



**Postgraduate Diploma
in
Strategic Business Information Technology**

**Module 4
Computer Networking and Management**

**Chapter 6
Multimedia Networking**

© NCC Education Limited, 2003

Modification History

Revision	Date	Revision Description
V1.0	January 2003	For issue

© NCC Education Limited, 2003

All Rights Reserved

The copyright in this document is vested in NCC Education Limited. The document must not be reproduced by any means, in whole or in part, or used for manufacturing purposes, except with the prior written permission of NCC Education Limited and then only on condition that this notice is included in any such reproduction.

Information contained in this document is believed to be accurate at the time of publication, but no liability whatsoever can be accepted by NCC Education Limited arising out of any use made of this information.

Trademarks

NCC Education acknowledge that the trademarks and registered trademarks of products mentioned in this material are held by the companies producing them. Use of a term in this material should not be regarded as affecting the validity of any trademark or service mark.

Copyright of any screen captures in this material are the property of the software's manufacturer.

This material may contain some clipart, which is copyright to the Corel Corporation.

Contents

Introduction – Multimedia Networking.....	5
Examples of Multimedia Applications.....	6
Streaming, Stored Audio and Video.....	6
Streaming of Live Audio and Video.....	7
Real-Time Interactive Audio and Video	7
Accessing Audio and Video from a Web Server.....	12
A Naive Implementation for Audio Streaming	13
Streaming from a Streaming Server	14
Real Time Streaming Protocol (RTSP).....	15
What RTSP Does Not Do.....	15
Summary.....	17

Introduction – Multimedia Networking

Visual 1 (NCC Course Title Visual)

Visual 2

Multimedia networking

<p><u>Chapter goals:</u></p> <ul style="list-style-type: none"> □ Understand service requirements for multimedia networking <ul style="list-style-type: none"> ○ delay ○ bandwidth ○ loss □ Learn about how to make the best of the best-effort Internet □ Learn about how the Internet might evolve to better support multimedia 	<p><u>Chapter overview:</u></p> <ul style="list-style-type: none"> □ Multimedia networking applications □ Streaming stored audio and video <ul style="list-style-type: none"> ○ RTSP □ Interactive real-time applications <ul style="list-style-type: none"> ○ Internet phone example □ RTP □ H.323 and SIP □ Beyond best effort <ul style="list-style-type: none"> ○ scheduling and policing ○ integrated services ○ differentiated services
--	---

In this chapter we consider networking applications whose data contains audio and video content. We refer to these applications as multimedia networking applications. Multimedia networking applications are typically highly sensitive to delay but are loss tolerant. After surveying and classifying different types of multimedia applications, we examine their deployment in a best-effort network, such as today's Internet.

Visual 3

Multimedia in networks (1)

<p><u>Fundamental characteristics:</u></p> <ul style="list-style-type: none"> □ Typically delay sensitive delay □ But loss tolerant: infrequent losses cause minor glitches that can be concealed □ Antithesis of data (programs, banking information, etc.) which are loss intolerant but delay tolerant □ Multimedia is also called 'continuous media' 	<p><u>Classes of MM applications:</u></p> <ul style="list-style-type: none"> □ Streaming stored audio and video □ Streaming live audio and video □ Real-time interactive video
--	---

The last few years have witnessed an explosive growth in the development and deployment of networked applications that transmit and receive audio and video content over the Internet. New multimedia networking applications (also referred to as continuous media applications) – entertainment video, IP telephony, Internet radio, multimedia

WWW sites, teleconferencing, interactive games, virtual worlds, distance learning, and much more – seem to be announced daily.

The service requirements of these applications differ significantly from those of traditional data-oriented applications such as the Web text/image, e-mail, FTP, and DNS applications. In particular, multimedia applications are highly sensitive to end-to-end delay and delay variation, but can tolerate occasional loss of data. These fundamentally different service requirements suggest that a network architecture that has been designed primarily for data communication may not be well suited for supporting multimedia applications.

