28

RTP, RTCP, and RTSP — Internet Protocols for Real-Time Multimedia Communication

Arjan Durresi
Louisiana State University

Raj Jain
Nayna Networks, Inc.

28.1 Multimedia over the Internet

The Internet was originally designed to support data communications. Most of the traffic initially included data such as e-mails and files. Voice traffic was served exclusively by telephone networks. As the Internet grew in terms of number of nodes, applications, and users, the need for multimedia communication over the Internet emerged. Animation, voice, and video clips are now common on the Internet. Multimedia networking products like Internet telephony, Internet TV, and video conferencing are available in the market. In the near future, people will enjoy other multimedia products in distance learning, distributed simulation, distributed work groups, and other areas. Multimedia traffic poses new requirements to the Internet architecture and its protocols. In response to this, researchers have designed a family of protocols, including Real-Time Transmission Protocol (RTP), its control part Real-Time Transmission Control Protocol (RTCP), and Real-Time Streaming Protocol (RTSP), that are the object of this chapter.

For transporting textual data, the best-effort service model of the IP-based Internet was shown to have been an adequate solution. In this service model, IP makes every effort to deliver packets, but does not provide guarantees. Thus, packets can be lost or, delivered out of order, and the delay is unpredictable. The advantage of such a model is that it keeps routers as simple as possible and keeps the network architecture very scalable — features that have been crucial for the spectacular success of the Internet. To increase the

0-8493-1985-4/05/$0.00+$1.50

reliability of the end-to-end services, data packets are transported by TCP, which is one of the most common transport layers used in IP networks. TCP has mechanisms to conceal the unreliability of IP and presents a reliable network channel to applications. For example, TCP retransmits lost packets and keeps them in order. TCP also has mechanisms to keep network congestion under control and to avoid congestion collapse. Hence, TCP is an optimal solution for data traffic. IP, which is implemented in all nodes, is simple and scalable, and TCP, which is implemented only at the end-user nodes, provides the needed reliability for the applications. This solution, however, has problems when attempting to transport multimedia traffic. Many multimedia applications are highly sensitive to end-to-end delay and delay variation (jitter); but can tolerate some data loss. TCP takes care of losses by retransmitting the lost packets, but does not guarantee the delay or its variation. That is why multimedia traffic does not work well with TCP. For example, if some packets carrying part of a video scene are lost, having them retransmitted by TCP could worsen the situation. Such retransmitted video traffic could overlap over a new video scene and may also damage it. To avoid such problems with TCP, most of the multimedia traffic is transported by UDP to make use of its multiplexing and checksum services. UDP is the other transport layer over IP.

Multimedia applications can be divided in two classes: real-time interactive applications and noninteractive streaming applications.

In real-time applications such as voice over the Internet and video conferencing, all communication has to be in real time and the data cannot be buffered for long to smooth out network delay or jitter. There are hundreds of Internet interactive multimedia products currently available such as Microsoft NetMeeting, PC-to-phone and PC-to-PC, H-323 based applications, and MBone tools [1]. Such applications are very delay and jitter sensitive. For example, voice application cannot accept delays more than 250–300 ms and jitter more than 75 ms; otherwise, the conversation becomes almost unintelligible. Some new types of applications are emerging in this category, such as Distributed Virtual Environments (DVE) for collaboration, scientific exploration, instrument operation, and data visualization by supporting continuous media flows, data transfers, and data manipulation tools within virtual environment interfaces. Other specialized equipment, such as haptic devices or head-mounted displays, is used to enable manipulation of instruments or navigation within virtual worlds. These collaborative distributed immersive environments pose new requirements to the network in terms of data transfer bit-rates, short one-way delay for real-time operation, and reliable transfer of data. One example of DVE application is a system developed by the National Tele-immersion Initiative, which uses 3D real-time acquisition to capture accurate dynamic models of humans. The system combines these models with a static 3D background (3D model of an office, e.g.), and it also adds synthetic 3D graphic objects that may not exist at all. These objects may be used as a basis for a collaborative design process that users immersed in the system can work on. The system can also be described as a mix of virtual reality and 3D videoconferencing, and can be used in tele-diagnosis and tele-medicine.

