[MS-RTP]:

Real-time Transport Protocol (RTP) Extensions

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation ("this documentation") for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **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 can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
- **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
- **Patents**. Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft Open Specifications Promise or the Microsoft Community Promise. If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **License Programs**. To see all of the protocols in scope under a specific license program and the associated patents, visit the Patent Map.
- **Trademarks**. The names of companies and products contained in this documentation might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names**. The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are 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 as specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications documentation does 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 documents are intended for use in conjunction with publicly available standards specifications and network programming art and, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

Support. For questions and support, please contact dochelp@microsoft.com.

Preliminary Documentation. This particular Open Specifications document provides documentation for past and current releases and/or for the pre-release version of this technology. This document provides final documentation for past and current releases and preliminary documentation, as applicable and specifically noted in this document, for the pre-release version. Microsoft will release final documentation in connection with the commercial release of the updated or new version of this technology. Because this documentation might change between the pre-release version and the final

version of this technology, there are risks in relying on this preliminary documentation. To the extent that you incur additional development obligations or any other costs as a result of relying on this preliminary documentation, you do so at your own risk.

Revision Summary

Date	Revision History	Revision Class	Comments
4/4/2008	0.1	New	Initial version
4/25/2008	0.2	Minor	Revised and edited technical content
6/27/2008	1.0	Major	Revised and edited technical content
8/15/2008	1.01	Minor	Revised and edited technical content
12/12/2008	2.0	Major	Revised and edited technical content
2/13/2009	2.01	Minor	Revised and edited technical content
3/13/2009	2.02	Minor	Revised and edited technical content
7/13/2009	2.03	Major	Revised and edited the technical content
8/28/2009	2.04	Editorial	Revised and edited the technical content
11/6/2009	2.05	Editorial	Revised and edited the technical content
2/19/2010	2.06	Editorial	Revised and edited the technical content
3/31/2010	2.07	Major	Updated and revised the technical content
4/30/2010	2.08	Editorial	Revised and edited the technical content
6/7/2010	2.09	Editorial	Revised and edited the technical content
6/29/2010	2.10	Editorial	Changed language and formatting in the technical content.
7/23/2010	2.10	None	No changes to the meaning, language, or formatting of the technical content.
9/27/2010	3.0	Major	Significantly changed the technical content.
11/15/2010	3.0	None	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	3.0	None	No changes to the meaning, language, or formatting of the technical content.
3/18/2011	3.0	None	No changes to the meaning, language, or formatting of the technical content.
6/10/2011	3.0	None	No changes to the meaning, language, or formatting of the technical content.
1/20/2012	4.0	Major	Significantly changed the technical content.
4/11/2012	4.0	None	No changes to the meaning, language, or formatting of the technical content.
7/16/2012	4.0	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	4.1	Minor	Clarified the meaning of the technical content.
2/11/2013	4.2	Minor	Clarified the meaning of the technical content.
7/30/2013	4.2	None	No changes to the meaning, language, or formatting of the

Date	Revision History	Revision Class	Comments
			technical content.
11/18/2013	4.2	None	No changes to the meaning, language, or formatting of the technical content.
2/10/2014	4.2	None	No changes to the meaning, language, or formatting of the technical content.
4/30/2014	4.3	Minor	Clarified the meaning of the technical content.
7/31/2014	4.3	None	No changes to the meaning, language, or formatting of the technical content.
10/30/2014	4.4	Minor	Clarified the meaning of the technical content.
3/30/2015	5.0	Major	Significantly changed the technical content.
6/30/2015	5.0	None	No changes to the meaning, language, or formatting of the technical content.
9/4/2015	6.0	Major	Significantly changed the technical content.
7/1/2016	7.0	Major	Significantly changed the technical content.
9/14/2016	7.0	None	No changes to the meaning, language, or formatting of the technical content.
9/29/2016	7.0	None	No changes to the meaning, language, or formatting of the technical content.
9/15/2017	8.0	Major	Significantly changed the technical content.
12/12/2017	8.0	None	No changes to the meaning, language, or formatting of the technical content.
4/27/2018	9.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

The Real-Time Transport Protocol (RTP) Extensions specifies 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. this protocol also provides bandwidth estimation, dominant speaker notification, video-packet loss recovery, and enhanced robustness for receivers.

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

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**.

audio video profile (AVP): A Real-Time Transport Protocol (RTP) profile that is used specifically with audio and video, as described in [RFC3551]. It provides interpretations of generic fields that are suitable for audio and video media sessions.

codec: An algorithm that is used to convert media between digital formats, especially between raw media data and a format that is more suitable for a specific purpose. Encoding converts the raw data to a digital format. Decoding reverses the process.

Comfort Noise payload: A description of the noise level of comfort noise. The description can also contain spectral information in the form of reflection coefficients for an all-pole model of the noise.

Common Intermediate Format (CIF): A picture format, described in the H.263 standard, that is used to specify the horizontal and vertical resolutions of pixels in YCbCr sequences in video signals.

conference: A **Real-Time Transport Protocol (RTP)** session that includes more than one **participant**.

connectionless protocol: A transport protocol that enables **endpoints (2)** 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 (2)** 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).

contributing source (CSRC): A source of a **stream** of **RTP packets** that has contributed to the combined **stream** produced by an RTP **mixer**. The **mixer** inserts a list of the synchronization source (SSRC) identifiers of the sources that contributed to the generation of a particular packet into the RTP header of that packet. This list is called the CSRC list. An example application is audio conferencing where a **mixer** indicates all the talkers whose speech was combined to produce the outgoing packet, allowing the receiver to indicate the current talker, even though all the audio packets contain the same SSRC identifier (that of the **mixer**). See [RFC3550] section 3.

datagram: A style of communication offered by a network transport protocol where each message is contained within a single network packet. In this style, there is no requirement for establishing a session prior to communication, as opposed to a connection-oriented style.

dominant speaker: A **participant** 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.

dual-tone multi-frequency (DTMF): In telephony systems, a signaling system in which each digit is associated with two specific frequencies. This system typically is associated with touch-tone keypads for telephones.

encryption: In cryptography, the process of obscuring information to make it unreadable without special knowledge.

endpoint: (1) A client that is on a network and is requesting access to a network access server (NAS).

