Internet Engineering Task Force
INTERNET-DRAFT draft-ietf-avt-rtp-03.ps

Audio-Video Transport WG
H. Schulzrinne/S. Casner
AT&T/ISI

September 15, 1993
Expires: 11/01/93

RTP: A Transport Protocol for Real-Time Applications

Status of this Memo

This document is an Internet Draft. Internet Drafts are working documents of the Internet Engineering Task Force (IETF), its Areas, and its Working Groups. Note that other groups may also distribute working documents as Internet Drafts.

Internet Drafts are draft documents valid for a maximum of six months. Internet Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet Drafts as reference material or to cite them other than as a "working draft" or "work in progress."

Please check the I-D abstract listing contained in each Internet Draft directory to learn the current status of this or any other Internet Draft.

Distribution of this document is unlimited.

Abstract

This memorandum describes a protocol called RTP suitable for the end-to-end network transport of real-time data, such as audio, video or simulation data for both multicast and unicast transport services. The data transport is augmented by a control protocol (RTCP) designed to provide minimal control and identification functionality particularly in multicast networks. RTP and RTCP are designed to be independent of the underlying transport and network layers. The protocol supports the use of RTP-level translators and bridges. Within multicast associations, sites can direct control messages to individual sites. The protocol does not address resource reservation and does not guarantee quality-of-service for real-time services.

This specification is a product of the Audio-Video Transport working group within the Internet Engineering Task Force. Comments are solicited and should be addressed to the working group's mailing list at `rem-conf@es.net` and/or the authors.

Contents

1 Introduction

This memorandum specifies a transport protocol for real-time applications. A discussion of real-time services and algorithms for their implementation and some of the RTP design decisions can be found in the current version of the companion Internet draft `draft-ietf-avt-issues`. The transport protocol provides end-to-end delivery services for data with real-time characteristics, for example, interactive audio and video. RTP itself does not provide any mechanism to ensure timely delivery or provide other quality-of-service guarantees, but relies on lower-layer services to do so. It does *not* guarantee delivery or prevent out-of-order delivery, nor does it assume that the underlying network is reliable and delivers packets in sequence. The sequence numbers included in RTP allow the end system to reconstruct the sender's packet sequence, but sequence numbers might also be used to determine the proper location of a packet, for example in video decoding, without necessarily decoding packets in sequence. RTP does not provide quality-of-service guarantees. RTP is designed to run on top of a variety of network and transport protocols, for example, IP, ST-II or UDP.[1] RTP transfers data in a single direction, possibly to multiple destinations if supported by the underlying network. A mechanism for sending control data in the opposite direction, reversing the path traversed by regular data, is provided.

While RTP is primarily designed to satisfy the needs of multi-participant multimedia conferences, it is not limited to that particular application. Storage of continuous data, interactive distributed simulation, active badge, and control and measurement applications may also find RTP applicable. Profiles are used to instantiate certain header fields and options for particular sets of applications. A profile for audio and video data may be found in the companion Internet draft `draft-ietf-avt-profile`.

The current Internet does not support the widespread use of real-time services. High-bandwidth services using RTP, such as video, can potentially seriously degrade other network services. Thus, implementors should take appropriate precautions to limit accidental bandwidth usage. Application documentation should clearly outline the limitations and possible operational impact of high-bandwidth real-time services on the Internet and other network services.

This document defines a packet format shared by two protocols:

- the real-time transport protocol (RTP), for exchanging data that hs real-time properties. The RTP header consists of a fixed-length portion plus optional control fields;
- the RTP control protocol (RTCP), for conveying information about the participants in an on-going session. RTCP consists of additional header options that may be ignored without affecting the ability to receive data correctly. RTCP is used for "loosely controlled" sessions, i.e., where there is no explicit membership control and set-up. Its functionality may be subsumed by a session control protocol, which is beyond the scope of this document.

2 RTP Protocol Use Scenarios

The following sections describe some aspects of the use of RTP. The examples were chosen to illustrate the basic operation of applications using RTP, not to limit what RTP may be used for. In these examples, RTP is carried on top of IP and UDP, and follows the conventions established by the profile for audio and video specified in the companion Internet draft `draft-ietf-avt-profile`.

[1] For most applications, RTP offers insufficient demultiplexing to run directly on IP.

2.1 Simple Multicast Audio Conference

A working group of the IETF meets to discuss the latest protocol draft, using the IP multicast services of the Internet for voice communications. Through some allocation mechanism, the working group chair obtains a multicast group address; all participants use the destination UDP port specified by the profile. The multicast address and port are distributed, say, by electronic mail, to all intended participants. The mechanisms for discovering available multicast addresses and distributing the information to participants are beyond the scope of RTP.

The audio conferencing application used by each conference participant sends audio data in small chunks of, say, 20 ms duration. Each chunk of audio data is preceded by an RTP header; RTP header and data are in turn contained in a UDP packet. The Internet, like other packet networks, occasionally loses and reorders packets and delays them by variable amounts of time. To cope with these impairments, the RTP header contains timing information and a sequence number that allow the receivers to reconstruct the timing seen by the source, so that, in our case, a chunk of audio is delivered to the speaker every 20 ms. The sequence number can also be used by the receiver to estimate how many packets are being lost. Each RTP packet also indicates what type of audio encoding (such as PCM, ADPCM or GSM) is being used, so that senders can change the encoding during a conference, for example, to accommodate a new participant that is connected through a low-bandwidth link.

Since members of the working group join and leave during the conference, it is useful to know who is participating at any moment. For that purpose, each instance of the audio application in the conference periodically multicasts the name, email address and other information of its user. Such control information is carried as RTCP SDES options within RTP messages, with or without audio data (see Section 6.2). These periodic messages also provide some indication as to whether the network connection is still functioning. A site sends the RTCP BYE (Section 6.3) option when it leaves a conference. The RTCP QOS (Section 6.4) option indicates how well the current speaker is being received and may be used to control adaptive encodings.

2.2 Bridges

So far, we have assumed that all sites want to receive audio data in the same format. However, this may not always be appropriate. Consider the case where participants in one area are connected through a low-speed link to the majority of the conference participants, who enjoy high-speed network access. Instead of forcing everyone to use a lower-bandwidth, reduced-quality audio encoding, a *bridge* is placed near the low-bandwidth area. This bridge resynchronizes incoming audio packets to reconstruct the constant 20 ms spacing generated by the sender, mixes these reconstructed audio streams, translates the audio encoding to a lower-bandwidth one and forwards the lower-bandwidth packet stream to the low-bandwidth sites.

After the mixing, the identity of the high-speed site that is speaking can no longer be determined from the network origin of the packet. Therefore, the bridge inserts a CSRC option (Section 5.2.1) into the packet containing a list of short site identifiers to indicate which site(s) “contributed” to that mixed packet. An example of this is shown for bridge B1 in Fig. 1. As name and location information is received by the bridge in SDES options from the high-speed sites, that information is passed on to the receivers along with a mapping to the CSRC identifiers.

