Chunlei Liu. “Multimedia Over IP: RSVP, RTP, RTCP, RTSP”
*Handbook of Emerging Communications Technologies: The Next Decade.*
Ed. Saba Zamir
Boca Raton: CRC Press LLC, 2000

# 2 Multimedia Over IP: RSVP, RTP, RTCP, RTSP

*Chunlei Liu*

## CONTENTS



The future Integrated Services Internet will provide means to transmit real-time multimedia data across networks. RSVP, RTP, RTCP and RTSP are the foundation of real-time services. This paper is a detailed survey of the four related protocols.

## 2.1 MULTIMEDIA NETWORKING: GOAL AND CHALLENGE

Computer networks were designed to connect computers at different locations so that they can share data and communicate. In the old days, most of the data carried on networks was textual data. Today, with the rise of multimedia and network technologies, multimedia has become an indispensable feature on the Internet. Animation, voice, and video clips become more and more popular on the Internet. Multimedia networking products such as Internet telephony, Internet TV, and videoconferencing have appeared on the market. In the future, people will enjoy other multimedia products in distance learning, distributed simulation, distributed work groups, and other areas.

For networkers, multimedia networking is to build the hardware and software infrastructure and application tools to support multimedia transport on networks so that users can communicate in multimedia. Multimedia networking will greatly boost the use the of computers as a communication tool. We believe that someday multimedia networks will replace telephone, television, and other inventions that have dramatically changed our lives.

### 2.1.1 The Real-time Challenge

Multimedia networking is not a trivial task. We can expect at least three difficulties. First, compared with traditional textual applications, multimedia applications usually require much higher bandwidth. A typical piece of 25-s $320 \times 240$ QuickTime movie could take 2.3MB, which is equivalent to about 1000 screens of textual data. This would have been unimaginable in the old days when only textual data was transmitted on the net.

Second, most multimedia applications require real-time traffic. Audio and video data must be played back continuously at the rate they were sampled. If the data does not arrive in time, the playback will stop and human ears and eyes can easily pick up the artifact. In Internet telephony, human beings can tolerate a latency of about 250 msec. If the latency exceeds this limit, the voice will sound like a call routed over a long satellite circuit and users will complain about the quality of the call. In addition to the delay, network congestion also has more serious effects on real time traffic. If the network is congested, the only effect on nonreal-time traffic is that the transfer takes longer to complete, but real-time data becomes obsolete and will be dropped if it doesn't arrive in time. If no proper action is taken, the retransmission of lost packets would aggravate the situation and jam the network.

Third, a multimedia data stream is usually bursty. Just increasing the bandwidth will not solve the burstiness problem. For most multimedia applications, the receiver has a limited buffer. If no measure is taken to smooth the data stream, it may overflow or underflow the application buffer. When data arrives too quickly, the buffer will overflow and some data packets will be lost, resulting in poor quality. When data arrives too slowly, the buffer will underflow and the application will starve.

Contrary to the high bandwidth, real-time and bursty traffic of multimedia data in real life networks are shared by thousands and millions of users and have limited

bandwidth and unpredictable delay and availability. How to solve these conflicts is a challenge multimedia networking must face.

The possibility of answering this challenge comes from the existing network software architecture and emerging hardware. The basis of the Internet, TCP/IP and UDP/IP, provides a range of services that multimedia applications can use. Fast networks like Gigabit Ethernet, FDDI, and ATM provide the high bandwidth required by digital audio and video.

Consequently the design of real-time protocols for multimedia networking becomes imperative before the multimedia age comes.

### 2.1.2 Multimedia over the Internet

There are other ways to transmit multimedia data, such as over dedicated links, cables, and ATM. However, the idea of running multimedia over the Internet is extremely attractive.

Dedicated links and cables are not practical because they require special installation and new software. Without an existing technology like LAN or WAN, the software development will be extremely expensive. ATM was said to be the ultimate solution for multimedia because it supports very high bandwidth, is connection-oriented, and can tailor different levels of quality of service to different types of applications. But at this moment, very few users have ATM networks reaching their organization, and even fewer have an ATM connection to their desktops.

