Unit-3

Multimedia Networking

New multimedia networking applications (also referred to as continuous-media applications)-streaming video, IP telephony, Internet radio, teleconferencing, interactive games, virtual worlds, distance learning, and much more--seem to be announced daily).

Many multimedia applications are highly sensitive to end-to-end delay and delay variation but can tolerate occasional loss of data.

Multimedia Networking Applications

Two axes-timing consideration and tolerance of data loss-are particularly important for networked multimedia applications. Timing considerations are important because many multimedia applications are highly delay-sensitive.

Networked multimedia applications are for the nmst part lost-tolerant-occasional loss only causes occasional glitches in the audio/video playback, and these losses can often be partially or cully concealed. These delay-sensitive but loss-tolerant characteristics are clearly different from those of elastic applications such as the Web, e-mail, FTP, and Telnet. For elastic applications, long delays are annoying but not particularly harmful, and the completeness and integrity of the transferred data is of paramount importance.

Examples of Multimedia Applications

The Internet carries a large variety of exciting multimedia applications. Three broad classes of multimedia applications: streaming stored audio/video, streaming live audio/video, and real-time interactive audio/video.

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. Stored video files might contain video of a professor's lecture, full-length movies, prerecorded television shows, documentaries, video archives of historical events, cartoons, or music video clips.

This class of applications has three key distinguishing features.

Stored media - The multimedia content has been prerecorded and is stored at the server.

Streaming - In a streaming stored audio/video applications, a client typically begins playout of the audio/video a few seconds after it begins receiving the file froom the server. Streaming, avoids having to sownload the entire file before beginning playout.

Continuous playout - Once playout of the multimedia content begins, it should process according to the original timing of the recording.

Streaming Live Audio and Video

This class of applications is similar to traditional braodcast 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 cormer of the world.

Since streaming live audio/video is not stored, a client cannot fast-forward through the media. With local storage of received data, other interactive operations such as pausing and rewinding through live multimedia transmissions are possible in some applications. Live, broadcast-like applications often havy many clients who are receiving the same audio/video program. Distribution of live audio/video to many receivers can be efficiently accoomplished using the IP multicasting technique. At the time of this writing, live audio/video distribution is more often accomplished through multiple separate unicast streams. As with streaming stored multimedia, continuous playout is required, although the timing constraints are less stringent that for real-time interactive applications.

Real-Time Interactive Audio and Video

This class of applications allows people to use audio/video to communicate with each other in real time. Real-time interactive audio over the Internet is often referred to a Internet phone. Internet phone can potentially provide private branch exchange (PBX), local, and long-distance telephone service at very low cost. It can also facilitate the deployment of new services that are not easily supported by the traditional circuit-switched networks. With real-time interactive video, also called video-conferencing, individuals communicate visually as well as orally. In a real-time interactive audio/video application, a user can speak or move at any time.

Hurdles for Multimedia in Today's Internet

The IP protocol deployed in the Internet today provides a best-effort service to all the datagrams it carries. In other words, the Internet makes its best effort to move each datagram from sender to receiver as quickly as possible, but it does not make any promises whatsoever about 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, it follows that neither of these transport protocols makes and 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. Multimedia over the Internet has achieved significant but limited success.

Internet phone and real-time interactive video had been less successful than streaming stored audio/video. Real-time interactive vioce 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.

Streaming Stored Audio and Video

In audio/video streaming, clients request compressed audio/video files that reside on servers. These servers canbe ordinary Wed servers or can be special streaming servers tailored for the audio/video streaming application. The real-time protocol (RTP) is a public-domain standard for encapsulating such segments. The real-time streaming protocol (RTSP) is a public-domain protocol for providing user interactivity.

Audio/video playout is not integrated directly into today's Web clients, a separate helper application is required for playing out the audio/video. Helper applications are often called media players. The media player performs serveral functions, including the following:

Decompression - Audio/video is almost always compressed to save disk storage and network bandwidth.

Jitter removal - Packet jitter is the variability of source-to-destination delays of packets within the same packet stream.

Error correction - Due to unpredictable congestion in the Internet, a fraction fo packets in the packet stream can be lost.

The media player has a graphical user interface with control knobs. This is the actual interface that the user interacts with.

Accessing Audio and Video Through a Web Server

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.

A direct socket connections is made between the server process and the media player process. This is typically done by making use of a meta file, a file that provides information about the audio/video file that is to be streamed.

Sending Multimedia from a Streaming Server to a Helper Application

Audio/video can be sent over UDP using application-layer protocols that may be better tailored than HTTP to audio and video streaming.

