44100113-0: COMPUTER NETWORKS
FALL 2019

Video Stream Programming

Assigned: November 20, 2019
Due Date: 23:59, December 11, 2019

Section 1-2 describe the assignment.

Section 3-4 provide very useful information to finish this project.

1 ASSIGNMENT

You must work on this assignment individually.

This project is designed for you to:

- To be familiar with video streaming services and its underlying RTP/RTCP/RTSP protocols
- Learn to implement stream video player on PC
- Awaken your programming skills

In TASK-1, you need to finish the following jobs, which gives you basic ideas for RTP/RTCP/RTSP protocol family:

- Read the given programs located in *RTP* directory carefully. The sample code provides you with partially implemented RTSP protocol in the server and the RTP depacketization in the client. The code also takes care of displaying the transmitted frames.
- Modify the code to let the client continuously receive sequence of *.jpg* files from server and display on the screen on client side.
- You will need to implement RTSP in the client and RTP server.
- **You MUST finish this task in Python language using the provided code snippets.**

In TASK-2, you will be asked to create, based on the experience in TASK-1, a pair of functional **RTP server** and **RTP client** with the following compulsory characteristics:

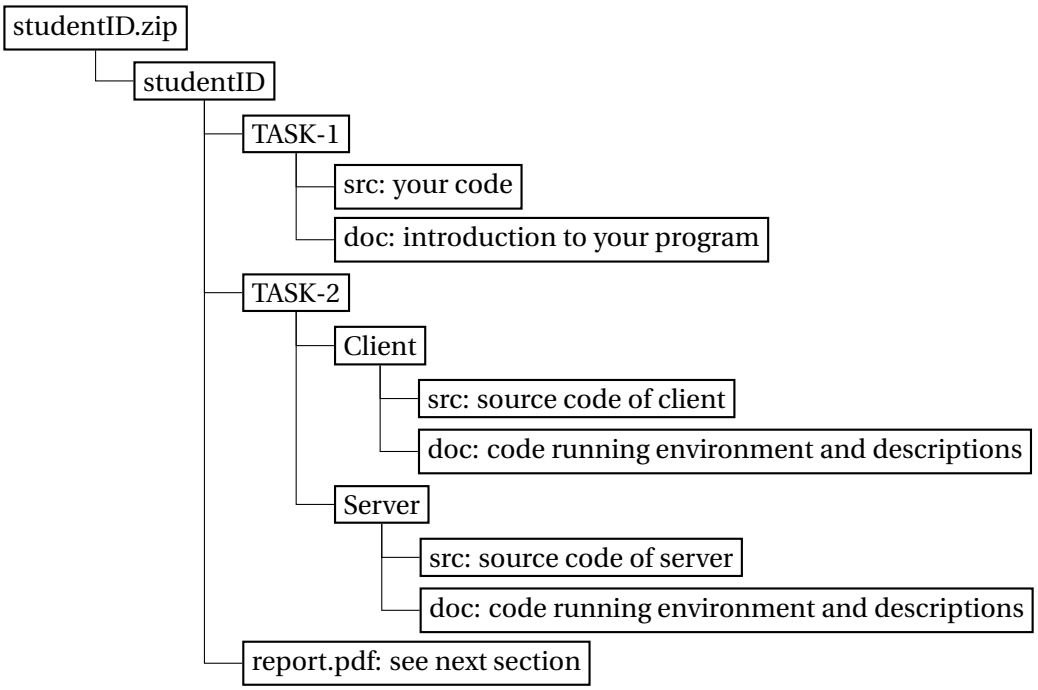
- Must program with Socet just like in Project1.
- You MUST create your server and client from scratch and MUST not reuse any open source media players.

- Support SETUP/PLAY/PAUSE/TEARDOWN command in RTSP protocol and support RTP encapsulation.
- Support repositioning of play point.
- Support change of play speed.
- Support at least .mp4 file format (the more the better).
- Support streams from multiple clients on a single server.
- **You can finish this task in any language you prefer.**

To make your implemented video stream system more user-friendy and readily usable, the following features are optional (note that only finish compulsory tasks will not bring you satisfying scores):

- Delay or advance audio track.
- Support subtitle demonstration and adjustment.
- Resume transmission after client reconnented.
- Informative play list.
- Search video by keyword or category on server.
- Video quality specification.
- Switch to full screen.
- Reliable transmission over UDP (use GBN/SR mechanisms).
- Dynamic video encoding (H.264/MJPEG/...) under different network conditions.
- Buffer mechanism for better users' experience.
- Other features that are worth noticing.

Your submitted files are to be organised as follows:



2 Grading Guidelines

- If your server dumps core during our testing, you will lose substantial credit. Handling invalid input reasonably and generating defensible error codes are fundamental parts of writing any server program.
- Poor design, documentation, or code structure will probably reduce your grade by making it hard for you to produce a working program and hard for the TA to understand it; egregious failures in

these areas will cause your grade to be lowered even if your implementation performs adequately. Putting all of your code in one module counts as an egregious design failure.

