# [MS-RTP]:
# Real-time Transport Protocol (RTP) Extensions

---

**Intellectual Property Rights Notice for Open Specifications Documentation**

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft's Open Specification Promise (available here: http://www.microsoft.com/interop/osp) or the Community Promise (available here: http://www.microsoft.com/interop/cp/default.mspx). If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

**Reservation of Rights.** All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

**Tools.** The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

## Revision Summary

| Date | Revision History | Revision Class | Comments |
|---|---|---|---|
| 04/04/2008 | 0.1 | | Initial version |
| 04/25/2008 | 0.2 | | Revised and edited technical content |
| 06/27/2008 | 1.0 | | Revised and edited technical content |
| 08/15/2008 | 1.01 | | Revised and edited technical content |
| 12/12/2008 | 2.0 | | Revised and edited technical content |
| 02/13/2009 | 2.01 | | Revised and edited technical content |
| 03/13/2009 | 2.02 | | Revised and edited technical content |
| 07/13/2009 | 2.03 | Major | Revised and edited the technical content |
| 08/28/2009 | 2.04 | Editorial | Revised and edited the technical content |
| 11/06/2009 | 2.05 | Editorial | Revised and edited the technical content |
| 02/19/2010 | 2.06 | Editorial | Revised and edited the technical content |
| 03/31/2010 | 2.07 | Major | Updated and revised the technical content |
| 04/30/2010 | 2.08 | Editorial | Revised and edited the technical content |
| 06/07/2010 | 2.09 | Editorial | Revised and edited the technical content |
| 06/29/2010 | 2.10 | Editorial | Changed language and formatting in the technical content. |
| 07/23/2010 | 2.10 | No change | No changes to the meaning, language, or formatting of the technical content. |
| 09/27/2010 | 3.0 | Major | Significantly changed the technical content. |
| 11/15/2010 | 3.0 | No change | No changes to the meaning, language, or formatting of the technical content. |
| 12/17/2010 | 3.0 | No change | No changes to the meaning, language, or formatting of the technical content. |
| 03/18/2011 | 3.0 | No change | No changes to the meaning, language, or formatting of the technical content. |
| 06/10/2011 | 3.0 | No change | No changes to the meaning, language, or formatting of the technical content. |

# Table of Contents


**1 Introduction ..... 5**
1.1 Glossary ..... 5
1.2 References ..... 6
1.2.1 Normative References ..... 6
1.2.2 Informative References ..... 7
1.3 Protocol Overview (Synopsis) ..... 7
1.4 Relationship to Other Protocols ..... 8
1.5 Prerequisites/Preconditions ..... 9
1.6 Applicability Statement ..... 9
1.7 Versioning and Capability Negotiation ..... 10
1.8 Vendor-Extensible Fields ..... 10
1.9 Standards Assignments ..... 10

**2 Messages ..... 11**
2.1 Transport ..... 11
2.2 Message Syntax ..... 11
2.2.1 RTP Packets ..... 11
2.2.2 RTCP Compound Packets ..... 12
2.2.3 RTCP Probe Packet ..... 13
2.2.4 RTCP Packet Pair Packet ..... 13
2.2.5 RTCP Packet Pair ..... 13
2.2.6 RTCP Packet Train Packet ..... 13
2.2.7 RTCP Packet Train ..... 13
2.2.8 RTCP Sender Report (SR) ..... 13
2.2.9 RTCP SDES ..... 13
2.2.9.1 SDES PRIV extension for media quality ..... 13
2.2.10 RTCP Profile Specific Extension ..... 15
2.2.10.1 RTCP Profile Specific Extension for Estimated Bandwidth ..... 15
2.2.10.2 RTCP Profile Specific Extension for Packet Loss Notification ..... 16
2.2.10.3 RTCP Profile Specific Extension for Video Preference ..... 17
2.2.10.4 RTCP Profile Specific Extension for Padding ..... 18
2.2.10.5 RTCP Profile Specific Extension for Policy Server Bandwidth ..... 18
2.2.10.6 RTCP Profile Specific Extension for TURN Server Bandwidth ..... 18
2.2.10.7 RTCP Profile Specific Extension for Audio Healer Metrics ..... 19
2.2.10.8 RTCP Profile Specific Extension for Receiver-side Bandwidth Limit ..... 20
2.2.10.9 RTCP Profile Specific Extension for Packet Train Packet ..... 21
2.2.10.10 RTCP Profile Specific Extension for Peer Info Exchange ..... 21

**3 Protocol Details ..... 23**
3.1 RTP Details ..... 23
3.1.1 Abstract Data Model ..... 23
3.1.2 Timers ..... 24
3.1.3 Initialization ..... 24
3.1.4 Higher-Layer Triggered Events ..... 25
3.1.5 Message Processing Events and Sequencing Rules ..... 25
3.1.6 Timer Events ..... 26
3.1.7 Other Local Events ..... 26
3.2 RTCP Details ..... 27
3.2.1 Abstract Data Model ..... 29
3.2.2 Timers ..... 30

3.2.3 Initialization ..... 30
3.2.4 Higher-Layer Triggered Events ..... 30
3.2.5 Message Processing Events and Sequencing Rules ..... 30
3.2.6 Timer Events ..... 32
3.2.7 Other Local Events ..... 32

**4 Protocol Examples ..... 33**
4.1 SSRC Change Throttling ..... 33
4.2 Dominant Speaker Notification ..... 33
4.3 Bandwidth Estimation ..... 34
4.4 Packet Loss Notification ..... 38
4.5 Video Preference ..... 38
4.6 Policy Server Bandwidth notification ..... 39
4.7 TURN Server Bandwidth Notification ..... 40
4.8 Audio Healer Metrics ..... 41
4.9 Receiver-side Bandwidth Limit ..... 42
4.10 SDES Private extension for media quality ..... 43

**5 Security ..... 44**
5.1 Security Considerations for Implementers ..... 44
5.2 Index of Security Parameters ..... 44

**6 Appendix A: Product Behavior ..... 45**

**7 Change Tracking ..... 48**

**8 Index ..... 49**

# 1 Introduction

This document specifies the Real-Time Transport Protocol (RTP) Extensions, which are a set of proprietary extensions to the base Real-Time Transport Protocol (RTP). RTP is a set of network transport functions suitable for applications transmitting real-time data, such as audio and video, across multimedia endpoints.

## 1.1 Glossary

The following terms are defined in [MS-GLOS]:

**common name (CN)**
**datagram**
**encryption**
**Internet Protocol version 4 (IPv4)**
**network address translation (NAT)**
**Transmission Control Protocol (TCP)**
**User Datagram Protocol (UDP)**

The following terms are defined in [MS-OFCGLOS]:

**audio video profile (AVP)**
**codec**
**Common Intermediate Format (CIF)**
**conference**
**contributing source (CSRC)**
**dual-tone multi-frequency (DTMF)**
**endpoint**
**forward error correction (FEC)**
**I-frame**
**Interactive Connectivity Establishment (ICE)**
**jitter**
**mixer**
**packetization time (P-time)**
**participant**
**Real-Time Transport Control Protocol (RTCP)**
**Real-Time Transport Protocol (RTP)**
**RTCP packet**
**RTP packet**
**RTP payload**
**RTP session**
**RTVideo**
**Session Description Protocol (SDP)**
**Session Initiation Protocol (SIP)**
**stream**
**Super P-frame (SP-frame)**
**Synchronization Source (SSRC)**
**Traversal Using Relay NAT (TURN)**
**TURN server**
**video frame**

The following terms are specific to this document:

**audio healer:** One or more digital signal processing algorithms designed to mask or conceal human-perceptible audio distortions that are caused by packet loss and jitter.

**connectionless protocol:** A transport protocol that enables endpoints (5) to communicate without a previous connection arrangement and that treats each packet independently as a datagram. Examples of connectionless protocols are Internet Protocol (IP) and User Datagram Protocol (UDP).

**connection-oriented transport protocol:** A transport protocol that enables endpoints (5) to communicate after first establishing a connection and that treats each packet according to the connection state. An example of a connection-oriented transport protocol is Transmission Control Protocol (TCP).

**dominant speaker:** A participant (2) whose speech is both detected by a mixer and perceived to be dominant at a specific moment. Heuristics typically are used to determine the dominant speaker.

**FEC distance:** A number that specifies an offset from the current packet to a previous audio packet that is to be sent as redundant audio data.

**silence suppression:** A mechanism for conserving bandwidth by detecting silence in the audio input and not sending packets that contain only silence.

**video encapsulation:** A mechanism for transporting video payload and metadata in Real-Time Transport Protocol (RTP) packets.

**MAY, SHOULD, MUST, SHOULD NOT, MUST NOT:** These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

## 1.2 References

### 1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[MS-RTASPF] Microsoft Corporation, "RTP Payload Format for Application Sharing Extensions"

[MS-TURN] Microsoft Corporation, "Traversal Using Relay NAT (TURN) Extensions"

[MS-TURNBWM] Microsoft Corporation, "Traversal using Relay NAT (TURN) Bandwidth Management Extensions"

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.rfc-editor.org/rfc/rfc2119.txt

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and Jacobson, V., "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003, http://www.ietf.org/rfc/rfc3550.txt

[RFC3551] Schulzrinne, H., and Casner, S., "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003, http://www.ietf.org/rfc/rfc3551.txt

### 1.2.2 Informative References

[MS-DTMF] Microsoft Corporation, "RTP Payload for DTMF Digits, Telephony Tones, and Telephony Signals Extensions"

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary".

[MS-H263PF] Microsoft Corporation, "RTP Payload Format for H.263 Video Streams Extensions"

[MS-ICE] Microsoft Corporation, "Interactive Connectivity Establishment (ICE) Extensions"

[MS-ICE2] Microsoft Corporation, "Interactive Connectivity Establishment (ICE) Extensions 2.0"

[MS-OFCGLOS] Microsoft Corporation, "Microsoft Office Master Glossary".

[MS-RTPRADEX] Microsoft Corporation, "RTP Payload for Redundant Audio Data Extensions"

[MS-RTVPF] Microsoft Corporation, "RTP Payload Format for RT Video Streams Extensions"

[MS-SDPEXT] Microsoft Corporation, "Session Description Protocol (SDP) Version 2.0 Extensions"

[MS-SIPRE] Microsoft Corporation, "Session Initiation Protocol (SIP) Routing Extensions"

[MS-SRTP] Microsoft Corporation, "Secure Real-time Transport Protocol (SRTP) Extensions"

[RFC3389] Zopf, R., "Real-Time Transport Protocol (RTP) Payload for Comfort Noise (CN)", RFC 3389, September 2002, http://www.rfc-editor.org/rfc/rfc3389.txt

[RFC4571] Lazzaro, J., "Framing Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP) Packets over Connection-Oriented Transport", RFC 4571, July 2006, http://www.ietf.org/rfc/rfc4571.txt

[RFC4733] Schulzrinne, H., "RTP Payload for DTMF Digits, Telephony Tones and Telephony Signals", RFC 4733, December 2006, http://www.ietf.org/rfc/rfc4733.txt

## 1.3 Protocol Overview (Synopsis)

This document specifies proprietary extensions to the **Real-Time Transport Protocol (RTP)** and the **audio video profile (AVP)**. RTP provides end-to-end delivery services for data with real-time characteristics. The AVP defines the AV-specific interpretations of profile-dependent fields. RTP extensions define packet formats to convey additional information and behavioral changes to enhance host security. These extensions include:

- **Dominant speaker** notification: Extends the standard RTP "Active Contributor" mechanism, the **contributing source (CSRC)** list, by assigning a special meaning to the first element of the list.
- **Synchronization Source (SSRC)**/Sequence Number change throttling: Limits the number of times the SSRC or sequence number of a **participant (2)** can change over a short period in time. The intention of this limit is to avoid attacks that seek to artificially increase resource usage on a host by flooding it with values that could force a costly re-initialization of receiver components.
- **Bandwidth estimation**: Defines a new mechanism to estimate and communicate the bandwidth on a channel. One host sends two or more "probe packets", and the other host can use the time interval between them to estimate the bandwidth, which is then communicated back through a **Real-Time Transport Control Protocol (RTCP)** profile extension.

- **Packet loss notification**: Defines an RTCP profile extension that allows a receiver to quickly notify the sender of the loss of a specific packet. The sender can then use this information to hasten recovery, such as by generating a new **I-frame** or **Super P-frame (SP-frame)** in the case of a video **stream (2)** encapsulated through extensions described in [MS-RTVPF].
- **Application Sharing**: Defines an application sharing profile, described in [MS-RTASPF], to support desktop/application sharing over RTP.
- **Video Preference**: Defines an RTCP profile extension that allows a receiver to request a sender to change the video resolution in the middle of the call, such as by generating a new I-frame of the desired resolution of a video stream (2) encapsulated through extensions described in [MS-RTVPF].
- **Policy Server Bandwidth**: Defines an RTCP profile extension that allows a host to send its bandwidth provisioned by the policy server, as obtained through the **Traversal Using Relay NAT (TURN)** protocol to the remote host.
- **TURN Server Bandwidth**: Defines an RTCP profile extension that allows a host to send its bandwidth provisioned by the **TURN server** as obtained through the TURN protocol to the remote host.
- **SDES PRIV extension for media quality:** Defines a private Source Descriptions for Media Streams (SDES) extension for sending the media quality from the CSRC or SSRC to a media receiver. The receiver can use this information to show which source is causing quality issues.
- **Receiver-side audio healer report:** Defines an RTCP profile extension that allows a host to send its receiver-side **audio healer** metrics, local network receive quality, and **FEC distance** request to the sender to help the sender drive the audio **forward error correction (FEC)**.
- **Receiver-side bandwidth limit:** Defines an RTCP profile extension that allows a host to send its receiver-side bandwidth limit request to the remote host to let the remote host know the maximum bandwidth it is capable of receiving.
- **Peer info exchange:** Defines an RTCP profile extension that enables a host to send its inbound and outbound network bandwidth throughput limit to the remote host.

## 1.4 Relationship to Other Protocols

Sessions for this protocol are usually initiated through **Session Initiation Protocol (SIP)** Routing Extensions, as described in [MS-SIPRE] section 3.14. RTP transport parameters, such as protocol, IP, and port, for sessions established through SIP are usually communicated through **Session Description Protocol (SDP)** extensions for audio and video, as described in [MS-SDPEXT] section 3.1.5.17.

A host can negotiate multiple transport parameters, in which case the selection can be made by means of an advanced connectivity mechanism such as the **Interactive Connectivity Establishment (ICE)** protocol, as described in [MS-ICE] section 3.1.4.8.3 and [MS-ICE2] section 3.1.4.8.3. The ICE negotiation can use **User Datagram Protocol (UDP)**, **Transmission Control Protocol (TCP)**, or an assortment of **network address translation (NAT)** traversal mechanisms. If a **connection-oriented transport protocol**, such as TCP, is used, the framing specified in [RFC4571] section 2 is used.

This protocol is based, in large part, on the RTP protocol, as described in [RFC3550] and [RFC3551].**RTP packets** can be encrypted and authenticated through the secure RTP protocol, as described in [MS-SRTP] section 3.1.3.3. For audio communications, RTP supports a redundancy

mechanism for FEC, as described in [MS-RTPRADEX] section 3, as well as a mechanism for communicating **dual-tone multi-frequency (DTMF)** events, as described in [MS-DTMF] section 3.