On the other hand, the Internet is growing exponentially. The well-established LAN and WAN technologies based on the IP protocol suite connect bigger and bigger networks all over the world to the Internet. In fact, the Internet has become the platform of most networking activities. This is the primary reason to develop multimedia protocols over the Internet. Another benefit of running multimedia over IP is that users can have integrated data and multimedia service over one single network, without investing in another network and building the interface between two networks.

At the current time, IP and Ethernet seem to be more favored in desktops and LANs, with ATM in WANs.

As a shared datagram network, the Internet is not naturally suitable for real-time traffic. To run multimedia over the Internet, several issues must be solved. First, multimedia means extremely dense data and heavy traffic. The hardware has to provide enough bandwidth.

Second, multimedia applications are usually related to multicast, i.e., the same data stream, not multiple copies, is sent a group of receivers. For example, in videoconferencing, the video data needs to be sent to all participants at the same time. Live video can be sent to thousands of recipients. The protocols designed for multimedia applications must take into account multicast in order to reduce the traffic.

Third, the cost attached to shared network resources is the unpredictable availability. However, real-time applications require guaranteed bandwidth when the transmission takes place, so there must be some mechanism for real-time applications to reserve resources along the transmission path.

Fourth, the Internet is a packet-switching datagram network where packets are routed independently across shared networks. The current technologies cannot guarantee that real-time data will reach the destination without being jumbled and jerky. Some new transport protocols must be used to take care of the timing issues so that audio and video data can be played back continuously with correct timing and synchronization.

Fifth, there should be some standard operations for applications to manage the delivery and present the multimedia data.

The answers to the above issues are the protocols that are discussed in this chapter.

### 2.1.3 The Solution

The Internet carries all types of traffic, each of which has different characteristics and requirements. For example, a file transfer application requires that some quantity of data be transferred in an acceptable amount of time, while Internet telephony requires that most packets get to the receiver in less than 0.3 seconds. If enough bandwidth is available, best-effort service fulfills all of these requirements. When resources are scarce, however, real-time traffic will suffer from the congestion.

The solution for multimedia over IP is to classify all traffic, allocate priority for different applications, and make reservations. The Integrated Services working group in the IETF (Internet Engineering Task Force) developed an enhanced Internet service model called Integrated Services Internet that includes best-effort service and real-time service.[2] The real-time service will enable IP networks to provide quality of service to multimedia applications. Resource ReSerVation Protocol (RSVP), together with Real-time Transport Protocol (RTP), Real-Time Control Protocol (RTCP), and Real-Time Streaming Protocol (RTSP), provides a working foundation for real-time services. Integrated Services allows applications to configure and manage a single infrastructure for multimedia applications and traditional applications. It is a comprehensive approach to providing applications with the type of service they need and the quality they choose. This chapter is a detailed review of these four protocols.

## 2.2 RSVP — Resource Reservation Protocol

RSVP is the network control protocol that allows the data receiver to request a special end-to-end quality of service for its data flows. Real-time applications use RSVP to reserve necessary resources at routers along the transmission paths so that the requested bandwidth can be available when the transmission actually takes place. RSVP is a main component of the future Integrated Services Internet which can provide both best-effort and real-time service.[2]

### 2.2.1 Development

RSVP design has been a joint effort of Xerox Corporation's Palo Alto Research Center (PARC), MIT, and Information Sciences Institute of the University of California (ISI).[1] The RSVP specification was submitted to the Internet Engineering

Steering Group (IESG) for consideration as a Proposed RFC in November 1994. In September 1997, RSVP Version 1 Functional Specification and the following other, related Internet drafts were approved as Proposed Standards:

- RFC 2205, Resource ReSerVation Protocol (RSVP) — Version 1 Functional Specification
- RFC 2206, RSVP Management Information Base using SMIv2
- RFC 2207, RSVP Extensions for IPSEC Data Flows
- RFC 2208, RSVP Version 1 Applicability Statement: Some Guidelines on Deployment
- RFC 2209, RSVP Version 1 Message Processing Rules

The RSVP working group of the IETF is developing other protocols to be used with RSVP.

