
18-345 Network Design Project

Real-Time Audio/Video Conferencing Transport Service and Protocol Design Specification Final

ABSTRACT

The specification describes a protocol and transport service that can be used to deliver real-time audio and video conferencing streams between pairs of IP hosts. It combines elements of TCP, UDP and RTP to deliver a service that provides a connection-oriented, unidirectional, best-effort message stream with facilities to detect, correct and report packet loss, estimate one-way trip delay and jitter, and reconstruct source message timing on the destination to maintain accurate playback.

Copyright © 2001 by Chris Lord and Ryan Dhuse. All Rights Reserved.

Authors:	Chris Lord, Ryan Dhuse
Software Version:	Proposed
Document Version:	0.0.003
Edit Number:	17
Last Revised:	11/30/2001 12:19 PM
Classification:	Internal Use Only
Location:	/afs/clord/Documents/18-345/RAVC Design.doc
Date Printed:	5/7/2002 10:39 PM

Table Of Contents

1	Introduction.....	1
1.1	Requirements.....	1
1.1.1	Application Factors and Constraints	1
1.1.2	Host Factors and Constraints	1
1.1.3	Network Factors and Constraints	1
1.1.3.1	Campus Infrastructure and Interconnections	1
1.1.3.2	Internet Packet Error and Loss.....	2
1.1.3.3	Internet Packet Delay.....	2
1.2	Transport Services and Protocol Features	4
1.2.1	Unidirectional Connection Orientation	4
1.2.2	Message Orientation	4
1.2.3	Segmentation and Sequencing	4
1.2.4	Rate Preservation and Delay Bounds	5
1.2.5	Error Detection and Correction.....	5
1.2.6	Payload Identification	5
1.2.7	Time Stamps and Delay	5
1.2.8	Delivery and Rate Monitoring	5
1.2.9	Congestion Control	5
2	Protocol Design.....	6
2.1	General Protocol Overview	6
2.1.1	Protocol Entities.....	6
2.1.2	Message Summary	6
2.1.3	Message Sequence	7
2.1.4	Protocol State Transitions	8
2.1.5	Retransmissions and Timeouts.....	8
2.1.6	Common Message Format	8
2.1.6.1	Byte Order, Alignment, and Data Formats	8
2.1.6.2	Fixed Header Format	9
2.2	Connection Establishment	9
2.2.1	Connect and Connect Confirm Messages	10
2.3	Connection Termination.....	10
2.3.1	Disconnect and Disconnect Confirm Message.....	11
2.4	Bulk Data Flow	11
2.4.1	Data Message	11
2.4.2	Data Confirm Message.....	12
2.5	Connection Monitoring and Maintenance	12
2.5.1	Keepalive Message	12
2.5.2	Resynchronization Message.....	13
2.5.3	Status Report Message.....	13
2.6	Failure Behavior.....	13
3	Transport Service Interface	14
3.1	Event Notifications.....	14
4	System Design.....	15
4.1	Producer (Master) Design.....	15
4.1.1	Packet Queuing and Rate Control	15
4.1.2	Connection Monitoring	15
4.1.3	FEC Generation	15
4.1.4	Congestion Control	16
4.2	Consumer (Slave) Design.....	16

4.2.1	Error Detection and Correction.....	16
4.2.2	Connection Monitoring.....	16
4.2.3	Statistics Collection.....	16
4.2.4	Jitter-Buffer.....	17

Revision History

Date	Version	Authors	Description
22-Nov-01	0.0.001	CCL	Initial
27-Nov-01	0.0.002	RPD	Added producer FEC and congestion control
28-Nov-01	0.0.003	CCL/RPD	Editing

Related Web Sites

Organization	Site	Comment
IANA	www.iana.org	Internet Assigned Numbers Authority.
IETF	www.ietf.org	Internet Engineering Task Force standards body.
RFC Archive	www.faqs.org	Internet FAQs, RFCs and Standards archive.
TCP-Friendly Research	/www.psc.edu/networking/tcp_friendly.html	Pittsburgh Supercomputing Center Research Repository on TCP-Friendly congestion control for non-TCP protocols.
Video Quality Research	http://www.its.bldrdoc.gov/n3/video/	
Cooperative Association for Internet Data Analysis	http://www.caida.org/cgi-bin/skitter_summary/main.pl	Reports of roundtrip delays throughout the Internet
National Laboratory for Applied Network Research	http://watt.nlanr.net/	Reports of roundtrip delays throughout the Internet

References, Related Documents and Standards

- [1] Baker, F., "Requirements for IP Version 4 Routers," RFC 1812, June 1995.
- [2] Basith, S., "Digital Video: An Introduction," May 24, 1996.
- [3] Braden, R., "Requirements for Internet Hosts—Communication Layers," RFC 1122, October 1989.
- [4] Comer, D., *Internetworking with TCP/IP: Volume 1 - Principles, Protocols and Architecture*, 1995.
- [5] Finger, R., "Measuring Quality in Videoconferencing Systems," Intel Corporation, November 1, 1997.
- [6] Floyd, S., et al, "Equation-Based Congestion Control for Unicast Applications." *SIGCOMM 2000 Proceedings*, May 2000.
- [7] Floyd, S., Fall, K., "Promoting the Use of End-to-End Congestion Control in the Internet," *IEEE/ACM Transactions on Networking*, May 3, 1999.
- [8] Lin, D., "Real-Time Voice Transmissions Over the Internet," Thesis, 1999.
- [9] Mills, D., "Network Time Protocol Version 3," RFC 1305, March 1992.
- [10] Postel, J. and J. Reynolds. "File Transfer Protocol," STD 9, RFC 959, October 1985.
- [11] Schulzrinne, et al., "RTP: A Transport Protocol for Real-Time Applications," Internet Draft, draft-ietf-avt-rtp-new-10.ps, July 20, 2001.
- [12] Stevens, W. Richard, *TCP/IP Illustrated: Volume 1 The Protocols*, 1994.
- [13] Video Coding Experts Group, "Report of the Ad Hoc Committee on Consideration of Simulation Conditions and Evaluations for Error Resilience Testing," October 22, 1999.

Many of these references are available online in [/afs/clord/Documents/18-345/References](http://afs/clord/Documents/18-345/References). The list does not include the associated RFCs which can be found at online repositories such as www.faqs.org.

1 Introduction

The specification describes a protocol and transport service that can be used to deliver real-time audio and video conferencing streams between pairs of IP hosts. It combines elements of TCP, UDP and RTP to deliver a service that provides a connection-oriented, unidirectional, best-effort message stream with facilities to detect, correct and report packet loss, estimate one-way trip delay and jitter, and reconstruct source message timing on the destination to maintain accurate playback.

1.1 Requirements

The requirements affecting the design of the protocol fall into three broad categories: application, host and network.

1.1.1 Application Factors and Constraints

Frame rates in video conferencing are different than television and conventional video, where a new video frame is presented between 24 and 30 times per second. Videoconferencing frame rates are constantly changing. The degree of motion in the subject, the amount of detail, and the percentage of the screen that changes from one frame to the next can influence frame rate [5]. Therefore, the protocol must support varying message rates and preserve the rate at which these messages are delivered to the receiver.