This architecture requires two servers. One server, the HTTP server, serves Web pages. The second server, the streaming server, servers the audio/video files. The two servers can run on the same end system or on two distinct end systems. 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

The audio/video is sent over UDP at a constant rate equal to the drain rate at the receiver.

This is the same as the first option, but the media player delays playout delays for two to five seconds in order to eliminate network-induced jitter.

The media is sent over TCP. The server pushes the media file into the TCP socket as quickly as it can; the client reads from; the TCP socket as quickly as it can and places the compressed video into the media player buffer.

Real-Time Streaming Protocol (RTSP)

RTSP allows a media player to control the transmission of a media stream. Control actions include pause/resume, repositioning of playback, fast-forward, and rewind. RTSP is an out-of-band protocol. The RSTP messages are sent out-of-band, whereas the media stream, whose packet structure is not defined by RTSP, is considered "in-band." RTSP message use a different port number, 544, from the media stream.

The file transfer protocol (FTP) also uses the out-of-band notion. FTP uses tow client/server pairs of sockets, each pair that transports control information; the other client/server socket pair supports a TCP connection that actually transports the file. The RTSP channel is in many ways similar to FTP's control channel.

The Web browser first requests a presentation description file from a Web server. The presentation description file can have references to several continuous-media files as well as directives for synchronization of the continuous-media files. Each reference to a continuous-media file begins with the URL method, rtsp://.

The Web server encapsulates the presentation description file in an HTTP response message and sends the message to the browser. When the browser receives the HTTP response message, the browser invokes a media player based on the content-type field of the message. The presentation description file includes references to media streams, using the URL method rtsp://.

It is interesting to note the similarities between HTTP and RTSP. All request and response message are in ASCII text, the client employs standardized methods, and the server

responds with standardized reply codes. One important difference, however, is that the RTSP server keeps track of the state of the client for each ongoing RTSP session.

The client initiates the session with the SETUP request, providing the URL of the file to be streamed and the RTSP version. The setup message includes the client port number to which the media should be sent over UDP using the RTP packetization protocol.

RTSP has facilities that allows clients to stream toward the server.

Making the Best of the Best-Effort Service: An Internet Phone Example

The Internet's network-layer protocol, IP, provides a best-effort service. That is to say that the service makes it best effort to move each datagram from source to destination as quickly as possible. It does not make any promises whatsoever about the extent of the end-to-end delay fro an individual packet, or about the extent of packet jitter and packet loss within the packet stream. The lack of guarantees about delay and packet jitter poses significant challenges to the design of real-time multimedia applications.

The Limitations of a Best-Effort Service

Best-effort service can lead to packet loss, excessive end-to-end delay, and packet jitter.

Packet Loss

Consider one of the UDP segments generated by our Internet phone application. The UDP segment is encapsulated in an IP datagram. As the datagram wanders through the network, it passes through buffers in the routers in order to access outbound links. It is possible that one or more of the buffers in the route from sender to receiver is full and cannot admit the IP datagram. The IP datagram is discarded, never to arrive at the receiving application.

Loss could be eliminated by sending packets the packets over TCP rather than over UDP. Recall that TCP retransmits packets that do not arrive at the destination. Retransmission mechanisms are often considered unacceptable for interactive real-time audio applications such as Internet phone, because they increase end-to-end delay.

End-to-End Delay

End-to-end delay is the accumulation of transmission, processing, and queuing delays in routers; propagation delays in the links; and end-system processing delays.

Packet Jitter

A crucial component of end-to-end delay is the random queuing delays in the routers. Because of these varying delays within the network, the time from when a packet is

generated at the source until it is received at the receiver can fluctuate from packet to packet. This phenomenon is called jitter.

Jitter can often be removed by using sequence numbers, timestamps, and a playout delay.

Removing Jitter at the Receiver for Audio

For a voice application the receiver should attempt to provide synchronous playout of voice chunks in the presence of random network jitter. This is typically done by combining the following three mechanisms:

Prefacing each chunk with a sequence number.

Prefacing each chunk with a timestamp.

Delaying playout of chunks at the receiver.

These three mechanisms, when combined, can alleviate or even eliminate the effects of jitters. Examine two playback strategies: fixed play-out delay and adaptive play-out delay.

Fixed Playout Delay

With the fixed-delay strategy, the receiver attempts to play out each chunk exactly q msecs after the chunk is generated, If a chunk is timestamped at time t, the receiver plays out the chunk at time t+q. Packets that arrive after their scheduled playout times are discarded and considered lost.