Examples of Multimedia Applications

Visual 4

Multimedia in networks (2)

<p><u>Streaming stored MM</u></p> <ul style="list-style-type: none"> □ Clients request audio/video files from servers and pipeline reception over the network and display □ Interactive: user can control operation (similar to VCR: pause, resume, fast forward, rewind, etc.) □ Delay: from client request until display start can be 1 to 10 seconds 	<p><u>Unidirectional real-time</u></p> <ul style="list-style-type: none"> □ Similar to existing TV and radio stations, but delivery over the Internet □ Non-interactive, just listen/view <p><u>Interactive real-time</u></p> <ul style="list-style-type: none"> □ Phone or video conference □ More stringent delay requirement than streaming & unidirectional because of real-time nature □ Video: < 150 msec acceptable □ Audio: < 150 msec good, <400 msec acceptable
--	--

Streaming, Stored Audio and Video

In this class of applications, clients request on-demand compressed audio or video files that are stored on servers. Stored audio files might contain audio from a professor's lecture, rock songs, symphonies, archives of famous radio broadcasts, or archived historical recordings. Stored video files might contain video of a professor's lecture, full-length movies, pre-recorded television shows, documentaries, video archives of historical events, cartoons, or music video clips. There are three key distinguishing features of this class of application:

- **Stored media** – the multimedia content has been prerecorded and is stored at the server. As a result, a user may pause, rewind, fast-forward or index through the multimedia content. The time from when a client makes such a request until the action manifests itself at the client should be on the order of 1 to 10 seconds for acceptable responsiveness.
- **Streaming** – in most stored audio/video applications, a client begins playout of the audio/video a few seconds after it begins receiving the file from the server. This means that the client will be playing out audio/video from one location in the file while it is receiving later parts of the file from the server. This technique, known as streaming, avoids having to download the entire file (and incurring a potentially long delay) before beginning playout. There

are many streaming multimedia products, including RealPlayer from RealNetworks and Microsoft's Windows Media [Microsoft Windows Media 2000]. There are also applications such as Kazaa, however, that require an entire audio file to be downloaded before playout begins.

- **Continuous playout** – once play out of the multimedia begins, it should proceed according to the original timing of the recording. This places critical delay constraints on data delivery. Data must be received from the server in time for its play out at the client; otherwise, it is considered useless. The end-to-end delay constraints for streaming, stored media are typically less stringent than those for live, interactive applications such as Internet telephony and video conferencing.

Streaming of Live Audio and Video

This class of application is similar to traditional broadcast radio and television, except that transmission takes place over the Internet. These applications allow a user to receive a live radio or television transmission emitted from any corner of the world.

Since streaming live audio/video is not stored, a client cannot fast forward through the media. However, with local storage of received data, other interactive operations such as pausing and rewinding though live multimedia transmissions are possible in some applications. Live, broadcast-like applications often have many clients who are receiving the same audio/video program. Distribution of live audio/video to many receivers can be efficiently accomplished using the multicasting techniques. This type of distribution is more often accomplished through multiple separate unicast streams. As with streaming stored multimedia, continuous play out is required, although the timing constraints are less stringent than for live interactive applications. Delays of up to tens of seconds from when the user requests the delivery/playout of a live transmission to when playout begins can be tolerated.

Real-Time Interactive Audio and Video

This class of application allows people to use audio/video to communicate with each other in real time. Real-time interactive audio is often referred to as Internet phone, since, from the user's perspective, it is similar to traditional circuit-switched telephone service. Internet phone can potentially provide PBX, local, and long-distance telephone service at very low cost. It can also facilitate computer-telephone integration (CTI), group real-time communication, directory services, caller identification, caller filtering, and more.

There are many Internet telephone products currently available. With real-time interactive video, also called video conferencing, individuals communicate visually as well as orally. There are also many real-time interactive video products currently available for the Internet, including Microsoft's NetMeeting. Note that in a real-time interactive audio/video application, a user can speak or move at anytime.

For a conversation with interaction among multiple speakers, the delay from when a user speaks or moves until the action is manifested at the receiving hosts should be less than a few hundred milliseconds. For voice, delays smaller than 150 milliseconds are not perceived by a human listener, delays between 150 and 400 milliseconds can be acceptable, and delays exceeding 400 milliseconds can result in frustrating, if not completely unintelligible, voice conversations.