Furthermore, the application has the ability to alter the frame rate based on conditions reported by the underlying protocol layer. Therefore, the transport service must support a mechanism of measuring the rate and delivery of frames and providing feedback to the application so the application can adjust the frame rate to prevailing conditions. This feedback must be both timely and accurate.

Insufficient frame rate is one of the most noticeable video deficiencies. Without a minimum of 24 frames per second, the video will be noticeably jerky. In addition, the missing frames will contain important audio synchronization data. In other words, if the movement of a person's lips is missing due to dropped frames during capture or playback, it is impossible for a human user to match the audio correctly with the video [2]. The application does have the ability to handle missing frames and partial frames, although this will be visible to the user. The greater the loss, the more poorer the perceived quality.

The abrupt jerkiness that is characteristic of poor quality digital video can also be caused by sudden variations in the rate which frames drawn on the screen. Linearity is a measure of how consistent the video frame rate is maintained and is another critical factor in video quality.

1.1.2 Host Factors and Constraints

There are no specific requirements imposed by the hosts implementing this protocol. Hosts are assumed to have ample memory and processing power to support any protocol satisfying the application bounds on the delay and throughput.

1.1.3 Network Factors and Constraints

1.1.3.1 Campus Infrastructure and Interconnections

The target campus network infrastructure provides local connections between 10 and 100 Mb/s. Campuses are interconnected via single connections to the Internet through commercial carriers of fixed capacities between 1.5 and 45 Mb/s.

There are predictable time-based load variations in campus traffic with most congestion between campuses occurring on the connections to the commercial carriers. Because of the higher LAN speeds within the campus, the time-based load variations are more noticeable between campuses as shown in Figure 1-1.

1.1.3.2 Internet Packet Error and Loss

The protocol must operate across the existing IP-based Internet. From an extensive study of the Internet backbone conducted in 1999 by the ITU-T Video Coding Experts Group [13], the following are key observations affecting real-time audio and video streams:

- There are virtually no bit errors in IP packets.
- There is no observed relationship between the packet size and packet loss rate.
- There is no observed relationship between the bit rate and the packet loss rate.
- Packet losses typically occur randomly or in very short bursts of two or three packets.
- Packet loss rates are dependent on the connection provider and time of day.

The above observations would seem to imply that congestion control is not required since the tests were conducted without any end-to-end congestion control and did not experience significant loss. If a protocol were to take advantage of this, it would penalize the majority of IP best-effort traffic which is congestion-control by TCP. There has been much recent research into the negative impacts and extreme unfairness against TCP traffic by protocols that do not employ (or implement inadequate) congestion control mechanisms [7]. The result has been proposals to identify and restrict disproportional bandwidth and unresponsive flows within the Internet at times of congestion. Unresponsive flows are those that fail to reduce their offered load in response to increased packet drops; disproportionate bandwidth flows are those that use considerably more bandwidth than others during congestion. [6]. This imposes a “good-neighbor” requirement, namely, the protocol should be TCP-friendly.

This does not, however, mean that it must employ the same TCP congestion control mechanism: reducing the sending rate in half in response to a single packet drop would severely impair streaming performance. It does mean that the protocol must monitor flows for potential congestion using rate and packet delivery information, it must provide feedback to the application which may be able to reduce its offered load, and it must monitor the application to ensure compliance with the feedback and, if necessary, reduce offered load unilaterally through delays and drops.

1.1.3.3 Internet Packet Delay

The videoconferencing application will initially be deployed between the Carnegie Mellon and Stanford campuses. To better understand the delays involved in Internet communication between the campuses, 30 days of observations for the month of October 2001 were obtained from the National Laboratory for Applied Network Research (NLANR). These observations measured the roundtrip time using ICMP between the two campuses every minute for a total of 43,200 samples. These samples were analyzed to understand delay variation across days of the week and times of the day. There was little variation across days, but significant variation with time as illustrated in Figure 1-1.

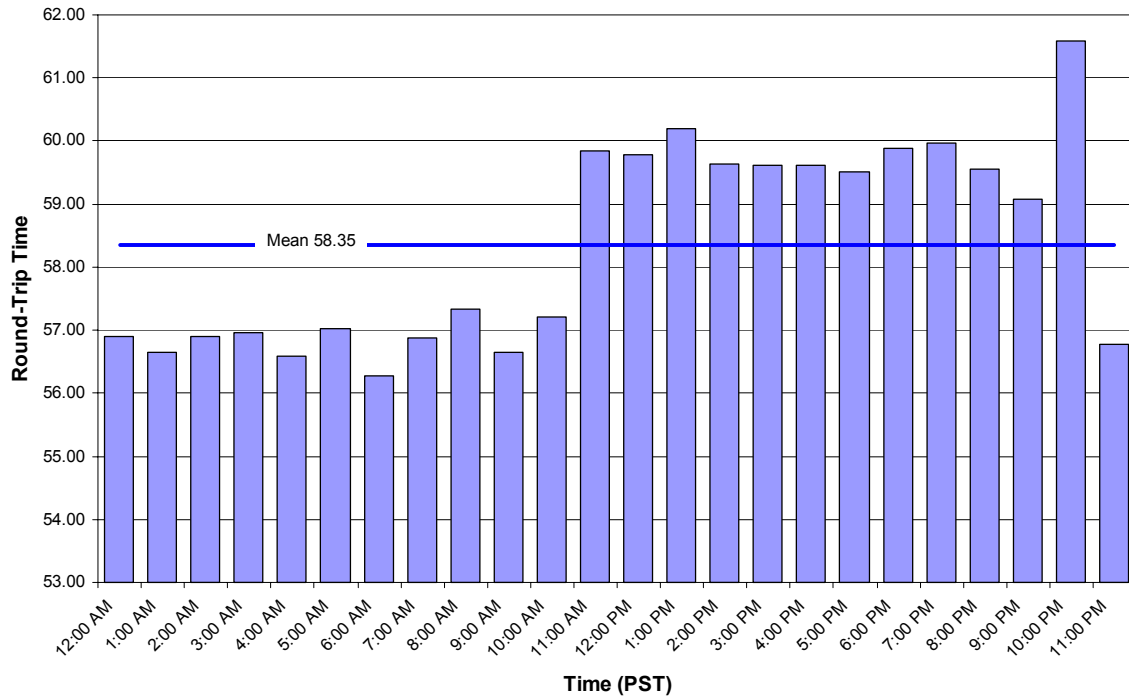


Figure 1-1: Roundtrip Time Distributions Between Stanford and CMU

The mean roundtrip time of 58.35ms was significantly less than the general Internet roundtrip times observed across the Internet by the CAIDA skitter probes during a week within the observation period, as shown in Figure 1-2..

