. Hier möchte ich den Windows Media Player als ASF- Player, den Windows Media Author als Tool zu Erstellen von ASF-Präsentationen und den Windows Media ASF Indexer vorstellen. Die praktische Vorführung wird sich auf das Abspielen von ASF-Dateien und das Erstellen einer ASF-Präsentation beschränken.

7.1 Windows Media Player

Der Windows Media Player ist ein Tool zum Abspielen von Videos, sowie auch Audio. Er beschränkt sich dabei nicht nur auf ASF-Dateien, sondern unterstützt viele andere Video- und Audioformate. Wir werden den Windows Media Player in diesem Vortrag zum Abspielen von ASF-Dateien verwenden. Dabei ist vorgesehen, eine ASF-Datei über das Internet abzuspielen, wenn es die technischen Gegebenheiten erlauben. Des weiteren werden wir eine ASF-Datei lokal von der Festplatte abspielen.

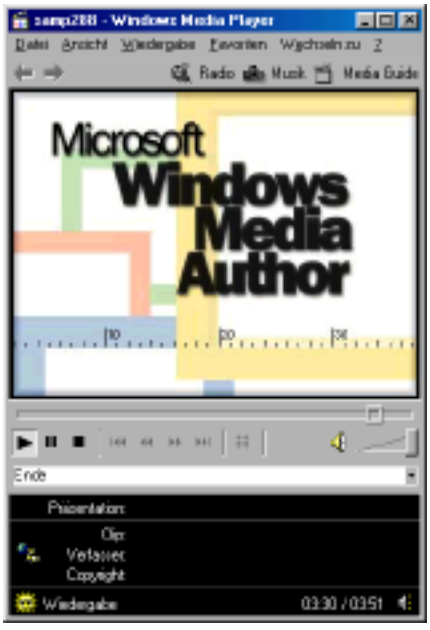


Abbildung 7-1 : Windows Media Player

7.1.1 Beispiele zu ASF-Video-streaming-Dateien

http://www.attheworld.com

http://www.streamingmedia.net/streamingmedia/west/webcast.html

http://www.kkrs.net/netmoviemaniahome/netmoviemaniahome.html

http://www.windowsmedia.com

http://www.videodetective.com/home.asp?list=3

http://disney.go.com/features/distribution/media/index.html

http://www.bloomberg.com/tv/tv.html

7.2 Windows Media Author

Der Windows Media Author ist ein Tool um ASF-Präsentationen zu erstellen. Wir werden das anhand einer fertigen ASF-Präsentation sehen. Der Windows Media Author macht es dem Anwender sehr leicht, eine ASF-Präsentation zu erstellen. Zuerst wählt man alle Bild-, Video- und Sounddateien aus, die man in die Präsentation einfügen möchte. Diese werden dann alle in einer Liste aufgeführt. Nun schiebt man diese an die richtige Position auf der Zeitachse. Jetzt können noch beliebige Markierungen angegeben werden, wie z.B. Überschriften oder Internetseiten, die zu einem bestimmten Zeitpunkt geladen werden sollen. Zu den angegebenen Marken kann später beliebig gesprungen werden. Deswegen ist es sinnvoll mit dem Setzen dieser Marken nicht kleinlich zu sein.