Visual 5

Multimedia in networks - challenges

- TCP/UDP/IP suite provides best-effort, no guarantees on delay or delay variation
 - streaming apps with initial delay of 5-10 seconds are now commonplace, but performance deteriorates if links are congested (transoceanic)
 - real-time interactive apps have rigid requirements for packet delay and **jitter**
 - **jitter** is the variability of packet delays within the same packet stream
- Design of multimedia apps would be easier if there were some 1st and 2nd class services
 - but in the public Internet, all packets receive equal service
 - packets containing real-time interactive audio and video stand in line, like everyone else
- Continued efforts to provide differentiated service

Today, the Internet's network-layer protocol provides a best-effort service to all the datagrams it carries. However, best-effort service does not make any promises whatsoever about the end-to-end delay for an individual packet. Nor does the service make any promises about the variation of packet delay within a packet stream. Because TCP and UDP run over IP, neither of these protocols can make any delay guarantees to invoking applications. Due to the lack of any special effort to deliver packets in a timely manner, it is an extremely challenging problem to develop successful multimedia networking applications for the Internet.

For example, streaming stored audio/video with user-interactivity delays of five-to-ten seconds is now commonplace in the Internet. Internet phone and real-time interactive video has, to date, been less successful than streaming stored audio/video. Indeed, real-time interactive voice and video impose rigid constraints on packet delay and packet jitter. Packet jitter is the variability of packet delays within the same packet stream. Real-time voice and video can work well in regions where bandwidth is plentiful, and hence delay and jitter are minimal. But quality can deteriorate to unacceptable levels as soon as the real-time voice or video packet stream hits a moderately congested link.

Visual 6

Multimedia in networks - making the best of best effort

To mitigate impact of 'best-effort' Internet

- Use UDP to avoid TCP and its slow-start phase...
- Buffer content at client and control playback to remedy jitter
- Timestamp packets, so that receiver knows when the packets should be played back
- Adapt compression level to available bandwidth
- Send redundant packets to mitigate the effects of packet loss

➔ We will discuss all these 'tricks'

The design of multimedia applications would certainly be more straightforward if there were some sort of first-class and second-class Internet services, whereby first-class packets are limited in number and receive priority service in router queues. Such a first-class service could be satisfactory for delay-sensitive applications.

All packets receive equal service; no packets, including delay-sensitive audio and video packets, receive special priority in the router queues. No matter how much money you have or how important you are, you must join the end of the line and wait your turn!

So for the time being we have to live with best-effort service. But given this constraint, we can make several design decisions and employ a few tricks to improve the user-perceived quality of a multimedia networking application. For example, we can send the audio and video over UDP, and thereby circumvent TCP's low throughput when TCP enters its slow-start phase. We can delay playback at the receiver by 100 msec or more in order to diminish the effects of network-induced jitter. We can timestamp packets at the sender so that the receiver knows when the packets should be played back. For stored audio/video we can prefetch data during playback when client storage and extra bandwidth is available. We can even send redundant information in order to mitigate the effects of network-induced packet loss.

Visual 7

[How should the Internet evolve to better support multimedia? \(1\)](#)

<p><u>Integrated services philosophy:</u></p> <ul style="list-style-type: none"> □ Change Internet protocols so that applications can reserve end-to-end bandwidth <ul style="list-style-type: none"> ○ need to deploy protocol that reserves bandwidth ○ must modify scheduling policies in routers to honour reservations ○ application must provide the network with a description of its traffic, and must further abide to this description □ Requires new, complex software in hosts and routers 	<p><u>Differentiated services philosophy:</u></p> <ul style="list-style-type: none"> □ Fewer changes to Internet infrastructure, yet provide 1st and 2nd class service □ Datagrams are marked □ User pays more to send/receive 1st class packets □ ISPs pay more to backbones to send/receive 1st class packets
---	--

Today there is a tremendous – and sometimes ferocious – debate about how the Internet should evolve in order to better accommodate multimedia traffic with its rigid timing constraints. At one extreme, some researchers argue that it isn't necessary to make any fundamental changes to best-effort service and the underlying Internet protocols. Instead, they argue that it is only necessary to add more bandwidth to the links along with network caching for stored information and multicast support for one-to-many real-time streaming. Opponents of this viewpoint argue that additional bandwidth can be costly, and that as soon as it is put in place it will be eaten up by new bandwidth-hungry applications for example, high-definition video on demand.