RTP supports Comfort Noise (CN) payload, as described in[RFC3389] section 4. **CN** is used for audio **codecs** that do not support CN, such as G.711, G.722.1, G.726, GSM 6.10, G.722, Siren, and RT Audio, for optimal audio quality. The clock rate of CN is the same as the clock rate of the audio codec.

RTP also supports the application sharing payload, as described in [MS-RTASPF] section 3.2.5 for sending an RDP payload for application and desktop sharing.

Negotiation for these and other payload properties, including supported codecs, sampling rates, and dynamic payload type mappings, can also be done through SDP. For video communications, because data for a single frame can sometimes span more than one RTP packet, various **video encapsulation** methods can be used, such as **RTVideo**, as described in [MS-RTVPF] section 2.2.

The following diagram illustrates this hierarchy between protocols. SIP and SDP are not represented because they are parallel to RTP.

CN Events, CN over RTP, G.722, RDP Payload, and RDP over RTP are not uniformly supported across all **endpoints (5)**.

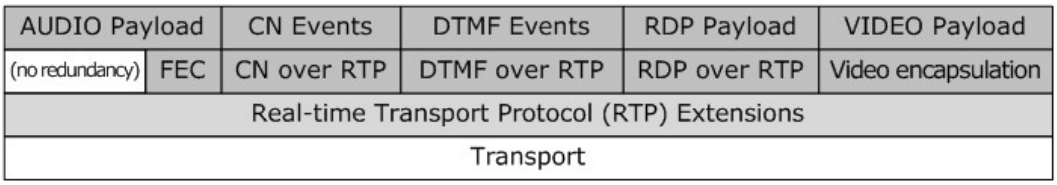



**Figure 1: Protocol hierarchy of RTP with the extension**

## 1.5 Prerequisites/Preconditions

To establish a session for this protocol, the whole negotiation for transport (for example, protocol, address, and port), payload (for example, codec, payload type mapping, sampling rate, bit rate, and video resolution) and **encryption** (for example, protocol, algorithm, and key) parameters has to take place by non-RTP means, such as SIP or SDP. At least one connection path at transport level has to be established, either directly or through a connectivity mechanism such as ICE.

## 1.6 Applicability Statement

This protocol is intended to be a streaming protocol only, carrying just the payload and the minimum metadata needed for real-time rendering. Even RTCP is intentionally limited in negotiation and session control capabilities. Except for these few exceptions, all capability negotiation, session establishment and session control is supposed to be done by non-RTP means, through another protocol, which is usually SIP or SDP.

This protocol is a best effort protocol and, when run over unreliable transport, does not provide reliable transmission of every packet. Redundancy mechanisms, such as the one described in [MS-RTPRADEX] section 3, can reduce the impact of packet loss, but not eliminate it.

This protocol is extremely time-sensitive, especially for voice communications. The quality of the experience is very dependent upon the quality of the underlying network. Issues such as long delays, **jitter**, and high packet loss all negatively affect the end-user experience. The choice of protocol, connectionless or connection-oriented, or connection path, direct or through an intermediate host, also affects user experience.

Although the packet loss extension is generic, because it only includes a sequence number, its use is only specified for video streams (2) in this document.

## 1.7 Versioning and Capability Negotiation

This protocol has the following versioning and capability negotiation constraints:

- **Supported Transports**: RTP is transport agnostic, and can be implemented over any connectionless or connection-oriented transport protocol. Common protocols used are TCP, UDP, and NAT traversal mechanisms such as ICE.
- **Protocol Versions**: The version of the RTP protocol is explicitly indicated on the **V** field of every RTP packet. Only version 2 of the RTP protocol is specified in this document.
- **Capability Negotiation**: Capability negotiation is done by non-RTP means, usually through a higher level application layer protocol such as SIP and SDP.

## 1.8 Vendor-Extensible Fields

The standard method for selecting codecs in this protocol is through payload types. [RFC3551] section 6 provides a default mapping for audio and video codecs that includes a range from 0x60 to 0x7F, or 96 to 127, to be used for dynamic codec mapping. For each session of this protocol using a dynamically mapped codec, a mapping between a number inside this range and a specific codec MUST be negotiated through non-RTP means, such as through SDP. Although there are no reserved or assigned numbers within this dynamic payload type range, some codecs are typically mapped to specific payload types. Some examples of dynamic payload type conventions can be found in section 2.2.1.

## 1.9 Standards Assignments

None.

# 2 Messages

## 2.1 Transport

This protocol MUST be supported over UDP and ICE, as described in [MS-ICE] and [MS-ICE2]<1>, over **Internet Protocol version 4 (IPv4)** only. When running over **connectionless protocols** such as UDP, each RTP packet MUST be transported in exactly one **datagram**. The total size of a single RTP packet, including all transport, network, and link-layer headers, MUST NOT exceed 1500 bytes. If the underlying transport is disconnected, or becomes inactive for more than 30 seconds, the **RTP session** SHOULD<2> be terminated.

## 2.2 Message Syntax

### 2.2.1 RTP Packets

The syntax of the RTP header is as specified in [RFC3550] section 5.1.The fields of the fixed RTP header have their usual meaning, which is specified in [RFC3550] section 5.1 and [RFC3551] section 2, with the following additional notes:

**Marker bit (M)**: In audio streams (2), if **silence suppression** is enabled, the marker bit (**M**) SHOULD be 1 for the first packet of a talk spurt and 0 for all other packets. Failure to do so can result in reduced audio quality at the receiving end. If silence suppression is disabled, the marker bit can be 1 for the first packet in the stream (2), but SHOULD<3> be 0 for all other packets. In video streams (2), the marker bit MUST be 1 for the last packet sent for each **video frame**, and 0 for all other packets.

**Payload type (PT)**: The payload type field identifies the format of the **RTP payload**, and determines its interpretation by the application. Codecs that are not assigned to static payload types MUST be assigned to a payload type within the dynamic range, which is between 0x60 and 0x7f.

Codecs with payload type number in the dynamic range can use different payload type number for send and receive.<4>

Codecs with payload type numbers in the static range MUST be used as specified in the following table. Codecs with payload types in the dynamic range can use a different payload type number, but MUST be used with the clock rate, **packetization times (P-times)**, and number of channels specified in the following table.

The following table lists audio codecs with payload type numbers, clock rates, P-times, and channels:

| Payload type | Codec | Clock rate | P-times | Channels |
|---|---|---|---|---|
| 0 | G.711 μ-Law<5> | 8000 | 10, 20, 40, 60 | 1 |
| 3 | GSM 6.10<6> | 8000 | 20, 40, 60 | 1 |
| 4 | G.723.1 | 8000 | 30, 60, 90 | 1 |
| 8 | G.711 A-Law<7> | 8000 | 10, 20, 40, 60 | 1 |
| 9 | G.722<8> | 8000 | 20, 40, 60 | 1 |
| 13 | Comfort Noise<9> | 8000 | Not Applicable | 1 |
| 111 | Siren | 16000 | 20, 40, 60, 100, 200 | 1 |

| Payload type | Codec | Clock rate | P-times | Channels |
|---|---|---|---|---|
| 112 | G.722.1 | 16000 | 20, 40, 60 | 1 |
| 114 | RT Audio | 16000 | 20, 40, 60 | 1 |
| 115 | RT Audio | 8000 | 20, 40, 60 | 1 |
| 116 | G.726 | 8000 | 20, 40, 60 | 1 |
| 118 | Comfort Noise<10> | 16000 | Not Applicable | 1 |

The following table lists video codecs with payload type numbers and clock rates:

| Payload type | Codec | Clock rate |
|---|---|---|
| 34 | H.263 [MS-H263PF] | 90000 |
| 121 | RT Video | 90000 |

The following table lists data codecs with payload type numbers and clock rates<11>:

| Payload type | Codec | Clock rate |
|---|---|---|
| 127 | x-data | 90000 |

**SSRC**: This field identifies the synchronization source. This identifier SHOULD be chosen randomly, but MUST not be zero. The loop detection and collision resolution algorithms from [RFC3550] section 8.2 can be used, but MUST NOT detect a loop in the case when the receiver's own SSRC appears as the first CSRC in a packet from a **mixer**. See the following definition for the CSRC list for details. Regardless of loops or collisions, **SSRC** SHOULD NOT be changed within 2 seconds of the start of the stream (2) or a previous **SSRC** change, to prevent packets from being ignored by the throttling algorithm described in section 3.1.

**CSRC list**: This list identifies the contributing sources for the payload contained in this packet, as defined by [RFC3550] Section 5.1. Additionally, for audio packets coming from mixers, the first CSRC in the list SHOULD be the dominant speaker at the moment in which the packet was generated, even if its audio stream (2) is not included in the mix. For example, the packet sent by the mixer to the dominant speaker itself has its own SSRC on the CSRC list, even though its audio is not actually mixed in that packet. This means that a receiver MUST be able to handle receiving its own SSRC on the first position of the CSRC list without detecting a loop. CSRC list positions other than the first maintain their usual meaning, and a receiver can detect a loop if it receives its own SSRC in those positions.

### 2.2.2 RTCP Compound Packets

RTCP compound packets are a concatenation of simple **RTCP packets**, as specified in [RFC3550] section 6. However, RTCP SDES, RTCP BYE, RTCP SR and RTCP RR can also be sent as simple packets, which means that they are sent as only one RTCP packet, instead of a concatenation of two or more. Accordingly, this protocol modifies the RTCP validation algorithm in [RFC3550] section A.2 to accept simple RTCP SDES, RTCP BYE, RTCP SR and RTCP RR packets. RTCP compound packets can carry one or more of the RTCP profile specific extensions described in section 2.2.10.

### 2.2.3 RTCP Probe Packet

The RTCP probe packet MUST be a simple, non-compound, SR packet with zero report blocks. This packet is used as the first packet of an RTCP packet pair for bandwidth estimation purposes.

### 2.2.4 RTCP Packet Pair Packet

An RTCP packet pair packet is an RTCP compound packet containing an RTCP SR or RR packet. An RTCP probe packet MUST be received previously and there MUST not be any other RTCP packets between the RTCP probe packet and this packet. The receiver MUST ignore those other RTCP packets for bandwidth estimation purpose.

An RTCP packet pair packet MAY contain an RTCP padding profile extension to pad the packet to a specific length.

### 2.2.5 RTCP Packet Pair

An RTCP packet pair is formed by an RTCP probe packet and an RTCP packet pair packet. These packets are sent back to back for bandwidth estimation purposes.

### 2.2.6 RTCP Packet Train Packet

An RTCP packet train packet is an RTCP compound packet containing an RTCP RR packet. The RTCP RR packet MUST contain a packet train packet profile extension. The RTCP RR packet SHOULD also contain an RTCP padding profile extension to pad the total length of the RTCP RR packet to a specific length.

### 2.2.7 RTCP Packet Train

An RTCP packet train is formed by an RTCP packet pair and 5 RTCP packet train packets. These 7 packets are sent back to back for bandwidth estimation purposes. The RTCP packet pair packet and the 5 RTCP packet train packets SHOULD have the same packet size. RTCP padding profile extension CAN be used to pad the packets into the specific size.

### 2.2.8 RTCP Sender Report (SR)

The syntax of the RTCP sender report is as specified in [RFC3550] section 6.4.1, with the following additional notes:

**Sender's packet count, sender's octet count:** The packet and octet counts SHOULD NOT include packet duplicates intentionally sent. For example, packet duplicates can be the retransmission of DTMF end packets, as specified in [RFC4733] section 2.5.1.4.

### 2.2.9 RTCP SDES

The RTCP SDES packets are as specified in [RFC3550] section 6.5, with the exception that all text is null terminated, except for SDES PRIV fields.

#### 2.2.9.1 SDES PRIV extension for media quality

The SDES private extension for media quality follows SDES PRIV, as specified in [RFC3550] Section 6.5.8. The format for media quality SDES PRIV extension is as follows. <12>

- Prefix string MUST be "MS-EVT" and MUST NOT be null terminated.

- Value string MUST NOT be null terminated and MUST follow the following format:

"v=V m=R…RMMMMMMMM q=R…RQQQQQQQQ"

**V:** Version of the extension MUST be 1 (v=1).

**R:** Reserved bits MUST be ignored by the receiver. Might be added in future releases.

**MMMMMMMM:** Bitmask, represented in 8-digit lower case Hexadecimal, indicating which media qualities are known. Each bit can be zero (0) for unknown and 1 for known.

The following table shows the component values for the **m** bitmask.

| Bitmask | Description |
|---|---|
| 0x1 | Send network quality. |
| 0x2 | Receive network quality. |
| 0x4 | Network latency. |
| 0x8 | Network bandwidth. |
| 0x80 | **Received video rate matching.** |
| 0x70 | Reserved for future use. |
| 0x100 | Audio capture device is not functioning. |
| 0x200 | Audio render device is not functioning. |
| 0x400 | Audio render glitch. |
| 0x800 | Low signal to noise ratio on device. |
| 0x1000 | Low speech level on device. |
| 0x2000 | Microphone clipping. |
| 0x4000 | Echo. |
| 0x8000 | Near echo to echo ratio. |
| 0x10000 | Device is in half duplex mode. |
| 0x20000 | Multiple audio endpoints. |
| 0x40000 | Device howling detected. |
| 0xF8000 | Reserved for future use. |
| 0x100000 | Low CPU cycles available. |
| 0xFE0000 | Reserved for future use. |

When a bit mask from the previous table that is listed as reserved is applied to **m** bits, the resulting value MUST be zero (0).

**QQQQQQQQ:** Bitmask, represented in 8-digit lower case Hexadecimal, indicating which media quality is good (0) and which media quality is bad (1). If the **m** bitmask is unknown (0), **Q** bitmask SHOULD be set to zero (0) and MUST be ignored by the receiver.

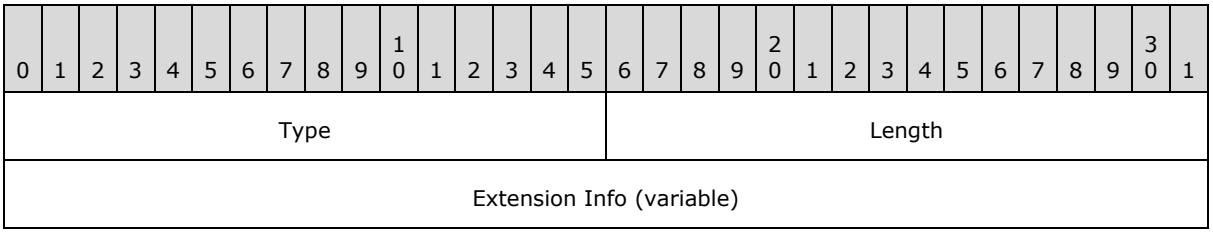
Additional fields, separated by a space and indicated by the same **name=value** syntax, might be added in future releases. These additional fields SHOULD be ignored.

Additional digits (**R**) in the **m** and **q** fields might be added in future releases. However, the least significant 8 digits MUST follow the preceding definition for the **m** and **q** bitmask. Any additional digits SHOULD be ignored.

### 2.2.10 RTCP Profile Specific Extension

The RTCP profile specific extension is appended to the RTCP SR or RR reports and is used to carry additional information not contained in the RTCP SR or RR reports. It is a block of data that immediately follows the RTCP SR or RR report packets. As with the rest of the RTP and RTCP fields, all integer fields on profile specific extensions are in network byte order, with the most significant byte first.