### 2.2.2 How Does RSVP Work?

RSVP is used to set up reservations for network resources. When an application in a host (the data stream receiver) requests a specific quality of service (QoS) for its data stream, it uses RSVP to deliver its request to routers along the data stream paths. RSVP is responsible for the negotiation of connection parameters with these routers. If the reservation is set up, RSVP is also responsible for maintaining router and host states to provide the requested service.

Each node capable of resource reservation has several local procedures for reservation setup and enforcement (see Figure 2.1). *Policy control* determines whether the user has administrative permission to make the reservation. In the future, authentication, access control, and reservation accounting will also be implemented by policy control. *Admission control* keeps track of the system resources and determines whether the node has sufficient resources to supply the requested QoS.

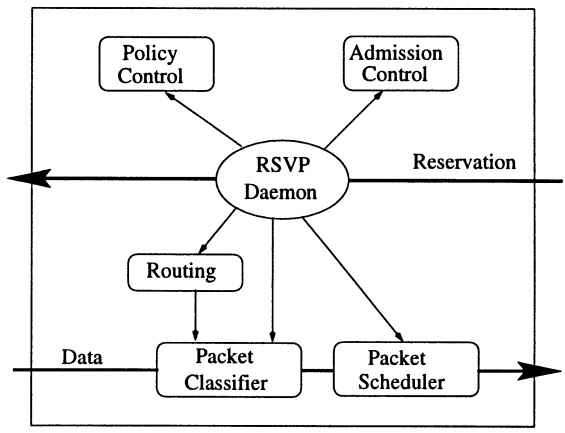


**Figure 2.1** Reservation at a node on the data flow path

The RSVP daemon checks with both procedures. If either check fails, the RSVP program returns an error notification to the application that originated the request. If both checks succeed, the RSVP daemon sets parameters in the packet classifier and packet scheduler to obtain the requested QoS. The *packet classifier* determines the QoS class for each packet and the *packet scheduler* orders packet transmission to achieve the promised QoS for each stream. The RSVP daemon also communicates with the routing process to determine the path to send its reservation requests and to handle changing memberships and routes.

This reservation procedure is repeated at routers along the reverse data stream path until the reservation merges with another reservation for the same source stream.

Reservations are implemented through two types of RSVP messages: PATH and RESV. The PATH messages are sent periodically from the sender to the multicast address. A PATH message contains *flow spec* to describe sender template (data format, source address, source port) and traffic characteristics. This information is used by receivers to find the reverse path to the sender and to determine what resources should be reserved. Receivers must join the multicast group in order to receive PATH messages. RESV messages are generated by the receivers and contain reservation parameters including *flow spec* and *filter spec*. The filter spec defines what packets in the flow should be used by the packet classifier. The flow spec is used in the packet scheduler and its content depends on the service. RESV messages follow the exact reverse path of PATH messages, setting up reservations for one or more senders at every node.

The reservation states RSVP builds at the routers are *soft states*. The RSVP daemon needs to send refresh messages periodically to maintain the reservation states. The absence of a refresh message within a certain time will destroy the reservation state. By using soft states, RSVP can easily handle changing memberships and routes.

The reservation requests are initiated by the receivers. They do not need to travel all the way to the source of the sender. Instead, they travel upstream until they meet another reservation request for the same source stream then merge with that reservation. Figure 2.2 shows how the reservation requests merge as they progress up the multicast tree.

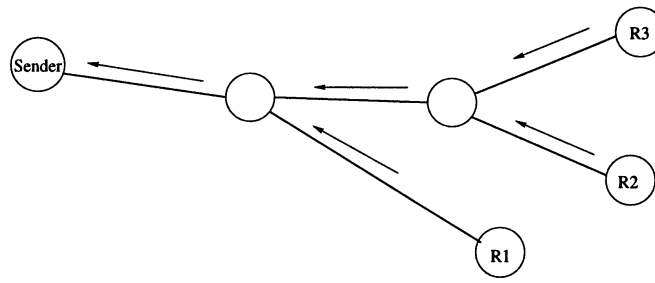


**Figure 2.2** Reservation merging

This reservation merging leads to the primary advantage of RSVP: scalability A large number of users can be added to a multicast group without increasing the data traffic significantly. Consequently, RSVP can scale to large multicast groups, and the average protocol overhead decreases as the number of participants increases.