Noninteractive applications class can be divided into two subclasses: streaming stored audio/video and streaming live audio/video. Streaming stored applications deliver audio and video streams stored in a server to clients. Examples of such stored files include lectures, songs, movies, and video clips. In streaming stored applications, the user begins to play out of the file, while the file is being received. The main requirement of such applications is that once the playout begins, it should proceed according to the original timing of the recording. This is translated in a requirement for the end-to-end network delay; but such a requirement is less stringent compared to interactive applications. Because the playout is not in real time, the file can be buffered to smooth out the effects of random network delay. Examples of streaming stored applications include RealPlayer [2], Apple's Quick Time [3], and Microsoft Windows Media [4]. Streaming live audio and video applications are similar to radio and television broadcasts. Even though the files are not stored in servers, they can be stored locally at clients. In the streaming stored applications, local buffering allows reducing the network jitter and delay compared to interactive applications.

Among the multimedia requirements discussed so far, the requirements for delay and jitter from interactive, real-time applications are the most stringent. The best-effort service of today's Internet cannot satisfy these requirements, and this is one of the reasons why multimedia applications are not yet the new killer applications on the Internet. There is a vivid debate among researchers about how to satisfy such multimedia requirements [5]. Some researchers argue that the Internet's service model should be changed to allow resource reservations, which would enable predictable delays and jitter. A large amount of research work is dedicated to such solutions and protocols, such as RSVP (part of IntServ), that signal the needed resource reservations. The main problem with this approach is that it increases the complexity of protocols and routers, which could lead to scalability issues. Other researchers propose to use more bandwidth (overprovisioning) with the best-effort service. This is still an open debate but for the time being, the proposal to use reservations or other QoS solutions is not being applied and multimedia applications use the best-effort service of the Internet.

Besides the limited delay and jitter, multimedia applications need some transport services with characteristics different from those of TCP, and with more functionality than UDP. The protocol designed to provide services for data with real-time characteristics is called Real-time Transport Protocol (RTP). In the following, we discuss some common needs of real-time applications considered in the design of RTP.

The large variety of multimedia applications of both classes use different coding schemes, each with its own trade-offs between quality, bandwidth, and computational cost. One of the basic requirements such applications have is the ability to interoperate with each other. For example, it should be possible for two independently developed Voice over IP applications to talk to each other. Instead of imposing a given voice-encoding scheme, RTP enables these applications to negotiate and select a scheme that is available to both parties. The same is true for video applications.

In the best-effort service provided by the Internet, the delay in routers is random and the end-to-end delays fluctuate; this phenomenon is called jitter. Jitter can change the original spacing between multimedia packets. If the receiver ignores the presence of jitter and plays the voice or video as soon as they arrive, the result could be unintelligible. Fortunately, jitter can be smoothed out by using a playback buffer, where the arriving packets are stored before being played. In order to play these packets at the right time, the receiver needs to know the timing relationships among the received packets. RTP enables the recipient of the packet stream to determine this information.

Related to the media timing is the need of multimedia applications for synchronization among various media, for example, between voice and video streams in a conference. These streams could have originated at one or more sources. Another important requirement of multimedia applications is to identify packet loss. Then, the specific application can deal in an appropriate manner with such losses.

The output data rate of multimedia applications depends on the information content. Because such applications do not use TCP, they are not congestion sensitive. Therefore, they do not reduce their data rate in response to network congestion, leading to packet loss and more congestion. Yet, many multimedia applications are capable of responding to congestion, for example, by changing parameters in coding schemes to reduce the consumed bandwidth during congestion periods. In order for this to occur, the sender needs the congestion feedback from receivers.

Another requirement for multimedia applications is the concept of frame indication at the application level. For example, it is important to notify a receiver video application that a set of packets belong to the same video scene, so that they can be treated in an appropriate way.

Based on the experience with existing multimedia products over the Internet, researchers have designed RTP.