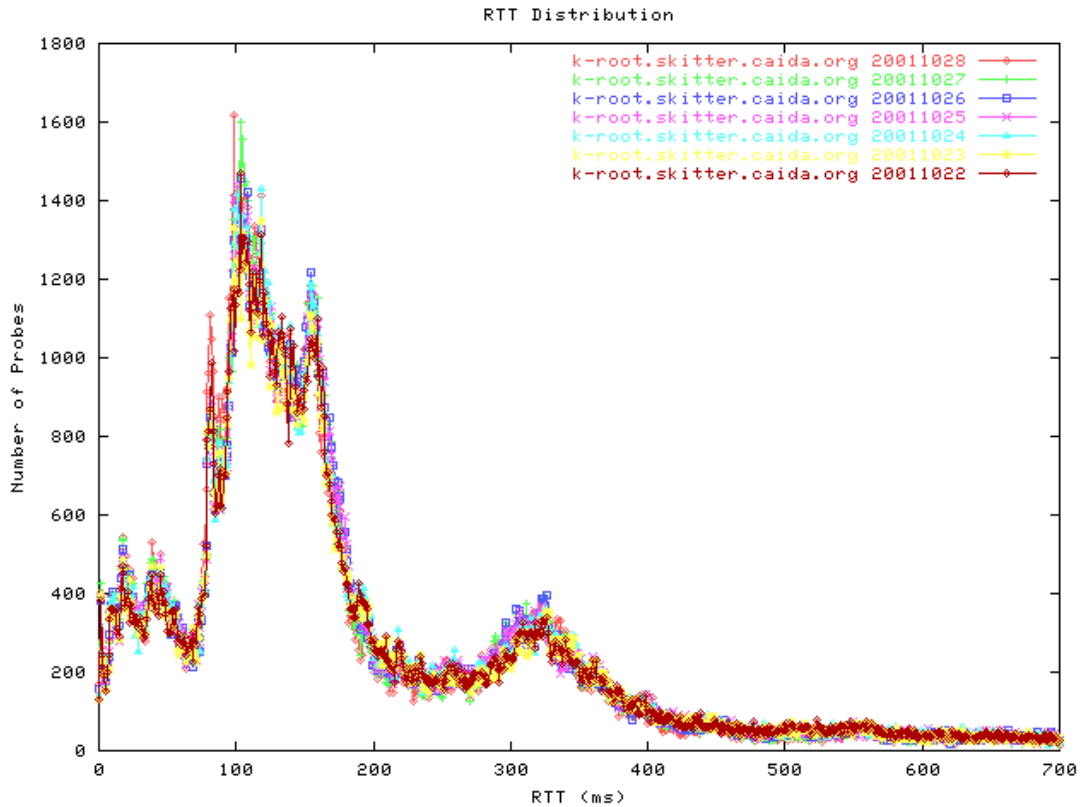


Figure 1-2: Internet Roundtrip Time Distribution

These results reveal that it is possible to satisfy the maximum end-to-end delay (100 ms) in playback-quality constraint for the majority of Internet sites. Most of the probe locations experiencing roundtrip times greater than 200ms lie outside the United States. Conferencing with these sites will violate the delay constraint and degrade the quality of service. Since an application has no way to control the transmission delay, it can only reduce the processing and buffering delays at both ends and hopefully achieve an acceptable voice quality.

1.2 Transport Services and Protocol Features

This section summarizes the specific features and services provided by the transport service using the RAVC protocol in light of the requirements imposed by the application and the operating environment. The descriptions are given in terms of the following:

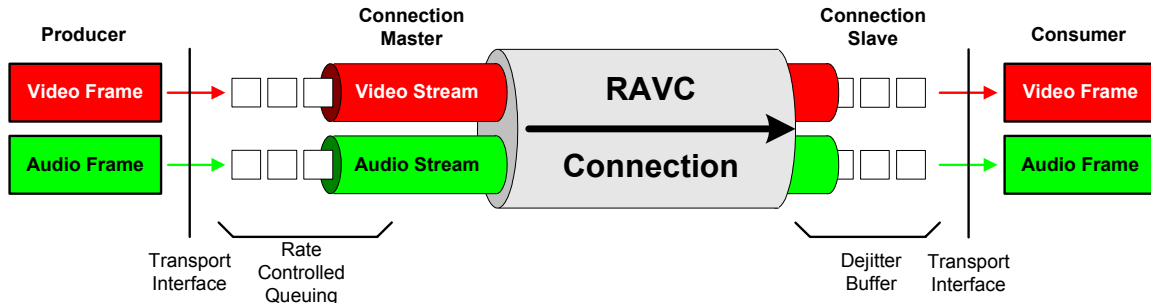


Figure 1-3: Transport and Protocol Overview

1.2.1 Unidirectional Connection Orientation

The transport service provides a unidirectional connection-oriented service between a source and destination. The connection establishment and teardown follow a similar sequence to that of TCP and uses similar state transitions. Unlike TCP, protocol packets are asymmetric between the master (the side performing the active open) and the slave (the side performing the passive open). The connection master is considered to be the source of conferencing data and the slave is considered the destination. Bidirectional communication (where both act as data sources) requires the establishment of two connections in opposite directions. Note that it is recommended that another protocol, such as TCP, be used for the stream destination to authenticate with the stream source and to negotiate features and options.

1.2.2 Message Orientation

The transport service provides for best-effort delivery of complete messages. These consist of either video or audio frames (or application-specific units) or control data. The source specifies a delivery flag on each message, which in turn specifies whether receipt of a partial message is acceptable or if messages must be received in their entirety. Because the protocol only supports best-effort delivery, uncorrectable packet loss in the first case will result in a sparse message being delivered to the destination application (missing data is guaranteed to be zero). In the second case, the message with uncorrectable packet loss will be discarded. It is up to the application to provide detection and recovery of this event, if necessary.

1.2.3 Segmentation and Sequencing

Application message or frame data is segmented into packets are sized to meet the maximum transmission unit between the source and destination and to avoid the overhead of IP-level fragmentation and reassembly. Path MTU discovery is performed using the same mechanisms as TCP.

Each packet contains a message sequence number which identifies the byte position of the payload in the owning message. Message order is preserved between the source and destination (with the exception of discarded messages) using a combination of a message identifier (a monotonically increasing message sequence number) and packet timestamps.

1.2.4 Rate Preservation and Delay Bounds

To the maximum extent possible, messages are delivered to the destination application at intervals that match the rate at which messages were received from the source application. This is subject to the bounds of the maximum acceptable delay established during the passive open which determines the maximum size of the dejitter buffer. Messages and packets that cannot be delivered within the delay bounds are discarded. The delay must be large enough to accommodate the one-way transmission time of packets while still delivering acceptable service.

1.2.5 Error Detection and Correction

The protocol supports error detection using a per-packet additive checksum that covers the protocol header and payload. This checksum is identical to that used in TCP and is described in [12]. Packets that fail the checksum are discarded.