(2) A device that is connected to a computer network.

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.

forward error correction (FEC): A process in which a sender uses redundancy to enable a receiver to recover from packet loss.

I-frame: A **video frame** that is encoded as a single image, such that it can be decoded without any dependencies on previous frames. Also referred to as Intra-Coded frame, Intra frame, and key frame.

Interactive Connectivity Establishment (ICE): A methodology that was established by the Internet Engineering Task Force (IETF) to facilitate the traversal of network address translation (NAT) by media.

Internet Protocol version 4 (IPv4): An Internet protocol that has 32-bit source and destination addresses. IPv4 is the predecessor of IPv6.

Internet Protocol version 6 (IPv6): A revised version of the Internet Protocol (IP) designed to address growth on the Internet. Improvements include a 128-bit IP address size, expanded routing capabilities, and support for authentication and privacy.

jitter: A variation in a network delay that is perceived by the receiver of each packet.

Media Source ID (MSI): A 32-bit identifier that uniquely identifies an audio or video source in a conference.

mixer: An intermediate system that receives a set of media **streams** of the same type, combines the media in a type-specific manner, and redistributes the result to each **participant**.

network address translation (NAT): The process of converting between IP addresses used within an intranet, or other private network, and Internet IP addresses.

packetization time (P-time): The amount, in milliseconds, of audio data that is sent in a single Real-Time Transport Protocol (RTP) packet.

participant: A user who is participating in a **conference** or peer-to-peer call, or the object that is used to represent that user.

Real-Time Transport Control Protocol (RTCP): A network transport protocol that enables monitoring of Real-Time Transport Protocol (RTP) data delivery and provides minimal control and identification functionality, as described in [RFC3550].

Real-Time Transport Protocol (RTP): A network transport protocol that provides end-to-end transport functions that are suitable for applications that transmit real-time data, such as audio and video, as described in [RFC3550].

RTCP packet: A control packet consisting of a fixed header part similar to that of **RTP packets**, followed by structured elements that vary depending upon the RTCP packet type. Typically, multiple RTCP packets are sent together as a compound RTCP packet in a single packet of the underlying protocol; this is enabled by the length field in the fixed header of each RTCP packet. See [RFC3550] section 3.

RTP packet: A data packet consisting of the fixed RTP header, a possibly empty list of contributing sources, and the payload data. Some underlying protocols may require an encapsulation of the RTP packet to be defined. Typically one packet of the underlying protocol contains a single RTP packet, but several RTP packets can be contained if permitted by the encapsulation method. See [RFC3550] section 3.

RTP payload: The data transported by **RTP** in a packet, for example audio samples or compressed video data. For more information, see [RFC3550] section 3.

RTP session: An association among a set of **participants** who are communicating by using the **Real-Time Transport Protocol (RTP)**, as described in [RFC3550]. Each RTP session maintains a full, separate space of **Synchronization Source (SSRC)** identifiers.

RTVideo: A video **stream** that carries an RTVC1 bit stream.

Session Description Protocol (SDP): A protocol that is used for session announcement, session invitation, and other forms of multimedia session initiation. For more information see [MS-SDP] and [RFC3264].

Session Initiation Protocol (SIP): An application-layer control (signaling) protocol for creating, modifying, and terminating sessions with one or more participants. **SIP** is defined in [RFC3261].

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

stream: A flow of data from one host to another host, or the data that flows between two hosts.

Super P-frame (SP-frame): A special P-frame that uses the previous cached frame instead of the previous P-frame or **I-frame** as a reference frame.

Synchronization Source (SSRC): A 32-bit identifier that uniquely identifies a media **stream** in a **Real-Time Transport Protocol (RTP)** session. An SSRC value is part of an **RTP packet** header, as described in [RFC3550].

Transmission Control Protocol (TCP): A protocol used with the Internet Protocol (IP) to send data in the form of message units between computers over the Internet. TCP handles keeping track of the individual units of data (called packets) that a message is divided into for efficient routing through the Internet.

Traversal Using Relay NAT (TURN): A protocol that is used to allocate a public IP address and port on a globally reachable server for the purpose of relaying media from one **endpoint (2)** to another **endpoint (2)**.

TURN server: An **endpoint (2)** that receives **Traversal Using Relay NAT (TURN)** request messages and sends TURN response messages. The protocol server acts as a data relay, receiving data on the public address that is allocated to a protocol client and forwarding that data to the client.

User Datagram Protocol (UDP): The connectionless protocol within TCP/IP that corresponds to the transport layer in the ISO/OSI reference model.

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

video frame: A single still image that is shown as part of a quick succession of images in a video.

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

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

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.

[MS-H264PF] Microsoft Corporation, "RTP Payload Format for H.264 Video Streams Extensions".

[MS-H26XPF] Microsoft Corporation, "Real-Time Transport Protocol (RTP/RTCP): H.261 and H.263 Video Streams Extensions".