The common header for such extensions is defined as follows:




**Type (2 bytes):** The extension type.

**Length (2 bytes):** The extension length in bytes, including this header.

**Extension info (variable):** Dependent on the extension type.

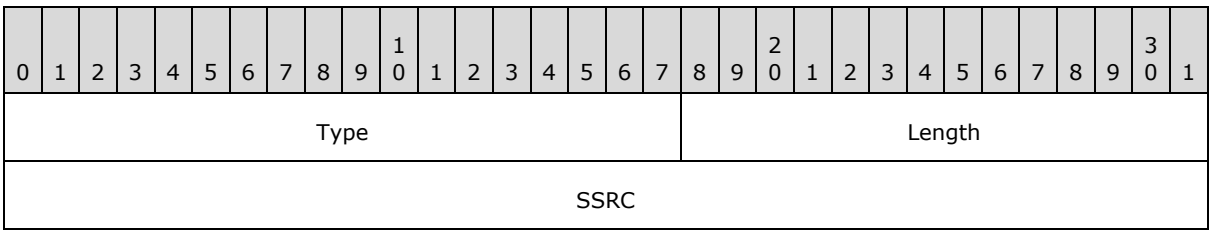
Any profile extension that is not recognized MUST be ignored by using the length field to skip the **Extension Info**. Other **Type** values are not used by any servers and are reserved for future use.

The number of profile extensions in one RTCP SR or RR report MUST be less than or equal to 20.<13>

The extensions defined are described in the next sections.

#### 2.2.10.1 RTCP Profile Specific Extension for Estimated Bandwidth

The format of the RTCP profile specific extension for estimated bandwidth is as follows.



| Bandwidth | | | |
|---|---|---|---|
| Confidence Level | Reserve1 | Reserve2 | Reserve3 |

**Type (2 bytes):** The extension type. Set to 0x0001 (1).

**Length (2 bytes):** The extension length in bytes, including this header. Set to 0x000C (12) or 0x0010 (16).

**SSRC (4 bytes):** The SSRC for which the bandwidth estimated is being reported.

**Bandwidth (4 bytes):** The estimated bandwidth in bits per second. A value of 0xFFFFFFFD (-3) means that this host does not yet have enough measurements to generate a bandwidth estimate and it also indicates the host supports packet pair receiving. A value of 0xFFFFFFFB (-5) means the host supports packet train receiving and it does not have enough measurements to generate a bandwidth estimate<14>. A value of 0xFFFFFFFA (-6) means this host supports packet train receiving and it signals the remote host to send packet train whenever possible<15>.

**Confidence Level (4 bits)** <16>**:** The confidence level of the bandwidth. 0 means the estimated bandwidth is of the lowest confidence or least reliable. 15 means the estimated bandwidth is of the highest confidence or most reliable. A larger confidence level value indicates a more reliable estimated bandwidth.

**Reserve1 (4 bits):** Reserved for future use. The sender SHOULD set it to 0. The receiver MUST ignore it.

**Reserve2 (1 byte):** Reserved for future use. The sender SHOULD set it to 0. The receiver MUST ignore it.

**Reserve3 (2 bytes):** Reserved for future use. The sender SHOULD set it to 0. The receiver MUST ignore it.

The last 4 bytes include the Confidence Level field is optional. The presence of the last 4 bytes MUST be consistent with the Length field. Length of 0x000C (12) indicates the last 4 bytes is not present. Length of 0x0010 (16) indicates the last 4 bytes is present. If the Confidence Level field is not present, then the confidence level of the estimated bandwidth SHOULD be treated as unknown.

### 2.2.10.2 RTCP Profile Specific Extension for Packet Loss Notification

The format of the RTCP profile specific extension for packet loss notification is as follows.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Type | | | | | | | | | | | | | | | | Length | | | | | | | | | | | | | | | |
| Reserved 1 | | | | | | | | | | | | | | | | Reserved 2 | | | | | | | | | | | | | | | |
| Sequence Number | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Type (2 bytes):** The extension type. Set to 0x0004 (4).

**Length (2 bytes):** The extension length in bytes, including this header. Set to 0x0008 (8).

**Reserved 1 (1 byte):** Reserved for future extensions and MUST be set to zero (0). MUST be ignored by the receiver.

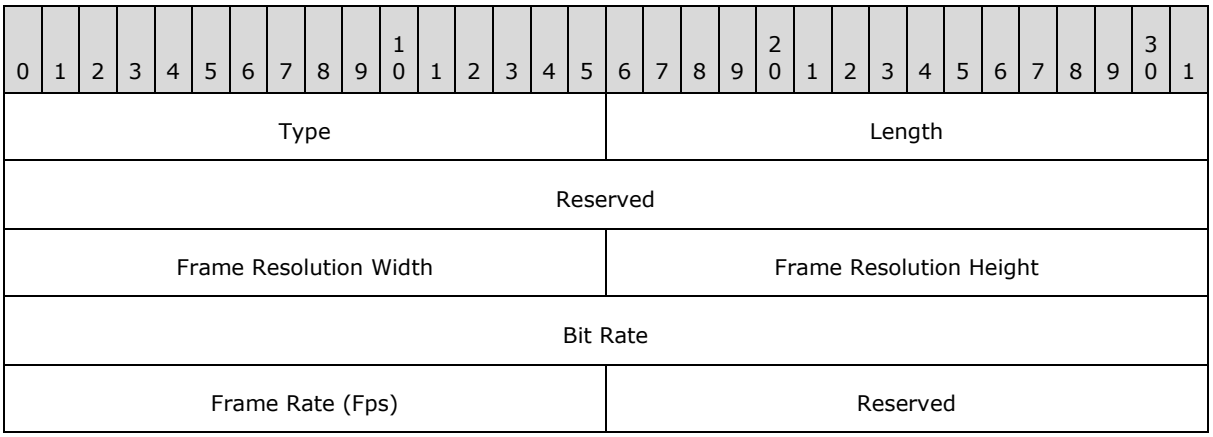
**Reserved 2 (1 byte):** Reserved for future extensions and SHOULD be set to zero (0). MUST be ignored by the receiver.

**Sequence Number (2 bytes):** This is the sequence number of the packet that is being reported as lost.

The frequency at which this request is sent SHOULD NOT be higher than once every 500 milliseconds.

### 2.2.10.3 RTCP Profile Specific Extension for Video Preference

The format of the RTCP profile specific extension for video preference is as follows: <17>




**Type (2 bytes):** The extension type. Set to 0x0005 (5).

**Length (2 bytes):** The extension length in bytes, including this header. Set to 0x0014 (20).

**Reserved (4 bytes):** Reserved for future extensions and SHOULD be set to zero (0). MUST be ignored by the receiver.

**Frame Resolution Width (2 bytes):** The requested width of the video frame in number of pixels. For example, the frame resolution width is 352 for **CIF** video.

**Frame Resolution Height (2 bytes):** The requested height of the video frame in number of pixels. For example, the frame resolution height is 288 for CIF video.

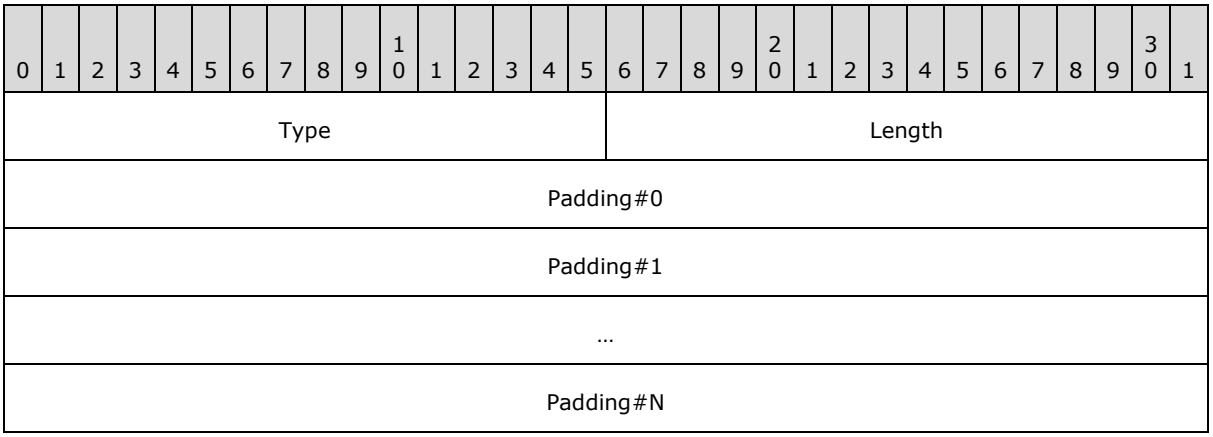
**Bit Rate (4 bytes):** The requested bit rate in kilobits per second. It is reserved for future extensions and SHOULD be set to zero (0). MUST be ignored by the receiver.

**Frame Rate (2 bytes):** The requested frame rate in frames per second. It is reserved for future extensions and SHOULD be set to zero (0). MUST be ignored by the receiver.

**Reserved (2 bytes):** Reserved for future extensions and SHOULD be set to zero (0). MUST be ignored by the receiver.

### 2.2.10.4 RTCP Profile Specific Extension for Padding

The format of the RTCP profile specific extension for padding is as follows.<18>



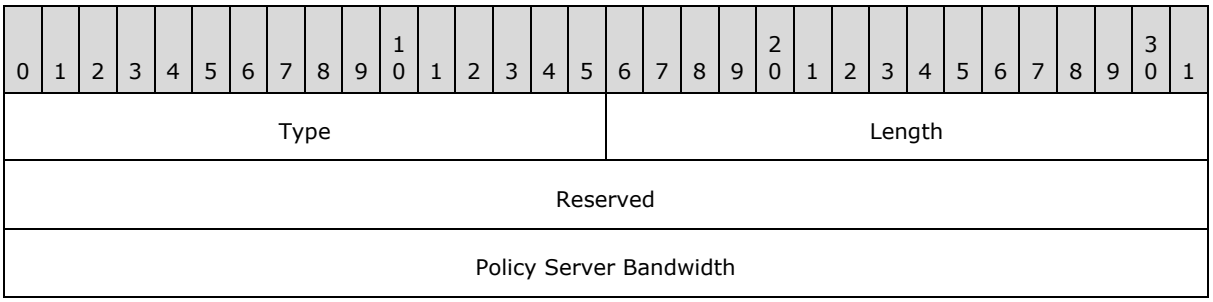

**Type (2 bytes):** The extension type. Set to 0x0006 (6).

**Length (2 bytes):** The extension length in bytes, including this header. The value varies depending on how many padding fields.

**Padding#N (4 bytes):** The sender CAN set any value. The Receiver MUST ignore it. The number of the padding fields MUST be equal to or larger than 0 and smaller than 16383.

### 2.2.10.5 RTCP Profile Specific Extension for Policy Server Bandwidth

The format of the RTCP profile specific extension for policy server bandwidth is as follows.<19>




**Type (2 bytes):** The extension type. Set to 0x0007 (7).

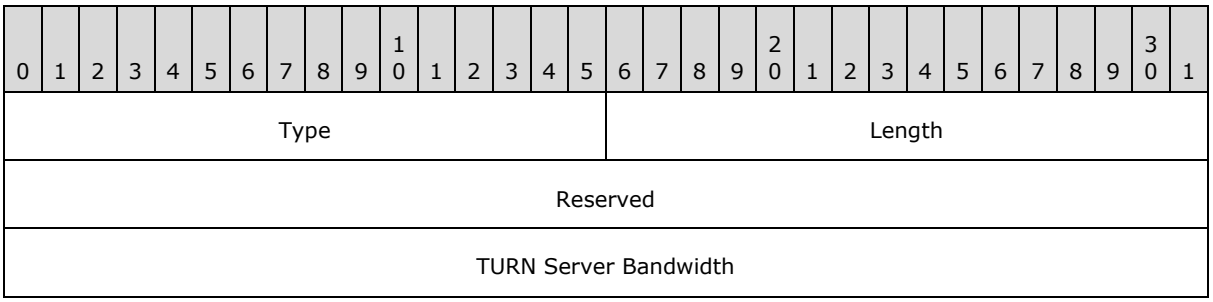
**Length (2 bytes):** The extension length in bytes, including this header. Set to 0x000C (12).

**Reserved (4 bytes):** Reserved for future extensions and SHOULD be set to zero (0). MUST be ignored by the receiver.

**Policy Server Bandwidth (4 bytes):** The maximum bandwidth in bits per second set by the policy server for this stream (2).

### 2.2.10.6 RTCP Profile Specific Extension for TURN Server Bandwidth

The format of the RTCP profile specific extension for TURN server bandwidth is as follows:<20>




**Type (2 bytes):** The extension type. Set to 0x0008 (8).

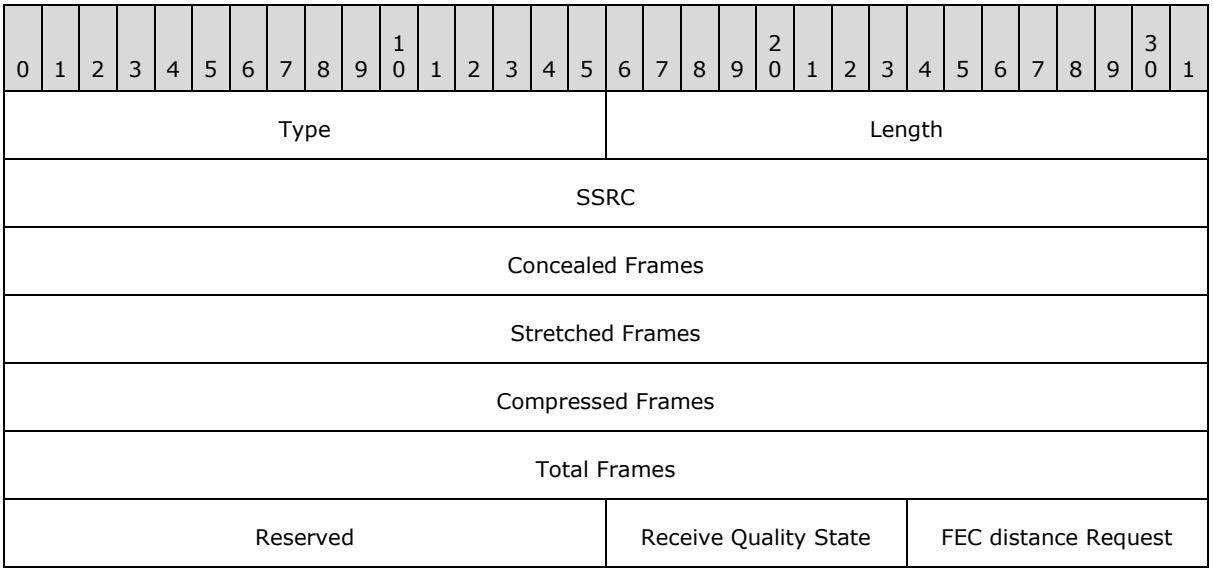
**Length (2 bytes):** The extension length in bytes, including this header. Set to 0x000C (12).

**Reserved (4 bytes):** Reserved for future extensions and SHOULD be set to zero (0). MUST be ignored by the receiver.

**TURN Server Bandwidth (4 bytes):** The maximum bandwidth, in bits per second, set by the TURN server for this stream (2).