The protocol provides forward error correction (FEC) using a redundancy packet that consists of an XOR of all packets to the start of the message or the last FEC packet. The interval of FEC packets is based on half the maximum allowed delay with at least one FEC packet sent per message. The FEC packet allows for the complete reconstruction of a single missing packet with only a minor penalty on delay. Because of the tight bounds on conferencing delay, there is insufficient time to detect and retransmit missing packets. Furthermore, the characteristic loss of only single packets means most errors will be corrected at much lower cost.

1.2.6 Payload Identification

Each packet belongs to one of three possible message types: audio, video and control. These types are supplied by the application layer on the source and delivered to the application layer on the destination. These allow multiple non-duplicated but related streams to be multiplexed across the same connection without appending additional data to the frame contents. Application-layer communication of non-stream data can be done using control messages.

1.2.7 Time Stamps and Delay

The connection master establishes the base time for all future packets during connection setup. All packets from the source to the destination are stamped with a delta time from this initial time (and therefore independent) measured in milliseconds. These delta times are used to monitor the condition of the connection (discussed in the next section) and to establish the intervals within the bounds of the dejitter buffering. It is not necessary for the master and slave clocks to be synchronized since the slave is only responsible for coordinating the inter-arrival times with the inter-departure times.

1.2.8 Delivery and Rate Monitoring

The connection slave reports the estimated one-way trip time and jitter as well as a packet count and byte statistics on a regular basis of not less than once every message. Reports can be requested more frequently by the connection master if desired. The protocol also provides a mechanism to obtain more detailed statistics as necessary.

1.2.9 Congestion Control

The protocol is TCP-friendly in the sense that it detects congestion conditions decreases offered load in the face of such congestion. This is accomplished from reports sent by the connection slave containing information about the estimated one-way transmission time, the number of lost, corrupt and discard frames, and the amount of time covered by buffered but undelivered data, if any.

2 Protocol Design

Neither TCP nor UDP are well suited for the transmission of real-time data audio and video streams. TCP provides a useful connection-oriented service, but the reliable delivery introduces unacceptable delays and the byte-orientation requires does not preserve the natural frame boundaries or separate inter-arrival times. UDP, on the other hand, is a connectionless best-effort service that preserves message boundaries—but not timing or ordering.

Most of the current real-time applications use a third protocol, the Real-time Transport Protocol (RTP) [11]. Despite the name, this is more of an application protocol than transport protocol and is frequently layered on UDP. A key feature provided by RTP is packet timestamps so that random delays resulting from other network traffic can be compensated for at a destination. That is, by using time stamps, packets can be buffered and removed from the buffers in a correct sequence.

The RAVC protocol design described here draws heavily on all three protocols to provide a connection-oriented, unidirectional, best-effort message stream between two IP endpoints with facilities to detect, correct and report packet loss, monitor delay and jitter, and reconstruct source message timing on the destination.

2.1 General Protocol Overview

Murphy's Law applies to networking. Any networking protocol must be well-behaved in the face of malformed packets, misinformation, and occasional failures of links, routers and systems. The data channel protocol should perform gracefully in response to willful management and configuration mistakes. Putting this requirement another way, RAVC continues the Internet tradition of being conservative in what is sent, but liberal in what is received.

2.1.1 Protocol Entities

There are two distinct protocol entities in RAVC a producer and a consumer. Connections are asymmetric to better reflect the differences between audio and video stream producers (also called the connection master) and audio and video stream consumers (also called the connection slave).

As indicated earlier, there may be cases where the slave initiates a conferencing session (as would be likely in a distance learning application where remote sites “subscribe” to a session in progress). While RAVC can be used to do this, the unidirectionality and asymmetry would require two connections: Data packets with control payloads sent from consumer to producer via one and from producer to consumer over the other. Furthermore, control data will be unnecessarily delayed by the hold time on the slave even though they are unaffiliated with data streams. These complications suggests that RAVC is best used in concert with TCP or some other protocol to handle session negotiation, setup, authentication and otherwise facilitate consumer communication with producer.

2.1.2 Message Summary

The following table summarizes the packets supported by the RAVC protocol. The directionality is indicated by Master (for data producer), Slave (for data consumer) or Both for symmetric packet initiation.

Message	Source	Description
Connect	Master	Establishes a connection between a data producer and a potential data consumer.
Connect Confirm	Master	Confirms to a consumer that a connection is fully established.
Data	Master	Delivers data from the producer to the consumer with the ability to identify message boundaries and reconstruct intermessage times.
Data Confirm	Slave	Confirms the receipt of a complete message and provides updated statistics on the quality of the connection.
Report	Master	Requests a report of all statistics on the quality of service the connection is providing. This is a superset of the Data Confirm response.
Report Confirm	Slave	Provides point-in-time statistics at the time the response is generated.
Resync	Master	Reestablishes fundamental parameters set when the connection is created without destroying the connection.
Resync Confirm	Slave	Sets the slave parameters on a connection resync.

Disconnect	Both	Initiates the termination of a connection.
Discon Confirm	Both	Confirms that a connection is fully terminated.

Each of these is described in greater detail in the sections that follow.

2.1.3 Message Sequence

The typical protocol packet sequence is illustrated in Figure 2-1.

1. Connections are always initiated by the system that will be producing the data, the producer or connection master. The exchange follows a three-way handshake (Connect, Ack, Confirm) so both sides agree on that the connection is established. The data consumer must have previously executed a passive open (listen) and be expecting this connection at the time it occurs. The connection must be acknowledged by the consumer.
2. Data packets flow from the producer to the consumer as messages are generated. The packets are interleaved in a manner that preserves (within a granularity determined by the transmission time for the MTU) the intermessage time at the transport interface. This intermessage timing is recorded in each packet so the consumer can reconstruct it within the bounds of the jitter buffer and hold time.
3. At the end of each application-layer message, the slave response with a Data Confirm packet which reports updated jitter, one-way trip and loss statistics. This is separate from the Data Ack packets which can be requested at any point by the master by setting the ARQ flag.
4. During periods of inactivity, the master must send zero-length data packets at a rate determined at connection setup to keep the connection alive and to ensure that statistics reflect the relatively recent (rather than potentially ancient) past.
5. Sometimes the statistics returned in each Data Confirm packet may not be sufficient to supply applications (or users) with all the information they need to understand the quality of service being delivered to the consumer. Under these circumstances, the Report packet can request a more detailed statistics report.
6. Connection termination is the only symmetric portion of the protocol: either end of a connection may initiate a disconnect. Like connect, this sequence follows a three-way handshake (Disconnect, Ack, Confirm) to ensure both sides agree on the termination.