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

[MSFT-H264UCConfig] Microsoft Corporation, "Unified Communication Specification for H.264 AVC and SVC UCConfig Modes V1.1", 2011, http://download.microsoft.com/download/A/F/B/AFBF8CBE-3A45-472A-93F3-AD8521FBD502/UC_Specification_for_H264_AVC_and_SVC_encoder.pdf

[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

[RFC4585] Ott, J., Wenger, S., Sato, N., et al., "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006, http://www.rfc-editor.org/rfc/rfc4585.txt

[RFC5285] D. Singer, H. Desineni, "A General Mechanism for RTP Header Extensions", http://tools.ietf.org/html/rfc5285

[RFC5506] Johansson, I., and Westerlund, M., "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", April 2009, http://www.ietf.org/rfc/rfc5506.txt

1.2.2 Informative References

[abs-send-time] WebRTC, "RTP Header Extension for Absolute Sender Time", http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time

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

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

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

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

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

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

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

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

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

[MS-TURN] Microsoft Corporation, "Traversal Using Relay NAT (TURN) 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., and Taylor, T., "RTP Payload for DTMF Digits, Telephony Tones and Telephony Signals", RFC 4733, December 2006, http://www.ietf.org/rfc/rfc4733.txt

[RFC5761] Perkins, C., and Westerlund M., "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010, http://www.rfc-editor.org/rfc/rfc5761.txt

1.3 Overview

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** 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** 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 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.
- **Picture loss indication:** Defines an RTCP feedback message that enables a receiver to indicate there is a packet loss to the sender.
- **Video source request:** Defines an RTCP feedback message that enables a receiver to request video from a specified video source.
- **Dominant speaker history notification:** Defines an RTCP feedback message that enables a **mixer** to notify the receivers the history of dominant speakers.

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.

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 described 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.

This protocol supports the **RTCP** protocol, as described in [RFC3550]. This protocol also supports the RTCP-based feedback protocol, as described in [RFC4585], and the reduced-size RTCP protocol, as described in [RFC5506] with extensions described in section 2.2.12.

RTP supports **Comfort Noise payload**, as described in [RFC3389] section 4. Comfort Noise is used for audio **codecs** that do not support Comfort Noise, 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 Comfort Noise 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** and H264, as described in [MS-RTVPF] and [MS-H264PF].

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

CN Events, CN over RTP, G.722, G.722/2 (see section 2.2.1.1), H264, H263, RDP Payload, and RDP over RTP are not uniformly supported across all **endpoints (2)**.

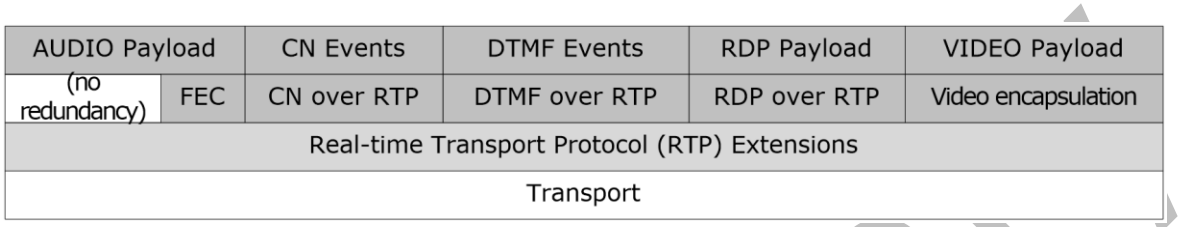


Figure 1: Protocol hierarchy of RTP with the extension

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** in this document.

1.7 Versioning and Capability Negotiation

This protocol does not change the versioning negotiation of RTP.

This protocol does not change the capability negotiation of RTP.

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 **TCP** for **Internet Protocol version 4 (IPv4)**/**Internet Protocol version 6 (IPv6) endpoints (2)**.<1> 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.

This protocol MAY have multiple RTP sessions sharing the same transport. When sharing the same transport, each RTP session MUST use **SSRCs** in the range specified in [MS-SDPEXT] section 3.1.5.31. These SSRC ranges MUST NOT overlap.

This protocol SHOULD multiplex RTP packets and RTCP packets on one single transport, as described in [RFC5761].

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**, if **silence suppression** is enabled, the marker bit (**M**) SHOULD be one for the first packet of a talk spurt and zero 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 one for the first packet in the stream, but SHOULD<3> be zero for all other packets. In video streams, the marker bit MUST be one for the last packet sent for each **video frame**, and zero 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 numbers in the dynamic range can use a 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

Payload type	Codec	Clock rate	P-times	Channels
9 or 117	G.722<8>	8000	20, 40, 60	1
13	Comfort Noise<9>	8000	Not Applicable	1
103	Silk<10>	8000	60	1
104	Silk	16000	20,60,100	1
106	OPUS	48000	20,60	2
111	Siren	16000	20, 40, 60, 100, 200	1
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
117	G.722<11>	8000	20,40,60	2
118	Comfort Noise<12>	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-H26XPF]<13>	90000
121	RT Video	90000
122	H.264 [MS-H264PF]<14>	90000
123	H.264 FEC [MS-H264PF]<15>	90000

The following table lists data codecs with payload type numbers and clock rates<16>:

Payload type	Codec	Clock rate
127	x-data	90000

SSRC: This field identifies the synchronization source. This identifier SHOULD be chosen randomly, or SHOULD be configured by methods specified in [MS-SDPEXT] section 3.1.5.31. The **SSRC** value MUST NOT be zero. The loop detection and collision resolution algorithms from [RFC3550] section 8.2 MUST NOT be used because this protocol MAY use **Media Source ID (MSI)** in the **CSRC** list in a packet from a **mixer**. See the following definition for the CSRC list for details. When **SSRC** is not configured, **SSRC** SHOULD NOT be changed within 2 seconds of the start of the stream 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. This protocol differs from [RFC3550] in CSRC values. It uses MSI instead of **SSRC** of the contributing sources<17>. 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 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.

2.2.1.1 G722 Encoding

This protocol differs from [RFC3551], as it supports both the mono (1 channel) and stereo (2 channels) G722 encoding. G722 mono encoding is specified in [RFC3551] section 4.5.2.