28.2 RTP

RTP is the Internet-standard protocol for the transport of real-time data, including audio and video [6, 7]. It can be used for media-on-demand as well as interactive services such as Internet telephony. RTP was developed by the Internet Engineering Task Force (IETF) and is in widespread use. The RTP standard actually defines a pair of protocols: RTP and RTCP. RTP is used for the exchange of multimedia data, while RTCP is the control part and is used to periodically obtain feedback control information regarding the quality of transmission associated with the data flows. RTP usually runs over UDP/IP; but efforts are under way to make it transport-independent so that it could be used over other protocols. RTP and the associated RTCP use consecutive transport-layer ports, when used over UDP. Figure 28.1 shows a schematic diagram illustrating the use of RTP. The video/audio is digitized using a particular codec. The blocks formed by such bit streams are encapsulated in RTP packets and then in UDP and IP packets.

The data part of RTP is a thin protocol providing support for applications with real-time properties such as continuous media (e.g., audio and video), including timing reconstruction, loss detection, security, and content identification. RTP does not reserve bandwidth or guarantee Quality of Service (QoS).

The control part of the protocol, RTCP, provides support for real-time conferencing of groups of any size within the Internet. It offers QoS feedback from receivers to the multicast group as well as support for the synchronization of different media streams. RTCP also conveys information about participants in a group session.

Attempts to send voiceover networks began in the early 1970s. Several patents on packet transmission of speech, time stamp, and sequence numbering were granted in the 1970s and 1980s. In 1991, a series of voice experiments were completed on DARTnet. In August 1991, the Network Research Group of Lawrence Berkeley National Laboratory released an audio conference tool **vat** for DARTnet use. The protocol used was referred later as RTP version 0.

In December 1992, Henning Schulzrinne, then at GMD Berlin, published RTP version 1. It underwent several states of Internet Drafts, and was finally approved as a Proposed Standard on November 22, 1995 by the IESG. This version was called RTP version 2 and was published as

- RFC 1889, updated in July 2003 by RFC 3550 RTP: A Transport Protocol for Real-Time Applications.
- RFC 1890, updated in July 2003 by RFC 3551 RTP Profile for Audio and Video Conferences with Minimal Control.

On January 31, 1996, Netscape announced "Netscape LiveMedia" based on RTP and other standards. Microsoft NetMeeting conferencing software also supports RTP.

The design of RTP was made following an architectural principle known as Application Level Framing (ALF). This principle was proposed by Clark and Tennenhouse in 1990 [8] as a new way to design protocols for emerging multimedia applications. ALF is based on the belief that applications understand their own needs better, which means that the intelligence should be placed in applications and the network should be kept simple. For example, an MPEG video application knows better how to recover from lost frames, and how to react to it if an I or a B frame is lost.

RTP is designed to support a wide variety of applications. It provides flexible mechanisms by which new applications can be developed without repeatedly revising RTP itself. For each class of applications, RTP defines a profile and one or more payload formats. The profile provides a range of information that ensures a common understanding of the fields in the RTP header for that application class. A profile can

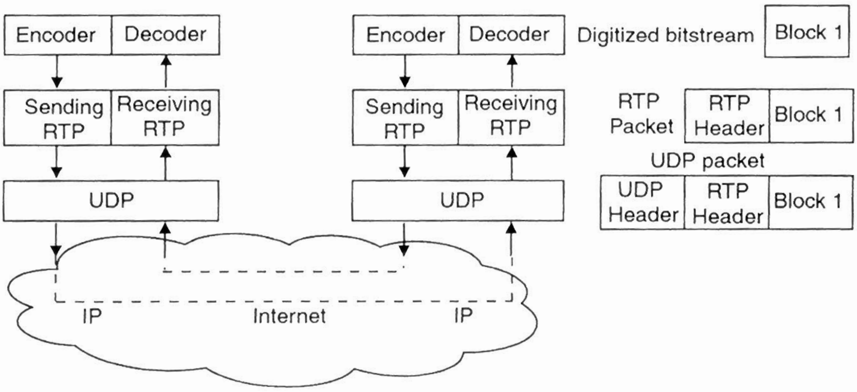

FIGURE 28.1 Real-time Transport Protocol.

also define extensions or modifications to RTP that are specific to a particular class of applications. The payload format specification explains how the data that follow the RTP header are to be interpreted.