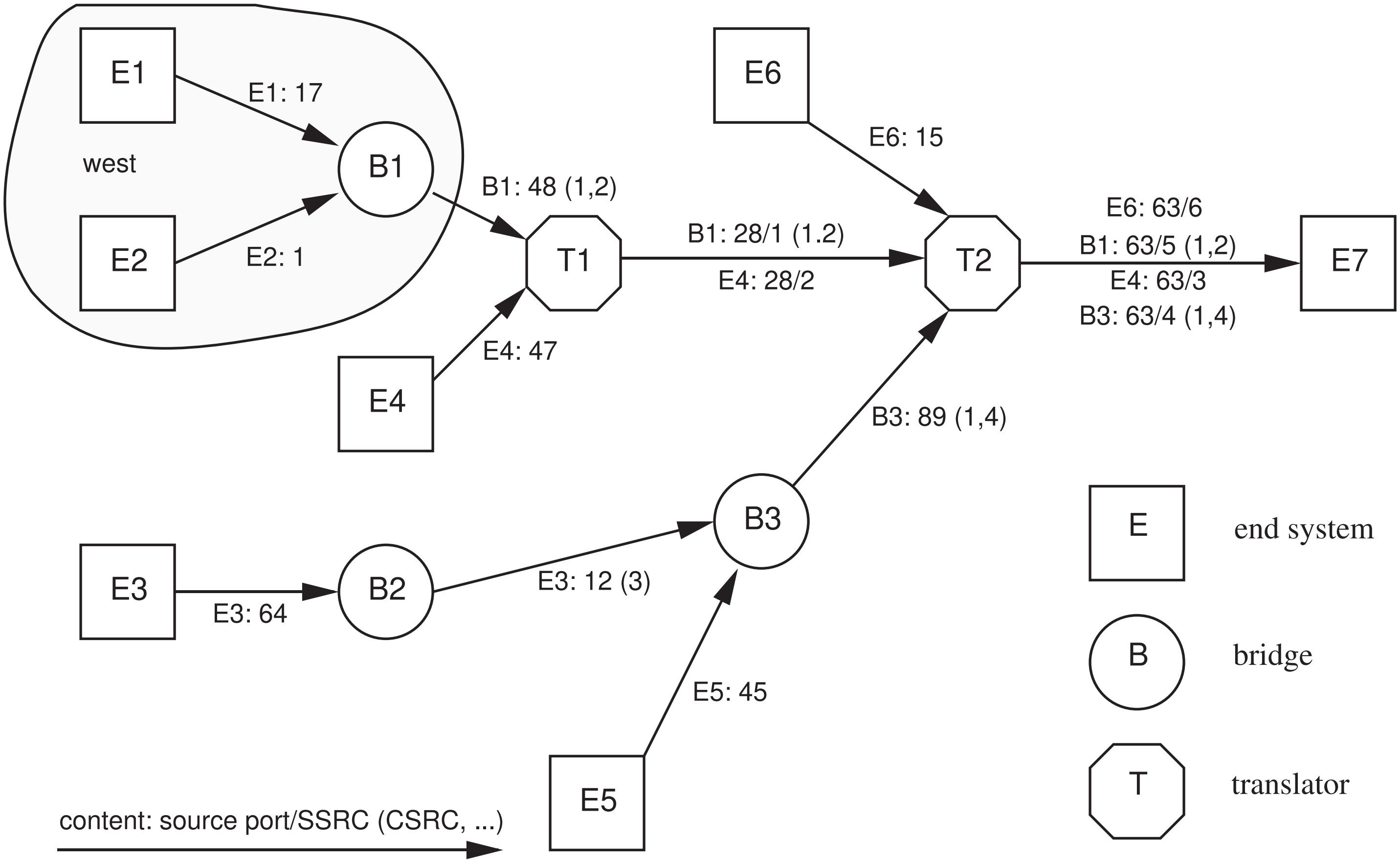

Figure 1: Sample RTP network with end systems, bridges and translators

2.3 Translators

Not all sites are directly accessible through IP multicast. For these sites, mixing may not necessary, but a translation of the underlying transport protocol is. RTP-level gateways that do not restore timing or mix packets from different sources are called *translators* in this document. Application-level firewalls, for example, will not let any IP packets pass. Two translators are installed, one on either side of the firewall, the outside one funneling all multicast packets received through the secure connection to the translator inside the firewall. The translator inside the firewall sends them again as multicast packets to a multicast group restricted to the site's internal network. Other examples include the connection of a group of hosts speaking only IP/UDP to a group of hosts that understand only ST-II.

After RTP packets have passed through a translator, they all carry the network source address of the translator, making it impossible for the receiver to distinguish packets from different speakers based on network source addresses. Since each sending site has its own sequence number space and slightly offset timestamp space, the receiver could not properly mix the audio packets. (For video, it could not properly separate them into distinct displays.) Instead of forcing all senders to include some globally unique identifier in each packet, a translator inserts an SSRC option (Section 5.2.2) with a short identifier for the source that is locally unique to the translator. This also works if an RTP packet has to travel through several translators, with the SSRC value being mapped into a new locally unique value at each translator. An example is shown in Fig. 1, where hosts T1 and T2 are translators. The RTP packets from host E4 are identified with SSRC value 2, while those coming from bridge B1 are labeled with SSRC value 1. Similarly, translator T2 labels packets from E6, B1, E4 and B3 with SSRC values 6, 5, 3 and 4, respectively (or some other unique values).

2.4 Security

Conference participants would often like to ensure that nobody else can listen to their deliberations. Encryption, indicated by the presence of the ENC option (Section 5.4.1), provides that privacy. The encryption method and key can be changed during the conference by indexing into a table. For example, a meeting may go into executive session, protected by a different encryption key accessible only to a subset of the meeting participants.

For authentication, a number of methods are provided, depending on needs and computational capabilities. All these message integrity check (MIC) options (Sections 5.4.3 and following) compute cryptographic checksums, also known as message digests, over the RTP data.

3 Definitions

Payload is the data following the RTP fixed header and the RTP/RTCP options. The payload format and interpretation are beyond the scope of this memo. RTP packets without payload are valid. Examples of payload include audio samples and video data.

An *RTP packet* consists of the encapsulation specific to a particular underlying protocol, the fixed RTP header, RTP and RTCP options (if any) and the payload, if any. A single packet of the underlying protocol may contain several RTP packets if permitted by the encapsulation method.

A *(protocol) port* is the "abstraction that transport protocols use to distinguish among multiple destinations within a given host computer. TCP/IP protocols identify ports using small positive integers." [1] The transport selectors (TSEL) used by the OSI transport layer are equivalent to ports.

A *transport address* denotes the combination of network address, e.g., the 4-octet IP Version 4 address, and the transport protocol port, e.g., the UDP port. In OSI systems, the transport address is called transport service access point or TSAP. The destination transport address may be a unicast or multicast address.

A *content source* is the actual source of the data carried in an RTP packet, for example, the application that originally generated some audio data. Data from one or more content sources may be combined into a single RTP packet by a bridge, which becomes the synchronization source (see next paragraph). Content sources identify the logical source of the data, for example, to highlight the current speaker in an audio conference; they have no effect on the delivery or playout timing of the data itself. In Fig. 1, E1 and E2 are the content sources of the data received by E7 from bridge B1, while B1 is the synchronization source.

A *synchronization source* is the combination of one or more content sources with its own timing. Each synchronization source has its own sequence number space. The audio coming from a single microphone and the video from a camera are examples of synchronization sources. The receiver groups packets by synchronization source for playback. Typically a single synchronization source emits a single medium (e.g., audio or video). A synchronization source may change its data format, e.g., audio encoding, over time. Synchronization sources are identified by their transport address and the identifier carried in the SSRC option. If the SSRC option is absent, a value of zero is assumed for that identifier.

A *transport source* is the transport-level origin of the RTP packets as seen by the receiving end system. In Fig. 1, host T2, port 63 is the transport source of all packets received by end system E7.

A *channel* comprises all synchronization sources sending to the same destination transport address using the same RTP channel identifier.

An *end system* generates the content to be sent in RTP packets and consumes the content of received RTP. An end system can act as one or more synchronization sources. (Most end systems are expected to be a single synchronization source.)