The reservation process does not actually transmit the data and provide the requested QoS. However, through reservation RSVP guarantees the network resources will be available when the transmission actually takes place.

Although RSVP sits on top of IP in the protocol stack, it is not a routing protocol, but rather an Internet control protocol. Actually, RSVP relies on the underlying routing protocols to find where it should deliver the reservation requests. RSVP is also intended to cooperate with unicast and multicast routing protocols. When the RSVP-managed flow changes its path, the routing module will notify the RSVP module of the route changes. Therefore, RSVP can quickly adjust the resource reservation to new routes.

The delivery of reservation parameters is different from the determination of these parameters. How to set the connection parameters to achieve the requested QoS is the task of QoS control devices; the role of RSVP is just a general facility to distribute these parameters. Since different applications may have different QoS control devices, RSVP is designed to treat these QoS parameters as opaque data to be delivered to and interpreted by the control modules at the routers. This logical separation of QoS control devices and distribution facility simplifies the design of RSVP and makes it more adaptive to new network technologies and applications.

### 2.2.3 RSVP Features

- RSVP flows are simplex.

RSVP distinguishes senders and receivers. Although in many cases a host can act both as a sender and as a receiver, one RSVP reservation only reserves resources for data streams in one direction.

- RSVP supports both multicast and unicast and adapts to changing memberships and routes.

RSVP is designed for both multicast and unicast. Since the reservations are initiated by the receivers and the reservation states are soft, RSVP can easily handle changing memberships and routes. A host can send IGMP (Internet Group Management Protocol) messages to join a multicast group. Reservation merging enables RSVP to scale to large multicast groups without causing heavy overhead for the sender.

- RSVP is receiver-oriented and handles heterogeneous receivers.

In heterogeneous multicast groups, receivers have different capacities and levels of QoS. The receiver-oriented RSVP reservation requests facilitate the handling of heterogeneous multicast groups. Receivers are responsible for choosing their own level of QoS, initiating the reservation, and keeping it active as long as it wants. The senders

divide traffic in several flows, each of which is a separate RSVP flow with a different level of QoS. Each RSVP flow is homogeneous and receivers can choose to join one or more flows. This approach makes it possible for heterogeneous receivers to request different QoS tailored to their particular capacities and requirements.

- RSVP has good compatibility.

Efforts have been made to run RSVP over both IPv4 and IPv6. It provides opaque transport of traffic-control and policy-control messages in order to be more adaptive to new technologies. It also provides transparent operation through nonsupporting regions.

### 2.2.4 RSVP Interfaces

RSVP communicates with both applications at the end hosts and various components inside network routers. For application programmers, the most important part is the client application programming interface. In an implementation developed by the ISI and Sun Microsystems, the following fundamental functions make up an RSVP client library called RAPI (RSVP Application Programming Interface):

- rapi_session() creates and initiates an API session and returns a session handle for further reference.
- rapi_sender() is called by a sender application to specify the parameters of its data flow. The RSVP daemon at the sender will use this flow spec to create PATH messages for this flow.
- rapi_reserve() is used to make, modify, or delete a reservation.
- rapi_release() asks the RSVP daemon to tear down the reservation.

For more information about RAPI, see the Internet Draft[4] by Braden and Hoffman.

## 2.3 RTP — REAL-TIME TRANSPORT PROTOCOL

RTP is an IP-based protocol providing support for the transport of real-time data such as video and audio streams. The services provided by RTP include time reconstruction, loss detection, security, and content identification. RTP is primarily designed for multicast of real-time data, but it can be also used in unicast. It can be used for one-way transport, such as video-on-demand, as well as interactive services such as Internet telephony.