### 2.2.10.7 RTCP Profile Specific Extension for Audio Healer Metrics

The RTCP profile specific extension for audio healer metrics is appended to the RTCP RR reports. The format of this extension is as follows. <21>




**Type (2 bytes):** The extension type. Set to 0x0009 (9).

**Length (2 bytes):** The extension length in bytes, including this header. Set to 0x001C (28).

**SSRC (4 bytes):** The SSRC for which the audio healer metrics is being reported.

**Concealed frames (4 bytes):** The total number of concealed audio frames that have been generated during the call. A concealed frame is a frame of audio data that contains generated or reconstructed audio that is intended to conceal lost or missing audio. For this metric, each

frame consists of 10 milliseconds of non-overlapping audio data with one or more frames which are encoded in each RTP packet.

**Stretched frames (4 bytes):** The total number of stretched frames that have been generated during the call. A stretched frame is a frame of audio data that is modified to require more time to play out than the original audio. For this metric, each frame consists of 10 milliseconds of non-overlapping audio data with one or more frames which are encoded in each RTP packet

**Compressed frames (4 bytes):** The total number of compressed frames that have been generated during the call. A compressed frame is a frame of audio data that is modified to require less time to play out than the original audio. For this metric, each frame consists of 10 milliseconds of non-overlapping audio data with one or more frames which are encoded in each RTP packet.

**Total frame (4 bytes):** The total number of frames that have been generated during the call. For this metric, each frame consists of 10 milliseconds of non-overlapping audio data with one or more frames which are encoded in each RTP packet.

**Reserved (2 bytes):** Reserved for future extensions and SHOULD be set to zero (0). MUST be ignored by the receiver.

**Received quality state (1 byte):** The received audio quality so far in the call. It MUST be set to one of the following values.

- 0: Unknown quality.
- 1: Good quality.
- 2: Poor quality.
- 3: Bad quality.

Other values MUST be mapped to "Unknown quality".

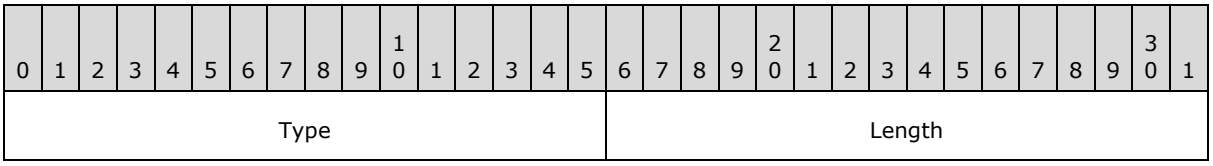
**FEC distance Request (2 bytes):** The FEC distance requested by the receiver from the sender. It MUST be set to one of the following values:

- 0: No FEC requested.
- 1: FEC distance of 1 is requested.
- 2: FEC distance of 2 is requested.
- 3: FEC distance of 3 is requested.

Other values MUST be set to zero (0).

### 2.2.10.8 RTCP Profile Specific Extension for Receiver-side Bandwidth Limit

The format of the RTCP profile specific extension for receiver-side bandwidth limit is as follows: <22>



| Reserved |
| --- |
| Receiver side Bandwidth limit |

**Type (2 bytes):** The extension type. Set to 0x000A (10).

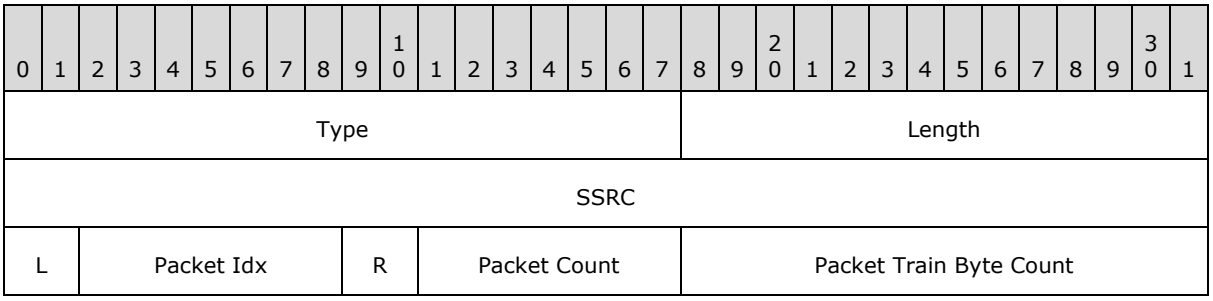
**Length (2 bytes):** The extension length in bytes, including this header. Set to 0x000C (12).

**Reserved (4 bytes):** Reserved for future extensions and SHOULD be set to zero (0). MUST be ignored by the receiver.

**Receiver-side Bandwidth Limit (4 bytes):** The maximum bandwidth, in bits per second, set by the receiver of this stream.

### 2.2.10.9 RTCP Profile Specific Extension for Packet Train Packet

The format of the RTCP profile specific extension for packet train packet is as follows: <23>




**Type (2 bytes):** The extension type. Set to 0x000B (11).

**Length (2 bytes):** The extension length in bytes, including this header. Set to 0x000C (12).

**SSRC (4 bytes):** The SSRC from which the packet train packet is sent.

**L (1 bit):** The last packet train packet flag. This field MUST be set to 1 if the packet is the last packet train packet in the packet train. It MUST be set to 0 otherwise.

**Packet Idx (7 bits):** The index of the packet train packet in the packet train. It starts from 0.

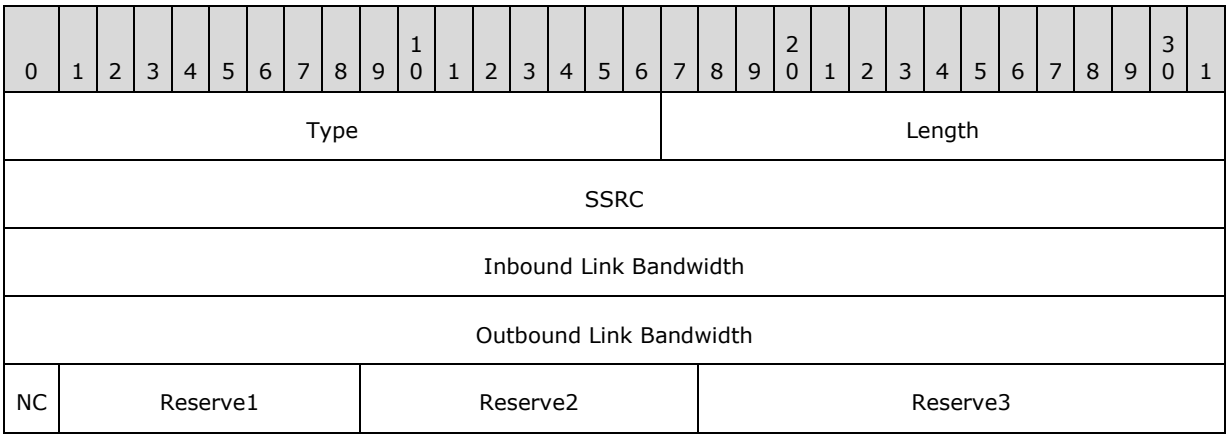
**R (1 bit):** Reserved. The sender SHOULD set it to 0. The receiver MUST ignore it.

**Packet Count (7 bits):** The total number of packet train packets in the packet train.

**Packet Train Byte Count (2 bytes):** The accumulated number of bytes in the RTCP RR packets containing the packet train packet counting from the first packet train packet to this packet train packet.

### 2.2.10.10 RTCP Profile Specific Extension for Peer Info Exchange

The format of the RTCP profile specific extension for peer info exchange is as follows: <24>




**Type (2 bytes):** The extension type. Set to 0x000C (12).

**Length (2 bytes):** The extension length in bytes, including this header. Set to 0x0014(20).

**SSRC (4 bytes):** The SSRC from which the Peer Info Extension packet is sent.

**Inbound Link Bandwidth (4 bytes):** The maximum inbound bandwidth the host CAN support**.**

**Outbound Link Bandwidth (4 bytes):** The maximum outbound bandwidth the host CAN support**.**

**NC (1 bit):** No cache flag. It indicates the inbound and outbound link bandwidth carries in the profile extension MUST not be cached and used beyond this session.

**Reserve1 (7 bits):** Reserved fields. The sender SHOULD set it to 0. The receiver MUST ignore it.

**Reserve1 (1 byte):** Reserved fields. The sender SHOULD set it to 0. The receiver MUST ignore it.

**Reserve1 (2 bytes):** Reserved fields. The sender SHOULD set it to 0. The receiver MUST ignore it.

# 3 Protocol Details

## 3.1 RTP Details

The SSRC throttling mechanism works by means of two states, which are called normal mode and throttling mode. Every time the SSRC changes, the receiver enters the throttling mode, in which further SSRC changes are restricted, and packets with unexpected SSRCs are dropped. The receiver goes back to the normal mode only after a given amount of time has passed without any SSRC change. A high-level overview of this behavior is illustrated in the following diagram. Detailed specifications of the states, transitions, and actions are given in sections 3.1.1 to 3.1.7.

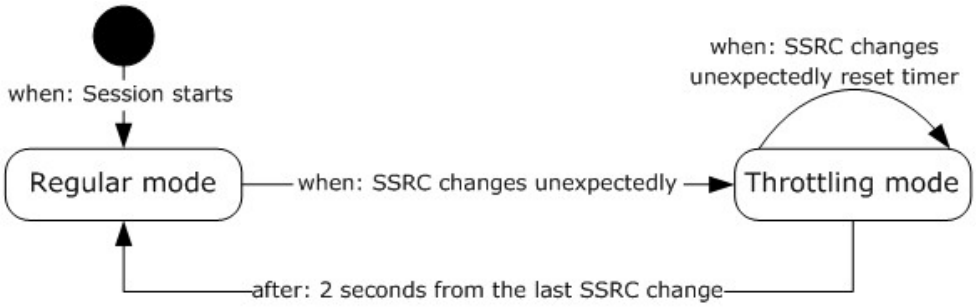



**Figure 2: Synchronization source throttling**

The sequence number throttling mechanism is analogous to the SSRC throttling mechanism, but is done using the values stored on the RTP participant (2), which means one set per SSRC, while the SSRC throttling is global for the RTP session.

A throttling algorithm SHOULD be used on the receiver side. Another example of a throttling mechanism is the probation mechanism specified in [RFC3550] sections 6.2.1 and A.1.

To get correct dominant speaker notification on the protocol clients, the mixer MUST use the first element in the CSRC list as that of the dominant speaker. Receivers can implement special treatment, such as visual indication on the interface, for the dominant speaker, which is the first CSRC on the CSRC list. Senders SHOULD implement an algorithm to select the dominant speaker. For an example, see section 4.2. The nature and behavior of the dominant speaker selection algorithm on the mixer and its usage by the protocol clients is up to the implementation and out of the scope of this specification.

Either the participant (2) time-out timer specified in section 3.1.2 or the time-out algorithm in [RFC3550] section 6.3.5 MUST be used.

### 3.1.1 Abstract Data Model

Common to both throttling mechanisms are the following variables per session:

- **ThrottlingMode**: "True" if the receiver is in throttling mode, which means that the throttling mode timer has not expired.

SSRC throttling extension variables per session:

- **LastGoodSSRC**: Stores the SSRC to be accepted as valid, which is the current SSRC for the stream (2).

- **ResyncSSRC**: Stores the last SSRC that was received out of throttling mode, and was different from **LastGoodSSRC**. If **ResyncSSRC** is seen again, it replaces **LastGoodSSRC**.
- **LastBadSSRC**: Stores the last SSRC that was received inside throttling mode, and was different from both the **LastGoodSSRC** and **ResyncSSRC**.

Sequence number throttling extension variables per participant (2):

- **NextGoodSeqNum:** Stores the next sequence number to be accepted as valid.
- **ResyncSeqNum**: The next number, or successor, in the sequence of numbers received from throttling mode that is not equal to the value in **NextGoodSeqNum**. If **ResyncSeqNum** is seen again, the value **ResyncSeqNum** + 1 replaces **NextGoodSeqNum**.
- **NextBadSeqNum**: The next number, or successor, in the sequence of numbers received from throttling mode that is not equal to the value in either **NextGoodSeqNum** or **ResyncSeqNum**.

Dominant speaker notification extension variables:

- **DominantSpeaker**: Last SSRC received as first CSRC on an RTP packet.
- **IsDominantSpeakerValid**: "True" if the SSRC in **DominantSpeakerSSRC** is valid. To be valid, the dominant speaker expiration timer has not expired and a packet with an empty CSRC list was not received.

### 3.1.2 Timers

This protocol has the following RTP-related timers, in addition to those specified in [RFC3550] section 6.3:

- **Throttling mode timer**: This timer is used by the throttling mechanism to establish how long a receiver stays in throttling mode. There SHOULD be a single throttling mode timer per RTP session, used for both SSRC and sequence number throttling. This timer SHOULD be set to 2 seconds, and MUST be less than or equal to this value.
- **Dominant speaker expiration timer**: Receivers SHOULD have a timer that expires sometime after the last RTP packet was received, to avoid keeping stale dominant speaker information during silent periods if a mixer uses silence suppression. This timer SHOULD be set to 3 seconds. It MUST be greater than the maximum allowed P-time on the RTP session.
- **Participant time-out timer**: This timer or the time-out algorithm in [RFC3550] section 6.3.5 MUST be used to time-out inactive participants (2). This timer SHUOLD be set to 50 seconds. There MUST be one participant (2) time-out timer per participant (2).

### 3.1.3 Initialization

If the SSRC throttling mechanism is used, all related variables MUST be initialized to invalid values at the start of a session. However the very first RTP packet on a stream (2) SHOULD NOT<25> trigger the throttling mode as if it were an SSRC change. For an implementation example, see section 4.1.

Similarly, if the sequence number throttling mechanism is used, all related variables MUST be initialized to invalid values at the start of a session. However, if the probation algorithm is used, it MUST update **NextGoodSeqNum** during the probation stage. For an implementation example, see [RFC3550], sections 6.2.1 and A.1.

**IsDominantSpeakerValid** MUST be initialized to "false". Mixers MUST initialize dominant speaker information according to their specific algorithms. For an example algorithm, see section 4.2.

### 3.1.4 Higher-Layer Triggered Events

This protocol has one RTP-related higher-layer triggered event, in addition to those specified in [RFC3550] and [RFC3551].

If the audio mixer has enough mixed audio data to send an RTP packet, packets are sent as specified in [RFC3550] and [RFC3551], except that if the dominant speaker extension is being used, it MUST run the dominant speaker detection algorithm, either standalone or as part of the audio mixing. It MUST then move the dominant speaker to the first position of the CSRC list, or add it in the first position if the dominant speaker's CSRC is not in the list.