Recovering from Packet Loss

Schemes that attempt to preserve acceptable audio quality in the presence of packet loss. Such schemes are called loss recovery schemes. Retransmitting lost packets is generally not appropriate in an interactive real-time application. Retransmitting a packet that has missed its playout deadline serves absolutely no purpose. And retransmitting a packet that overflowed a router queue cannot normally be accomplished quickly enough.

Two types of loss anticipation schemes are forwards error correction (FEC) and interleaving.

Forward Error Correction (FEC)

The basic idea of FEC is to add redundant information to the original packet stream. For the cost of marginally increasing the transmission rate of the audio of the stream, the

redundant information can be used to reconstruct approximations or exact versions of some of the lost packets.

Interleaving

Interleaving can significantly improve the perceived quality of an audio stream. It also has low overhead. The obvious disadvantage of interleaving is that it increases latency. This limits its use for interactive applications although it can perform well stored audio. A major advantage of interleaving is that it does not increase the bandwidth requirements of a stream.

Receiver-Based Repair of Damaged Audio Streams

Receiver-based recovery schemes attempt to produce a replacement for a lost packet that is similar to the original. This is possible since audio signals, and in particular speech, exhibit large amounts of short-term self-similarity. These techniques work for relatively small loss rates length of a phoneme these techniques break down, since whole phonemes may be missed by the listener.

Perhaps the simplest form of receiver-based recovery is packet repetition. Packet repetition replaces lost packets with copies of the packets that arrived immediately before the loss. It has low computational complexity and performs reasonably well. Another form of receiver-based recovery is interpolation, which uses audio before and after the loss to interpolate a suitable packet to cover the loss. Interpolation performs somewhat better than packet repetition but is significantly more computationally intensive.

Protocols for Real-Time Interactive Applications

Real-time interactive applications promise to drive much of the future Internet growth. It is therefore not surprising that standards bodies have been busy for many years at hammering out standards for this class of applications. With the appropriate standards in place fro real-time interactive applications, independent companies will be able to create new compelling products that interoperate with each other.

RTP

RTP can be used for transporting common formats for sound and MPEG and H.263 for video. It can also be used for transporting proprietary sound and video formats. RTP enjoys widespread implementation in hundreds of products and research prototypes. It is also complementary to other important real-time interactive protocols.

RTP Basics

RTP typically runs on top of UDP. The sending side encapsulates a media chunk within an RTP packet, then encapsulates the packet in a UDP segment, and then hands the segment to IP. The receiving side extracts the RTP packet from the UDP segment, then extracts the media chunk from the RTP packet, and then passes the chunk to the media player for decoding and rendering.

If an application incorporate RTP--instead of a proprietary scheme to provide payload type, sequence numbers, of timestamps--then the application will more easily interoperate with other networked multimedia applications.

It should be emphasized that RTP does not provide any mechanism to ensure timely delivery of data or provide other quality-of-service (QoS) guarantees; it does not even guarantee delivery of packets or prevent out-of-order delivery of packets. RTP encapsulation is seen only at the end systems. Routers do not distinguish between IP datagrams that carry RTP packets and IP datagrams that don't.

RTP allows each source to be assigned its own independent RTP stream of packets.

RTP packets are not limited to unicast applications. They can also be sent over one-to-many and many-to-many multicast trees. For a many-to-many multicast session, all of the session's senders and sources typically use the same multicast group for sending their RTP streams. RTP multicast streams belonging together belong to an RTP session.

RTP Packet Header Fields

The four main RTP packet header fields are the payload type, sequence number, timestamp, and the source identifier fields.

The payload-type field in the RTP packet is 7 bits long. For an audio stream, the payload-type field is used to indicate the type of audio encoding that is being used. If a sender decides to change the encoding in the middle of a session, the sender can inform the receiver of the change through this payload-type field. The sender may want to change the encoding in order to increase the audio quality or to decrease the RTP stream bit rate.

Important fields

- Sequence numbers-The sequence number field is 16 bits long. The sequence number increments by one for each RTP packet sent, and may be used by the receiver to detect packet loss and to restore packet sequence.
- Timestamp field-The timestamp field is 32 bits long. It reflects the sampling instant of the first byte in the RTP data packet.
- Synchronization source identifier (SSRC). The SRRC is 32 bits long. It identifies the source of the RTP stream. Each stream in an RTP session has a distinct SSRC. The SSRC is not the IP address of the sender, but instead is a number that the source assigns randomly when the new stream is started.

Developing Software Applications with RTP

- There are two approaches to developing an RTP-based networked application. The first approach is for the application developer to incorporate RTP by hand to write the code that performs RTP encapsulation at the sender side and RTP unravelling at the receiver side. The second approach is for the application developer to use existing RTP libraries and Java classes which performs the encapsulation and unravelling for the application.