At the other extreme, some researchers argue that fundamental changes should be made to the Internet so that applications can explicitly reserve end-to-end bandwidth. These researchers feel, for example, that if a user wants to make an Internet phone call from host A to host B, then the user's Internet phone application should be able to explicitly reserve

bandwidth in each link along a route from host A to host B. However, allowing applications to make reservations and requiring the network to honour the reservations requires some big changes. First we need a protocol that, on the behalf of applications, reserves bandwidth from the senders to their receivers. Second, we must modify scheduling policies in the router queues so that bandwidth reservations can be honoured.

Visual 8

How should the Internet evolve to better support multimedia? (2)

<u>Laissez-faire philosophy</u>	<u>Virtual Private Networks (VPNs)</u>
<ul style="list-style-type: none"> <input type="checkbox"/> No reservations, no datagram marking <input type="checkbox"/> As demand increases, provision more bandwidth <input type="checkbox"/> Place stored content at edge of network: <ul style="list-style-type: none"> <input type="checkbox"/> ISPs and backbones add caches <input type="checkbox"/> content providers put content in CDN nodes <input type="checkbox"/> P2P: choose nearby peer with content 	<ul style="list-style-type: none"> <input type="checkbox"/> Reserve permanent blocks of bandwidth for enterprises <input type="checkbox"/> Routers distinguish VPN traffic using IP addresses <input type="checkbox"/> Routers use special scheduling policies to provide reserved bandwidth

With these new scheduling policies, not all packets get equal treatment; instead, those that reserve (and pay) more get more. Third, in order to honour reservations, the applications must give the network a description of the traffic that they intend to send into the network. The network must then police each application's traffic to make sure that it abides by the description. Finally, the network must have a means of determining whether it has sufficient available bandwidth to support any new reservation request. These mechanisms, when combined, require new and complex software in the hosts and routers as well as new types of services.

There is a camp between the two extremes – the so-called differentiated services camp. This camp wants to make relatively small changes at the network and transport layers, and introduce simple pricing and policing schemes at the edge of the network that is, at the interface between the user and the user's ISP. The idea is:

- to introduce a small number of classes – possibly just two;
- assign each datagram to one of the classes;
- give datagrams different levels of service according to their class in the router queues;
- charge users according to the class of packets that they are sending into the network.

Visual 9

<u>Streaming stored audio and video</u>	
Streaming stored media:	Media player:
<ul style="list-style-type: none"> <input type="checkbox"/> Audio/video file is stored in a server <input type="checkbox"/> Users request audio/video file on demand <input type="checkbox"/> Audio/video is rendered within, say, 10 s after request <input type="checkbox"/> Interactivity (pause, re-positioning, etc.) is allowed 	<ul style="list-style-type: none"> <input type="checkbox"/> Removes jitter <input type="checkbox"/> Decompresses <input type="checkbox"/> Error correction <input type="checkbox"/> Graphical user interface with controls for interactivity <input type="checkbox"/> Plug-ins may be used to embed the media player into the browser window

In audio/video streaming, clients request compressed audio/video files that are resident on servers. These servers can be ‘ordinary’ Web servers, or can be special streaming servers tailored for the audio/video streaming application. Upon client request, the server directs an audio/video file to the client by sending the file into a socket.

Both TCP and UDP socket connections are used in practice. Before sending the audio/video file into the network, the file is segmented, and the segments are typically encapsulated with special headers appropriate for audio/video traffic. The Real-time protocol (RTP) is a public-domain standard for encapsulating such segments. Once the client begins to receive the requested audio/video file, the client begins to render the file (typically) within a few seconds. Most existing products also provide for user interactivity, for example, pause/resume and temporal jumps within the audio/video file. This user interactivity also requires a protocol for client/server interaction. Real-time streaming protocol (RTSP), is a public-domain protocol for providing user interactivity.