### 3.1.5 Message Processing Events and Sequencing Rules

This protocol processes RTP-related packets as specified in [RFC3550] section 6 and section A.1, with the following additions:

- If RTP and RTCP are being multiplexed, as in the case of TCP, the payload type field MUST be used to differentiate between RTP and RTCP.
- For every received RTP packet, the participant (2) time-out timer of the participant (2) respective to its SSRC MUST be restarted.
- If the throttling mechanism is used, the following actions SHOULD be executed on receipt of every RTP packet:

This code follows the product behavior in footnote <26>.

```
IF SSRC != LastGoodSSRC THEN
     IF SSRC = ResyncSSRC THEN
          SET LastGoodSSRC = SSRC
     ELSE
          IF ThrottlingMode is on THEN
               IF SSRC !=LastBadSSRC THEN
                    SET LastBadSSRC = SSRC
                    RESTART throttling mode timer
               ENDIF
               DROP packet
          ELSE
               SET ResyncSSRC = SSRC
               START throttling mode timer
          ENDIF
     ENDIF
ENDIF
GET participant's information from SSRC
IF SeqNum made a large jump from NextGoodSeqNum (according to [RFC3550] section A.1) THEN
     IF SeqNum = ResyncSeqNum
          SET NextGoodSeqNum = SeqNum's successor
     ELSE
          IF ThrottlingMode is on THEN
               IF SeqNum != NextBadSeqNum THEN
                    SET NextBadSeqNum = SeqNum's successor
                    RESTART throttling mode timer
               ELSE
                    SET NextBadSeqNum = SeqNum's successor
```

```
                ENDIF
                DROP packet
            ELSE
                SET ResyncSeqNum = SeqNum's successor
                START throttling mode timer
            ENDIF
        ENDIF
    ENDIF
```

- If the dominant speaker notification extension is used, the following actions MUST be executed on receipt of every RTP packet from an audio mixer:

```
IF CSRC list is empty
     SET IsDominantSpeakerValid to false
     NOTIFY upper layer that stream has no dominant speaker
ELSE
     SET IsDominantSpeakerValid to true
     START dominant speaker expiration timer
     IF first CSRC != DominantSpeaker
          SET DominantSpeaker = first CSRC
          NOTIFY upper layer that dominant speaker has changed
     ENDIF
ENDIF
```

- If the inter-arrival jitter estimation is computed, the following action SHOULD be executed on receipt of every RTP packet from the network: <27>

```
IF THE PACKET IS NOT DTMF
     CALCULATE JITTER per algorithm in [RFC3550] Section 6.4.1
ELSE
     IGNORE THIS PACKET FOR JITTER CALCULATION
ENDIF
```

### 3.1.6 Timer Events

This protocol has the following RTP-related timer event processing rules, in addition to those specified in [RFC3550] section 6.3:

**Throttling mode timer expires**: No action. The algorithm in section 3.1.5 detects that the timer expired and switches back to normal mode when the next RTP packet is received.

**Dominant speaker expiration timer expires**: The receiver MUST set **IsDominantSpeakerValid** to "false", and notify the upper layer that the stream (2) has no dominant speaker.

**Participant time-out timer expires**: The receiver MUST delete the respective participant (2) object.

### 3.1.7 Other Local Events

This protocol has no additional local RTP-related events, beyond those specified in [RFC3550] section 6 and section A.1.

## 3.2 RTCP Details

RTCP packets SHOULD be sent on every RTP session. Failure to do so can result in loss of functionality on the remote end, because channel statistics such as loss rate and jitter are not communicated, and possibly termination of the session by time-out, if silence suppression is enabled and there is a long period of silence, as specified in section 3.1 of this document or [RFC3550] section 6.3.5.

The RTCP SDES PRIV extension for media quality, as specified in section 2.2.9.1, works as follows:<28>

- The **m** and **q** bits are initialized to zero (0) at the start of the call.
- With every RTCP report an audio host, which is not a mixer, sends an SDES PRIV extension for media quality to the remote host or mixer with the RTCP report.
- If the host detects that the media quality state is good or bad, it SHOULD update the **m** and **q** bits by setting **m** to 1, and **q** to 0 (good) or 1 (bad) and SHOULD send the updated media quality PRIV extension to the remote Host.
- If the mixer receives the SDES private extension from any host, it SHOULD send the SDES private extension for the host to all the other hosts with the regular RTCP reports.
- If the mixer detects a quality state on behalf of a source, it SHOULD combine those bits with the extension received from that host or, if the host hasn't sent an extension, build a new extension with the detected bits. Failure to do so can result in loss of functionality on the remote end because the media quality information is not available at the remote host.

To prevent SDES broadcast flooding from mixer, because it can receive an RTCP PRIV extension for media quality from the same host every few seconds with every RTCP report coming from that host, a mixer SHOULD do the following:

- If the **m** or **q** bits have not changed for a host, the mixer SHOULD send the RTCP PRIV extension for media quality, which contains media quality information for this host, to other hosts every 30 seconds.
- If the **m** or **q** bits have changed for a host and the mixer has sent RTCP PRIV extension for media quality for this host to another host in the last 10 seconds, the mixer SHOULD NOT send the RTCP PRIV extension for media quality to the other host. After the 10 seconds, the mixer SHOULD send the latest **m** and **q** bits. During these 10 seconds, the mixer can receive RTCP PRIV extension for media quality from the same host.

The bandwidth estimation, as specified in section 2.2.10.1, works as follows:

1. One host sends a pair of packets to another host, back to back.
2. The receiver calculates the bandwidth on the link, based on the reception times and packet sizes.
3. The receiver combines multiple measurements to arrive at a bandwidth estimate that is communicated back to the sender through an extension to the RTCP report.

To accelerate bandwidth estimation, the session starts in a "packet pair fast" RTCP sending rate. Once enough RTCP packet pairs have been sent, or the receiver has successfully estimated the bandwidth, the session changes to the "normal" RTCP sending rate. A high-level overview of this behavior is illustrated in the following diagram. Detailed specifications of the states, transitions, and actions are given in sections 3.2.1 to 3.2.7.

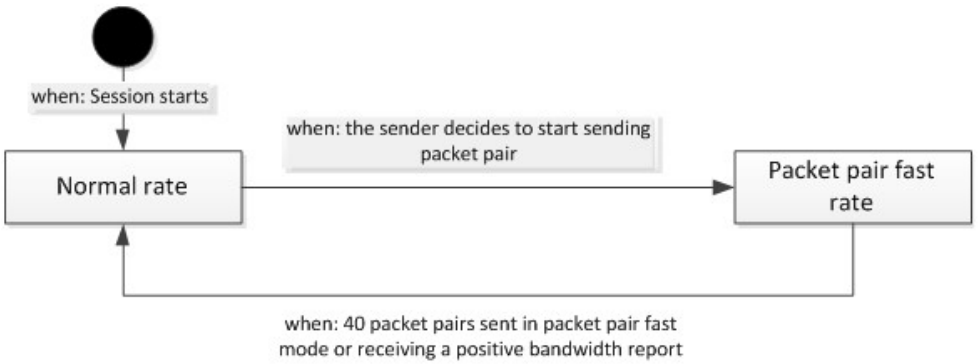



Figure : Behavior of packet pair bandwidth acceleration

There is a similar pattern for packet train bandwidth estimation. The sender starts sending packet train in a "fast" packet train sending rate (one packet train in one second). Once enough RTCP packet trains have been sent, the session changes to the "normal" packet train sending rate. In "normal" RTCP send rate, packet train SHOULD not be sent more than once every 5 seconds. A high-level overview of this behavior is illustrated in the following diagram. Detailed specifications of the states, transitions, and actions are given in sections 3.2.1 to 3.2.7.

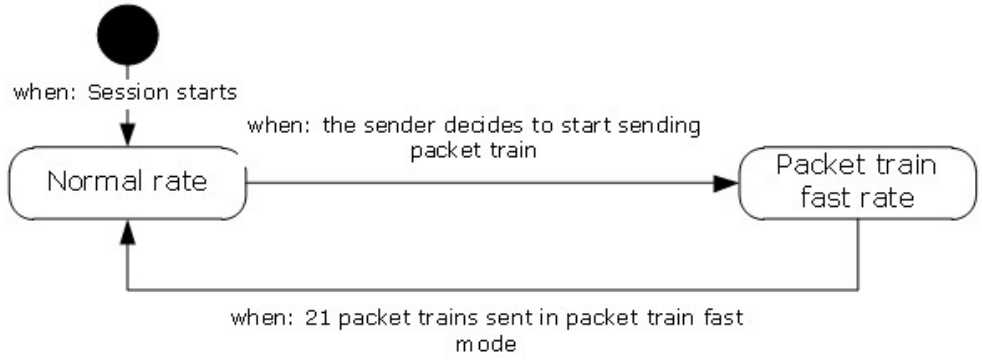



**Figure 3: Behavior of packet train bandwidth acceleration**

If RTCP packet pairs or packet trains are not sent as specified in this document, the receiver cannot send a bandwidth estimate back. Bandwidth estimatesMUST be sent through the profile extension. Failure to do so can result in reduced functionality on the remote end for features that need a bandwidth estimate. RTCP packet pairs and packet trains MUST be correctly received and parsed, but MAY not be used by the bandwidth calculation algorithm.

Packet loss notification extensions SHOULD be sent when video packet loss is detected, and MUST be received and parsed correctly, but the receiver cannot take any action in response to the notification.

A video preference extension, as specified in section 2.2.10.3, SHOULD<29> be sent to the video source when the video receiver prefers to receive a different video resolution. It MUST be received and parsed correctly, but the video source can decide not to take any action in response to this notification. This extension SHOULD be sent at least 10 times for a specific resolution to ensure that the preference is not lost on the wire. The received video preference SHOULD be mapped to the closest resolution in the video capabilities negotiated through SDP extensions for audio and video, as described in [MS-SDPEXT].

Policy server bandwidth policy SHOULD be sent through the policy server bandwidth extension. Failure to do so can result in reduced functionality on the remote end for features that need the policy server bandwidth policy.<30> If a host receives this bandwidth policy as described in [MS-TURNBWM] section 2, it MUST send this to the remote host. This extension SHOULD be sent at least 5 times for a specific bandwidth to ensure that this profile extension is not lost on the wire.

TURN server bandwidth policy SHOULD be sent through the TURN server bandwidth extension. Failure to do so can result in reduced functionality on the remote end for features that need the TURN server bandwidth policy<31>. If a host receives this bandwidth policy, as described in [MS-TURN] section 2.2.2.9, it MUST send this to the remote host. This extension SHOULD be sent at least 5 times for a specific bandwidth to ensure that this profile extension is not lost on the wire.

Audio healer profile specific extension SHOULD be sent from the host receiving audio to the host sending audio in every report if the metric is available. Failure to do so can result in reduced FEC functionality on the send side under packet loss.<32> It MUST be parsed correctly, but the audio source can decide not to take any action in response to this report.

Receiver-side bandwidth limit SHOULD be sent through the Receiver-side bandwidth limit extension. Failure to do so can result in reduced functionality on the remote end for features that need the receiver-side bandwidth limit<33>. It MUST be received and parsed correctly for audio, video and Application sharing profiles. The host SHOULD not send the stream more than this limit to the receiver for application sharing profile. This extension SHOULD be sent at least 5 times for a specific bandwidth to ensure that this profile extension is not lost on the wire.

Peer info exchange extension SHOULD be sent from the host to the remote host in every RTCP packet after the session starts. Enough RTCP packets containing peer info extension SHOULD be sent to avoid the accidental loss of RTCP packets. It is recommended that peer info extension continue to be sent until receiving a bandwidth estimation extension containing a positive bandwidth value from the remote host.

A packet train packet extension MUST be sent in an RTCP packet train packet. An RTCP packet train packet MUST contain one and only one packet train packet extension.

A padding extension SHOULD be used to pad an RTCP packet pair packet or an RTCP packet train packet to a specific size.

### 3.2.1 Abstract Data Model

Common to both the sending rates are the following variables (per session).

**RTCPSendingRate:** Defines the rate at which RTCP reports are sent. Reports are sent either at a packet pair fast rate, at the normal rate, or at the packet train fast rate. The packet pair fast rate uses a fixed time interval, which is defined by the Fast RTCP packet pair sending timer. The normal rate uses a random time interval based on a value that scales with the number of SSRCs in the **conference**, as defined in [RFC3550] Section 6.2. The packet train fast rate also uses a fixed time interval, which is defined by the Fast RTCP packet train sending timer.

**FastRTCPPacketPairCount:** Keeps track of how many packet pairs have been sent at the fast RTCP send rate.

**ReceivingRTCPPacketPairs:** Indicates whether or not RTCP packet pairs have been received.

BandwidthEstSendingMode: Indicates whether RTCP packet pair or packet train is sent to estimate bandwidth.

FastRTCPPacketTrainCount: Keeps track of how many packet trains have been sent at the fast RTCP packet train send rate.

### 3.2.2 Timers

This protocol has the following RTCP-related timers, in addition to those specified in [RFC3550] and [RFC3551]:

**RTCP Send timer:** When the RTCP send rate is "normal", its next value is computed as specified in [RFC3550] Section 6.2. When the RTCP send rate is "packet pair fast", its next value SHOULD be set to 250 milliseconds. This timer is started each time an RTCP compound packet is sent, and is used to schedule the sending of the next RTCP packet pair. When the RTCP send rate is "packet train fast", its next value SHOULD be set to 1 second. This timer is started each time an RTCP compound packet is sent, and is used to schedule the sending of the next RTCP packet train.

**RTCP Bye timer:** This timer SHOULD be set to 20 seconds <34>. It is started when an RTCP BYE is received. There MUST be one timer per participant (2).

**Packet loss notification timer:**<35> This timer SHOULD be set to 200 milliseconds, and MUST be greater than or equal to this value. It is started when an RTCP packet is sent containing a packet loss notification extension.

### 3.2.3 Initialization

This protocol has the following RTCP-related initialization requirements, in addition to those specified in [RFC3550] and [RFC3551]:

**RTCPSendingRate:** Initialized to "normal" when the protocol starts.

**FastRTCPPacketPairCount:** Initialized to zero (0) when the protocol starts.

**ReceivingRTCPPacketPairs:** Initialized to "false" when the protocol starts.

**FastRTCPPacketTrainCount** Initialized to 0 when the protocol starts.

**BandwidthEstSendingMode:** Initialized to "PacketPair" when the protocol starts.

**RTCPPacketTrainSendingRate:** Initialized to "normal" when the protocol starts.

### 3.2.4 Higher-Layer Triggered Events

This protocol has the following RTCP-related higher-layer triggered events, in addition to those specified in [RFC3550] section 6.3:

**Application wishes to leave the RTP session:** RTCP BYE packet can be sent immediately. When the BYE packet is sent immediately, the algorithm described in [RFC3550] section 6.3.7 is not used.

### 3.2.5 Message Processing Events and Sequencing Rules

This protocol processes RTCP-related packets as specified in [RFC3550] section 6.3, with the following additions.

For every RTCP packet, the participant (2) time-out timer, as specified in section 3.1.2, corresponding to the packet's SSRC MUST be restarted.

The following rules apply to specific types of RTCP packets:

- **RTCP Probe Packet:** Arrival time is recorded and the packet is discarded.
- **RTCP SR or RR Packet:** The following rules apply:

- If the packet contains an SR or RR with a report block for the current send SSRC, BandwidthEstSendingMode is "packet pair", **FastRTCPPacketPairCount** is zero (0), and **RTCPSendingRate** is "normal", then **RTCPSendingRate** is set to "packet pair fast", and the RTCP send timer MUST be set to 250 milliseconds.

- If the received packet has a profile specific extension with a positive bandwidth report, **RTCPSendingRate** is "fast", and BandwidthEstSendingMode is "packet pair", then **RTCPSendingRate** is set to "normal".

- If the received packet has a bandwidth estimation extension with 0xFFFFFFFA (-6), the sender MAY decide to switch to send packet train. Then BandwidthEstSendingMode is set to "packet train", FastRTCPPacketTrainCount is set to 0, and RTCPPacketTrainSendingRate is set to "packet train fast", and the RTCP send timer is set 1 second.

- If there is a profile specific extension with a packet loss notification, and the RTP session is a video session, the receiver SHOULD use the sequence number field on this extension to choose a recovery procedure and instruct the video encoder accordingly. For example, the receiver could instruct the video encoder to immediately generate an SP-frame or I-frame.