Mixers and Translators

Besides the sender and receiver, RTP defines two other roles for systems, those of a translator and a mixer. They reside in between senders and receivers, and process the RTP packets as they pass through. The translators are used to translate from one payload to another. For example, suppose some of the participants in the audio–video conference have lower access bandwidth than that required for the conference. In this case, a translator could convert the stream to audio and video formats that require less bandwidth.

Mixers, on the other hand, are used to combine multiple source streams into one. An example is a video mixer that combines the images of individual people into one video stream to simulate a group scene.

Header Format

Figure 28.2 shows the header format used by RTP. The first 12 bytes are always present, whereas the contributing source identifiers are only used in certain circumstances. After this header, there may be optional header extensions. Finally, the RTP payload, whose format is determined by the application, follows the header. This header contains only the fields that are likely to be used by most of the multimedia applications. Information specific to a single application is carried in the RTP payload.

V — version: *2 bits*: The first two bits are a version identifier, which contains the value 2 for the RTP version used at the time of this writing.

P — padding: *1 bit*: If the padding bit is set, the packet contains one or more additional padding octets at the end, which are not part of the payload. The last octet of the padding contains a count of how many padding octets should be ignored, including itself. RTP data might be padded to fill up a block of certain size as required by an encryption algorithm.

X — extension: *1 bit*: If the extension bit is set, exactly one extension header follows the fixed header.

CC — CSRC count: *4 bits*: The CSRC count contains the number of CSRC identifiers that follow the fixed header. This number is more than one if the payload of the RTP packet contains data from several sources.

M — marker: *1 bit*: The interpretation of the marker is defined by a profile. The marker is intended to allow significant events such as frame boundaries to be marked in the packet stream.

PT — payload type: *7 bits*: PT identifies the format of the RTP payload and determines its interpretation by the application. One possible use of this field would be to enable an application to switch from one coding scheme to another based on the information about resource availability on the network of feedback on application quality. Default payload types are defined in RFC 3551, as shown in Table 28.1.

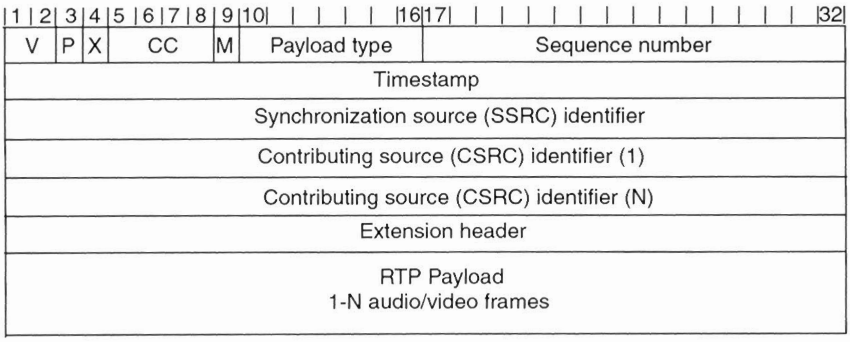
1
2
3
4
5
6
7
8
9
10
16
17
32
V
P
X
CC
M
Payload type
Sequence number
Timestamp
Synchronization source (SSRC) identifier
Contributing source (CSRC) identifier (1)
Contributing source (CSRC) identifier (N)
Extension header
RTP Payload
1-N audio/video frames

FIGURE 28.2 RTP header format.

Example specifications include PCM, MPEG1/MPEG2 audio and video, JPEG video, Sun CelB video, H.261 video streams, etc. More payload types can be added by providing a profile and payload format specification. The exact usages of marker bit and the payload type are determined by the application profile.

Sequence number: 16 bits: The sequence number increments by one for each RTP data packet sent, and can be used by the receiver to detect missing and misplaced packets. The initial value is randomly set to make crypto analysis attacks on possible encryption more difficult. RTP does not take any action when it detects a lost packet. It is left to the application to decide what to do when a packet is lost. For example, a video application could replay the previous frame if a frame is lost. Another application, however, could decide to change encoding parameter in order to reduce the needed bandwidth. These decisions need some intelligence, which is left to the applications themselves.