- Your project will be graded based on the following three parts:

  1. TASK-1: Implementation of basic RTP/RTCP/RTSP protocols (20%)
     If you implement the program correctly, you will get full mark of this part.
  2. TASK-2: Implementation of RTP server and client (60%)
     If you implement the server and it works, you will get 20 credits. If you implement the client and it works, you will get another 20 credits. Implementing more features will bring you additional scores. More pay for more work.
  3. Project report (20%)
     You should include a short introduction to your system, RTP/RTCP/RTSP commands you implemented, notable features of your system and screenshots in your report. Anything you think valuable, e.g., difficulties you encountered, bugs you fixed, can be included in the report. However, **the report should not be longer than FIVE pages.**

3 RTP/RTCP/RTSP BACKGROUND

The Real-time Transport Protocol (RTP) is a network protocol for delivering audio and video over IP networks. RTP is used in communication and entertainment systems that involve streaming media, such as telephony, video teleconference applications including WebRTC, television services and web-based push-to-talk features. RTP typically runs over User Datagram Protocol (UDP).

RTP is used in conjunction with the RTP Control Protocol (RTCP). While RTP carries the media streams (e.g., audio and video), RTCP is used to monitor transmission statistics and quality of service (QoS) and aids synchronization of multiple streams. RTP is one of the technical foundations of Voice over IP and in this context is often used in conjunction with a signaling protocol such as the Session Initiation Protocol (SIP) which establishes connections across the network.

The Real Time Streaming Protocol (RTSP) is a network control protocol designed for use in entertainment and communications systems to control streaming media servers. The protocol is used for establishing and controlling media sessions between end points. Clients of media servers issue VHS-style commands, such as play, record and pause, to facilitate real-time control of the media streaming from the server to a client (Video On Demand) or from a client to the server (Voice Recording).

3.1 RTP PROTOCOL

RTP is the Internet-standard protocol for the transport of real-time data, including audio and video. It can be used for media-on-demand as well as inte1 active 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. Fig. 3.1 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

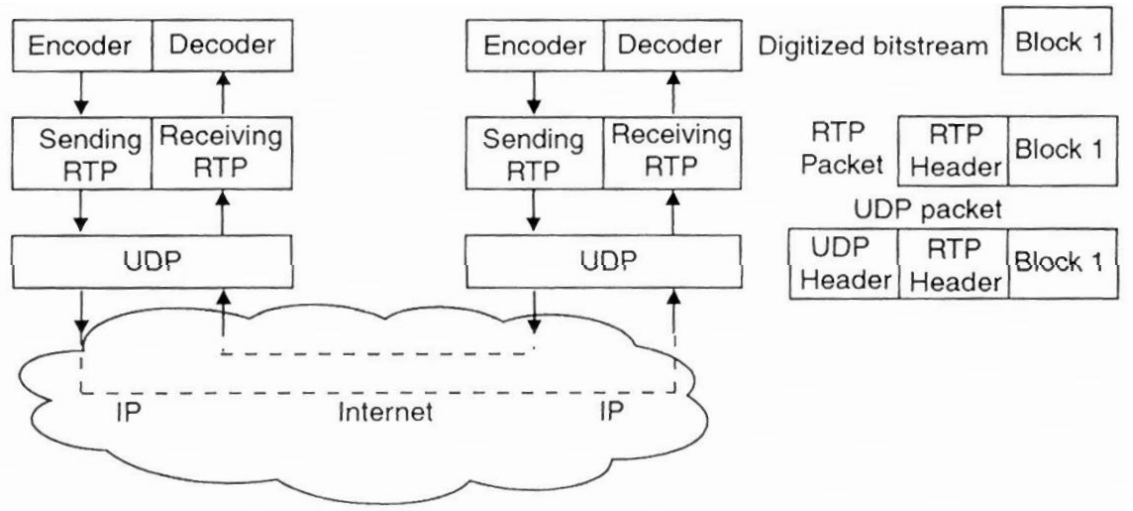


Figure 3.1: Real-time Transport Protocol.

control part of the protocol, RTCP, provides support for real-ti.me 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 Schulzrimle, 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 ransport 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 Applicati on Level Framing (ALF). This principle was proposed by Clark and Tennenhouse in 1990 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 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