- If there is a record of a previous RTCP probe packet, **ReceivingRTCPPacketPairs** is set to "true" and an arrival time gap is computed as the difference between the arrival time of this packet and the probe packet.

- The packet length of the RTCP compound packet includes all headers up to the network layer. For example, over UDP includes RTP, UDP, and IP headers.

- These two values are used to compute the bandwidth perceived by these two packets while traversing the path from their source up to their destination, as the RTCP compound packet length divided by the arrival time gap. How specific implementations to estimate bandwidth from individual calculations is outside the scope of this specification.

- **RTCP RR Packet:** The following rules apply:

  - If the RTCP RR packet contains a packet train packet extension, the arrival time is recorded. The packet train packet extension is parsed and used to validate the packet train. If the following conditions are met:

    - the packet train extension has the L bit set to 1

    - there is a previous RTCP probe packet

    - there is a packet pair packet received

    - all packet train packet extensions are received with "Packet Idx" in increasing sequential order and there is no gap between "Packet Idx".

    - The number of packet train packets is equal to the value specified in the packet count field

then sum up the packet length of the RTCP SR/RR containing the packet pair packet, all RTCP packet train packets, and headers up to the network layer for each packet. The sum is used to compute the bandwidth from their source to their destination, as the sum divided by the arrival time gap between the RTCP probe packet and the last RTCP packet train packet. The specific implementations to estimate bandwidth from individual calculations is outside the scope of this specification.

- **RTCP APP Packet:** This packet is ignored.

- **RTCP BYE:** The SSRC from which this packet was sent is designated as having sent an RTCP BYE, and its RTCP bye timer is started.
- **RTCP Packet Train Packet:** Arrival time is recorded. Packet train packet extension SHOULD be parsed and used to validate the packet train.

### 3.2.6 Timer Events

This protocol has the following RTCP-related timer event processing rules, in addition to those specified in [RFC3550] section 6.2.

**RTCP send timer expires:** If **BandwidthEstSendingMode** is "packet pair", a new RTCP packet pair is prepared and sent to the network destination. If **BandwidthEstSendingMode** is "packet train", a new RTCP packet train is prepared and sent to the network destination. All packets in the RTCP packet pair or RTCP packet train MUST be sent back-to-back, that is, the next one immediately after the previous one. Restart the timer. If the **RTCPSendingRate** is "normal", compute a new value for this timer according to [RFC3550] Section 6.2. If the **RTCPSendingRate** is "packet pair fast", set the timer to 250 milliseconds, increment **FastRTCPPacketPairCount** by 1, and if that counter reaches 40, set **RTCPSendingRate** to "normal". If the **RTCPSendingRate** is "packet train fast", set the timer to 1 second, increment **FastRTCPPacketTrainCount** by 1, and if that counter reaches 21, set **RTCPSendingRate** to "normal". If a report is being sent in the compound packet as a part of RTCP packet pair or RTCP packet train, a bandwidth estimation profile extension SHOULD be attached to each report. The sender can stop sending RTCP probe packets, which means it begins sending only RTCP Compound packets, if it determines that the receiver does not support processing of these packets.

**RTCP Bye timer expires:** The information associated with the SSRC that started this timer is deleted. If any packet from the same SSRC arrives after the timer has expired, this SSRC is treated as a new participant (2).

**Packet loss notification timer expires:** No action required. If a packet loss is detected after the timer has expired, the algorithm in section 3.2.7 detects that the timer is expired and sends a new packet loss notification.

### 3.2.7 Other Local Events

This protocol has the following RTCP-related local event processing rules, in addition to those specified in [RFC3550] section 6.

**Video packet loss detected**: If the loss of a video packet is detected and the packet loss notification timer is expired, an RTCP packet pair SHOULD be sent just as if the RTCP send timer had expired, as specified in section 3.2.6, and MUST include a packet loss notification extension containing the lost packet's sequence number. The packet loss notification timer MUST be restarted. The details of how and when to flag that a video packet has been lost are up to the implementation.

# 4 Protocol Examples

In the following examples, only the fields relevant to the extension being demonstrated are shown. SSRC are shown as 1, 2, 3, and 1000 and sequence numbers are shown starting from 1 for illustrative purposes. Real SSRCs are normally random, and sequence numbers normally start at a random value, as specified in section 2.2.

## 4.1 SSRC Change Throttling

The following diagram represents a flow of messages from the sender to the receiver using SSRC change throttling.

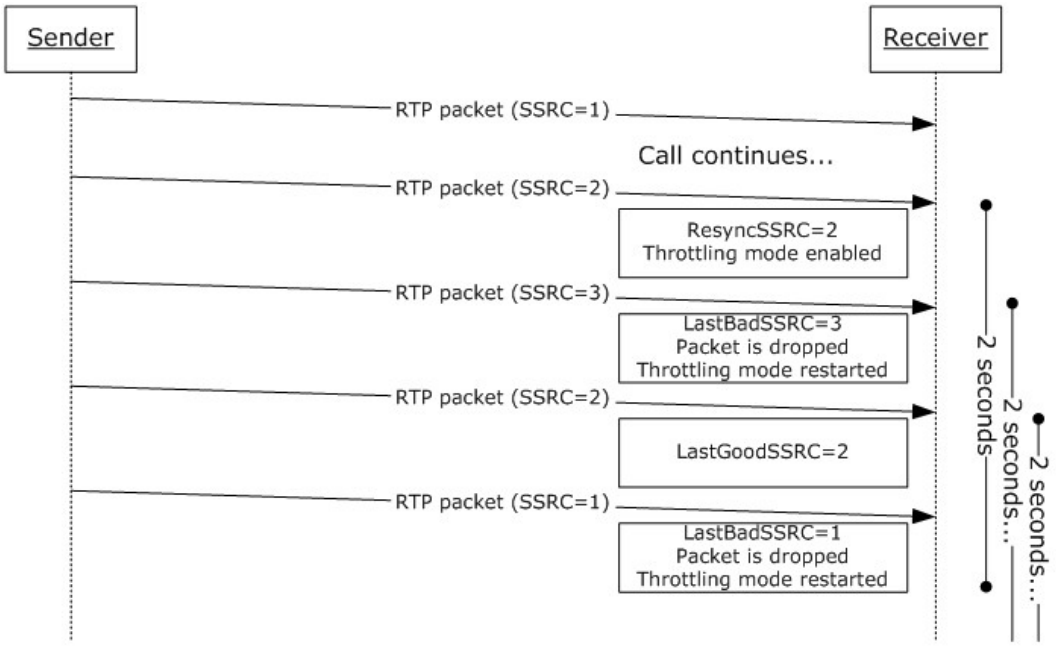



**Figure 4: Synchronization source change throttling**

At the first SSRC change, from SSRC=1 to SSRC=2, throttling mode is enabled. Then a packet with a third SSRC, SSRC=3, is received, which causes the receiver to drop the packet and to restart the throttling mode timer. A packet with the **ResyncSSRC** is received, causing it to be the new valid SSRC for the session. Then a packet with the first SSRC, SSRC=1, is received, but because the receiver has already switched to SSRC=2, this packet's SSRC is unknown, which causes the receiver to drop the packet again, and to restart the throttling mode timer.

## 4.2 Dominant Speaker Notification

The following diagram represents an exchange of messages between the protocol client and the mixer for dominant speaker notification.

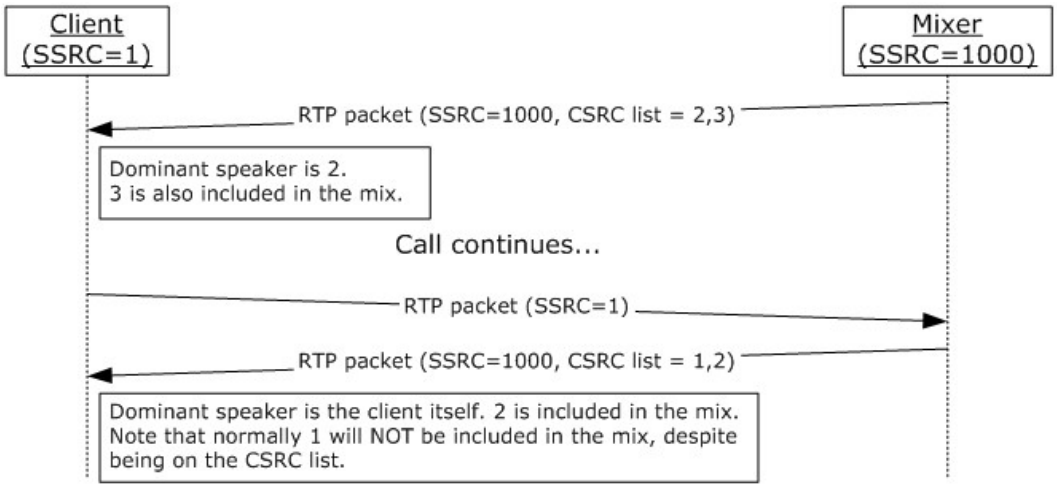



**Figure 5: Message exchange for dominant speaker notifications**

On the first message, the dominant speaker is notified to be the one with SSRC=2. The protocol client can then use this information, perhaps in conjunction with the SDES data communicated through RTCP for SSRC=2, to put a visual indicator beside protocol client 2's name on the user interface.

Then the protocol client is talking, and receives a packet from the mixer indicating that it is the current dominant speaker. The protocol client can use the information to put a visual indicator beside its own name on the user interface. The protocol client does not detect this as a loop.

## 4.3 Bandwidth Estimation

The following diagram represents an exchange of messages between two hosts for bandwidth estimation.

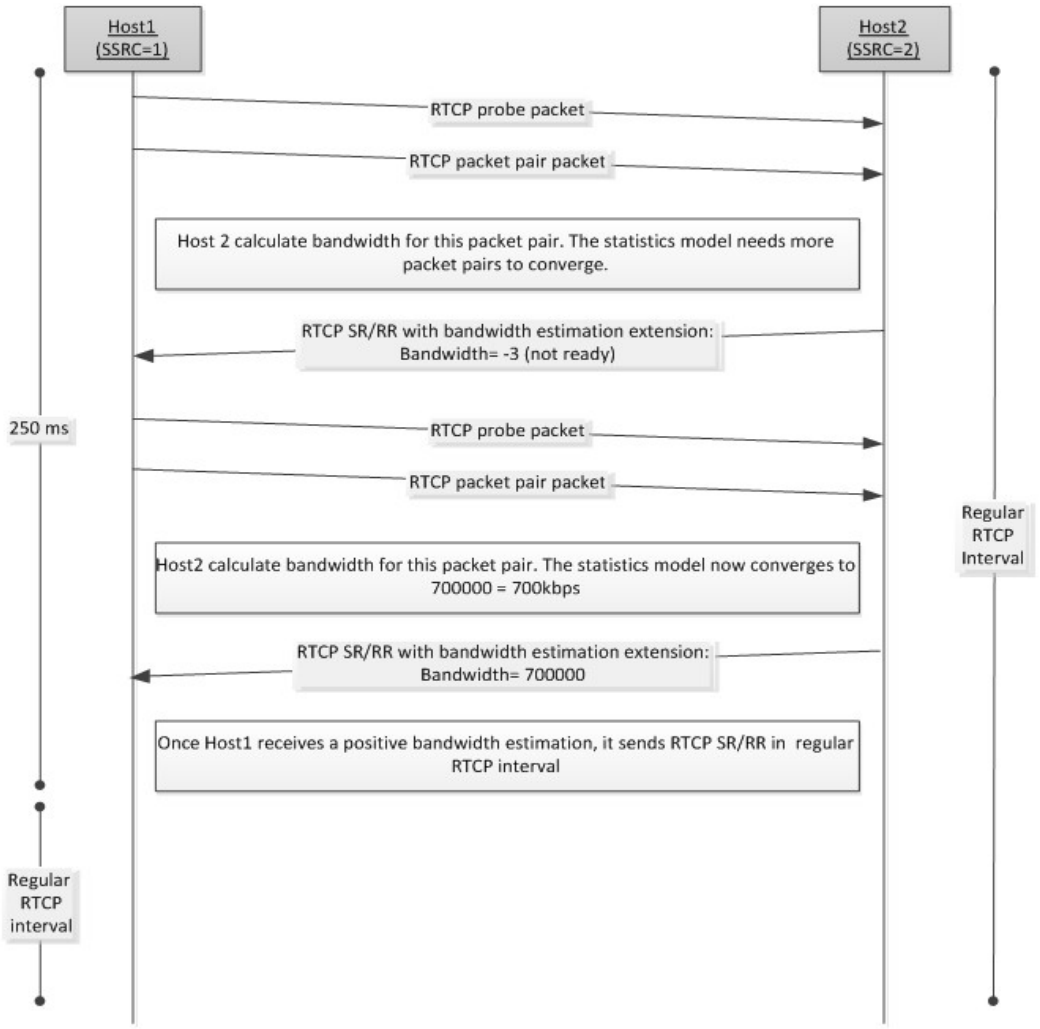



**Figure 6: Message exchange for bandwidth estimation for packet pair**

In the he above figure, Host2 does not support packet train. On receipt of the first RTCP packet pair, Host2 is able to calculate the bandwidth from the initial packet pair, but its particular statistical method needs a more packet pair to converge, so it sends 0xFFFFFFFB (-3) in the bandwidth report to indicate that the estimation is not ready and packet train is not supported. After it receives a few more RTCP packet pairs, the statistical method converges to to 700000. So Host2 sends a bandwidth estimation report with 700000. Host1 switches to the regular RTCP interval after it receives the positive bandwidth estimation.