G722 stereo encoding concatenates two G722 mono encodings, with the left channel's encoding at the beginning, followed by the right channel's encoding. It is a simple concatenation of two mono G722 frames with the following format:

Left channel's G722 encoding || Right channel's G722 encoding

G722 stereo encoding uses payload type 117. G722 mono encoding uses payload type 9 or 117.

2.2.1.2 RTP Header Extension

This protocol provides an extension mechanism to carry additional information in the **RTP packet** header, as specified in [RFC3550] section 5.3 and [RFC5285]<18>.

The only extension supported is the absolute send time extension as specified in [abs-send-time].<19>

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 mean 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 2.2.11.

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 MUST use the RTCP padding profile extension if there is a need 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 RR packet. The **RTCP** RR packet MUST contain a packet train packet profile extension. The RTCP RR packet MUST 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 five RTCP packet train packets. These seven packets are sent back to back for bandwidth estimation purposes. The RTCP packet pair packet and the five RTCP packet train packets SHOULD have the same packet size. **RTCP** padding profile extension MUST 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.

When there is more than one **stream** in the **RTP session**, this protocol only sends SR packets for the lowest **SSRC** in the SSRC range. The packet and octet counts SHOULD be the sum of packet and octet counts of all streams in the RTP session.

2.2.9 RTCP Receiver Report (RR)

The syntax of the **RTCP** receiver report is as specified in [RFC3550] section 6.4.2, with the following additional notes:

Last SR and Delay since Last SR: When there is more than one **stream** in the **RTP session**, this protocol uses the SR packet for the lowest **SSRC** in the SSRC range to compute LSR and DLSR in each report block.

When there is more than one stream in the RTP session, this protocol creates one report block for each stream. If there are more than 10 streams, this protocol puts the report blocks of the last 10 streams from which it receives **RTP packets**.

2.2.10 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.

This protocol differs from [RFC3550] in the value of the **CSRC** field in the SDES chunk. When a **mixer** uses this protocol to send SDES for contributing sources, the CSRC field in the SDES chunk is the **MSI** instead of **SSRC** from the contributing sources.

2.2.10.1 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. <20>

- 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 either zero (0) for unknown or 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).

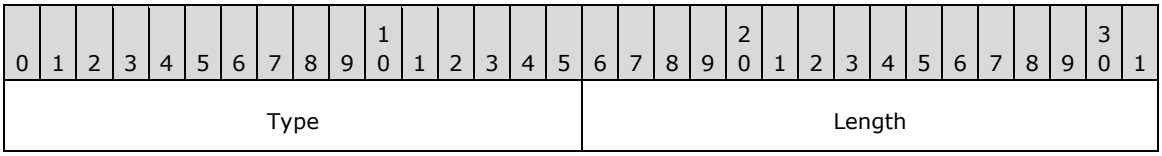
- **QQQQQQQQ:** Bitmask, represented in 8-digit lower case Hexadecimal, indicating which media quality is good (0) and which media quality is bad (1). If the **m** bitmask is unknown (0), **Q** bitmask SHOULD be set to zero (0) and MUST be ignored by the receiver.

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.

2.2.11 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:



Extension Info (variable)	

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.

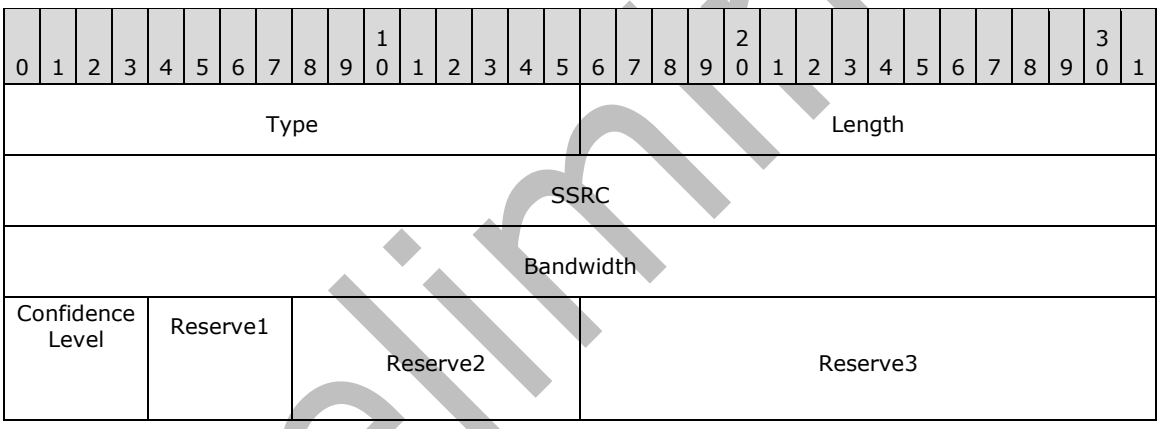
Any profile extension that is not recognized MUST be ignored by using the length field to skip the **Extension Info**. Other **Type** values are not used by any servers and are reserved for future use.

The number of profile extensions in one RTCP SR or RR report MUST be less than or equal to 20.<21>

The extensions defined are described in the next sections.

2.2.11.1 RTCP Profile Specific Extension for Estimated Bandwidth

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



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<22>. A value of 0xFFFFFFFA (-6) means this host supports packet train receiving and it signals the remote host to send packet train whenever possible<23>.

Confidence Level (4 bits) <24>**:** The confidence level of the bandwidth. A value of 0 means the estimated bandwidth is of the lowest confidence or least reliable. A value of 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.