Audio/video streaming is often requested by users through a Web client (that is, browser). But because audio/video playout is not integrated directly in today’s Web clients, a separate helper application is required for playing out the audio/video. The helper application is often called a media player, the most popular of which are currently RealNetworks’ Real Player and the Microsoft Windows Media Player. The media player performs several functions, including:

- **Decompression** – Audio/video is almost always compressed to save disk storage and network bandwidth. A media player must decompress the audio/video on the fly during playout.
- **Jitter removal** – Packet jitter is the variability of source-to-destination delays of packets within the same packet stream. Since audio and video must be played out with the same timing with which it was recorded, a receiver will buffer received packets for a short period of time to remove this jitter.
- **Error correction** – Due to unpredictable congestion in the Internet, a fraction of packets in the packet stream can be lost. If this fraction becomes too large, user-perceived audio/video quality becomes unacceptable. To this end, many streaming systems attempt to recover from losses by either (1)

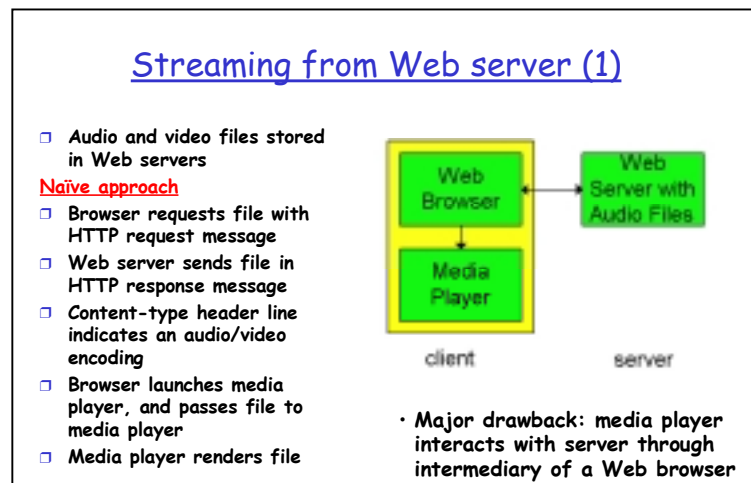
reconstructing lost packets through the transmission of redundant packets, (2) by having the client explicitly request retransmissions of lost packets, (3) masking loss by interpolating the missing data from the received data.

- **Graphical user interface with control knobs** – This is the actual interface that the user interacts with. It typically includes volume controls, pause/resume buttons, sliders for making temporal jumps in the audio/video stream, and so on.

Plug-ins may be used to embed the user interface of the media player within the window of the Web browser. For such embeddings, the browser reserves screen space on the current Web page, and it is up to the media player to manage the screen space. But either appearing in a separate window or within the browser window (as a plug-in), the media player is a program that is being executed separately from the browser.