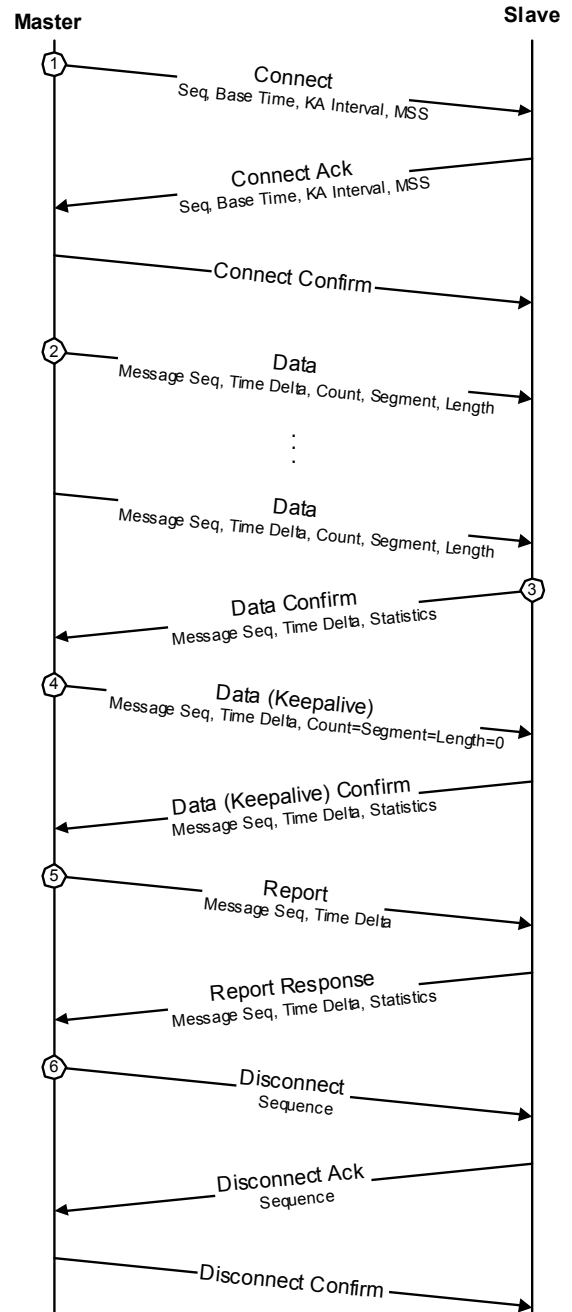


Figure 2-1: Typical Protocol Message Exchange

There are other variations that could exist in the packet sequence. The above is meant to illustrate a more typical pattern of exchange.

2.1.4 Protocol State Transitions

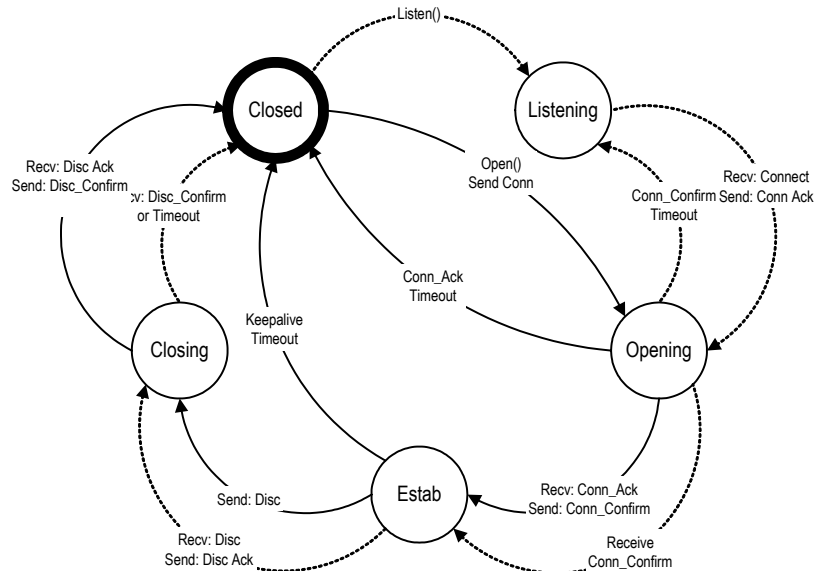


Figure 2-2: Protocol State Transitions

A connection can be in one of five legal states which determine the protocol packets sent and expected. These are illustrated in Figure 2-2. The dotted paths represent those taken by a passive open and passive disconnect.

2.1.5 Retransmissions and Timeouts

Most state transitions are protected by a retry limit and a timeout determined by the implementation. In general, the timeouts are set to the largest expected roundtrip time including processing overhead on the target (n). The recommended retry limit is two (which gives a total of three attempts with a state transition timeout of $3n$). Figure 2-3 illustrates this typical usage.

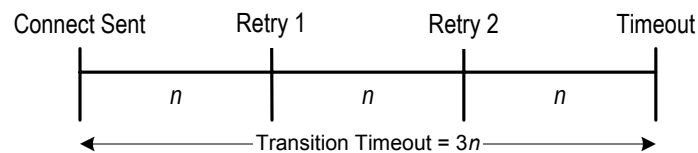


Figure 2-3: Retry Limit and State Transition Timeouts

2.1.6 Common Message Format

All protocol packets carry the same 20-byte protocol header. Each protocol packet type may have additional packet-specific fields following the fixed header. In all cases, however, the packet headers are of fixed length based on their type and therefore there is no packet header length.

2.1.6.1 Byte Order, Alignment, and Data Formats

Like TCP and UDP, all integer fields are carried in network byte order (also known as big-endian), that is, most significant byte first. Unless otherwise noted, numeric constants are in decimal. All header data is aligned to its natural length: 16-bit fields are aligned on even byte offsets and 32-bit fields are aligned at offsets divisible by four. All unused, reserved or padding fields are zero. Absolute time is represented using the timestamp format of the Network Time Protocol (NTP), which is in seconds relative to 0h UTC on 1 January 1900 [9]. The full resolution NTP timestamp is a 64-bit unsigned fixed-point number with the integer part in the first 32 bits and the fractional part in the last 32 bits. This provides precision down to approximately 200 picoseconds.

The conservative principle also applies: reserved fields must be zero when sent and must be ignored when received.

2.1.6.2 Fixed Header Format

The fixed header format is 20 bytes and must occur at the start of every packet. The format of this header is shown in Figure 2-4.