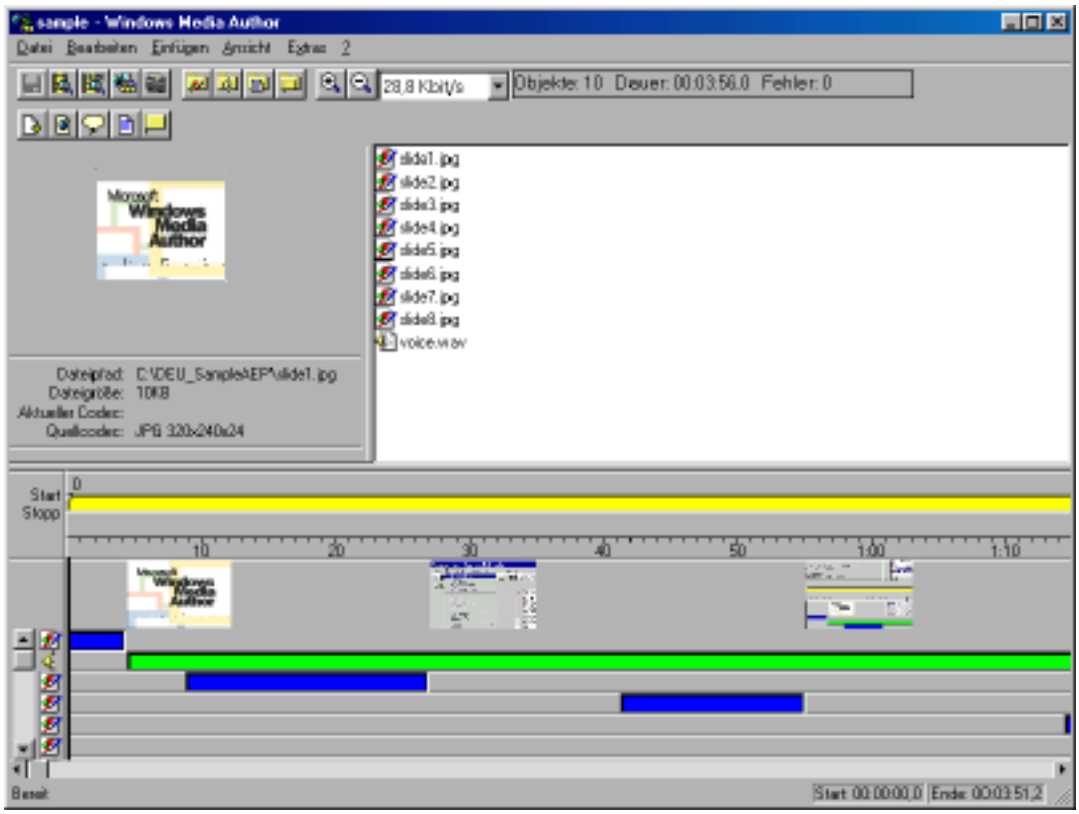


Abbildung 7-2: Windows Media Author

Zum Speichern der Präsentation stehen verschiedene Qualitäten bereit. Zum einen 28,8 Kbits/s zum andern 56 Kbits/s. Die Bildqualität der ASF- Präsentation ist nicht ganz so gut wie bei anderen Präsentationstools wie z.B. MS PowerPoint. Aber dafür kann sie ohne Weiteres über eine 28,8 Kbits/s Verbindung in Echtzeit übertragen werden.

7.3 Windows Media ASF Indexer

Wie schon erwähnt können in einer ASF- Datei Markierungen angegeben werden, zu denen dann auch gesprungen werden kann. Mit dem Windows Media ASF Indexer können diese Markierungen bearbeitet werden. Auch können die Eigenschaften einer ASF- Datei, wie z.B. der Titel, der Verfasser und eine Beschreibung angegeben werden.

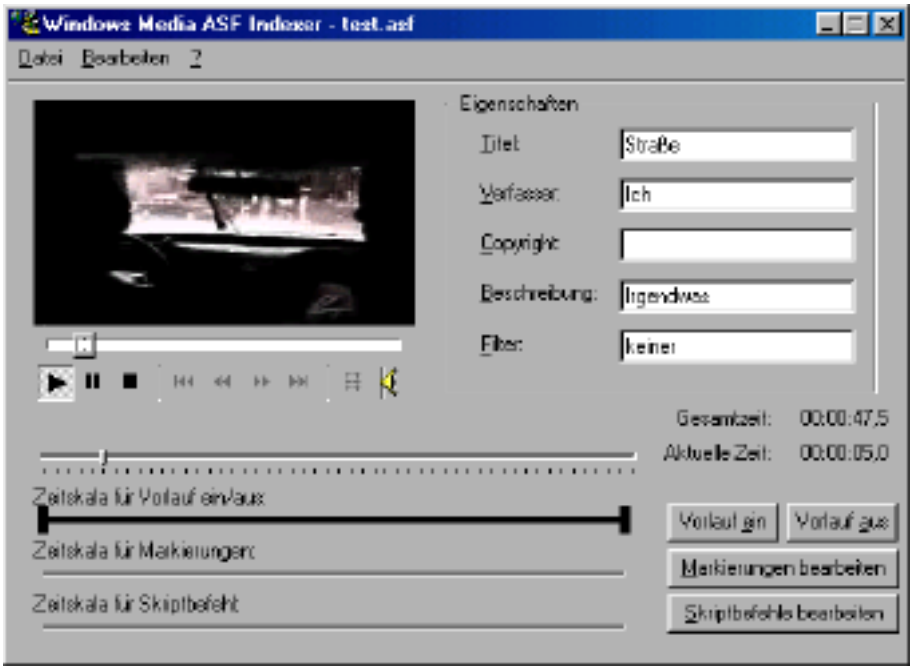


Abbildung 7-3: Windows Media Indexer

8 Literaturverzeichnis

[MSRN 98] Co-authored by Microsoft Corporation and RealNetwerks, inc.:
Advanced Streaming Format (ASF)
February 26, 1998, Public Specification Version 1.0
http://www.microsoft.com/ASF/specs.htm [Stand: 28.11.1999]

[Msof 98a] The Advanced Streaming Format Version 2 Software Developers Kit
Microsoft Corporation 18.02.1998

[Klem 98a] Anders Klemets:
RTP Payload Format for ASF Streams
http://www.microsoft.com/asf/resources/draft-klemets-asf-rtp-00.txt [Stand: 13.12.1999]

[Klem 98b] Anders Klemets:
Common Generic RTP Payload Format
http://www.microsoft.com/asf/resources/draft-klemets-generic-rtp-00.txt [Stand: 13.12.1999]

[FlPh 98] Eric Fleischmann, Philip A. Chou:
Advanced Streaming Format (ASF):

[Msof 98b] A universal container file Format for synchronized media:
Microsoft Corporation January 1998

[FlKl 98] Eric Fleischman, Anders Klemets:
Recording MBone Sessions to ASF Files:
http://www.microsoft.com/asf/resources/draft-fleischman-asf-record-00.txt [Stand: 28.11.1999]

[CFCC 98] Philip A. Chou, Eric Fleischman, Wie-ge Chen, Ming-Chieh Lee, Mark van Antwerp, Thomas Gadros, Donald Newwell:
The MPEG-4 Intermedia Format MIF as an Extension of ASF
http://www.microsoft.com/ASF/