RTP is designed to work in conjunction with the auxiliary control protocol RTCP to get feedback on quality of data transmission and information about participants in the ongoing session.

### 2.3.1 Development

Attempts to send voice over networks began in the early seventies. Several patents on packet transmission of speech, time stamp, and sequence numbering were granted in the seventies and eighties. 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, GMD Berlin, published RTP version 1. It went through several states of Internet drafts and was finally approved as a proposed standard on November 22, 1995 by the IESG. This version[5,6] was called RTP version 2 and was published as RFC 1889, *RTP: A Transport Protocol for Real-Time Applications* and RFC 1890, *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 claims that their NetMeeting Conferencing Software supports RTP. The latest extensions have been made by an industry alliance around Netscape Inc., which uses RTP as the basis of the RTSP.

### 2.3.2 How Does RTP Work?

As discussed in the first section, the Internet is a shared datagram network, and packets sent on the Internet have unpredictable delay and jitter. However, multimedia applications require appropriate timing in data transmission and playback. RTP provides timestamping, sequence numbering, and other mechanisms to take care of the timing issues. Through these mechanisms, RTP provides end-to-end transport for real-time data over a datagram network.

Timestamping is the most important information for real-time applications. The sender sets the timestamp according to the instant the first octet in the packet was sampled. Timestamps increase by the time covered by a packet. After receiving the data packets, the receiver uses the timestamp to reconstruct the original timing in order to play out the data at the correct rate. Timestamping is also used to synchronize different streams with timing properties, such as audio and video data in MPEG. However, RTP itself is not responsible for the synchronization. This has to be done in the application level.

UDP does not deliver packets in timely order, so sequence numbers are used to place the incoming data packets in the correct order. They are also used for packet loss detection. Notice that in some video formats, when a video frame is split into several RTP packets all of them can have the same timestamp. Consequently, timestamp alone is not enough to put the packets in order.

The payload type identifier specifies the payload format as well as the encoding/compression schemes. From this payload type identifier the receiving application knows how to interpret and play out the payload data. Default payload types are defined in RFC 1890.[6] Example specifications include PCM, MPEG1/MPEG2 audio and video, JPEG video, Sun CellB video, and H.261 video streams. More payload types can be added by providing a profile and payload format specification. At any given time of transmission, an RTP sender can send only one type of payload although the payload type may change during transmission, for example to adjust to network congestion.

Another function is source identification. It allows the receiving application to know where the data is coming from. For example, in an audio conference a user could tell who is talking from the source identifier.

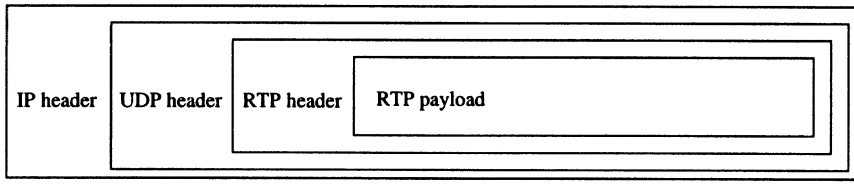


**Figure 2.3** RTP data in a UDP/IP packet

The above mechanisms are implemented through the RTP header. Figure 2.3 shows an RTP packet encapsulated in a UDP/IP packet.

RTP is typically run on top of UDP to make use of its multiplexing and checksum functions. TCP and UDP are the two most commonly used transport protocols on the Internet. TCP provides a connection-oriented and reliable flow between two hosts, while UDP provides a connectionless but unreliable datagram service over the network. UDP was chosen as the target transport protocol for RTP for two reasons. First, RTP is primarily designed for multicast; the connection-oriented TCP does not scale well and therefore is not suitable. Second, for real-time data, reliability is not as important as timely delivery. Even more, reliable transmission provided by retransmission as in TCP is not desirable. For example, in network congestion, some packets might get lost and the application would result in lower but acceptable quality. If the protocol insists on reliable transmission, the retransmitted packets could possibly increase the delay, jam the network, and eventually starve the receiving application.