Accessing Audio and Video from a Web Server

Visual 10

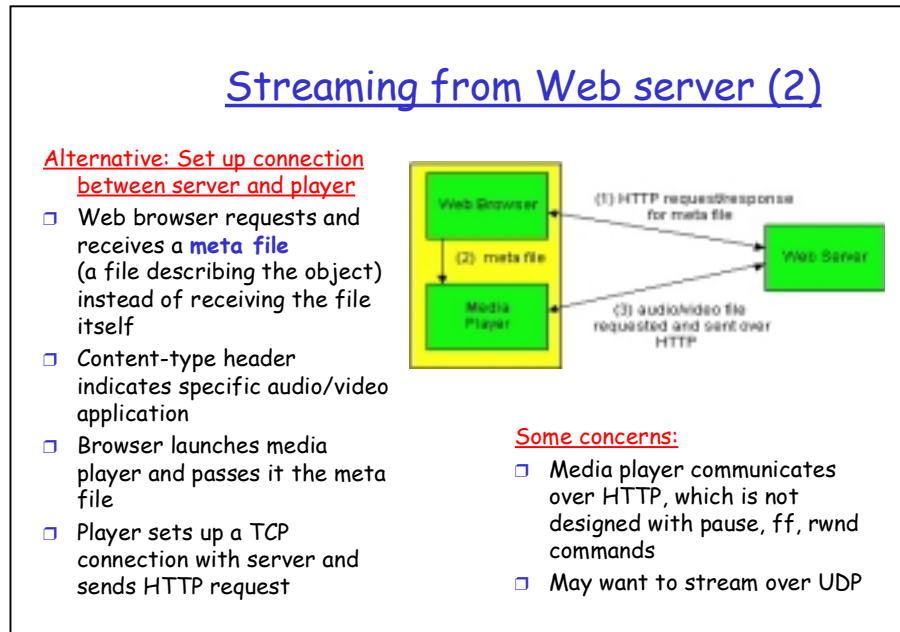


Stored audio/video can reside either on a Web server that delivers the audio/video to the client over HTTP, or on an audio/video streaming server that delivers the audio/video over non-HTTP protocols.

Consider first the case of audio streaming. When an audio file resides on a Web server, the audio file is an ordinary object in the server's file system, just as are HTML and JPEG files. When a user wants to hear the audio file, the user's host establishes a TCP connection with the Web server and sends an HTTP request for the object. Upon receiving a request, the Web server bundles the audio file in an HTTP response message and sends the response message back into the TCP connection. The case of video can be a little more tricky, because the audio and video parts of the 'video' may be stored in two different files, that is, they may be two different objects in the Web server's file system. In this case, two separate HTTP requests are sent to the server (over two separate TCP connections for HTTP/1.0), and the audio and video files arrive at the client in parallel. It is up to the client to manage the synchronization of the two streams. It is also possible that the audio and video are interleaved in the same file, so that only one object need be sent to

the client. To keep our discussion simple, for the case of ‘video’ we assume that the audio and video are contained in one file.

Visual 11



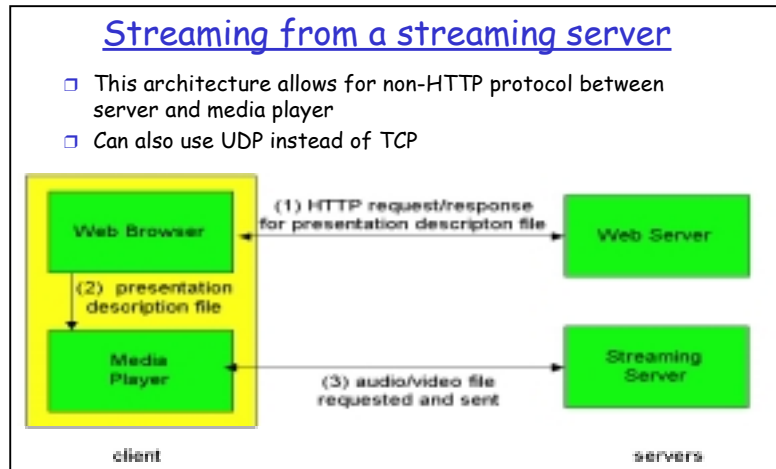
A Naive Implementation for Audio Streaming

1. The browser process establishes a TCP connection with the Web server and requests the audio/video file with an HTTP request message.
2. The Web server sends to the browser the audio/video file in an HTTP response message.
3. The content-type header line in the HTTP response message indicates a specific audio/video encoding. The client browser examines the content-type of the response message, launches the associated media player, and passes the file to the media player.
4. The media player then renders the audio/video file.

Although this approach is very simple, it has a major drawback: The media player (that is, the helper application) must interact with the server through the intermediary of a Web browser. This can lead to many problems. In particular, when the browser is an intermediary, the entire object must be downloaded before the browser passes the object to a helper application. The resulting delay before playout can begin is typically unacceptable for audio/video clips of moderate length. For this reason, audio/video streaming implementations typically have the server send the audio/video file directly to the media player process. In other words, a direct socket connection is made between the server process and the media player process.