An (RTP-level) *bridge* receives RTP packets from one or more sources, combines them in some manner and then forwards a new RTP packet. A bridge may change the data format. Since the timing among multiple input sources will not generally be synchronized, the bridge will make timing adjustments among the streams and generate its own timing for the combined stream. Therefore, bridges are synchronization sources, with each of the sources whose packets were combined into an outgoing RTP packet as the content sources for that outgoing packet. Audio bridges and media converters are examples of bridges. In Fig. 1, end systems E1 and E2 use the services of bridge B1. B1 inserts CSRC identifiers for E1 and E2 when they are active (e.g., talking in an audio conference). The RTP-level bridges described in this document are unrelated to the data link-layer bridges found in local area networks. If there is possibility for confusion, the term 'RTP-level bridge' should be used. The name bridge follows common telecommunication industry usage.

An (RTP-level) *translator* forwards RTP packets, but does not alter their sequence numbers or timestamps. Examples of its use include encoding conversion without mixing or retiming, conversion from multicast to unicast, and application-level filters in firewalls. A translator is neither a synchronization nor a content source. The properties of bridges and translators are summarized in Table 1. Checkmarks in parentheses designate possible, but unlikely actions. The options are explained in Sections 5.2, the RTCP options in Section 6.

	end sys.	bridge	translator
mix sources	–	√	–
change encoding	N/A	√	√
encrypt	√	√	(√)
sign for authentication	√	√	–
alter content	√	√	√
insert CSRC (RTP)	–	√	–
insert SSRC (RTP)	√	√	√
insert SDST (RTP)	√	√	–
insert SDES (RTCP)	√	√	–

Table 1: The properties of end systems, bridges and translators

A *synchronization unit* consists of one or more packets that are emitted contiguously by the sender. The most common synchronization units are talkspurts for voice and frames for video transmission. During playout synchronization, the receiver must reconstruct exactly the time difference between packets within a synchronization unit. The time difference between synchronization units may be changed by the receiver to compensate for clock drift or to adjust to changing network delay jitter. For example, if audio packets are generated at fixed intervals during talkspurts, the receiver has to play back packets with exactly the same spacing. However, if, for example, a silence period between synchronization units (talkspurts) lasts 600 ms, the receiver may adjust it to, say, 500 ms without this being noticed by the listener.

Non-RTP mechanisms refers to other protocols and mechanisms that may be needed to provide a useable service. In particular, for multimedia conferences, a conference control application may distribute encryption and authentication keys, negotiate the encryption algorithm to be used, and determine the mapping from the RTP format field to the actual data format used. For simple applications, electronic mail or a conference database may also be used. The specification of such mechanisms is outside the scope of this memorandum.

4 Byte Order, Alignment, and Reserved Values

All integer fields are carried in network byte order, that is, most significant byte (octet) first. This byte order is commonly known as big-endian. The transmission order is described in detail in [2], Appendix A. Unless otherwise noted, numeric constants are in decimal (base 10). Numeric constants prefixed by '0x' are in hexadecimal.

Fields within the fixed header and within options are aligned to the natural length of the field, i.e., 16-bit words are aligned on even addresses, 32-bit long words are aligned at addresses divisible by four, etc. Octets designated as padding have the value zero.

Textual information is encoded accorded to the UTF-2 encoding of the ISO standard 10646 (Annex F) [3,4]. US-ASCII is a subset of this encoding and requires no additional encoding. The presence of multi-octet encodings is indicated by setting the most significant bit to a value of one. An octet with a binary value of zero may be used as a string terminator for padding purposes. However, strings are not required to be zero terminated.

5 Real-Time Data Transfer Protocol – RTP

5.1 RTP Fixed Header Fields

The RTP header has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver| ChannelID |P|S|  format   |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     timestamp (seconds)       |     timestamp (fraction)      |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
| options ...                                                   |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

The first eight octets are present in every RTP packet and have the following meaning:

protocol version: 2 bits
Identifies the protocol version. The version number of the protocol defined in this memo is one (1).

channel ID: 6 bits
The channel identifier field forms part of the tuple identifying a channel (see definition in Section 3) to provide an additional level of multiplexing at the RTP layer. The channel ID field is convenient if several different channels are to receive the same treatment by the underlying layers or if a profile allows for the concatenation of several RTP packets on different channels into a single packet of the underlying protocol layer.

option present bit (P): 1 bit
This flag has a value of one (1) if the fixed RTP header is followed by one or more options and a value of zero otherwise.

end-of-synchronization-unit (S): 1 bit
This flag has a value of one in the last packet of a synchronization unit, a value of zero otherwise.[2]

format: 6 bits
The format field forms an index into a table defined through the RTCP FMT option or non-RTP mechanisms (see Section 3). The mapping establishes the format of the RTP payload and determines its interpretation by higher layers. If no mapping has been defined in this manner, a standard mapping is specified by the companion profile document, RFC TBD. Also, default formats may be defined by the current edition of the Assigned Numbers RFC.

sequence number: 16 bits
The sequence number counts RTP packets. The sequence number increments by one for each packet sent. The sequence number may be used by the receiver to detect packet loss, to restore packet sequence and to identify packets to the application.

timestamp: 32 bits
The timestamp reflects the wall clock time when the RTP packet was generated. Several consecutive RTP packets may have equal timestamps if they are generated at once. The timestamp consists of the middle 32 bits of a 64-bit NTP timestamp, as defined in RFC 1305 [5]. That is, it counts time since 0 hours UTC, January 1, 1900, with a resolution of 65536 ticks per second. (UTC is Coordinated Universal Time, approximately equal to the historical Greenwich Mean Time.) The RTP timestamp wraps around approximately every 18 hours.

The timestamp of the first packet within a synchronization unit is expected to closely reflect the actual sampling instant, measured by the local system clock. If possible, the local system clock should be controlled by a time synchronization protocol such as NTP. However, it is allowable to operate without synchronized time on those systems where it is not available, unless a profile or session protocol requires otherwise. It is not necessary to reference the local system clock to obtain the timestamp for the beginning of every synchronization unit, but the local clock should be referenced frequently enough so that clock drift between the synchronized system clock and the sampling clock can be compensated for gradually. Within one synchronization unit, it may be appropriate to compute timestamps based on the logical timing relationships between the packets. For audio samples, for example, the nominal sampling interval may be used.

5.2 The RTP Options

The packet header may be followed by options and then the payload. Each option consists of the F (final) bit, the option type designation, a one-octet length field denoting the total number of 32-bit words comprising the option (including F bit, type and length), followed by any option-specific

[2] As shown in Appendix A, the beginning of a synchronization unit can be readily established from this flag. If this flag were to signal the beginning of a synchronization unit instead, the end of a synchronization unit could not be established in real time.

data. The last option before the payload has the F bit set to one; for all other options this bit has a value of zero.

An application may discard options with types unknown to it. Private and experimental options should use option types 64 through 127. Fields designated as “reserved” or “R” are set aside for future use; they should be set to zero by senders and ignored by receivers.

Unless otherwise noted, each option may appear only once per packet. Each packet may contain any number of options. Options may appear in any order, unless specifically restricted by the option description. In particular, the position of some security options may have significance.

The RTP options have the following type values:

name	value
CSRC	0
SSRC	1
SDST	2
BOS	3
ENC	8
MIC	9
MICA	10
MICK	11
MICS	12

5.2.1 CSRC: Content source identifiers

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|   CSRC      |    length     | content source identifier   ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The content source option, inserted only by bridges, lists all sources that contributed to the packet. For example, for audio packets, all sources that were mixed together to create this packet are enumerated, allowing correct talker indication at the receiver. Each CSRC option may contain one or more 16-bit content source identifiers. The identifier values must be unique for all content sources received from a particular synchronization source on a particular channel; the value of binary zero is reserved and may not be used. If the number of content sources is even, the two octets needed to pad the list to a multiple of four octets are set to zero. There should only be a single CSRC option within a packet. If no CSRC option is present, the content source identifier is assumed to have a value of zero. CSRC options are not modified by RTP-level translators.