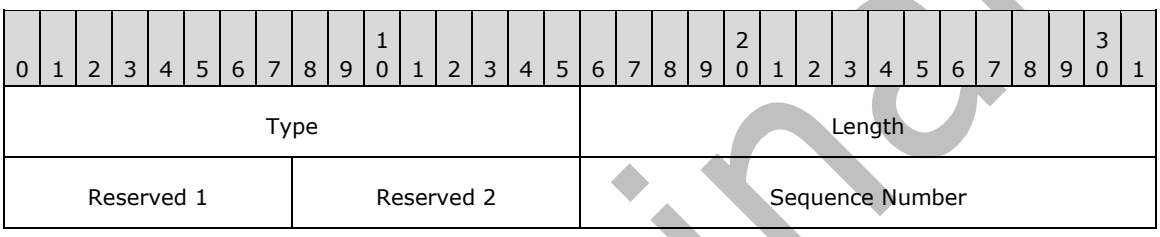
Reserve2 (1 byte): Reserved for future use. The sender SHOULD set it to 0. The receiver MUST ignore it.

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, which is optional. The presence of the last 4 bytes MUST be consistent with the Length field. A 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.

2.2.11.2 RTCP Profile Specific Extension for Packet Loss Notification

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



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.

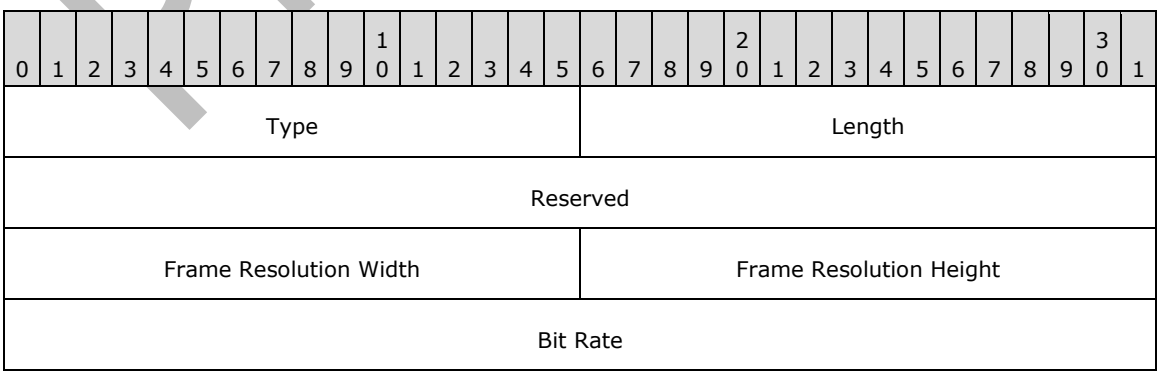
Reserved 2 (1 byte): Reserved for future extensions and SHOULD be set to zero (0). MUST be ignored by the receiver.

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.

2.2.11.3 RTCP Profile Specific Extension for Video Preference

The format of the **RTCP** profile specific extension for video preference is as follows:<25>



Frame Rate (Fps)	Reserved

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.

Frame Resolution Height (2 bytes): The requested height of the video frame in number of pixels.

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.

2.2.11.4 RTCP Profile Specific Extension for Padding

The format of the **RTCP** profile specific extension for padding is as follows.<26>

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															
Padding#0																															
Padding#1																															
…																															
Padding#N																															

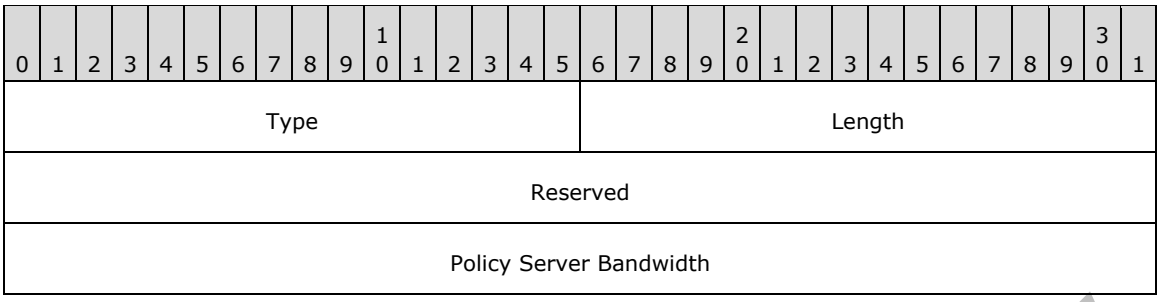
Type (2 bytes): The extension type. Set to 0x0006 (6).

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 MAY 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.

2.2.11.5 RTCP Profile Specific Extension for Policy Server Bandwidth

The format of the **RTCP** profile specific extension for policy server bandwidth is as follows.<27>



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

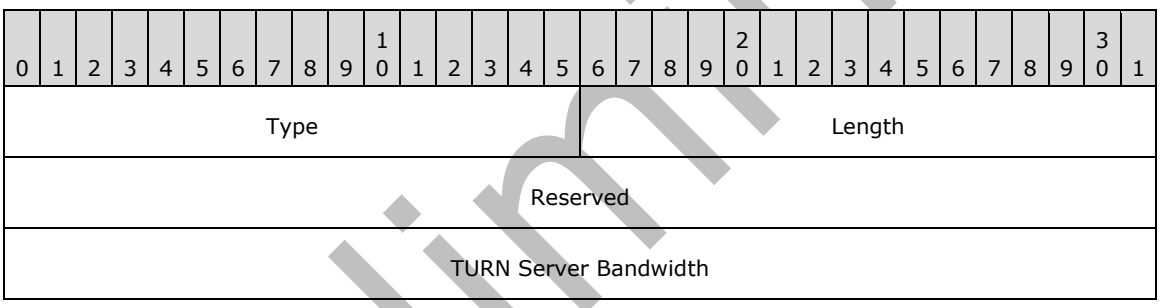
Length (2 bytes): The extension length in bytes, including this header. Set to 0x000C (12).