- The UDP API requires the sending process to set, for each UDP segment it sends, the destination IP address and the destination port number before popping the packet into the UDP socket. The UDP segment will then wander through the Internet and will eventually arrive at the door of the receiving process for the application. This door is fully addressed by the destination IP address and the destination port number.

- An alternative approach is to use a Java RTP class to implement the RTP operations. With this approach the application developer is given the impression that RTP is part of the transport layer, with an RTP/UDP API between the application layer and the transport layer.

RTP Control Protocol (RTCP)

RFC 1889 also specifies RTCP, a protocol that a networked multimedia application can use in conjunction with RTP. RTCP packets are transmitted by each participant in an RTP session to all other participants in the session using IP multicast. There is aq single multicast address and all RTP and RTCP packets belonging to the session use the multicast address. RTP and RTCP packets are distinguished from each other through the use of distinct port numbers.

RTCP packets do not encapsulate chunks of audio or video. Instead, RTCP packets are sent periodically and contain sender and/or receiver reports that announce statistics that can be useful to the application. These statistics include number of packets sent, number of packets lost, and interarrival jitter. The RTP does not dictate what the application should do with this feedback information; this is up to the application developer.

RTCP Packet Types

For each RTP stream that a receiver receives as part of a session, the receiver generates a reception report. The receiver aggregates its reception reports into a single RTCP packet. The packet is then sent into the multicast tree that connects all the session's participants. The reception reports include several fields, the most important of which are listed below.

- The SSRC of the RTP stream for which the reception report is being generated.
- The fraction of packets lost within the RTP stream. THe last sequence number received in the stream of RTP packets.
- The interarrival jitter, which is a smoothed estimate of the variation in the interarrival time between successive packets in the RTP stream.

For each RTP stream that a sender is transmitting, the sender creates and transmitts RTCP sender report packets. These packets include information about the RTP stream, including:

- The SSRC of the RTP stream
- The timestamp and wall clock time of the most recently generated RTP packet in the stream.
- The number of packets sent in the stream.
- The number of bytes sent in the stream.

RTCP Bandwidth Scaling

The number amount of RTP traffic sent into the multicast tree does not change as the number of receivers increases, whereas the amount of RTCP traffic grows linearly with the number of receivers. To solve this scaling problem. RTCP modifies the rate at which a participane sends RTCP packets into the multicast tree as a function of the number of participants in the session. Since each participant sends control packets to everyone else, each participant can setimate the total number of participants in the session. RTCP attempts to limit its traffic to 5 percent of the session bandwidth.

SIP

SIP is a lightweight protocol that does the following:

- It provides mechanisms for establishing calls between a caller and a callee over an IP network.
- It provieds mechanisms for the caller to derermine the current IP address of the callee.
- It provides mechanisms for call management, such as adding new media streams during the call, changing the encoding during the call, inviting new participants during the call, call transfer, and call holding.

SIP Address

Many--if not most--SIP addresses to resemble e-mail addresses. An intersting feature of SIP addresses is that they can be included in Web pages, just as people's e-mail addresses are included in Web pages with mailto URL.

H.323

As an alternative to SIP, H.323 is a popular standard for real-time audio and video conferencing among end systems on the Internet. The H.323 gatekeeper is a device similar to an SIP register.

The H.323 standard is an umbrella specification that includes the following specifications:

- A specification for how endpoints negotiate common audio/video encodings.
- A specification for how audio and video chunks is encapsulated and sent over the network.
- A specification for how endpoints communicate with their respective gatekeepers.
- A specification for how Internet phones communicates through a gateway with ordinary phones in the public circuit-switched telephone network.

Each H.323 endpoint must support the G.711 speech compression standard. G.711 uses PCM to generate digitized speech at either 56 kbps or 64 kbps. Although H.323 requires every endpoint to be voice capable, video capabilities are optional.

H.323 is a comprehensive umbrella standard, which, in addition to the standards and protocols mandates a H.245 control protocol, a Q.931 signalling channel, and anRAS protocol for registration with the gatekeeper.

Some of the most important differences between H.323 and SIP:

- H.323 is a complete, vertically integrated suite of protocols for multimedia conferencing: signalling, registration, admission control, transport, and codes.
- SIP addresses only session initiation and management and is a single component.
- H.323 comes from the ITU, whereas SIP comes from the IETF and borrows many concepts from the Web, DNS, and Internet e-mail.
- H.323 is large and complex. Sip uses the KISS principle: keep it simple, stupid.