A conformant RTP implementation does not have to be able to generate or interpret the CSRC option.

5.2.2 SSRC: Synchronization source identifier

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|    SSRC     |  length = 1   |          identifier           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The SSRC option may be inserted by RTP-level translators, end systems and bridges. It is typically used only by translators, but it may be used by an end system application to distinguish several sources sent with the same transport source address. Multiple synchronization sources with the same transport source address (e.g., the same IP address and UDP port) must each insert a distinct SSRC identifier. Conversely, synchronization sources that are distinguishable by their transport address do not require the use of SSRC options. The SSRC value zero is reserved; the receiver treats the packet as if the SSRC option were not present. If no SSRC option is present, the transport source address is assumed to indicate the synchronization source. There must be no more than one SSRC option per packet; thus, a translator must remap the SSRC identifier of an incoming packet into a new, locally unique SSRC identifier. The SSRC option can be viewed as an extension of the source port number in protocols like UDP, ST-II or TCP.

An RTP receiver must support the SSRC option. RTP senders only need to support this option if they intend to send more than one source to the same channel using the same source port.

5.2.3 BOS: Beginning of synchronization unit

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|     BOS     |   length = 1  |        sequence number        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The sequence number within the options contains the sequence number of the first packet within the current synchronization unit. The BOS option allows the receiver to compute the offset of a packet with respect to the beginning of the synchronization unit, even if the last packet of the previous synchronization unit was lost. It is expected that many applications will be able to tolerate such a loss, and so will not use the BOS option but rely on the S bit.

5.3 Reverse-Path Option

With two-party (unicast) communications, having a receiver of data relay back control information to the sender is straightforward. Similarly, for multicast communications, control information can easily be sent to all members of the group. It may, however, be desirable to send a unicast message to a single member of a multicast group, for example to request retransmission of a particular data frame or to request/send a reception quality report. For this particular use, RTP includes a mechanism for sending so-called reverse RTP packets. The format of reverse RTP packets is exactly the same as for regular RTP packets and they can make use of all the options defined in this memorandum, except SSRC, as appropriate. The support for and semantics of particular options are to be specified by a profile. Reverse RTP packets travel through the same translators as forward RTP packets. A site distinguishes reverse RTP packets from forward RTP packets by their arrival port. Reverse RTP packets arrive on the same port that the site uses as a source port for forward (data) RTP packets. Only reverse RTP packets carry the SDST option; if RTP packets are carried directly within IP or other network-layer protocols, the presence of the SDST option signals that the packet is a reverse RTP packet.

A receiver of reverse RTP packets cannot rely on sequence numbers being consecutive, as a sender is allowed to use the same sequence number space while communicating through this reverse path with several sites. In particular, a receiver of reverse RTP packets cannot tell by the sequence numbers whether it has received all reverse RTP packets sent to it. The sequence number space of reverse RTP packets has to be completely separate from that used for RTP packets sent to the multicast group. If the same sequence number space were used, the members of the multicast group not receiving reverse RTP packets would detect a gap in their received sequence number space. The sender of reverse RTP packets should ensure that sequence numbers are unique, modulo wrap-around, so that they can, if necessary, be used for matching request and response. (Currently, no such request-response mechanism has been defined.) As a hypothetical example, consider defining a request to pan the remote video camera. After completing the request, the receiver of the request would send a generic acknowledgement containing the sequence number of the request to the requestor as an option (not as the packet sequence number in the fixed header).

The timestamp should reflect the approximate sending time of the packet. The channel identifier must be the same as that used in the corresponding forward RTP packets.

If many receivers send a reverse RTP packet in response to a stimulus in the data stream, the simultaneous delivery of a large number of packets back to the data source can cause congestion for both the network and the destination (this is known as an "ack implosion"). Thus reverse RTP packets should be used with care, perhaps with mechanisms such as response rate limiting and random delays to spread out the simultaneous delivery.

5.3.1 SDST: Synchronization destination identifier

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|    SDST     |  length = 1   |          identifier           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The SDST option is only inserted by RTP end systems and bridges if they want to send unicast information to a particular site within the multicast group. Packets containing an SDST option must not contain an SSRC option and vice versa. Packets containing a SDST option are always reverse RTP packets. The SDST option may be used to distinguish reverse RTP packets from forward RTP packets if the port-number mechanism described earlier in this section is not available, e.g., because RTP packets are carried directly within IP packets, without UDP.

If a forward RTP packet carries SSRC identifier X when sent from A to B, where A and B may be either two translators or an end system and a translator, the unicast reverse RTP packet will carry an SDST option with identifier X from B to A.

Consider the topology shown in Fig. 1. Assume that all forward RTP packets are addressed to destination port 8000. For the case that B1 wants to send a reverse packet to E1, B1 simply sends to the source address and port, that is, port 17 in this example. E1 can tell by the arrival on port 17 that the packet is a reverse packet rather than a regular (forward) packet.

The mechanism is somewhat more complicated when translators intervene. We focus on end system E7. E7 receives, say, video from a range of sources, E1 through E6 as indicated by the arrows. The transmission from T2 to E7 could be either multicast or unicast. Assume that E7 wants to send a retransmission request, a request to pan the camera, etc., to end system E4 and only to E4. E7 may not be able to directly reach E4, as E4 may be using a network protocol unknown to E7 or be located behind a firewall. According to the figure, video transmissions from E4 reach E7 through T2 with source port 63 and SSRC identifier 3. For the reverse message, E7 sends a message to T2, with destination port 63 and SDST identifier 3. T2 can look up in its table that it sends forward data coming from T1 with that identifier 3. T2 also knows that those messages from T1 carry SSRC 2 and arrive with source port 28. Just like E7, T2 places the SSRC identifier, 2 in this case, into the SDST option and forwards the packet to T1 at port 28. Finally, translator T1 consults its table to find that it labels packets coming from E4, port 47 with SSRC value 2 and thus knows to forward the reverse packet to E4, port 47. T1 can either place SDST value zero or no SDST option into that packet. Note that E4 cannot directly determine that E7 sent the reverse packet, rather than, say, E6. If that is important, a global identifier as defined for the QOS option needs to be included in the reverse packet.

Only applications that need to send or receive reverse control RTP packets need to implement the SDST option.

5.4 Security Options

The security options below offer message integrity, authentication and privacy and the combination of the three. Support for the security options is not mandatory, but see the discussion for the ENC option. The four message integrity check options — MIC, MICA, MICK and MICS — are mutually exclusive, i.e., only one of them should be used in a single RTP packet.

Combinations of one of the message integrity check options (MIC, MICA, MICK, or MICS) and the encryption (ENC) option described below can be used to provide a variety of security services:

confidentiality: Confidentiality means that only the intended receiver(s) can decode the received RTP packets; for others, the RTP packet contains no useful information. Confidentiality of the content is achieved by encryption. The presence of encryption and the encryption initialization vector is indicated by the ENC option.[3]

authentication and message integrity: In combination with certificates[4], the receiver can ascertain that the claimed originator is indeed the originator of the data (authentication) and that the data has not been altered after leaving the sender (message integrity). These two security services are provided by the message integrity check options. Certificates for MICA must be distributed through means outside of RTP. The services offered by MICA and MIC/MICK/MICS differ: With MIC/MICK/MICS, the receiver can only verify that the message originated within the group holding the secret key, rather than authenticate the sender of the message, while the MICA option affords true authentication of the sender.

authentication, message integrity, and confidentiality: By carrying both the message integrity check and ENC option in RTP packets, the authenticity, message integrity and confidentiality of the packet can be assured (subject to the limitations discussed in the previous paragraph).