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**.

2.2.11.6 RTCP Profile Specific Extension for TURN Server Bandwidth

The format of the **RTCP** profile specific extension for **TURN server** bandwidth is as follows:<28>



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

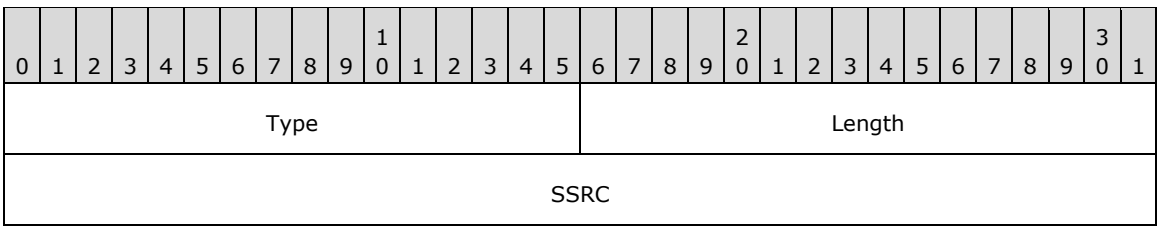
Length (2 bytes): The extension length in bytes, including this header. Set to 0x000C (12).

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**.

2.2.11.7 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. <29>



Concealed Frames		
Stretched Frames		
Compressed Frames		
Total Frames		
Reserved	Receive Quality State	FEC distance Request

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 that 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 that 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 that 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 that are encoded in each RTP packet.

Reserved (2 bytes): Reserved for future extensions and SHOULD be set to zero (0). This 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 (1 byte): The **FEC distance** requested by the receiver from the sender. It MUST be set to one of the following values:

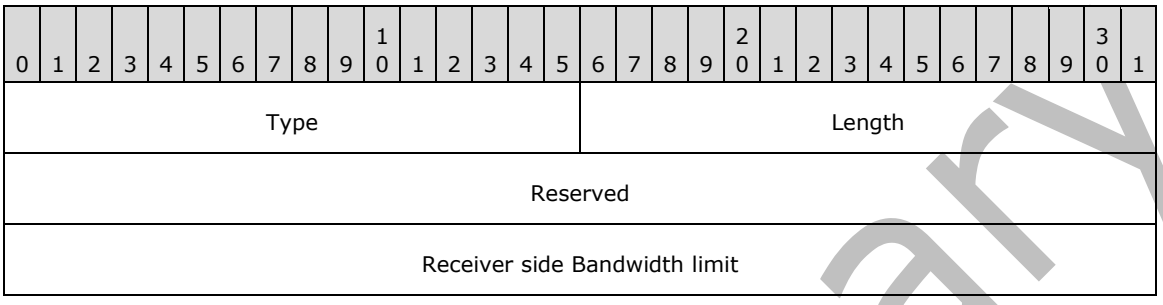
- 0: No **FEC** requested.
- 1: FEC distance of 1 is requested.

- 2: FEC distance of 2 is requested.
- 3: FEC distance of 3 is requested.

Other values MUST be set to zero (0).

2.2.11.8 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: <30>



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

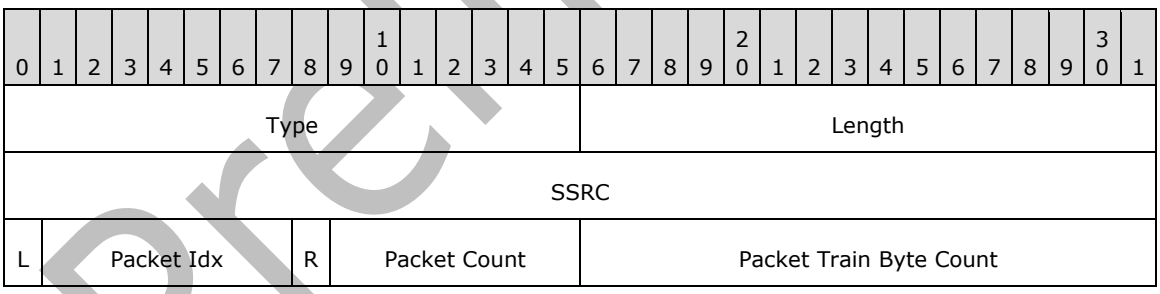
Length (2 bytes): The extension length in bytes, including this header. Set to 0x000C (12).

Reserved (4 bytes): Reserved for future extensions and SHOULD be set to zero (0). This 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**.

2.2.11.9 RTCP Profile Specific Extension for Packet Train Packet

The format of the **RTCP** profile specific extension for packet train packet is as follows: <31>



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.

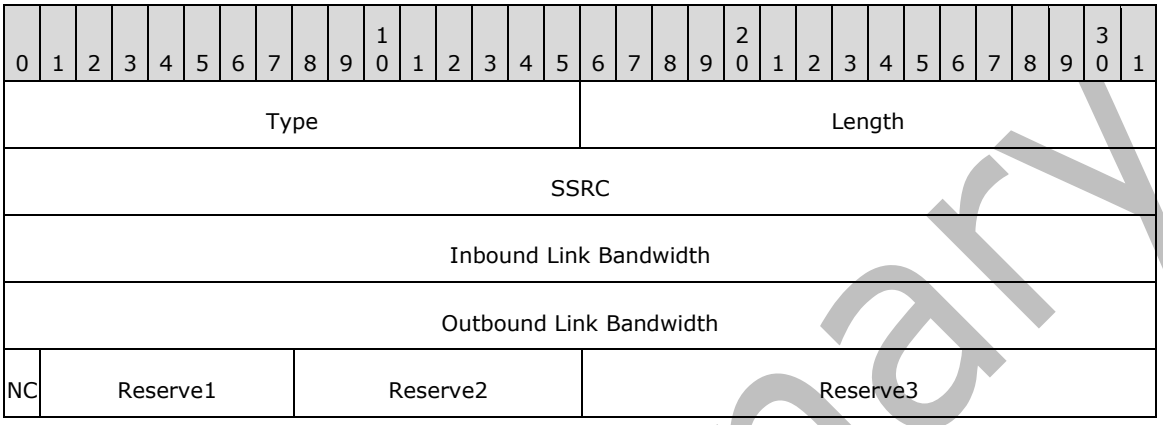
R (1 bit): Reserved. The sender SHOULD set it to 0. The receiver MUST ignore it.