Timestamp: 32 bits: The timestamp indicates the sampling instant of the first octet in the RTP data packet. Its function is to enable the receiver to play back samples at the appropriate intervals and to enable different media streams to be synchronized. It can also be used for calculations in jitter smoothing. The resolution of the clock used should be sufficient for the desired synchronization accuracy and for measuring packet jitter.

The initial value is randomly set. Because different applications may require different granularities of timing, RTP does not specify the units in which time is measured. The timestamp is just a counter of ticks, and the time between ticks is application specific. The clock granularity is specified in the RTP profile or payload format for an application.

SSRC — Synchronization source: 32 bits: This field identifies synchronization sources within the same RTP session. It is chosen randomly to avoid two sources in the same RTP session to have the same SSRC identifier. It indicates where the data were combined, or the source of the data if there is only one source. The sources could be in the same node or in different nodes, such as during a conference.

CSRC — Contributing source list: 0 to 15 items, 32 bits each: The CSRC list identifies the contributing sources for the payload contained in this packet. The CC field gives the number of identifiers. It is used

TABLE 28.1 Payload Types (PT) for Some Audio and Video Encodings

PT	Encoding name	Media type	Clock rate [kHz]
0	PCMU	Audio	8
3	GSM	Audio	8
4	G723	Audio	8
5	DVI4	Audio	8
6	DVI4	Audio	16
7	LPC	Audio	8
8	PCMA	Audio	8
9	G722	Audio	8
10	L16	Audio	44.1
11	L16	Audio	44.1
12	QCELP	Audio	8
13	CN	Audio	8
14	MPA	Audio	90
15	G728	Audio	8
16	DVI4	Audio	11.025
17	DVI4	Audio	22.05
18	G729	Audio	8
25	CelB	Video	90
26	JPEG	Video	90
31	H261	Video	90
32	MPV	Video	90
33	MP2T	Audio/Video	90
34	H263	Video	90

only when a number of RTP streams pass through a mixer. A mixer can be used, for example, in a conference to combine data received from many sources and sending it as a single stream to reduce the needed bandwidth.

To set up an RTP session, the application defines a particular pair of destination transport addresses (one network address plus a pair of ports for RTP and RTCP). In a multimedia session, each medium is carried in a separate RTP session, with its own RTCP packets reporting the reception quality for that session. For example, audio and video would travel on separate RTP sessions, enabling a receiver to select whether or not to receive a particular medium. An audio–video conferencing scenario presented below illustrates the use of RTP.

Example of an Audio and Video Conference

Audio and video media, used in the conference, are transmitted on separate RTP sessions. That is, separate RTP and RTCP packets are transmitted for each medium using two different UDP port pairs. In the pair of ports used for audio, one port is used for data and the other for RTCP. The same is true for other pair of ports used for video pair. The conference can also use Internet multicast service. In this case, two multicast group addresses are needed: one for voice and one for video. There is no direct coupling at the RTP level between the audio and the video sessions, except that a user participating in both sessions should use the same canonical name in the RTCP packets for both so that the sessions can be associated. This separation between audio and video allows users in the conference to receive only audio. The synchronization playback of a source's audio and video can be achieved using timing information carried in RTCP packets for both sessions.

Both audio and video applications used by each participant send chunks of data. Each of these chunks is carried in RTP packets. Each packet has an RTP header that indicates what type of audio or video encoding is contained in the packet. This information is used by the receiver to select the appropriate decoding scheme.

The RTP header contains timing information and a sequence number that allow the receivers to reconstruct the timing produced by the source for both audio and video. The timing reconstruction is done separately for each source of RTP packets in the conference.

Each participants' audio and video application periodically multicasts a reception report plus the identification information name of its user on the RTCP port. The reception report indicates the quality of reception of the current source and could be used to control the adaptive encodings. A participant sends an RTCP BYE packet when it leaves the conference.

28.3 Real-time Transport Control Protocol — RTCP

RTCP is the control protocol designed to work in conjunction with RTP. It is specified in RFC 3550. RTCP's relationship with other layers is shown in Figure 28.3.

In an RTP session, participants periodically send RTCP packets to all the members in the same RTP session using IP multicast. RTCP packets contain sender and/or receiver reports that announce statistics such as the number of packets sent, number of packets lost, and inter-arrival jitter. This function may be useful for adaptive applications that can use this feedback to send high- or low-quality data depending on the network congestion. Such applications will increase the compression ratio when there is little available bandwidth, and will reduce the compression ratio, which will result in higher multimedia quality when there is more available bandwidth. This feedback information can also be used for diagnostic purposes to localize eventual problems.