The message integrity check is applied first to all parts of the outgoing packet to be authenticated, and the message integrity check option is prepended to those parts. Then the packet including the message integrity check option is encrypted using the shared secret key. The ENC option must be followed immediately by the message integrity check option, without any other options in between. The receiver first decrypts the octets following the ENC option and then authenticates the decrypted data using the signature contained in the message integrity check option.

For this combination of security features and group authentication, the combination ENC and MIC is recommended (instead of MICS or MICK), as it yields the lowest processing overhead.

A message integrity check option followed by an ENC option should not be used. All message integrity check options are computed over the fixed header, the first four octets of the message

[3] For efficiency reasons, this specification does not insist that content encryption only be used in conjunction with message integrity and authentication mechanisms. In most cases, it will be obvious to the person receiving the data if he or she does not possess the right encryption key.

[4] For a description of certificates see, for example, RFC 1422 or [6].

integrity check option and the data, that is, the remaining header options and payload that follow the message integrity check option. The MICK option includes the whole MICK option itself in the message integrity check. The fixed header is protected to foil replay attacks and reassignment to a different channel.

The message integrity check options and the ENC option shall not cover the SSRC and SDST options, i.e., SSRC and SDST must be inserted between the fixed header and the ENC or message integrity check options; SSRC and SDST are subject to change by translators that likely do not possess the necessary descriptor table (see below) and encryption keys. Translators that have the necessary keys and descriptor translation table may modify the contents of the RTP packet, unless the MICA option is used (see MICA description in Section 5.4.3).

All security options carry a one-octet descriptor field. This descriptor is an index into two tables, one for the message integrity check options, one for the ENC option, established by non-RTP means, containing digest algorithms (MD2, MD5, etc.), encryption algorithms (DES variants) and encryption keys or shared secrets (for the MICK option). All sources within the same channel share the same table; this reduces per-site state information. The descriptor value may change during a session, for example, to switch to a different encryption key.

The descriptor value zero selects a set of default algorithms, namely, MD5 for the message digest algorithm, DES CBC for the encryption algorithm.

5.4.1 ENC: Encryption

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|     ENC     |  length = 3   |   reserved    |  descriptor   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      DES (CBC) initialization vector, bytes 0 through 3       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      DES (CBC) initialization vector, bytes 4 through 7       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|     ENC     |  length = 1   |   reserved    |  descriptor   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

All packet data after this option is encrypted, using the encryption key and symmetric encryption algorithm specified by the descriptor field. Every encrypted RTP packet must contain this option. Note that the fixed header is specifically not encrypted because some fields must be interpreted by translators that will not have access to the key. The descriptor value may change over time to accommodate varying security requirements or limit the amount of ciphertext using the same key. For example, in a job interview conducted across a network, the candidate and interviewers could share one key, with a second key set aside for the interviewers only. For symmetric keys, source-specific keys offer no advantage.

The descriptor value zero is reserved for a default mode using the Data Encryption Standard (DES) algorithm in CBC (cipher block chaining) mode, as described in Section 1.1 of RFC 1423 [7]. The padding specified in that section is to be used. The 8-octet initialization vector (IV) may be carried unencrypted within the ENC option, generated anew for each packet. If the ENC option does not contain an initialization vector (indicated by an option length of one), the fixed RTP header is used as the initalization vector. (Using the fixed RTP header as the initialization vector avoids regenerating the initialization vector for each packet and incurs less header overhead.) For details on the tradeoffs for CBC initialization vector use, see [8]. Support for encryption is not required. Implementations that do not support encryption should recognize the ENC option so that they can avoid processing encrypted messages and provide a meaningful failure indication. Implementations that support encryption should, at the minimum, always support the DES CBC algorithm.

5.4.2 MIC: Messsage integrity check

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|     MIC     |    length     |   reserved    |  descriptor   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                message digest (unencrypted)                  ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The MIC option option is used only in combination with the ENC option immediately preceding it to provide privacy and group membership authentication. The message integrity check uses the digest algorithm specified by the descriptor field. (A message digest is a cryptographic hash function that transforms a message of any length to a fixed-length byte string, where the fixed-length string has the property that it is computationally infeasible to generate another, different message with the same digest.) The value zero implies the use of the MD5 message digest. Note that the MIC option is not separately encrypted.

5.4.3 MICA: Message integrity check, asymmetric encryption

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|    MICA     |    length     |        message digest        ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    (asymmetrically encrypted)                ...
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Currently, only the use of the MD2 and MD5 message digest algorithms is defined, as described in RFC 1319 [9] (as corrected in Section 2.1 of RFC 1423) and RFC 1321 [10], respectively. The MD2 and MD5 message digests are 16 octets long.

RFC 1423, Section 2.1:

> To avoid any potential ambiguity regarding the ordering of the octets of an MD2 message digest that is input as a data value to another encryption process (e.g., RSAEncryption), the following holds true. The first (or left-most displayed, if one thinks in terms of a digest's "print" representation) octet of the digest (i.e., digest[0] as specified in RFC 1319), when considered as an RSA data value, has numerical weight 2**120. The last (or right-most displayed) octet (i.e., digest[15] as specified in RFC 1319) has numerical weight 2**0.

RFC 1423, Section 2.2:

> To avoid any potential ambiguity regarding the ordering of the octets of an MD5 message digest that is input as an RSA data value to the RSA encryption process, the following holds true. The first (or left-most displayed, if one thinks in terms of a digest's "print" representation) octet of the digest (i.e., the low-order octet of A as specified in RFC 1321), when considered as an RSA data value, has numerical weight 2**120. The last (or right-most displayed) octet (i.e., the high-order octet of D as specified in RFC 1321) has numerical weight 2**0.

The message digest is encrypted, using asymmetric keys, with the sender's private key using the algorithm described in Section 4.2.1 of RFC 1423:

> As described in PKCS #1, all quantities input as data values to the RSAEncryption process shall be properly justified and padded to the length of the modulus prior to the encryption process. In general, an RSAEncryption input value is formed by concatenating a leading NULL octet, a block type BT, a padding string PS, a NULL octet, and the data quantity D, that is, RSA input value = 0x00, BT, PS, 0x00, D. To prepare a MIC for RSAEncryption, the PKCS #1 "block type 01" encryption-block formatting scheme is employed. The block type BT is a single octet containing the value 0x01 and the padding string PS is one or more octets (enough octets to make the length of the complete RSA input value equal to the length of the modulus) each containing the value 0xFF. The data quantity D is comprised of the MIC and the MIC algorithm identifier.

The encoding is described in detail in RFC 1423. For encrypting MD2 and MD5, the data quantity D comprises the 16-octet checksum, preceded by the binary sequences shown here in hexadecimal:

0x30, 0x20, 0x30, 0x0C, 0x06, 0x08, 0x2A, 0x86, 0x48, 0x86, 0xF7, 0x0D, 0x02, 0x02, 0x05, 0x00, 0x04, 0x10 for MD2 and 0x30, 0x20, 0x30, 0x0C, 0x06, 0x08, 0x2A, 0x86, 0x48, 0x86, 0xF7, 0x0D, 0x02, 0x05, 0x05, 0x00, 0x04, 0x10 for MD5.

Contrary to what is specified in RFC 1423 for privacy enhanced mail, the asymmetrically signed MIC is carried in binary, *not* represented in the printable encoding of RFC 1421, Section 4.3.2.4. The encrypted length of the signature will be equal to the modulus of the RSA encryption used, rounded to the next integral octet count. The modulus and public key are conveyed to the receivers by non-RTP means. Asymmetric keys are used since symmetric keys would not allow authentication of the individual source in the multicast case.