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.

2.2.11.10 RTCP Profile Specific Extension for Peer Info Exchange

The format of the **RTCP** profile specific extension for peer info exchange is as follows: <32>



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 exchange is sent.

Inbound Link Bandwidth (4 bytes): The maximum inbound bandwidth supported by the host.

Outbound Link Bandwidth (4 bytes): The maximum outbound bandwidth supported by the host.

NC (1 bit): No cache flag. If NC is one, it indicates the inbound and outbound link bandwidth carried in the profile extension MUST NOT be cached and used beyond this session.

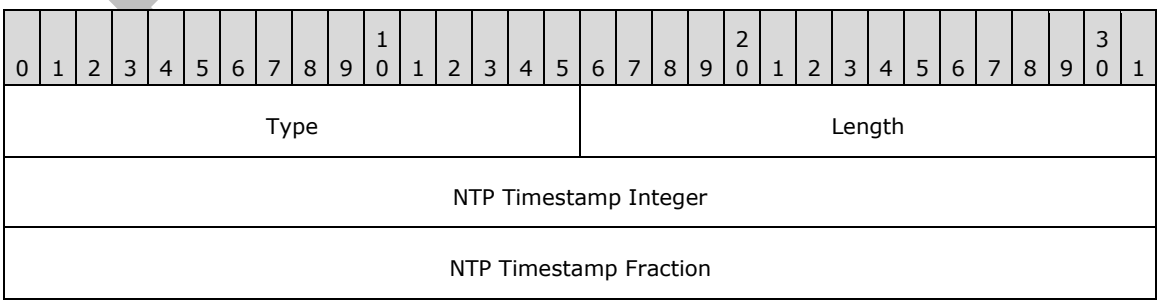
Reserve1 (7 bits): Reserved field. The sender SHOULD set it to zero. The receiver MUST ignore it.

Reserve2 (1 byte): Reserved field. The sender SHOULD set it to zero. The receiver MUST ignore it.

Reserve3 (2 bytes): Reserved field. The sender SHOULD set it to zero. The receiver MUST ignore it.

2.2.11.11 RTCP Profile Specific Extension for Network Congestion Notification

The format of the **RTCP** profile-specific extension for Network Congestion Notification is as follows<33>:



Congestion Info	Reserved

Type (2 bytes): The extension type. Set to 0x000D (13).

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

NTP Timestamp Integer (4 bytes): The NTP timestamp integer part. The full resolution NTP timestamp is a 64-bit unsigned fix-pointer number. This field has the integer part of the NTP timestamp.

NTP Timestamp Fraction (4 bytes): The NTP timestamp fraction part.

Congestion Info (1 byte): The bitmask of congestion state,

Bit 0: Uncongested according to relative one way delay;

Bit 1: Congested according to relative one way delay;

Bit 2: Uncongested according to packet loss rate;

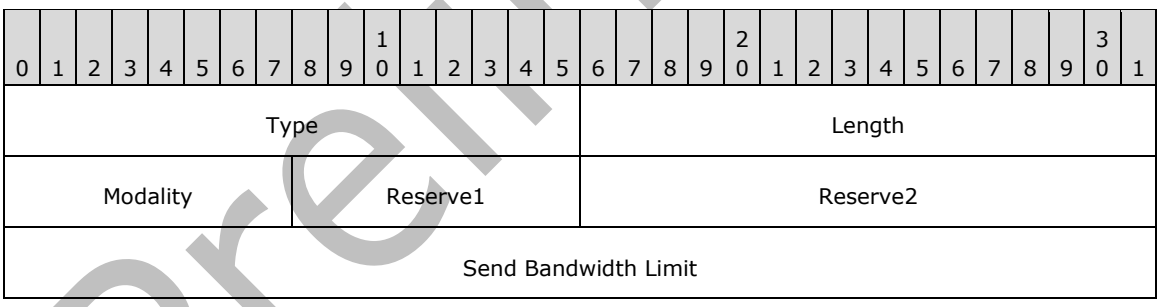
Bit 3: Congested according to packet loss rate;

Bit 4-7: Reserved, The sender SHOULD set it to 0. The receiver MUST ignore it.

Reserved (3 bytes): Reserved. The sender SHOULD set it to 0. The receiver MUST ignore it.

2.2.11.12 RTCP Profile Specific Extension for Modality Send Bandwidth Limit

The format of the **RTCP** profile-specific extension for Modality Send Bandwidth Limit is as follows<34>:



Type (2 bytes): The extension type. Set to 0x000E (14).

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

Modality (1 byte): Modality type. Currently supported modality type:

- 2: Video.

Reserve1 (1 byte): Reserved field. The sender SHOULD set it to zero. The receiver MUST ignore it.

Reserve2 (2 bytes): Reserved field. The sender SHOULD set it to zero. The receiver MUST ignore it.

Send Bandwidth Limit (4 bytes): The maximum outbound bandwidth in bits per second available for the specified modality type.