RTCP provides a way to correlate and synchronize different media streams that have come from the same sender. When collisions of SSRC occur, it is necessary to change the SSCR value of a given stream. This is done using RTCP.

In applications that involve separate multimedia streams, a common system clock is used for their synchronization. The system that initiates the session provides this function, and the RTCP messages enable all systems to use the same clock.

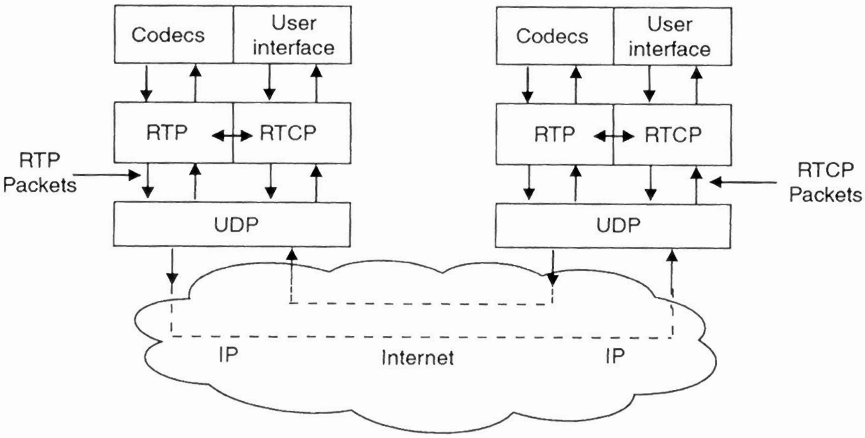

FIGURE 28.3 Real-time Transport Control Protocol.

RTCP is also used to deliver information about the membership in the session.

RFC 3550 defines five RTCP packet types to carry control information. These five types are as follows:

RR (Receiver Report): Receiver reports are generated by participants that are not active senders. They contain reception quality feedback about data delivery, including the highest packets number received, the number of packets lost, interarrival jitter, and timestamps to calculate the round-trip delay between the sender and the receiver.

SR (Sender Report): Sender reports are generated by active senders. In addition to the reception quality feedback as in RR, they contain a sender information section, providing information on inter-media synchronization, cumulative packet counters, and number of bytes sent.

SDES (Source Description Items): They contain information to describe the sources. In RTP data packets, sources are identified by randomly generated 32-bit identifiers. These identifiers are not convenient for human users. RTCP SDES (source description) packets contain textual information called *canonical names* as globally unique identifiers of the session participants. It may include the user's name, telephone number, e-mail address, and other information.

BYE: Indicates end of participation.

APP (Application specific functions): These are intended for experimental use as new applications and new features are developed.

RTCP packets are sent periodically among participants. When the number of participants increases, it is necessary to balance between getting up-to-date control information and limiting the control traffic. In order to scale up to large multicast groups, RTCP has to prevent the control traffic from overwhelming network resources. RTCP limits the control traffic to at most 5% of the overall session traffic. This is enforced by adjusting the RTCP packet generation rate according to the number of participants.

RTP Implementation Resources

RTP is an open protocol that does not provide preimplemented system calls. Implementation is tightly coupled to the application itself. Application developers have to add the complete functionality in the application layer by themselves. It is, however, always more efficient to share and reuse code rather than starting from scratch. The RFC 3550 specification itself contains numerous code segments that can be used directly in the applications. There are some implementations with source code available on the web for evaluation and educational purposes. Many modules in the source code can be used with minor modifications. The following is a list of useful resources:

vat (http://www-nrg.ee.lbl.gov/vat/),
rtptools (ftp://ftp.cs.columbia.edu/pub/schulzrinne/rtptools/),
NeVoT (http://www.cs.columbia.edu/~hgs/rtp/nevot.html).

RTP Library (http://www.bell-labs.com/project/RTPlib/DOCS/rtp_api.html) provides a high-level interface for developing applications that make use of the RTP.