The signature is padded as necessary. The value of the padding is left unspecified. The number of non-padding bits within the signature is known to the receiver as being equal to the key length. The MIC algorithm is identified through the octets prepended to the actual 16-octet signature.

A translator is not allowed to modify the parts of an RTP packet covered by the MICA option as the receiver would have no way of establishing the identity of the translator and thus could not verify the integrity of the RTP packet.

Support for sending or interpreting MICA options is not required.

5.4.4 MICK: Message integrity check, keyed

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|    MICS     |    length     |   reserved    |  descriptor   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           message digest (symmetrically encrypted)           ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

This message integrity check option does not require encryption. In addition to the RTP packet parts to be included in the message digest according to the introduction to this section, the shared secret is placed in the MICK option and included in the message digest. The shared secret is equivalent to the key used for the MICS and ENC options, but is 16 octets long, padded if needed with binary zeroes. The shared secret in the MICK option is then replaced by the computed 16-octet message digest.

The receiver stores the message digest contained in the MICK option, replaces it with the shared secret key and computes the message digest in the same manner as the sender. If the RTP packet has not been tampered with and has originated with one of the holders of the shared secret, the computed message digest will agree with the digest found on reception in the MICS option.[5]

5.4.5 MICS: Message integrity check, symmetric-key encrypted

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|    MICS     |     length    |   reserved    |  descriptor   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           message digest (symmetrically encrypted)          ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

This message integrity check encrypts the message digest using DES ECB mode as described in RFC 1423, Section 3.1.

6 Real Time Control Protocol — RTCP

The real-time control protocol (RTCP) conveys minimal control and advisory information during a session. It provides support for "loosely controlled" sessions, i.e., where participants enter and leave without membership control and parameter negotiation. The services provided by RTCP services augment RTP, but an end system does not have to implement RTCP features to participate in sessions. There is one exception to this rule: if an application sends FMT options, the receiver has to decode these in order to properly interpret the RTP payload. RTCP does not aim to provide the services of a session control protocol and does not provide some of the services desirable for two-party conversations. If a session control protocol is in use, the services of RTCP should not be required. (As of the writing of this document, a session or conference control protocol has not been specified within the Internet.)

RTCP options share the same structure and numbering space as RTP options, which are described in Section 5. Unless otherwise noted, control information is carried periodically as options within RTP packets, with or without payload. RTCP packets are sent to all members of a session. These packets are part of the same sequence number space as RTP packets not containing RTCP options. The period should be varied randomly to avoid synchronization of all sources and its mean should increase with the number of participants in the session to limit the growth of the overall network and host interrupt load. The length of the period determines, for example, how long a receiver

[5] The message integrity check follows the practice of SNMP Version 2, as described in RFC 1446, Section 1.5.1. The MICS option itself is covered by the digest in order to detect tampering with the descriptor field itself. Using the secret key in the signature instead of encrypting the MD5 message digest avoids the use of an encryption algorithm when only authentication is desired. However, the security of this approach has not been as well established as the authentication based on encrypting message digests used in the MICS, MIC and MICA options.

joining a session has to wait until it can identify the source. A receiver may remove from its list of active sites a site that it has not been heard from for a given time-out period; the time-out period may depend on the number of sites or the observed average interarrival time of RTCP messages. Note that not every periodic message has to contain all RTCP options; for example, the EMAIL part within the SDES option might only be sent every few messages. RTCP options should also be sent when information carried in RTCP options changes, but the generation of RTCP options should be rate-limited.

The option types are defined below:

6.1 FMT: Format description

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|     FMT     |    length     |R|R|  format   |    reserved   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   format-dependent data                       ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

format: 6 bits
The format field corresponds to the index value from the format field in the RTP fixed header, with values ranging from 0 to 63.

Format-dependent data: variable length
Format-dependent data may or may not appear in a FMT option. It is passed to the next layer and not interpreted by RTP.

A FMT mapping changes the interpretation of a given format value carried in the fixed RTP header starting at the packet containing the FMT option. The new interpretation applies only to packets from the same synchronization source as the packet containing the FMT option. If format mappings are changed through the FMT option, the option should be sent periodically as otherwise sites that did not receive the FMT option due to packet loss or joining the session after the FMT option was sent will not know how to interpret the particular format value.

6.2 SDES: Source descriptor

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|     SDES    |    length     |       source identifier       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  type = ADDR  |    length     |    reserved   | address type  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     network-layer address                    ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  type = ADDR  |   length = 2  |    reserved   | addr. type = 1|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          IPv4 address                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  type = PORT  |   length = 1  |             port              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  type = PORT  |   length > 1  |    reserved   |    reserved   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             port                             ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  type = CNAME |    length     | user and domain name         ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  type = EMAIL |    length     | electronic mail address      ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  type = NAME  |    length     | common name of source        ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   type = LOC  |    length     | geographic location of site  ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   type = TXT  |    length     | text describing source       ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The SDES option provides a mapping between a numeric source identifier and one or more items identifying the source.[6] For those applications where the size of a multi-item SDES option would be a concern, multiple SDES options may be formed with subsets of the items to be sent in separate

[6] Several attributes were combined into one option so that the receiver does not have to perform multiple mappings from identifiers to site data structures.

packets.

A bridge uses an identifier value of zero within the SDES option to refer to itself rather than content sources bridged by it. For each content source, a bridge forwards the SDES information received from that source, but changes the SDES source identifier to the value used in the CSRC option when identifying that content source. A bridge that contributes local data to outgoing packets should select another non-zero source identifier for that traffic and send CSRC and SDES options for it.

Translators do not modify or insert SDES options. The end system performs the same mapping it uses to identify the content sources (that is, the combination of network source, synchronization source and the source identifier within this SDES option) to identify a particular source. SDES information is specific to a particular channel, unless a profile or a higher-layer control protocol defines that all packets with the same source identifier (network and transport-level source addresses and the optional SSRC value) from a set of channels defined by the control protocol are described by the same SDES.

Currently, the items listed in Table 2 are defined. Each has a structure similar to that of RTCP and RTP options, that is, a type field followed by a length field, measured in multiples of four octets. No final bit (see Section 5.2) is needed since the overall length is known. Text items are encoded according to the rules in Section 4. All of the SDES items are optional; however, if quality-of-service monitoring is to be used, one ADDR item and the PORT item are mandatory, as described for the QOS option. Only the TXT item is expected to change during the duration of a session. Option types 128 through 255 are reserved for private or experimental extensions. Items are padded with the binary value zero to the next multiple of four octets. Each item may appear only once unless otherwise noted.

type	value	description
ADDR	1	network address of source
PORT	2	source port
CNAME	4	canonical user and host identifier, e.g., "doe@sleepy.megacorp.com" or "sleepy.megacorp.com"
EMAIL	5	user's electronic mail address, e.g., "John.Doe@megacorp.com"
NAME	6	common name describing the source, e.g.,"John Doe, Bit Recycler, Megacorp"
LOC	8	geographic user location, e.g., "Rm. 2A244, Murray Hill, NJ"
TXT	16	text describing the source, e.g., "out for lunch"

Table 2: Summary of SDES items

A more detailed description of the content of some of these items follows:

ADDR: This item contains the network address of the source, for example, the IP version 4 address or an NSAP. The address is carried in binary form, not as "dotted decimal" or similar human-readable form. A source may send several network addresses, but only one for each address type value. Address types are identified by the Domain Name Service Resource Record (RR) type, as specified in the current edition of the Assigned Numbers RFC.