Streaming from a Streaming Server

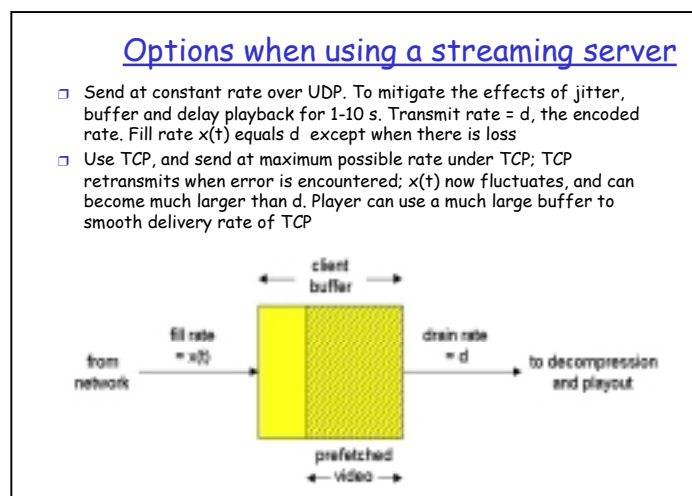
Visual 12



In order to get around HTTP and/or TCP, audio/video can be stored on and sent from a streaming server to the media player. This streaming server could be a proprietary streaming server, such as those marketed by RealNetworks and Microsoft, or could be a public-domain streaming server. With a streaming server, audio/video can be sent over UDP (rather than TCP) using application-layer protocols that may be better tailored than HTTP to audio/video streaming.

This architecture requires two servers. One server, the HTTP server, serves Web pages (including meta files). The second server, the streaming server, serves the audio/video files. The two servers can run on the same end system or on two distinct end systems. The steps for this architecture are similar to those described in the previous architecture. However, now the media player requests the file from a streaming server rather than from a Web server, and now the media player and streaming server can interact using their own protocols. These protocols can allow for rich user interaction with the audio/video stream.

Visual 13



There are many options for delivering the audio/video from the streaming server to the media player. A partial list of the options is given below:

- The audio/video is sent over UDP at a constant rate equal to the drain rate at the receiver (which is the encoded rate of the audio/video). For example, if the audio is compressed using GSM at a rate of 13 Kbps, then the server clocks out the compressed audio file at 13 Kbps. As soon as the client receives compressed audio/video from the network, it decompresses the audio/video and plays it back.
- This is the same as option 1, but the media player delays payout for 2-5 seconds in order to eliminate network-induced jitter. The client accomplishes this task by placing the compressed media that it receives from the network into a client buffer. Once the client has ‘prefetched’ a few seconds of the media, it begins to drain the buffer.

Real Time Streaming Protocol (RTSP)

Visual 14

Real Time Streaming Protocol (RTSP)

<p><u>HTTP</u></p> <ul style="list-style-type: none"> □ Designers of HTTP had fixed media in mind: HTML, images, applets, etc. □ HTTP does not target stored continuous media (i.e., audio, video, SMIL presentations, etc.) <p><u>RTSP: RFC 2326</u></p> <ul style="list-style-type: none"> □ Client-server application layer protocol. □ For user to control display: rewind, fast forward, pause, resume, repositioning, etc. 	<p><u>What it does not do:</u></p> <ul style="list-style-type: none"> □ Define how audio/video is encapsulated for streaming over network □ Restrict how streamed media is transported; it can be transported over UDP or TCP □ Specify how the media player buffers audio/video <p><u>RealNetworks</u></p> <ul style="list-style-type: none"> □ Server and player use RTSP to send control info to each other
--	--

Many Internet multimedia users (particularly those who grew up with a remote TV control in hand) will want to control the playback of continuous media by pausing playback, repositioning playback to a future or past point of time, visual fast-forwarding playback, visual rewinding playback, and so on. This functionality is similar to what a user has with a VCR when watching a video cassette or with a CD player when listening to a music CD. To allow a user to control playback, the media player and server need a protocol for exchanging playback control information. RTSP, defined in RFC 2326, is such a protocol.

What RTSP Does Not Do

RTSP does not:

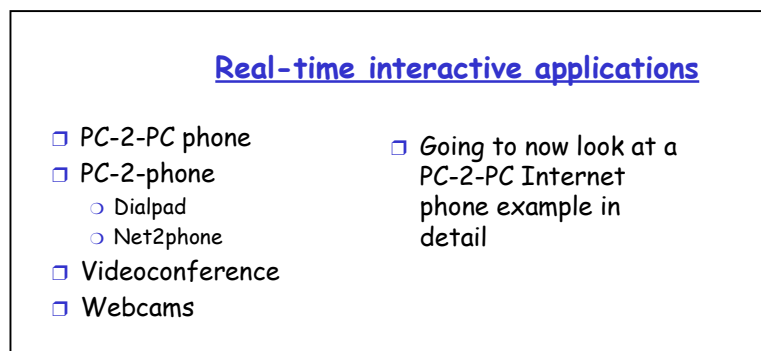
- define compression schemes for audio and video;
- define how audio and video is encapsulated in packets for transmission over a network; encapsulation for streaming media can be provided by RTP or by a proprietary protocol. For example, RealMedia’s G2 server and player use