RTP page (http://www/cs.columbia.edu/~hgs/rtp) maintained by Henning Schulzrinne is a very complete reference. Readers might also want to visit the Free Phone site [9], which documents an Internet phone application that uses RTP.

28.4 RTSP

RTSP, defined in RFC 2326, is an application-level *r* protocol that enables control over the delivery of data with real-time properties over IP. Such control includes pausing playback, repositioning playback to future or past point of time, fast forwarding, and rewinding playback. These functionalities are similar to those of a DVD player. RTSP does not typically deliver the continuous media itself, although interleaving of the continuous media stream with the control stream is possible. In the words of the authors: "RTSP acts as a network remote control for multimedia servers"[10]. Sources of data include both live data feeds and stored clips.

RTSP is a client–server multimedia presentation protocol. There is no notion of RTSP connection. Instead, a server maintains a session labeled by an identifier. An RTSP session is in no way tied to a transport-level protocol. During an RTSP session, an RTSP client may open and close many reliable transport connections to the server in order to issue RTSP requests. It may alternatively use a connectionless transport protocol such as UDP.

RTSP is designed to work with lower-level protocols like RTP and RSVP to provide a complete streaming service over the Internet. It provides a means for choosing delivery channels (such as UDP, multicast UDP, and TCP), and delivery mechanisms based on RTP. The RTSP messages are sent out-of-band of the media stream. RTSP works for large-audience multicast as well as single-viewer unicast.

RTSP was jointly developed by RealNetworks, Netscape Communications, and Columbia University. It was developed from the streaming practice and experience of RealNetworks' RealAudio and Netscape's LiveMedia. The first draft of RTSP protocol was submitted to IETF on October 9, 1996 for consideration as an Internet Standard. Since then, it has gone through significant changes, and it was approved by IETF as a Proposed Standard on April 1998.

Numerous products using RTSP are available today. Major online players, such as Netscape, Apple, IBM, Silicon Graphics, VXtreme, Sun, and other companies, have announced their support for RTSP.

RTSP establishes and controls streams of continuous audio and video media between the media servers and the clients. A *media server* provides playback or recording services for the media streams, while a *client* requests continuous media data from the media server. RTSP supports the following operations:

Retrieval of media from the media server: The client can request a presentation description via HTTP or some other method. If the presentation is being multicasted, the presentation description contains the multicast addresses and ports to be used for the continuous media. If the presentation is to be sent only to the client, the client provides the destination for security reasons. The client can also ask the server to set-up a session to send the requested data.

Invitation of a media server to a conference: A media server can be invited to join an existing conference, either to play back media into the presentation or to record all or a subset of the media in a presentation. This method is useful for distributed applications such as distance learning. Several parties in the conference may take turns "pushing the remote control buttons."

Adding media to an existing presentation: The server or the client can notify each other about any additional media becoming available. This is particularly useful for live presentations.

RTSP aims to provide the same services on streamed audio and video just as HTTP does for text and graphics. It is intentionally designed to have similar syntax and operations, so that most extension mechanisms to HTTP can be added to RTSP.

Each presentation and media stream in RTSP is identified by an RTSP URL. The overall presentation and the properties of the media are defined in a presentation description file, which may include the encoding, language, RTSP URLs, destination address, port, and other parameters. The client can also obtain the presentation description file using HTTP, e-mail, or other means.

RTSP, however, differs from HTTP in several aspects. First, while HTTP is a stateless protocol, an RTSP server has to maintain "session states" in order to correlate RTSP requests with a stream. Second, HTTP is an asymmetric protocol where the client issues requests and the server responds; but in RTSP, both the media server and the client can issue requests. For example, the server can issue a request to set playing back parameters of a stream.

The services and operations in the current version are supported through the following methods:

OPTIONS: The client or the server informs the other party about the options it can accept.

DESCRIBE: The client retrieves the description of a presentation or media object identified by the request URL from the server.

ANNOUNCE: When sent from client to server, Announce posts the description of a presentation or media object identified by the request URL to a server. When sent from server to client, Announce updates the session description in real time.

SETUP: The client asks the server to allocate resources for a stream and start an RTSP session.