PORT: If the length field is one, the transport selector, such as the UDP port number, is carried as octets three and four in the first and only word of the item. If the length field is greater than one, octets three and four are zero and the transport selector appears in words two and following of this item, in network byte order. The figure shows the use of the PORT item for the TCP and UDP protocols. There must be no more than one PORT item in an SDES option. The PORT item should immediately precede any ADDR items.[7]

CNAME: The CNAME item must have the format "user@host" or "host", where "host" is the fully qualified domain name of the host from which the real-time data originates, formatted according to the rules specified in RFC 1034, RFC 1035 and Section 2.1 of RFC 1123. The "host" form may be used if a user name is not available, for example on single-user systems. The user name should be in a form that a program such as "finger" or "talk" could use, i.e., it typically is the login name rather than the "real life" name. Note that the host name is not necessarily identical to the electronic mail address of the participant. The latter is provided through the EMAIL item.

LOC: Depending on the application, different degrees of detail are appropriate for this item. For conference applications, a string like "Murray Hill, New Jersey" may be sufficient, while, for an active badge system, strings like "Room 2A244, AT&T BL MH" might be appropriate. The degree of detail is left to the implementation and/or user, but format and content may be prescribed by a profile.

6.3 BYE: Goodbye

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|     BYE     | length = 1    |   content source identifier   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The BYE option indicates that a particular session participant is no longer active. A bridge sends BYE options with a (non-zero) content source value. An identifier value of zero indicates that the source indicated by the synchronization source (SSRC) option and transport address is no longer

[7] Multiple concurrent transport addresses are not meaningful. The ordering simplifies processing at the receiver, as the consecutive octet string of PORT followed by the first ADDR can be used as a globally unique identifier. The transport protocol does not need to be identified, as the receiver will only see one type of transport protocol for a session.

active. If a bridge shuts down, it should first send BYE options for all content sources it handles, followed by a BYE option with an identifier value of zero. Each RTCP message can contain one or more BYE messages. Multiple identifiers in a single BYE option are not allowed, to avoid ambiguities between the special value of zero and any necessary padding.

6.4 QOS: Quality of service measurement

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|F|     QOS     |    length     |    reserved   |    reserved   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       packets expected                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       packets received                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    minimum delay (seconds)    |    minimum delay (fraction)   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    maximum delay (seconds)    |    maximum delay (fraction)   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    average delay (seconds)    |    average delay (fraction)   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  type = PORT  |    length     |   transport address         ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  type = ADDR  |    length     |    reserved   | address type  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     network-layer address                   ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The QOS option conveys statistics of a single synchronization source belonging to the channel identified by the multicast address, destination port and channel identifier. The synchronization source is identified by appending the first of the ADDR items together with the PORT item from the SDES option. These SDES items are appended directly to the fixed-length part of the QOS option, with PORT preceding ADDR. For a description of these items, see the SDES option.

If the QOS option is used in reverse control packets, the destination port number identifies the channel, along with the channel identifier. For that reason, every multicast group should be associated with a unique source port.

The other fields of the option contain the number of packets received, the number of packets expected, the minimum delay, the maximum delay and the average delay. The expected number of packets may be computed according to the algorithm in Section A.5. The delay measures are in units of 1/65536 of a second, that is, with the same resolution as the timestamp in the fixed RTP header.

A single RTCP packet may contain several QOS options. It is left to the implementor to decide how often to transmit QOS options and which sources are to be included.

7 Security Considerations

Without the use of the security options described in section 5.4, RTP suffers from the same security deficiencies as the underlying protocols, for example, the ability of an impostor to fake source or destination network addresses, or to change header or payload without detection. For example, the SDES fields may be used to impersonate another participant.

IP multicast provides no direct means for a sender to know all the receivers of the data sent. RTP options make it easy for all participants in a session to identify themselves; if deemed important for a particular application, it is the responsibility of the application writer to make listening without identification difficult. It should be noted, however, that privacy of the payload can generally be assured only by encryption.

The periodic transmission of session messages may make it possible to detect denial-of-service attacks, as the receiver can detect the absence of these expected messages.

Unlike for other data, ciphertext-only attacks may be more *difficult* for compressed audio and video sources. Such data is very close to white noise, making statistics-based ciphertext-only attacks difficult. Even without message integrity check options, it may be difficult for an attacker to detect automatically when he or she has found the secret cryptographic key since the bit pattern after correct decryption may not look significantly different from one decrypted with the wrong key. However, the session information is more or less constant and predictable, allowing known-plaintext attacks. Chosen-plaintext attacks appear, in general, to be difficult.

The integrity of the timestamp in the fixed RTP header can be protected by the message integrity options. If clocks are known to be synchronized, an attacker only has a very limited time window of maybe a few seconds every 18 hours to replay recorded RTP without detection by the receiver.

Key distribution and certificates are outside the scope of this document.

8 RTP over Network and Transport Protocols

This section describes issues specific to carrying RTP packets within particular network and transport protocols.

8.1 Defaults

The following rules apply unless superseded by protocol-specific subsections in this section. The rules apply to both forward and reverse RTP packets.

RTP packets contain no length field or other delineation, so that a framing mechanism is needed if they are carried in underlying protocols that provide the abstraction of a continuous bit stream rather than messages (packets). TCP is an example of such a protocol. Framing is also needed if the underlying protocol may contain padding so that the extent of the RTP payload cannot be determined. For these cases, each RTP packet is prefixed by a 32-bit framing field containing the length of the RTP packet measured in octets, not including the framing field itself. If an RTP packet traverses a path over a mixture of octet-stream and message-oriented protocols, each RTP-level bridge between these protocols is responsible for adding and removing the framing field.

A profile may determine that this framing method is to be used even when RTP is carried in protocols that do provide framing in order to allow carrying several RTP packets in one lower-layer protocol data unit, such as a UDP packet. Carrying several RTP packets in one network or transport packet reduces header overhead and may simplify synchronization between different streams.

8.2 ST-II

When used in conjunction with RTP, ST-II [11] service access ports (SAPs) have a length of 16 bits. The next protocol field ("NextPCol", Section 4.2.2.10 in RFC 1190) is used to distinguish two encapsulations of RTP over ST-II. The first uses NextPCol value TBD and directly places the RTP packet into the ST-II data area. If NextPCol value TBD is used, the RTP header is preceded by a 32-bit header shown below. The octet count determines the number of octets in the RTP header and payload to be checksummed. The 16-bit checksum uses the TCP and UDP checksum algorithm.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| count of octets to be checked |          check sum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       RTP packet (fixed header, options and payload)         ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

A Implementation Notes

We describe aspects of the receiver implementation in this section. There may be other implementation methods that are faster in particular operating environments or have other advantages. These implementation notes are for informational purposes only.