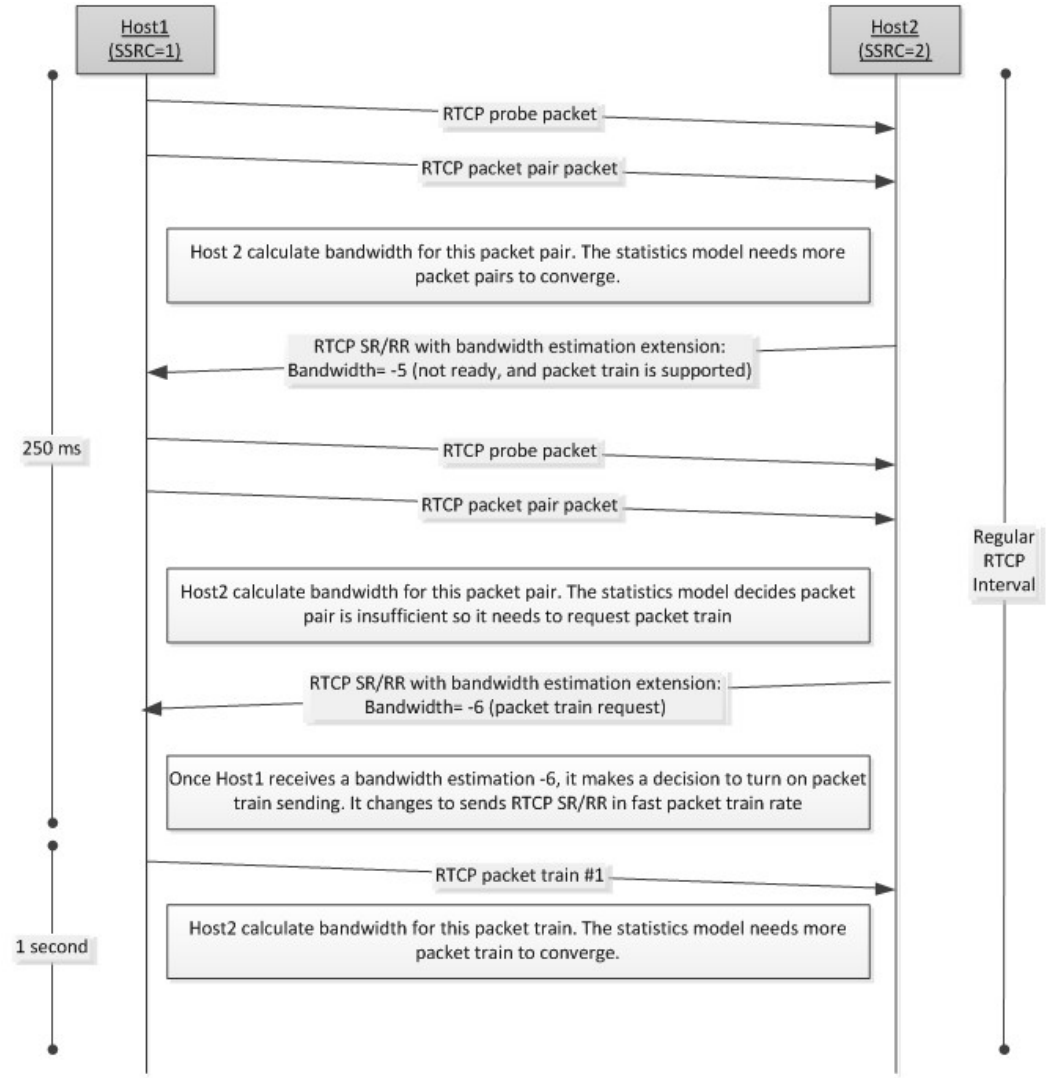



**Figure 7: Message exchange for bandwidth estimation for packet train (Part 1)**

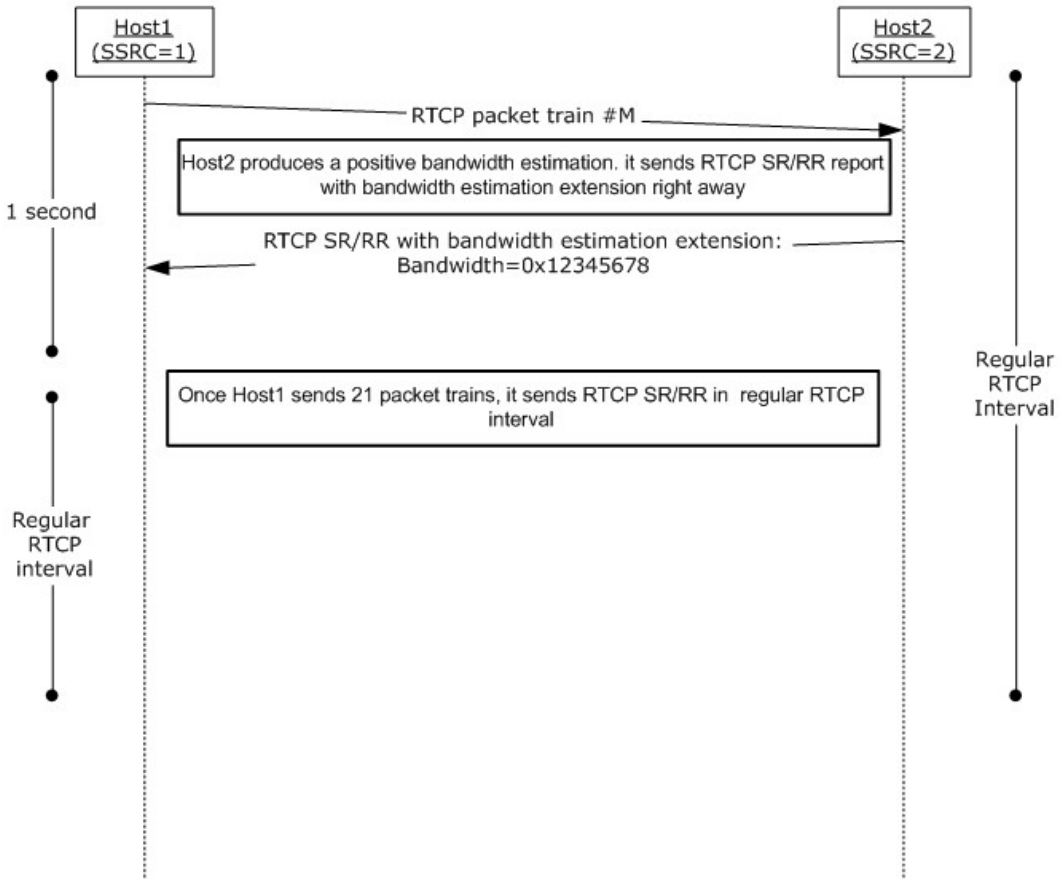



**Figure 8: Message exchange for bandwidth estimation for packet train (Part 2)**

On receipt of the first RTCP packet pair, Host2 is able to calculate the bandwidth from the initial packet pair, but its particular statistical method needs a more packet pair to converge, so it sends 0xFFFFFFFB (-5) in the bandwidth report to indicate that the estimation is not ready and packet train is supported. After it receives a few more RTCP packet pairs, the statistical method decides to request packet train so it sends 0xFFFFFFFA (-6) in the bandwidth report to indicate it would like Host1 to send packet train from then on. Host1 receives the bandwidth estimation report with 0xFFFFFFFA (-6). It makes a decision whether to switch packet train sending mode. If Host1 decides to send packet train, it switches to use 1 second RTCP interval. Once Host2 receives enough packet trains and its statistical method produces a positive bandwidth estimation, it sends a bandwidth report right away.

Host1 switches to the regular RTCP interval after it sends out 21 packet trains.

## 4.4 Packet Loss Notification

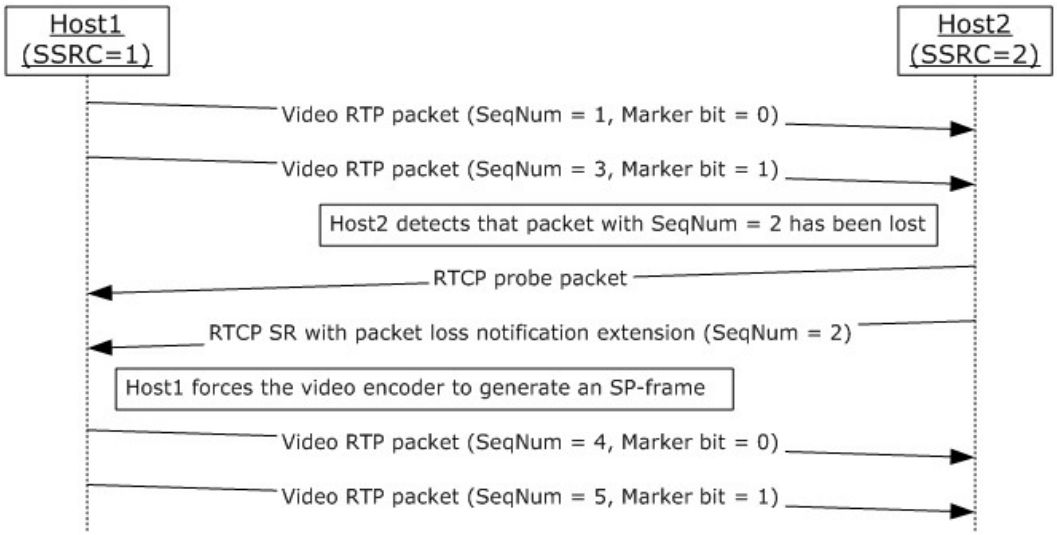



**Figure 9: Message exchange for packet loss notifications**

On receipt of the packet with sequence number 3, Host2 detects that the packet with sequence number 2 has not been received. It then sends an RTCP packet pair with a packet loss notification extension. Upon receiving this notification, Host1 causes the encoder to generate an SP-frame to be sent to Host2. The two subsequent video packets, sequence numbers 4 and 5, contain this SP-frame.

## 4.5 Video Preference

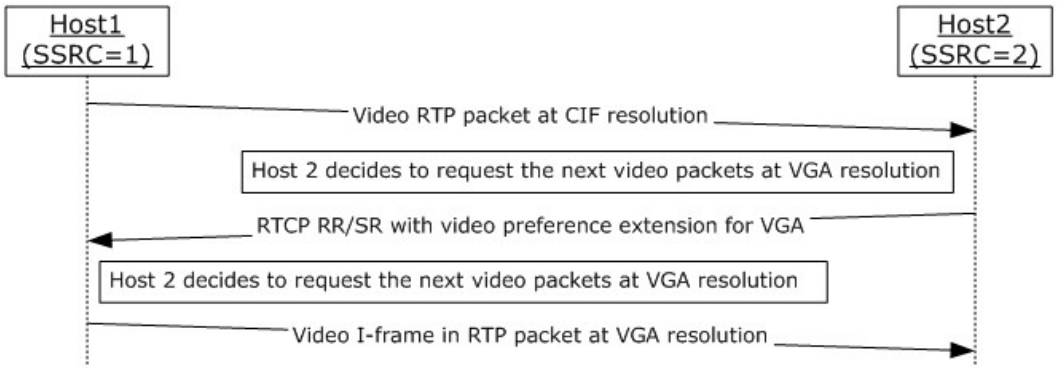



**Figure 10: Message exchange for video preference**

On receipt of the RTP video packet at Common Intermediate Format (CIF) resolution, Host2 decides to request the next video packets at VGA resolution. Host2 sends an RTCP packet with a video preference extension. Upon receiving this preference, Host1 asks the encoder to generate an I-frame at the preferred resolution to be sent to Host2. The encoder can decide to ignore this request if it cannot honor this video resolution. Reasons that the encoder cannot honor this request include:

- The bandwidth is not sufficient.

- The camera does not support the resolution.
- The computer is not powerful enough to honor this request.
- The resolution was not negotiated in video capability negotiation.

If the encoder can honor this request, the next subsequent video packets contain the I-frame of the new resolution.

## 4.6 Policy Server Bandwidth notification

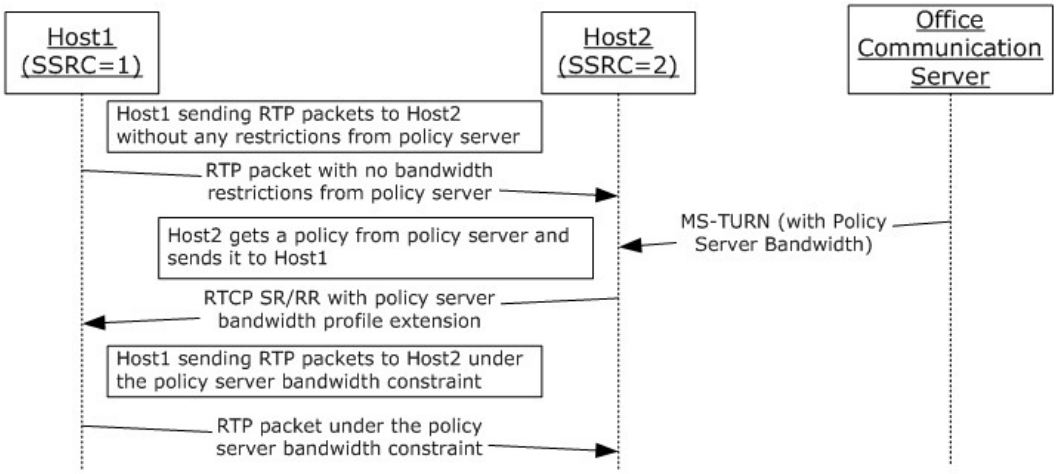



**Figure 11: Message Exchange for Policy server bandwidth extension**

On receipt of the policy server bandwidth via the TURN protocol, Host2 sends an RTCP packet with a policy server bandwidth policy extension to Host1. Upon receiving this extension, Host1 asks the encoder to generate the next frame beneath the policy server bandwidth constraints to be sent to Host2.

## 4.7 TURN Server Bandwidth Notification

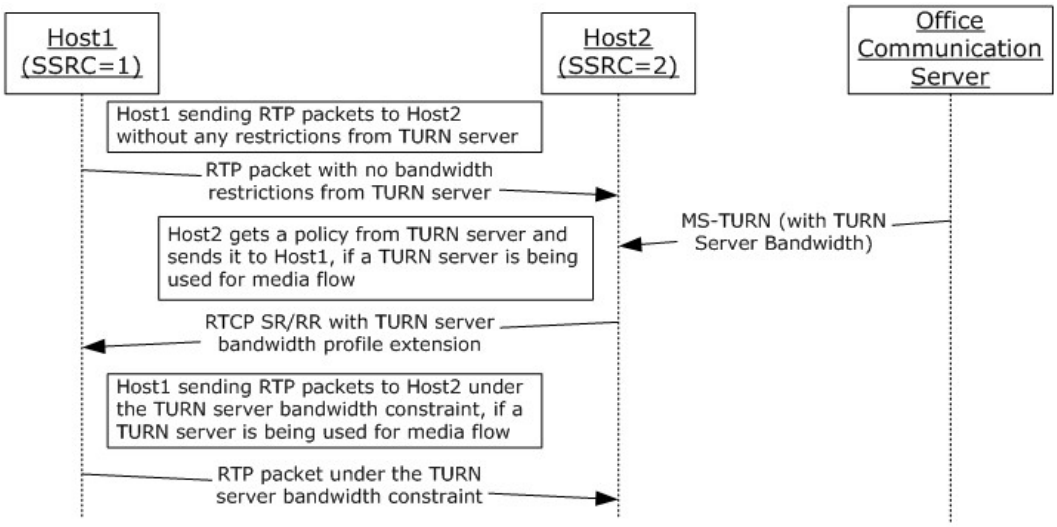



**Figure 12: Message exchange for TURN Server bandwidth extension**

On receipt of the TURN server bandwidth via the TURN protocol, Host2 sends an RTCP packet with a TURN server bandwidth policy extension to Host1. Upon receiving this extension, Host1 asks the encoder to generate the next frame beneath the TURN server bandwidth constraints to be sent to Host2.