PLAY: The client asks the server to start sending data on a stream allocated via SETUP.

PAUSE: The client temporarily halts the stream delivery without freeing server resources.

TEARDOWN: The client asks the server to stop delivery of the specified stream and free the resources associated with it.

GET_PARAMETER: Retrieves the value of a parameter of a presentation or a stream specified in the URI.

SET_PARAMETER: Sets the value of a parameter for a presentation or stream specified by the URI.

REDIRECT: The server informs the clients that it must connect to another server location. The mandatory location header indicates the URL that the client should connect to.

RECORD: The client initiates recording a range of media data according to the presentation description.

Note that some of these methods can either be sent from the server to the client or from the client to the server; but others can only be sent in one direction. Not all these methods, however, are necessary in a fully functional server. For example, a media server with live feeds may not support the PAUSE method.

RTSP requests are usually sent on a channel independent of the data channel. They can be transmitted in persistent transport connections, or as a one-connection per request/response transaction, or in connectionless mode.

RTSP Implementation Resources

Several RTSP implementations are available on the web. The following is a collection of useful implementation resources:

RTSP Reference Implementation (http://www.real.com/devzone/library/fireprot/rtsp/reference.html): This is a source code testbed for the standards community to experiment with RTSP compatibility.

RealMedia SDK (http://www.realnetworks.com/devzone/tools/index.html): This is an open, cross-platform, client–server system that allows implementers to create RTSP-based streaming applications. It includes a working RTSP client and server, as well as the components to quickly create RTSP-based applications that stream arbitrary data types and file formats.

W3C's Jigsaw (http://www.w3.org/Jigsaw/): A Java-based web server. The RTSP server in the latest beta version was written in Java.

IBM's RTSP Toolkit (http://www.research.ibm.com/rtsptoolkit/): IBM's toolkits derived from tools developed for ATM/video research and other applications in 1995–1996. Its shell-based implementation illustrates the usefulness of the RTSP protocol for nonmultimedia applications.

28.5 Summary

This chapter discusses the three related protocols for real-time multimedia data over the Internet.

RTP is the transport protocol for real-time data. It provides timestamp, sequence number, and other means to handle the timing issues in real-time data transport.

RTCP is the control part of RTP that helps with quality of service and membership management.

RTSP is a control protocol that initiates and directs delivery of streaming multimedia data from media servers. It is the "Internet VCR remote control protocol." Its role is to provide the remote control; however, the actual data delivery is done separately, most likely by RTP.

References

[1] Voice On the Net, http://von.com.

[2] RealNetworks, http://www.realnetworks.com.

[3] Apple's Quick Time, http://www.apple.com/quicktime.

[4] Microsoft's Windows Media, http://www.microsoft.com/windows/windowsmedia/.

[5] J.F. Kurose and K.W. Ross, *Computer Networking — A Top-down Approach Featuring the Internet*, 2nd ed., 2003.

[6] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, RTP: A Transport Protocol for Real-Time Applications, RFC3550, July 2003.

[7] H. Schulzrinne, S. Casner, RTP Profile for Audio and Video Conferences with Minimal Control, RFC3551, July 2003.

[8] D. Clark and D. Tennenhouse, Architectural considerations for a new generation of protocols, Proceedings of the SIGCOMM '90 Symposium, Sept., 1990, pp. 200–208.

[9] Why use the Plain Old Telephone when you can get so much better on the Internet? 1999, http://www-sop.inria.fr/rodeo/fphone/.

[10] H. Schulzrinne, A. Rao, and R. Lanphier, Real Time Streaming Protocol (RTSP), RFC 2326, April 1998.

[11] L.L. Peterson and B.S. Davie, *Computer Networks — a System Approach*, 3rd ed., Morgan Kaufmann, Los Attos, CA, 2003.

[12] Mark A. Miller, *Voice over IP Technologies*, M&T Books, 2002.

[13] Fred Halsall, *Multimedia Communications*, Addison-Wesley, Reading, MA, 2001.

[14] Alberto Leon-Garcia and Indra Widjaja, *Communication Networks*, 2nd ed., McGraw-Hill, New York, 2003.

[15] RTSP Resource Center, http://www.rtsp.org.