The following definitions are used for all examples; the structure definitions are valid for 32-bit big-endian architectures only. Bit fields are assumed to be packed tightly, with no additional padding.

```
#include <sys/types.h>

typedef double CLOCK_t;

typedef enum {
  RTP_CSRC   = 0,
  RTP_SSRC   = 1,
  RTP_SDST   = 2,
  RTP_BOS    = 3,
  RTP_ENC    = 8,
  RTP_MIC    = 9,
  RTP_MICA   = 10,
  RTP_MICK   = 11,
  RTP_MICS   = 12,
  RTP_FMT    = 32,
  RTP_SDES   = 34,
  RTP_BYE    = 35,
  RTP_QOS    = 36
} rtp_option_t;

typedef struct {
  unsigned int ver:2;       /* version number */
  unsigned int channel:6;   /* channel id */
  unsigned int o:1;         /* option present */
  unsigned int s:1;         /* sync bit */
  unsigned int format:6;    /* content type */
  u_short seq;              /* sequence number */
  u_long  ts;               /* time stamp */
} rtp_hdr_t;

typedef union {
  struct {
    int final:1;            /* final option */
    int type:7;             /* option type */
    u_char length;          /* length, including type/length */
    short id[1];
  } csrc;
  /* ... */
} rtp_t;
```

A.1 Timestamp Recovery

For some applications it is useful to have the receiver reconstruct the sender's high-order bits of the NTP timestamp from the received 32-bit RTP timestamp. The following code uses double-precision floating point numbers for whole numbers with a 48-bit range. Other type definitions of `CLOCK_t` may be appropriate for different operating environments, e.g., 64-bit architectures or systems with slow floating point support. The routine applies to any clock frequency, not just the RTP value of 65,536 Hz, and any clock starting point. It will reconstruct the correct high-order bits as long as the local clock `now` is within one half of wrap-around time of the 32-bit timestamp, e.g., approximately 9.2 hours for RTP timestamps.

```
#include <math.h>

#define MOD32bit 4294967296.
#define MAX31bit 0x7fffffff

CLOCK_t clock_extend(ts, now)
u_long ts;    /* in: timestamp, low-order 32 bits */
CLOCK_t now;  /* in: current local time */
{
  u_long high, low;   /* high and low order bits of 48-bit clock */

  low  = fmod(now, MOD32bit);
  high = now / MOD32bit;

  if (low > ts) {
    if (low - ts > MAX31bit) high++;
  }
  else {
    if (ts - low > MAX31bit) high--;
  }
  return high * MOD32bit + ts;
} /* extend_timestamp */
```

Using the full timestamp internally has the advantage that the remainder of the receiver code does not have to be concerned with modulo arithmetic. The current local time does not have to be derived directly from the system clock for every packet; a clock based on samples, e.g., incremented by the nominal audio frame duration, is sufficient. The whole seconds within NTP time stamps can be obtained by adding 2208988800 to the value of the standard Unix clock (generated, for example, by the gettimeofday system call), which starts from the year 1970. For the RTP time stamp, only the least significant 16 bits of the second are used.

A.2 Detecting the Beginning of a Synchronization Unit

RTP packets contain a bit flag indicating the end of a synchronization unit. The following code fragment determines, based on sequence numbers, if a packet is the beginning of a synchronization unit. It assumes that the packet header has been converted to host byte order.

```
static u_long seq_eos;
rtp_hdr_t *h;
static int flag;

if (h->s) {
  flag    = 1;
  seq_eos = h->seq;
}
/* handle wrap-around of sequence number */
else if (flag && (h->seq - seq_eos < 32768)) {
  flag = 0;
  /* handle beginning of synchronization unit */
}
```

A.3 Demultiplexing and Locating the Synchronization Source

The combination of destination address, destination port and channel identifier determines the channel. For each channel, the receiver maintains a list of all sources, content and synchronization sources alike, in a table or other suitable data structure. Synchronization sources are stored with a content source value of zero. When an RTP packet arrives, the receiver determines its network source address and port (from information returned by the operating system), synchronization source (SSRC option) and content source(s) (CSRC option). To locate the table entry containing timing information, mapping from content descriptor to actual encoding, etc., the receiver sets the content source to zero and locates a table entry based on the triple (transport source address, and synchronization source identifier, 0).

The receiver identifies the contributors to the packet (for example, the speaker who is heard in the packet) through the list of content sources carried in the CSRC option. To locate the table entry, it matches on the triple (network address and port, synchronization source identifier, content source).

Note that since network addresses are only generated locally at the receiver, the receiver can choose whatever format seems most appropriate for matching. For example, a Berkeley Unix-based system may use `struct sockaddr` data types if it expects network sources with non-IP addresses.

A.4 Parsing RTP Options

The following code segment walks through the RTP options, preventing infinite loops due to zero and invalid length fields. Structure definitions are valid for big-endian architectures only.

```
u_long len;       /* length of RTP packet in bytes */
u_long *pt;       /* pointer */
rtp_hdr_t *h;     /* fixed header */
rtp_t *r;         /* options */

if (h->o) {
  for (pt = (u_long *)(h+1);; pt += r->csrc.length) {
    r = (rtp_t *)pt;

    /* invalid length field */
    if ((char *)pt - (char *)h > len || r->csrc.length == 0) return -1;

    switch(r->csrc.type) {
      case RTP_BYE:
        /* handle BYE option */
        break;
      case RTP_CSRC:
        /* handle CSRC option */
        break;

        /* ... */

      default:
        /* undefined option */
        break;
    }
    if (r->csrc.final) break;
  }
}
```

A.5 Determining the Expected Number of RTP Packets

The number of packets expected can be computed by the receiver by tracking the first sequence number received (seq0), the last sequence number received, seq, and the number of complete sequence number cycles:

```
expected = cycles * 65536 + seq - seq0 + 1;
```

The cycle count is updated for each packet, where seq_prior is the sequence number of the prior packet:

```
unsigned long seq, seq_prior;

if (seq - seq_prior > 65536)
  cycle++;
else if (seq - seq_prior > 32768)
  cycle--;

seq_prior = seq;
```

Acknowledgments

This memorandum is based on discussions within the IETF audio-video transport working group chaired by Stephen Casner. The current protocol has its origins in the Network Voice Protocol and the Packet Video Protocol (Danny Cohen and Randy Cole) and the protocol implemented by the vat application (Van Jacobson and Steve McCanne). Stuart Stubblebine (ISI) helped with the security aspects of RTP. Ron Frederic (Xerox PARC) provided extensive editorial assistance.

B Addresses of Authors

Stephen Casner
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695
telephone: +1 310 822 1511 (extension 153)
electronic mail: casner@isi.edu

Henning Schulzrinne
AT&T Bell Laboratories

MH 2A244
600 Mountain Avenue
Murray Hill, NJ 07974-0636
telephone: +1 908 582 2262
facsimile: +1 908 582 5809
electronic mail: `hgs@research.att.com`

References

[1] D. E. Comer, *Internetworking with TCP/IP*, vol. 1. Englewood Cliffs, New Jersey: Prentice Hall, 1991.

[2] J. Postel, "Internet protocol," Network Working Group Request for Comments RFC 791, Information Sciences Institute, Sept. 1981.

[3] International Standards Organization, "ISO/IEC DIS 10646-1:1993 information technology – universal multiple-octet coded character set (UCS) – part I: Architecture and basic multilingual plane," 1993.

[4] The Unicode Consortium, *The Unicode Standard*. New York, New York: Addison-Wesley, 1991.

[5] D. L. Mills, "Network time protocol (version 3) – specification, implementation and analysis," Network Working Group Request for Comments RFC 1305, University of Delaware, Mar. 1992.

[6] S. Kent, "Understanding the Internet certification system," in *Proceedings of the International Networking Conference (INET)*, (San Francisco, California), pp. BAB–1 – BAB–10, Internet Society, Aug. 1993.

[7] D. Balenson, "Privacy enhancement for internet electronic mail: Part III: Algorithms, modes, and identifiers," Network Working Group Request for Comments RFC 1423, IETF, Feb. 1993.

[8] V. L. Voydock and S. T. Kent, "Security mechanisms in high-level network protocols," *ACM Computing Surveys*, vol. 15, pp. 135–171, June 1983.

[9] J. Kaliski, Burton S., "The MD2 message-digest algorithm," Network Working Group Request for Comments RFC 1319, RSA Laboratories, Apr. 1992.

[10] R. Rivest, "The MD5 message-digest algorithm," Network Working Group Request for Comments RFC 1321, IETF, Apr. 1992.

[11] C. Topolcic, S. Casner, C. Lynn, Jr., P. Park, and K. Schroder, "Experimental internet stream protocol, version 2 (ST-II)," Network Working Group Request for Comments RFC 1190, BBN Systems and Technologies, Oct. 1990.