RTSP to send control information to each other. But the media stream itself can be encapsulated in RTP packets or in some proprietary data format;

- restrict how streamed media is transported; it can be transported over UDP or TCP;
- restrict how the media player buffers the audio/video. The audio/video can be played out as soon as it begins to arrive at the client, it can be played out after a delay of a few seconds, or it can be downloaded in its entirety before playout.

So if RTSP does not do any of the above, what does RTSP do? RTSP is a protocol that allows a media player to control the transmission of a media stream. As mentioned above, control actions include pause/resume, repositioning of playback, fast forward and rewind. RTSP is a so-called out-of-band protocol. In particular, the RTSP messages are sent out-of-band, whereas the media stream, whose packet structure is not defined by RTSP, is considered ‘in-band.’ RTSP messages use a different port number, 544, than the media stream. The RTSP specification [RFC 2326] permits RTSP messages to be sent over either TCP or UDP.

Visual 15



The Internet’s network-layer protocol, IP, provides a best-effort service. That is to say that the Internet makes its best effort to move each datagram from source to destination as quickly as possible. However, best-effort service does not make any promises whatsoever on the extent of the end-to-end delay for an individual packet, or on the extent of packet jitter and packet loss within the packet stream.

Real-time interactive multimedia applications, such as Internet phone and real-time video conferencing, are acutely sensitive to packet delay, jitter, and loss. Fortunately, designers of these applications can introduce several useful mechanisms that can preserve good audio and video quality as long as delay, jitter, and loss are not excessive.

The speaker in our Internet phone application generates an audio signal consisting of alternating talk spurts and silent periods. In order to conserve bandwidth, our Internet phone application only generates packets during talk spurts. During a talk spurt the sender generates bytes at a rate of 8 Kbytes per second, and every 20 milliseconds the sender gathers bytes into chunks. Thus, the number of bytes in a chunk is $(20 \text{ msec}) - (8 \text{ Kbytes/sec}) = 160 \text{ bytes}$. A special header is attached to each chunk, the contents of

which is discussed below. The chunk and its header are encapsulated in a UDP segment, and then the UDP datagram is sent into the socket interface. Thus, during a talk spurt, a UDP segment is sent every 20 msec.

If each packet makes it to the receiver and has a small constant end-to-end delay, then packets arrive at the receiver periodically every 20 msec during a talk spurt. In these ideal conditions, the receiver can simply play back each chunk as soon as it arrives. But, unfortunately, some packets can be lost and most packets will not have the same end-to-end delay, even in a lightly congested Internet. For this reason, the receiver must take more care in (1) determining when to play back a chunk, and (2) determining what to do with a missing chunk.

Summary

Visual 16

Summary

- Multimedia networking - most exciting development in the Internet today
- Fewer people listening to radios and watching televisions - using the Internet to receive audio and video emissions
 - trend will continue
- Internet used to transport phone calls economically and also for numerous value-added services, such as video conferencing, online directory services, and voice messaging services

Multimedia networking is perhaps the most exciting development in the Internet today. People throughout the world are spending less time in front of their radios and televisions and are instead turning to the Internet to receive audio and video emissions, both live and pre-recorded. As high-speed access penetrates more residences, this trend will continue – couch potatoes throughout the world will access their favourite video programs through the Internet rather than through the traditional broadcast distribution channels. In addition to audio and video distribution, the Internet is also being used to transport phone calls. In fact, over the next 10 years the Internet may render the traditional circuit-switched telephone system nearly obsolete in many countries. The Internet will not only provide phone service for less money, but will also provide numerous value-added services, such as video conferencing, online directory services, and voice messaging services.