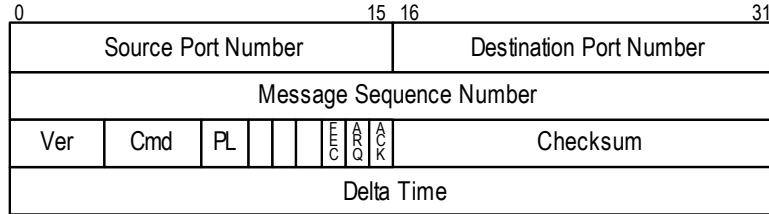


Figure 2-4: Fixed Packet Header Format

Common Header Format		
Field	Bits	Description
Source Port	16	The source port number identifies the sending application process. It is assigned by the transport layer during an active or passive open and is unique among all active port numbers on the local system.
Destination Port	16	The destination port number identifies the receiving application process. It is assigned by the transport layer during an active or passive open and is unique among all active port numbers on the local system.
Message Sequence Number	32	The message sequence number is a monotonically increasing sequence number established by the connection master and used by the connection slave to distinguish and order packets and to identify outdated packets. All packets belonging to the same message contain the same message sequence number. Because messages can be interleaved, it is possible for packets to arrive at the destination with alternating sequence numbers. The destination tracks the minimum acceptable sequence number.
Version	4	The version number is incremented for major incompatible revisions of the protocol. Most revisions can be handled without updating the protocol version.
Command	4	The command code identifies the specific packet type and hence packet format following the fixed-length header. There are eight command types currently defined (Connect, Connect Confirm, Data, Data Confirm, Resync, Resync Confirm, Report, Report Confirm, Disconnect, Disconnect Confirm). Each of these may optionally require an acknowledgement as discussed in the flags field.
Payload	2	The payload field identifies the type of payload in a data packet (control, video, audio) and is used to multiplex multiple related streams across the same connection. This is supplied by the sending application and delivered to the receiving application. It is not used by the transport.
Flags	6	There are currently only three flags defined: acknowledgement requested (ARQ), acknowledgement (ACK), and forward error correction (FEC). When the ARQ flag is set, the receiver should reflect the packet header with the ACK flag set and the ARQ flag cleared. When the FEC flag is set, it indicates this packet contains payload redundancy information for the packets since the start of the message or since the last FEC packet was sent.
Checksum	16	The checksum covers the entire protocol header and payload. [TBD]
Delta Time	32	The delta time conveys the delta from the absolute time established during connection setup. The very first protocol exchange is a Connect and Connect acknowledgment, both of which contain the absolute timestamp. This timestamp is used in data packets to reconstruct interframe timings and to calculate the one-way trip times. This timestamp is in units of milliseconds for a total field value of approximately 50 days. Sessions that exceed this length of time must be terminated and reestablished or send a Resync packet to reset the base time.
Total	128	

2.2 Connection Establishment

Connections are always initiated by the system that will be producing the data, the producer or connection master. The exchange follows a three-way handshake (Connect, Ack, Confirm) so both sides agree on that the connection is established. The data consumer must have previously executed a passive open (listen) and be expecting this connection at the time it occurs. The connection must be acknowledged by the consumer.

2.2.1 Connect and Connect Confirm Messages

The Connect and Connect Confirm packets share identical formats except for the command code. This is shown in Figure 2-5.

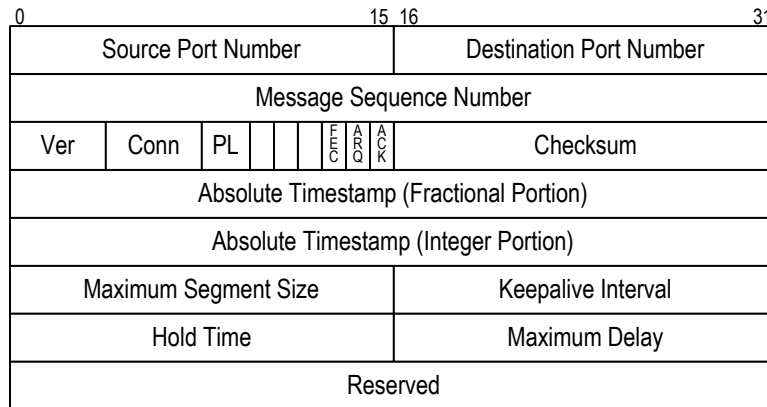


Figure 2-5: Connect and Connect Confirm Packet Header Format

Connect and Connect Confirm Format		
Field	Bits	Description
Fractional Time	32	In Connect packets, this is not a delta time but rather the fractional component of the absolute time.
Absolute Time	32	The absolute time is given as an NTP timestamps, a 64-bit unsigned fixed-point number, in seconds relative to 0 h on 1 January 1900. The integer part is in these 32 bits and the fraction part in prior 32 bits. The precision of this representation is about 200 picoseconds.
Maximum Segment Size	16	The maximum segment size (MSS) is the largest packet that can be sent (or received) by a protocol entity and is usually based on the maximum transmission unit that can be sent without incurring fragmentation. When a connection is established, each end advertises its MSS. The connection master uses the smaller of the advertised or received MSS. Since data transfer is always in segments of MSS and the maximum number of message segments is 16-bits, the maximum message size is 65535*MSS. With the default wide-area MTU of 576 bytes, this results in a message size over 33MB using a 532 byte payload (576 bytes less the 20 byte IP header and 24 byte RAVC data header).
Keepalive Interval	16	The keepalive interval is the rate, in milliseconds, at which keepalive packets should be sent by the connection master in the absence of all other traffic. When a connection is established, each end advertises its interval. The connection master uses the smaller of the advertised or received interval (likewise, the slave uses the small of it advertised or received interval). It is recommended that this interval be sufficiently high (on the order of seconds) to avoid generating unnecessary traffic.
Hold Time	16	The hold time, in milliseconds, as specified by the application on the consumer (producers can supply a recommended value, but the ultimate decision is the consumer and producers must always use the consumer's value).
Maximum Delay	16	The maximum delay, in milliseconds, as specified by the application on the consumer (producers can supply a recommended value, but the ultimate decision is the consumer and producers must always use the consumer's value). This is used by the producer to identify packets (and entire messages) that would arrive too late and therefore should not be transmitted at all.
Total	128	

Unlike TCP, there cannot be a simultaneous open. If two sides attempt to create a connection simultaneously, the result will be two separate unidirectional connections, not one.

2.3 Connection Termination

Connection termination is the only symmetric portion of the protocol: either end of a connection may initiate a disconnect. Like connect, this sequence follows a three-way handshake (Disconnect, Ack, Confirm) to ensure both sides agree on the termination.

2.3.1 Disconnect and Disconnect Confirm Message

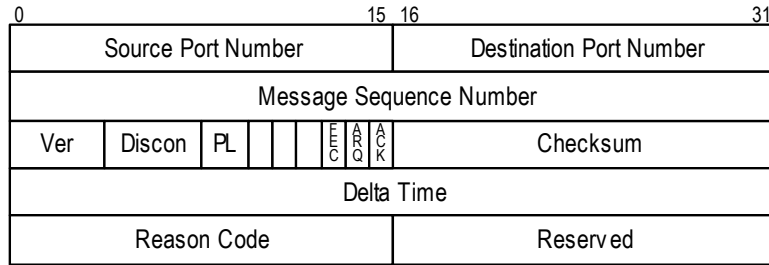


Figure 2-6: Disconnect and Disconnect Confirm Packet Header Format

Disconnect and Disconnect Confirm Format		
Field	Bits	Description
Reason Code	16	An optional application-specific reason code which identifies to the receiver why a connection was terminated. This is used for reporting and logging by the application; it has no meaning to the protocol.
Total	16	

2.4 Bulk Data Flow

Data packets flow from the producer to the consumer as messages are generated. The packets are interleaved in a manner that preserves (within a granularity determined by the transmission time for the MTU) the intermessage time at the transport interface. This intermessage timing is recorded in each packet so the consumer can reconstruct it within the bounds of the jitter buffer and hold time