RTP and RTCP packets are usually transmitted using UDP/IP service. However, efforts have been made to make them transport-independent so they can also be run on CLNP(Connectionless Network Protocol), IPX (Internetwork Packet Exchange), AAL5/ATM, or other protocols.

In practice, RTP is usually implemented within the application. Many issues, such as lost packet recovery and congestion control, have to be implemented in the application level.

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-conferencing scenario presented in RFC 1889[5] illustrates the use of RTP. Suppose each participant sends audio data in segments of 20-ms duration. Each segment of audio data is preceded by an RTP header, and the resulting RTP message is placed in a UDP packet. The RTP header indicates the type of audio encoding that is used, e.g., PCM. Users can opt to change the encoding during a conference in reaction to network congestion or, for example, to accommodate low-bandwidth requirements of a new conference participant. Timing information and a sequence number in the RTP header are used by the receivers to reconstruct the timing

produced by the source, so, in this example, audio segments are contiguously played out at the receiver every 20 ms.

### 2.3.3 RTP Fixed Header Fields

The RTP header has the format shown in Figure 2.4

The first twelve octets are present in every RTP packet, while the list of CSRC (contributing source) identifiers is present only when inserted by a mixer. The fields have the following meaning:

**version (V): 2 bits**. Version of RTP. The newest version is 2.

**padding (P): 1 bit**. If 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.

**extension (X): 1 bit**. If set, the fixed header is followed by exactly one header extension.

**CSRC count (CC): 4 bits**. Indicates 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.

**marker (M): 1 bit**. Defined by a profile, the marker is intended to allow significant events such as frame boundaries to be marked in the packet stream.

**payload type (PT): 7 bits**. Identifies the format of the RTP payload and determines its interpretation by the application.

**sequence number: 16 bits**. Increments by one for each RTP data packet sent. It may be used by the receiver to detect packet loss and to restore packet sequence. The initial value is randomly set.

**timestamp: 32 bits**. The sampling instant of the first octet in the RTP data packet. It may be used for synchronization and jitter calculations. The initial value is randomly set.

**SSRC: 32 bits**. SSRC is a randomly chosen number to distinguish synchronization sources within the same RTP session. It indicates where the data was combined or the source of the data if there is only one source.

**CSRC list: 0 to 15 items, 32 bits each**. Contributing sources for the payload contained in this packet. The number of identifiers is given by the CC field.

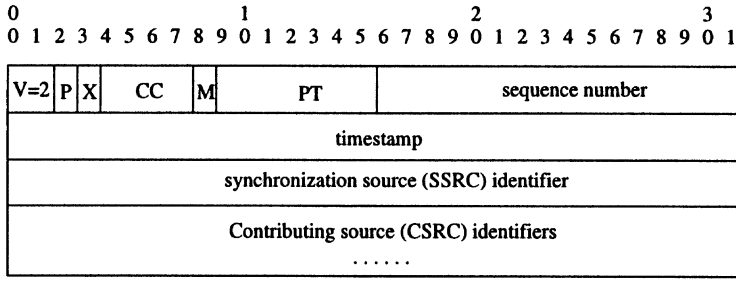


**Figure 2.4** RTP packet header

### 2.3.4 RTCP — Real-time Control Protocol

RTCP is the control protocol designed to work in conjunction with RTP. It is standardized in RFC 1889 and RFC 1890. In an RTP session, participants periodically send RTCP packets to convey feedback on quality of data delivery and information on membership. RFC 1889 defines five RTCP packet types to carry control information. These five types are:

- **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 packet 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 intermedia synchronization, cumulative packet counters, and number of bytes sent.
- **SDES** (source description items) — They contain information to describe the sources.
- **BYE** — Indicates end of participation.
- **APP** (application specific functions) — Is now intended for experimental use as new applications and new features are developed.

Through these control information packets, RTCP provides the following services:

- **QoS monitoring and congestion control**

This is the primary function of RTCP. RTCP provides feedback to an application about the quality of data distribution. The control information is useful to the senders, the receivers, and third-party monitors. The sender can adjust its transmission based on the receiver report feedback. The receivers can determine whether a congestion is local, regional, or global. Network managers can evaluate the network performance for multicast distribution.