## 4.8 Audio Healer Metrics

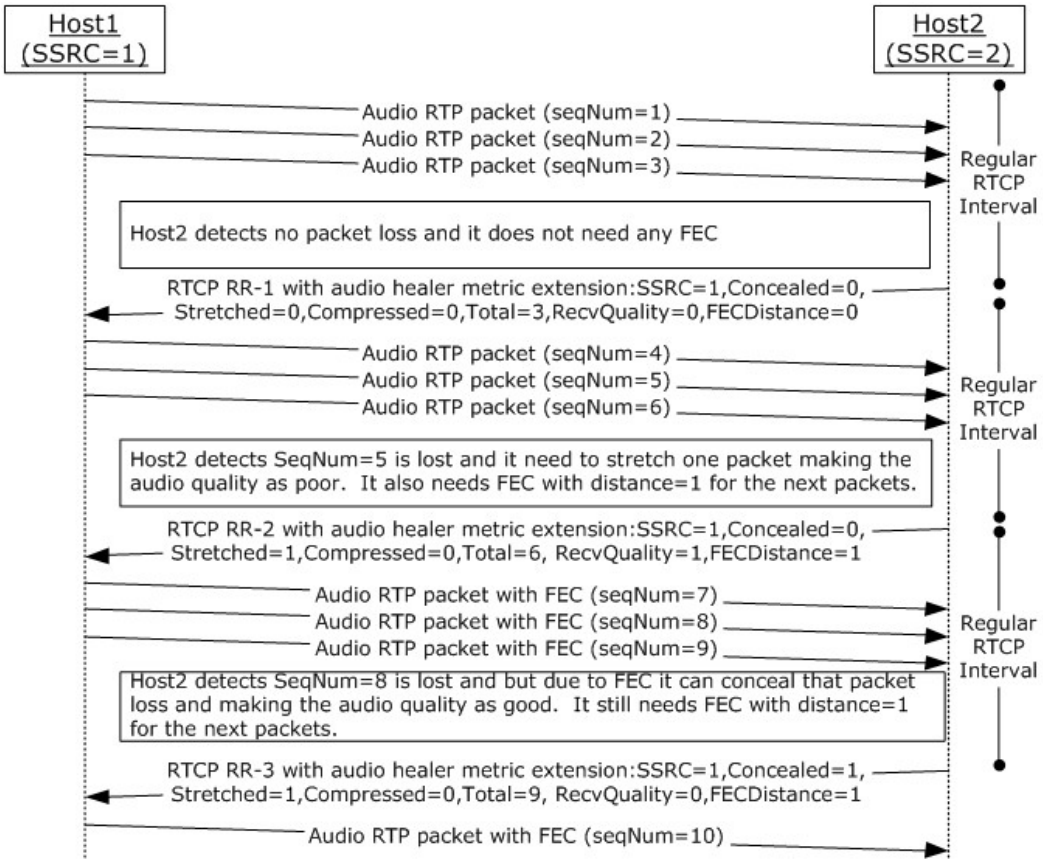



**Figure 13: Message exchange for audio healer metrics extension**

When Host2 is sending the regular RTCP RR, it adds the healer based profile extension for the current call.

- On sending the first RTCP RR, RTCP RR-1, Host2 detects that there is no packet loss and it does not need any FEC. Host2 sends this information in RTCP RR-1.
- On sending the second RTCP RR, RTCP RR-2, Host2 detects that there is a packet loss and it needs FEC with FEC distance =1. Host2 sends this information in RTCP RR-2. Host1 understands this extension and sends the following RTP packets with FEC distance =1.
- On sending the third RTCP RR, RTCP RR-3, Host2 detects that there is a packet loss, but it can recover because of FEC. It still needs FEC with FEC distance =1. Host2 sends this information in RTCP RR-1. Host1 understands this extension and sends the following RTP packets with FEC distance =1.

## 4.9 Receiver-side Bandwidth Limit

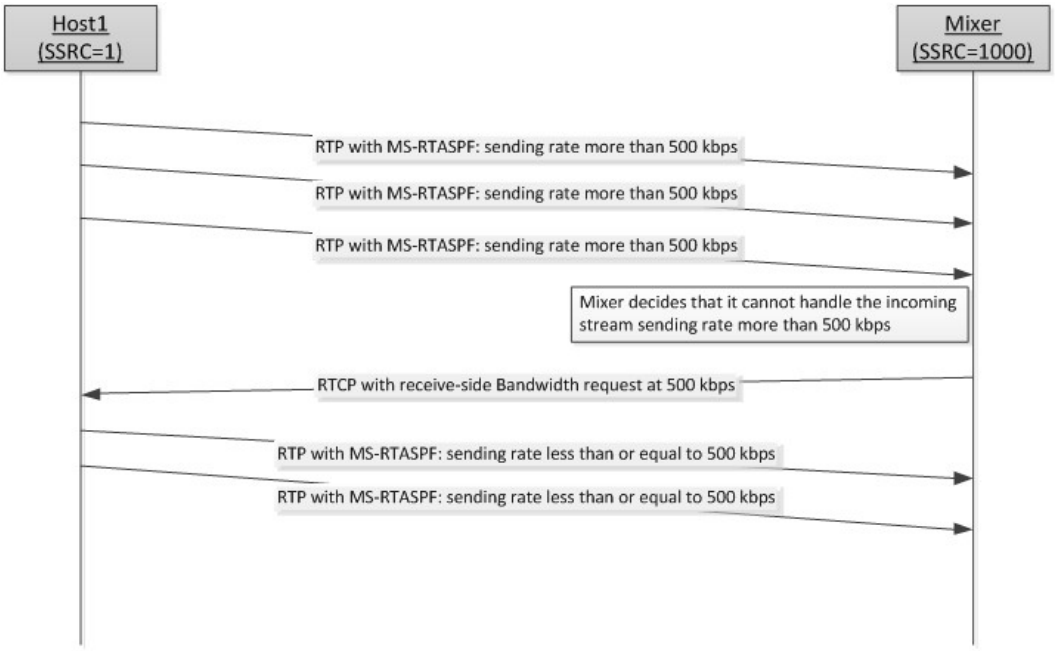



**Figure 14: Message exchange for receiver-side bandwidth limit**

Host 1 is sending Application sharing payload (as described in [MS-RTASPF] section 3.2.5) to the Mixer at a bitrate more than 500 kbps. The Mixer decides that it cannot handle the incoming stream from Host1 at this bitrate and asks the Host1 to send the stream at 500 kbps or lower bitrate by sending the RTCP packet with a receiver-side bandwidth limit extension to Host1. Upon receiving the extension, Host1 caps the outgoing stream to 500 kbps.

## 4.10 SDES Private extension for media quality

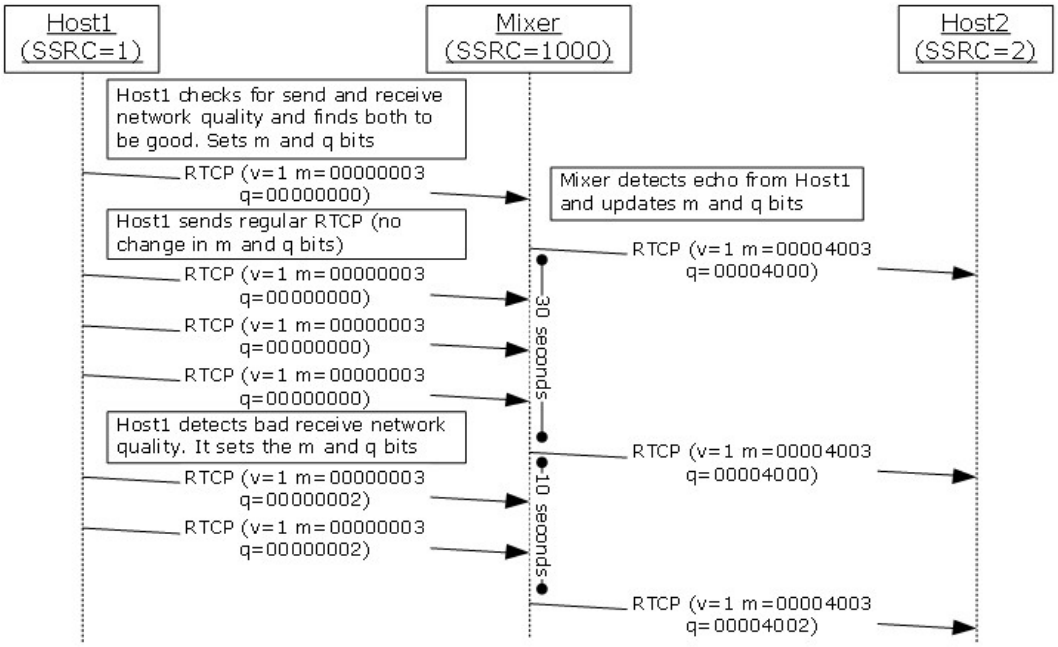



**Figure 15: SDES PRIV extension for media quality**

Host1 and Host2 are having a conference call using a mixer. Host1 checks the send and receive network quality and finds that both are good. Host1 fills in **m** by setting the **m** bitmask for send and receive network quality as 1 and **q** bits, with all q bits as zero (0). Host1 sends it to Mixer in regular RTCP reports. Mixer detects echo from Host1 and updates the **m** and **q** bits. Mixer sends RTCP SDES private extension of media quality to Host2 immediately, because Host1 has not sent SDES PRIV quality state before. After less than 30 seconds, Mixer receives another RTCP report from Host1 with the same **m** and **q** bits. Because the last report was sent less than 30 seconds before, Mixer does not send this extension to Host2. After 30 seconds, Mixer sends another RTCP SDES extension for media quality to Host2 with the same **m** and **q** bits.

After less than 10 seconds, Mixer receives another RTCP report from Host1 with the different **m** or **q** bits. Because the last RTCP SDES private extension was sent less than 10 seconds before, Mixer does not send the extension to Host2. Within the same 10 seconds, Mixer receives another extension from Host1 with different **m** or **q** bits. It updates the **m** and **q** bits for Host1 and, after 10 seconds, sends the latest **m** and **q** bit from Host1 to Host2.

# 5 Security

## 5.1 Security Considerations for Implementers

None.

## 5.2 Index of Security Parameters

None.

# 6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft® Office Communications Server 2007
- Microsoft® Office Communications Server 2007 R2
- Microsoft® Office Communicator 2007
- Microsoft® Office Communicator 2007 R2
- Microsoft® Lync™ Server 2010
- Microsoft® Lync™ 2010

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

<1> Section 2.1: Office Communications Server 2007, Office Communicator 2007: [MS-ICE2] is not supported.

<2> Section 2.1: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: RTP Session SHOULD be terminated between 30 and 40 seconds.

<3> Section 2.2.1: Office Communicator 2007, Office Communicator 2007 R2: Silence suppression cannot be disabled.

<4> Section 2.2.1: Office Communications Server 2007, Office Communicator 2007: DTMF payloads are required to use the same payload type for the send and receive directions.

<5> Section 2.2.1: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: Sending G711 μ-Law with 10 msec P-time is not supported.

<6> Section 2.2.1: Sending /receiving GSM 6.10 is supported for Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2 only.

<7> Section 2.2.1: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: Sending G711 A-Law with 10 msec P-time is not supported.

<8> Section 2.2.1: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<9> Section 2.2.1: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<10> Section 2.2.1: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<11> Section 2.2.1: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<12> Section 2.2.9.1: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<13> Section 2.2.10: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<14> Section 2.2.10.1: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<15> Section 2.2.10.1: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<16> Section 2.2.10.1: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<17> Section 2.2.10.3: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<18> Section 2.2.10.4: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<19> Section 2.2.10.5: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<20> Section 2.2.10.6: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<21> Section 2.2.10.7: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<22> Section 2.2.10.8: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<23> Section 2.2.10.9: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<24> Section 2.2.10.10: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<25> Section 3.1.3: Office Communications Server 2007, Office Communicator 2007: If the SSRC throttling mechanism is used, all related variables are required to be initialized to invalid values at the start of a session. The very first RTP packet on a stream (2) is required to trigger the throttling mode as if it were an SSRC change.

<26> Section 3.1.5: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<27> Section 3.1.5: Office Communications Server 2007, Office Communicator 2007: If the inter-arrival jitter estimation is computed, the jitter per algorithm is required to be calculated on receipt of every RTP packet from the network, which means no special handling for DTMF.

<28> Section 3.2: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<29> Section 3.2: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<30> Section 3.2: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<31> Section 3.2: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<32> Section 3.2: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<33> Section 3.2: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<34> Section 3.2.2: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: RTCP BYE timer should be set to 2 seconds.

<35> Section 3.2.2: Office Communications Server 2007, Office Communicator 2007: Packet loss notification timer: This timer is required to be greater than or equal to 500 milliseconds, with a recommended setting of 500 milliseconds. It is started when an RTCP packet is sent containing a packet loss notification extension.

# 7 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

# 8 Index


**A**

Abstract data model
  RTCP 29
  RTP 23
Applicability 9
Audio healer metrics
  example 41
  message 19

**B**

Bandwidth estimation
  example 34

**C**

Capability negotiation 10
Change tracking 48

**D**

Data model - abstract
  RTCP 29
  RTP 23
Dominant speaker notification
  example 33

**E**

Estimated bandwidth
  example 34
  message 15
Examples 33
  audio healer metrics 41
  bandwidth estimation 34
  dominant speaker notification 33
  Packet loss notification 38
  policy server bandwidth notification 39
  Receiver-side bandwidth limit 42
  SDES private extension 43
  SSRC change throttling 33
  TURN server bandwidth notification 40
  Video preference 38

**F**

Fields - vendor-extensible 10

**G**

Glossary 5

**H**

Higher-layer triggered events
  RTCP 30
  RTP 25

**I**

Implementer - security considerations 44
Index of security parameters 44
Informative references 7
Initialization
  RTCP 30
  RTP 24
Introduction 5

**L**

Local events
  RTCP 32
  RTP 26

**M**

Message processing
  RTCP 30
  RTP 25
Messages
  RTCP Compound Packets 12
  RTCP Packet Pair 13
  RTCP Packet Pair Packet 13
  RTCP Packet Train 13
  RTCP Packet Train Packet 13
  RTCP Probe Packet 13
  RTCP Profile Specific Extension 15
    audio healer metrics 19
    estimated bandwidth 15
    packet loss notification 16
    Packet Train Packet 21
    Peer Info Exchange 21
    Receiver-side bandwidth limit 20
    TURN server bandwidth 18
    video preference 17
  RTCP SDES 13
    SDES PRIV extension 13
  RTCP Sender Report (SR) 13
  RTP Packets 11
  transport 11

**N**

Normative references 6

**O**

Overview (synopsis) 7

**P**

Packet loss notification
  example 38
  message 16
Packet Train Packet
  message 21

Parameters - security index 44
Peer Info Exchange
  message 21
Policy server bandwidth
  example 39
Policy server bandwidth notification
  example 39
Preconditions 9
Prerequisites 9
Product behavior 45

## R

Receiver-side Bandwidth Limit 20
  example 42
References
  informative 7
  normative 6
Relationship to other protocols 8
RTCP
  abstract data model 29
  higher-layer triggered events 30
  initialization 30
  local events 32
  message processing 30
  overview 27
  sequencing rules 30
  timer events 32
  timers 30
RTCP Compound Packets message 12
RTCP Packet Pair message 13
RTCP Packet Pair Packet message 13
RTCP Packet Train message 13
RTCP Packet Train Packet message 13
RTCP Probe Packet message 13
RTCP Profile Specific Extension message 15
  audio healer metrics 19
  estimated bandwidth 15
  packet loss notification 16
  Packet Train Packet 21
  Peer Info Exchange 21
  Receiver-side bandwidth limit 20
  TURN server bandwidth 18
  video preference 17
RTCP SDES message 13
  SDES PRIV extension 13
RTCP Sender Report (SR) message 13
RTP
  abstract data model 23
  higher-layer triggered events 25
  initialization 24
  local events 26
  message processing 25
  overview 23
  sequencing rules 25
  timer events 26
  timers 24
RTP Packets message 11

## S

SDES private extension
  example 43
  message 13
Security
  implementer considerations 44
  parameter index 44
Sequencing rules
  RTCP 30
  RTP 25
SSRC change throttling
  example 33
Standards assignments 10

## T

Timer events
  RTCP 32
  RTP 26
Timers
  RTCP 30
  RTP 24
Tracking changes 48
Transport 11
Triggered events
  RTCP 30
  RTP 25
TURN server bandwidth
  example 40
  message 18
TURN server bandwidth notification
  example 40

## V

Vendor-extensible fields 10
Versioning 10
Video preference
  example 38
  message 17