2.4.1 Data Message

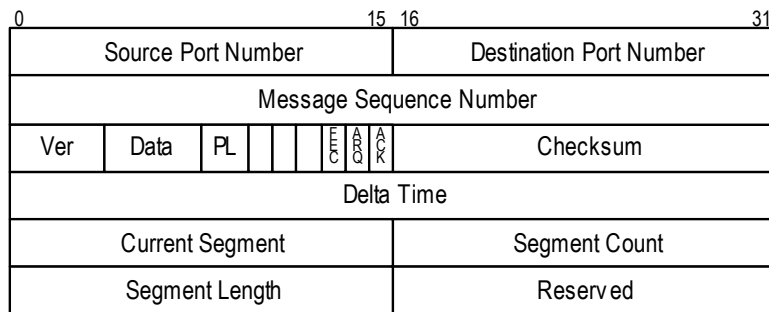


Figure 2-7: Data Packet Header Format

Data Format		
Field	Bits	Description
Current Segment	16	The current segment number. This must be between 1 and the segment count, inclusive. All segments except the last must carry a payload equal to the size of the negotiated MTU so that packets can arrive out of order and still be directly copied to their eventual buffer location.
Segment Count	16	The total number of segments in the message identified by the message sequence number.
Segment Length	16	The length of this segment. For all but the last segment, this should be equal to the MTU.
FEC Flag	1	The FEC flag identifies this packet as a FEC packet for the message sequence number. This can be used by the receiver to replace a lost or discarded packet or in place of a packet that might be received out of order (for example, when the FEC packet arrives before a message packet).
Total	49	

2.4.2 Data Confirm Message

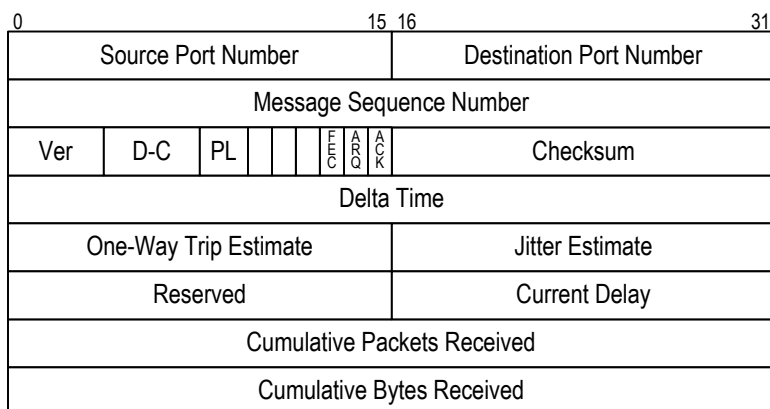


Figure 2-8: Data Confirm Packet Header Format

Data Confirm Format		
Field	Bits	Description
One-Way Trip Estimate	16	The one-way trip time, in milliseconds, as estimated on the receiver using a smoothed calculation similar to that used by TCP to estimate the RTT time. The seed values for this calculation come from comparing the interpacket arrival times on the consumer with the difference between the interpacket departure times in the packet time stamps. The difference represents the one-way trip time.
Jitter Estimate	16	An estimate of the statistical variance of the packet interarrival time, in milliseconds. The jitter estimate is defined to be the mean deviation (smoothed absolute value) of the difference in packet spacing at the receiver compared to the sender for a pair of packets. This is equivalent to the difference in the "relative transit time" for the two packets; the relative transit time is the difference between a packet's timestamp and the packet arrival time.
Current Delay	16	The current delay, in milliseconds, at the time the data confirm message was sent. This corresponds to the size of the data in the dejitter buffer and can be compared to the maximum delay reported at connection setup.
Cumulative Packets Received	32	The total number of packets received. This can be compared with the total number of packets sent on the producer to detect possible congestion conditions. This is value will wrap at the maximum. If wrapping isn't acceptable, it can be reset via the Resync packet.
Cumulative Bytes Received	32	The total number of bytes received. This can be compared with the total number of bytes sent on the producer to detect possible congestion conditions. This is value will wrap at the maximum. If wrapping isn't acceptable, it can be reset via the Resync packet.
Total	112	

2.5 Connection Monitoring and Maintenance

2.5.1 Keepalive Message

Keepalives must be sent in the absence of data from the master to the slave at an interval not less than that negotiated at connection establishment. These allow both ends to monitor the condition of the path, keep the state of the one-way trip time current, and to detect errors in either the path or the systems that would require application cleanup and intervention.

There is no specific keepalive packet. Rather, a keepalive is simply a Data packet with a message length of zero (segment count, current segment and segment length are all zero). The connection slave will respond to this with a Data_Confirm packet as soon as it is received. Keepalive packets are not subject to congestion control or traffic shaping.

Note that this also provides a mechanism for the master to obtain connection statistics during bulk data transfer at intervals greater than those determined by the length of the application-level messages. A “keepalive” packet can be sent at any time to obtain “quick” statistics on the connection. More detailed information is available through the status report packets.

2.5.2 Resynchronization Message

There are situations where the connection master will need to resynchronize a consumer. This can occur when streams have been interrupted at the source. It can also occur when fundamental parameters (such as MSS, base time or keepalive interval) governing the current session have changed. It is also necessary when a connection has been active for more than 49 days (such that the delta time may wrap). Rather than teardown and reestablish the connection which would likely involve application intervention, the protocol provides a means of resynchronizing within an existing connection.

The format of this packet is identical to that of a Connect packet and has a matching Confirm response (except for the command codes). Upon receipt of a Resync, the slave will reset the base time and other parameters to the lesser of those supplied or those returned in the Confirm response. All message sequence numbers logically less than the supplied value will be discarded after a Resync. This message also resets all protocol estimates and counters.

2.5.3 Status Report Message

The connection master can request a detailed status report at any time. This is a superset of the information provided in the Data Confirm packet. In the current iteration of the protocol, however, it is identical to the Data Confirm response which provides sufficient feedback to the producer.

2.6 Failure Behavior

Packets that are invalid for the current protocol state are counted and discarded since they are likely the result of either excessively delayed packets, protocol violations or malicious attacks.

3 Transport Service Interface

The transport service interface is modeled after the traditional socket interface (in particular, the Windows Sockets Version 2 API) with extensions to support notifications of rate change events and specification of the maximum acceptable hold-time. Compatible aspects of the API are not duplicated here, rather a summary of the various functions is provided with details on how usage differs when operating with a transport service using RAVC.

3.1 Event Notifications

The standard event notification functions are supported by the transport interface. The implication is that the interface is designed to be used by a threaded application which supports synchronization mechanisms and asynchronous input/output operations. Common event management functions are listed below.

Function	Description
CloseEvent	Destroys an event object.
CreateEvent	Creates an event object.
EnumNetworkEvents	Discovers occurrences of network events.
EventSelect	Associates network events with an event object.
ResetEvent	Resets an event object.
SetEvent	Sets an event object.
WaitForEvents	Blocks on a single event objects.
WaitForMultipleEvents	Blocks on multiple event objects.