- **Source identification**

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

- **Inter-media synchronization**

RTCP sender reports contain an indication of real time and the corresponding RTP timestamp. This can be used in intermedia synchronization, such as lip synchronization in video.

- **Control information scaling**

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. RTP limits the control traffic to at most 5% of the overall session traffic. This is enforced by adjusting the RTCP generating rate according to the number of participants.

### 2.3.5 RTP Features

- RTP provides end-to-end delivery services for data with real-time characteristics, such as interactive audio and video. But RTP itself does not provide any mechanism to ensure timely delivery. It needs support from lower layers that actually have control over resources in switches and routers. RTP depends on RSVP to reserve resources and to provide the requested quality of service.
- RTP does not assume anything about the underlying network, except that it provides framing. RTP is typically run on the top of UDP to make use of its multiplexing and checksum service, but efforts have been made to make RTP compatible with other transport protocols, such as ATM AAL5, and IPv6.
- Unlike usual data transmission, RTP does not offer any form of reliability or flow/congestion control. It provides timestamps and sequence numbers as hooks for adding reliability and flow/congestion control, but how to implement is totally left to the application.
- RTP is a protocol framework that is deliberately not complete. It is open to new payload formats and new multimedia software. By adding new profile and payload format specifications, one can tailor RTP to new data formats and new applications.
- RTP/RTCP provides functionality and control mechanisms necessary for carrying real-time content, but it is not responsible for the higher-level tasks such as assembly and synchronization. These have to be done at the application level.
- The flow and congestion control information of RTP is provided by RTCP sender and receiver reports.

### 2.3.6 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 must add the complete functionality in the application layer by themselves. However, it is always more efficient to share and reuse code rather than start from scratch. The RFC 1889 specification[5] itself contains numerous code segments that can be borrowed directly by the applications. Some implementations with source code are available on the web for evaluation and educational purposes. Many

modules in the source code can be usable with minor modifications. The following is a list of useful resources:

- **vat** — http://www-nrg.ee.lbl.gov/vat/
- **tptools** — ftp://ftp.cs.columbia.edu/pub/schulzrinne/rtptools/
- **NeVoT** — http://www.cs.columbia.edu/~{}hgs/rtp/nevot.html
- **RTP Page** — http://www/cs.columbia.edu/~{}hgs/rtp — maintained by Henning Schulzrinne and a very complete reference
- **RTP Library** — http://www.iasi.rm.cnr.it/iasi/netlab/gettingSoftware.html — by E. A. Mastromartino and offers convenient ways to incorporate RTP functionality into C++ Internet applications

## 2.4 RTSP — REAL-TIME STREAMING PROTOCOL

Instead of being sent as a single large multimedia file that must be stored then played back, multimedia data is usually sent across the network in streams. Streaming breaks data into packets with a size suitable for transmission between the servers and clients. The real-time data flows through the transmission, decompresses and plays back pipeline just like a water stream. A client can play the first packet and decompress the second while receiving the third. Thus the user can start enjoying the multimedia without waiting until the end of transmission.

RTSP is a client-server multimedia presentation protocol to enable controlled delivery of streamed multimedia data over an IP network. It provides "VCR-style" remote control functionality for audio and video streams, such as pause, fast forward, reverse, and absolute positioning. Sources of data include both live data feeds and stored clips.

RTSP is an application-level protocol 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 upon RTP. It works for large audience multicast as well as single-viewer unicast.

### 2.4.1 DEVELOPMENT

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 the RTSP protocol was submitted to IETF on October 9, 1996 for consideration as an Internet Standard. Since then, it has gone through significant changes. The latest version is draft-ietf-mmusic-rtsp-07.[7]

Although RTSP is still an Internet Draft at IETF and is likely to have significant changes, products using RTSP are available today. The major online players, Netscape, Apple, IBM, Silicon Graphics, VXtreme, Sun, and other companies, have claimed their support to RTSP, although Microsoft's absence is conspicuous.