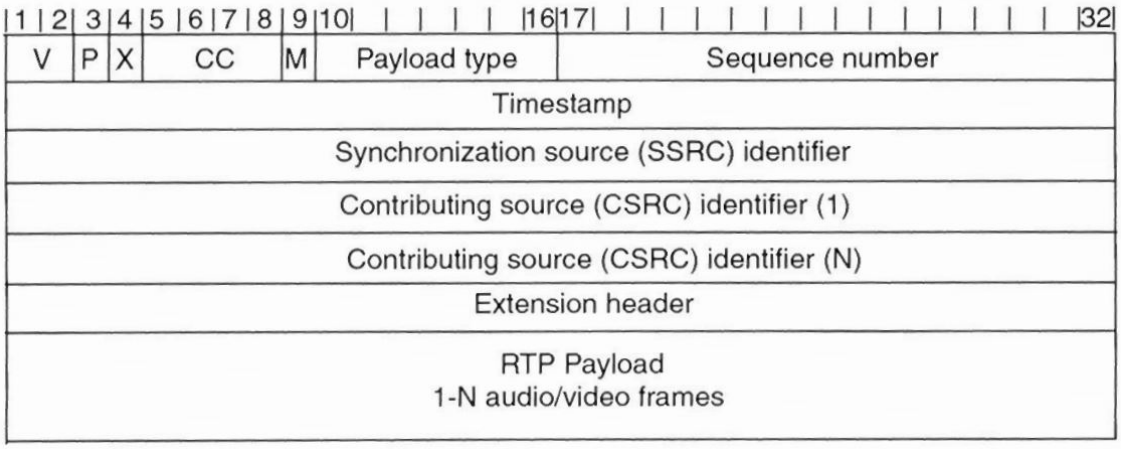


Figure 3.2: RTP header format.

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 requir‘ e 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
Fig. 3.2 shows the header format used by RTP. The first 12 bytes are always present whereas the con- tributing 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 apph.cations. Information specific to a single application is carried in the RTP payload.

V - nersion: 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 Fig. 3.3.

Example specifications include PCM, MPEGI/MPEG2 audio and video MPEG video, Sun CelB vicleo 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

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

Figure 3.3: Payload types (PT) for some Audio and Video encodings.

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 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.

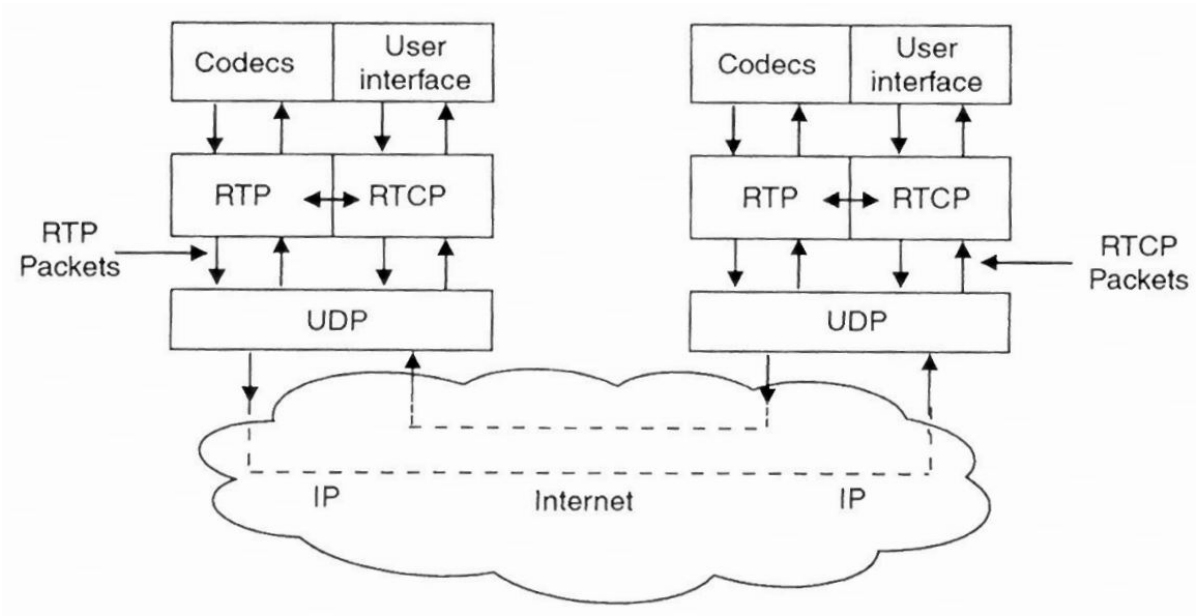


Figure 3.4: Real-time Transport Control Protocol.

3.2 RTCP PROTOCOL

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 Fig. 3.4.

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. RTCP is also used to deliver informatl on 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 pack-

ets, 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 specifc 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.

3.3 RTSP PROTOCOL

RTSP defined in RFC 2326 (the latest version is RTSP v2.0 in RFC 7826), is an application-level protocol that enables control over the delivery of data with real-time properties over IR Such control includes pausing playback, repositioning playback to future or past point of time, fast forwarding, and rewinding playback. These functionalities are simll ar 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". 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 descn ption vl a 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.

4 Helpful Links

- RTP and RTCP protocol `https://tools.ietf.org/html/rfc3550`
- RTSP protocol v2.0 `https://tools.ietf.org/html/rfc7826`

- RTP payload specifications https://tools.ietf.org/html/rfc3551
- RTP payload for H.264 video https://tools.ietf.org/html/rfc6184
- RTP payload for text conversations https://tools.ietf.org/html/rfc4103