The transport service supports three events that can be used by the conferencing application:

Event	User	Description
Connect	Consumer	Indicates a passive open has completed and a new connection fro the producer is available.
Status Change	Both	Indicates the quality of the connection has changed (in either a positive or negative direction). This is provided primarily for the producer so that the application can adjust the rate of deliver to the consumer. Possible causes for this notification include congestion or improvement following congestion. The application can use one of the status query functions to obtained detailed information about the event.
Receive	Consumer	A message is available. The message notification is delayed on the receiver by an amount equal to the hold time specified at the passive open. Separate message notifications are provided at a rate that matches the message times on the sender.
Error	Both	An error has occur on the connection.

4 System Design

The information in this chapter constitutes implementation guidelines and recommendations. The RAVC protocol provides the raw material for providing a streaming service, but it is the proper use of the protocol that determines whether that service meets the needs of an application.

4.1 Producer (Master) Design

This section describes characteristics of the producer of data streams.

4.1.1 Packet Queuing and Rate Control

It is expected that the application will deliver complete messages consisting of audio and video data for transmission every T ms. However, these messages will be of various sizes due to the encoding method that is used. The transport service uses a method of segmenting messages into packet and interleaving packets of distinct messages in order to smooth the arrival rate of complete messages on the receiver.

Four variables are used by the sender's rate control algorithm. The first, P_x , is the number of packets that have yet to be sent in message x . The oldest message from the application layer with unsent packets is denoted by $x = 1$. When P_1 reaches 0, each P value is shifted and P_1 once again denotes the number of packets remaining from the oldest message from the application layer. H is the current estimate of the receiver's hold time. T is the current estimate of the one-way trip time. In addition, a correction factor, C , is used to account for errors in estimates or future congestion. A C value of 1 represents no adjustment while a value of .8 represents a 20% adjustment.

Using these variables, we find the estimated number of packets that can be sent before the current hold time expires

is given by $\left\lfloor \frac{H}{T} \right\rfloor$. For each message x that has been received from the application layer for transmission, the value

$Z = \frac{P_x}{C \cdot X \cdot \left\lfloor \frac{H}{T} \right\rfloor}$ is computed. A single packet from the message with the highest Z value is then transmitted. If

two or more Z values are equal, a packet from the oldest message is transmitted. This algorithm has the effect of delaying smaller messages in order to deliver larger messages sooner. Overall, complete messages are received at a regular rate regardless of size.

4.1.2 Connection Monitoring

The keepalive protocol packet serves as a way to gather information about the current state of the connection. The data confirm message that is sent in response to the keepalive packet supplies the sender with the updated information it needs to make decisions. This information is essential to the rate control algorithm and useful when deciding how many redundant FEC packets to send.

4.1.3 FEC Generation

Forward error correction (FEC) provides a way to correct errors in a message on the receiver's side without retransmission. It is most useful in mediums where a high loss rate is commonplace, such as wireless networks.

Most data loss during transmission manifests itself as lost packets rather than bit errors within packets that are received. To this end, the RAVC protocol does not use FEC on a packet level, but rather on a message level. When a message is passed from the application to the transport layer, it is used as a seed for the FEC algorithm. The FEC algorithm then generates $N + K$ packets, where N is the minimum number required to transmit the message and K is a fixed number of redundant FEC packets. These packets are then transmitted. Studies show that when packet loss

occurs, it is usually limited to one or two packets [13]. In the event of such a packet loss, most of the redundant packets will still be received. Due to the nature of FEC, it does not matter which N packets are received, only that N packets are in fact received. The entire message can then be reconstructed by the consumer for delivery to the application layer.

4.1.4 Congestion Control

The RAVC protocol contains no explicit form of congestion notification. The producer can, however, learn that conditions are deteriorating by monitoring trends in the one-way trip time estimate that is reported in each data confirm message. When the one-way trip time estimate exceeds acceptable levels, the RAVC protocol sends a congestion warning event to the application layer in hopes that the application will use this information to adjust its rate and/or volume of transmission.

After the congestion warning event has been sent, further congestion warnings are temporarily disabled. A keepalive packet is inserted at the end of the current send queue and its sequence number is noted. When the data confirm message for this keepalive packet is received (or has timed-out), congestion warnings are re-enabled. The updated one-way trip time from the data confirm message of this keepalive packet may trigger a congestion warning. This prevents multiple congestion warning events from occurring before the application has a chance to respond.

In the event that congestion persists for after the application layer has been notified several times, the sender should begin to drop messages in an attempt to alleviate the congestion. Packets which belong to messages that have already exceeded their maximum acceptable delay should be dropped first. These packets are late even before they are sent; they have no hope of meeting their deadline. Next, the video portion of the smallest message in the queue should be dropped. Although dropping small messages does little to alleviate congestion on the grand scale of things, a small message implies a small amount of encoded video data. This usually means that there is very little change from the previous frame in the video stream. Therefore, dropping these frames produces small and sometimes unperceivable glitches in the video output. The dropping algorithm will also avoid dropping sequential messages if possible.

4.2 Consumer (Slave) Design

4.2.1 Error Detection and Correction

When the receiver detects lost packets due to gaps in message segment numbers, it may still be able to recover the message due to redundant FEC packets from the sender. The sender generates an FEC packet every so many Data packets. If Data packet is lost or discarded, the receiver can wait until it receives the next FEC packet to reconstruct a single missing packet.

4.2.2 Connection Monitoring

In addition to maintaining an idle connection, the keepalive protocol packet provides a way for the consumer to update the producer with its current status. It is vital that the consumer keep the producer informed of the current one-way trip time estimate in order for the consumer to react to increasing congestion.

4.2.3 Statistics Collection

The characteristics of Internet traffic are changing constantly. To make the different parts of a system work in a dynamic network, statistics collection is needed. This works in conjunction with the protocol headers and packets which facilitate gathering and reporting of statistics.

At the receiver, the arrival time of each packet is recorded. From the time stamp in each packet header and the arrival time, end-to-end delay and jitter can be estimated and fed into the jitter manager. The receiver can also estimate packet loss based on missing message segments, but it cannot know all lost packets since it is unable to

know the number of packets in a message sequence if all packets of missing sequence are lost. Fortunately, this rarely occurs so the estimate is reasonable for diagnostic purposes. The accurate tracking of lost packets is handled by the sender; the receiver always reports the cumulative packets received in the status reports.

4.2.4 Jitter-Buffer

The purpose of a buffering algorithm at the receiver is to delay the first synchronized audio and video frame by a hold time before it is played. The hold time is critical because it determines the playback time of the first frame therefore the playback times of all subsequent frames. If it is set too large, there will be unnecessary end-to-end delay. If it is set too small, then the playback quality will suffer due to the burstiness of the received stream.

The maximum acceptable delay is the maximum amount of time that playback can be delayed and still be considered acceptable. This value is a bound on the hold time and therefore is always greater than the hold time. From a system design perspective, the delay essentially determines the size of the dejitter buffer; the hold time determines the skew between wallclock time on the receiver when a frame is played back and the equivalent wallclock time when the frame was generated. The system design strives to maintain a constant hold time.