### 2.4.2 RTSP Operations and Methods

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 is the "network remote control" between the server and the client. It provides the following operations:

- Retrieval of media from the media server: The client can request a presentation description and ask the server to set up a session to send the requested data.
- Invitation of a media server to a conference: The media server can be invited to the conference to play back media or to record a presentation.
- Adding media to an existing presentation: The server or the client can notify each other about any additional media becoming available.

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

In RTSP, each presentation and media stream 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 presentation description file can be obtained by the client using HTTP, e-mail, or other means.

RTSP 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 basically 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 playback parameters of a stream.

In the current version, the services and operations are supported through the following methods:

- **OPTIONS** — The client or the server tells the other party 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 realtime.
- **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 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 be sent either from the server to the client or from the client to the server, but others can be sent in only one direction. Not all these methods 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, as a one-connection-per-request/response transaction, or in connectionless mode.

### 2.4.3 RTSP Features

- RTSP is an application-level protocol with syntax and operations similar to HTTP but for audio and video. It uses URLs like those in HTTP.
- An RTSP server needs to maintain states, using SETUP, TEARDOWN, and other methods.
- RTSP messages are carried out-of-band. The protocol for RTSP may be different from the data delivery protocol.
- Unlike HTTP, in RTSP both servers and clients can issue requests.
- RTSP is implemented on multiple operating system platforms; it allows interoperability between clients and servers from different manufacturers.

### 2.4.4 RTSP Implementation Resources

Although RTSP is still an IETF draft, there are a few implementations already available on the web. The following is a collection of useful implementation resources:

- RTSP Reference Implementation — http://www6.real.com/rtsp/reference.html

This is a source code testbed for the standards community to experiment with RTSP compatibility.

- RealMedia SDK — http://www6.real.com/realmedia/index.html

This is an open, cross platform, client-server system where implementors can create RTSP-based streaming applications. It includes a working RTSP client and server, as well as the components to quickly create RTSP-based applications which 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.

## 2.5 SUMMARY

This chapter discusses the four related protocols for real-time multimedia data in the future Integrated Services Internet.

RSVP is the protocol that deals with lower layers that have direct control over network resources to reserve resources for real-time applications at the routers on the path. It does not deliver the data.

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. It relies on RSVP for resource reservation to provide quality of service. 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; the actual data delivery is done separately, most likely by RTP.

## ACKNOWLEDGMENT

This chapter was originally written as a paper in Professor Raj Jain's "Recent Advances in Networking" class in summer, 1997 at Ohio State University. The author sincerely thanks Professor Jain for his helpful guidance.

## REFERENCES

1. L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, RSVP: A New Resource ReSerVation Protocol, *IEEE Network*, Vol 7, no 5, pp 8-18, September 1993.
2. R. Braden, D. Clark, and S. Shenker, Integrated Services in the Internet Architecture: an Overview, ftp://ds.internic.net/rfc/rfc1633.txt, RFC1633, ISI, MIT, and PARC, June 1994.

3. L. Zhang, S. Berson, S. Herzog, and S. Jamin, Resource ReSerVation Protocol(RSVP)--Version 1 Functional Specification, ftp://ds.internic.net/rfc/rfc2205.txt, RFC2205, September 1997.
4. R. Braden and D. Hoffman, RAPI — An RSVP Application Programming Interface, ftp://ftp.ietf.org/internet-drafts/draft-ietf-rsvp-rapi-00.txt, Internet Draft, June 1997.
5. H. Schulzrinne and S. Casner, R. Frederick, and V. Jacobson, RTP: A Transport Protocol for Real-Time Applications, ftp://ds.internic.net/rfc/rfc1889.txt, RFC1889, January, 1996.
6. H. Schulzrinne, RTP Profile for Audio and Video Conferences with Minimal Control, ftp://ds.internic.net/rfc/rfc1890.txt, RFC1890, January 1996.
7. H. Schulzrinne, A. Rao, and R. Lanphier, Real Time Streaming Protocol (RTSP), ftp://ds.internic.net/internet-drafts/draft-ietf-mmusic-rtsp-07.txt, Internet Draft, January 1998.