2.2.12 RTCP Feedback Message

The syntax of the **RTCP** feedback message defined in this protocol follows the syntax specified in [RFC4585]<35>. The RTCP feedback message is configured by the method specified in [MS-SDPEXT] section 3.1.5.30.

Details of the RTCP feedback message common fields, including PT, FMT, and so on, are specified in [RFC4585] section 6.

This protocol defines one payload-specific feedback message:

- Picture Loss Indication (PLI)

This protocol defines two application layer feedback messages:

- Video Source Request (VSR)
- Dominant Speaker History Notification (DSH)

This protocol MUST send out RTCP feedback messages as reduced-size RTCP messages, in the format specified in [RFC5506] section 4.1. This protocol extends [RFC3550] in the timer behavior, as specified in sections 3.2.2 and 3.2.6.

2.2.12.1 Picture Loss Indication (PLI)

The Picture Loss Indication (PLI) feedback message is the only payload-specific message this protocol supports. It is identified by PT=PSFB and FMT=1, specified in [RFC4585] section 6.3.1.

A PLI feedback message MAY be used to request sync frames or IDR on one or more H264 **streams** in a simulcast stream.

This protocol supports two types of PLI messages: standard PLI and extended PLI. The type of PLI to use is configured by the method specified in [MS-SDPEXT] section 3.1.5.30.

The standard PLI message format is specified in [RFC4585] section 6.3.1.

The extended PLI defined in this protocol differs from the standard PLI that it contains one Feedback Control Information (FCI) field. All other fields in the common packet format are same as defined in [RFC4585] section 6.1.

The FCI field is defined as follows:

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Request Id																Reserved															
SFR0										SFR1										SFR2										SFR3									
SFR4										SFR5										SFR6										SFR7									

Request Id (2 bytes): A 16-bit unsigned number that uniquely identifies the PLI. A PLI can be sent multiple times to improve redundancy. All the retransmitted PLIs MUST carry the same Request Id. A different PLI MUST carry a different Request Id.

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

SFR0 (1 byte): Sync frame request byte #0. Each bit in a SFR byte corresponds to one Priority Id which identifies one H264 stream. The bit value of 1 indicates a sync frame is requested on the H264 stream identified by the corresponding Priority Id. The bit value of 0 indicates a sync frame is not requested for the H264 stream identified by the corresponding Priority Id. The more significant bit corresponds to a higher Priority Id value. The most significant bit in SFR0 corresponds to Priority Id 7. The least significant bit in SFR0 corresponds to Priority Id 0.

SFR1 (1 byte): Sync frame request byte #1. This SFR byte defines the sync frame request for Priority ID 8~15.

SFR2 (1 byte): Sync frame request byte #2. This SFR byte defines the sync frame request for Priority ID 16~23.

SFR3 (1 byte): Sync frame request byte #3. This SFR byte defines the sync frame request for Priority ID 24~31.

SFR4 (1 byte): Sync frame request byte #4. This SFR byte defines the sync frame request for Priority ID 32~39.

SFR5 (1 byte): Sync frame request byte #5. This SFR byte defines the sync frame request for Priority ID 40~47.

SFR6 (1 byte): Sync frame request byte #6. This SFR byte defines the sync frame request for Priority ID 48~55.

SFR7 (1 byte): Sync frame request byte #7. This SFR byte defines the sync frame request for Priority ID 56~63.

2.2.12.2 Video Source Request (VSR)

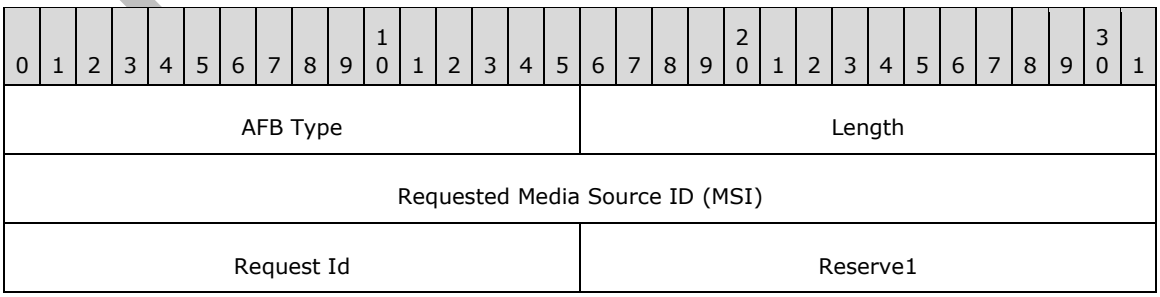
The Video Source Request (VSR) feedback message is an application-layer feedback message. It is identified by PT=PSFB, FMT=15, and Type=1.

Application layer feedback message details are specified in [RFC4585] section 6.4.

The VSR message MAY be used to notify the sender that the receiver wants to watch a specific video source with the video parameters in the message. It contains one Feedback Control Information (FCI) field in addition to the application feedback message common field.

The VSR FCI field contains one VSR header and up to 20 VSR entries. The VSR feedback message can be used by the **mixer** to forward the VSR it receives to the real video source. In this scenario, the VSR sent by the mixer is the aggregated result of multiple VSRs the mixer receives. When these VSRs received by the mixer have different video parameters, the aggregated VSR MAY have multiple VSR entries. The algorithm used by the mixer is implementation-defined and not prescribed by this specification.

The VSR header is defined as follows:



Version	A	Reserve2	B	C
Reserve3				

AFB Type (2 bytes): The application layer feedback message type. Set to 0x1.

Length (2 bytes): The FCI size, including the AFB Type and Length field. A VSR MUST have less than or equal to 20 VSR entries.

Requested Media Source ID (MSI) (4 bytes): The **MSI** of the video source to be watched. The following constant values have special meanings:

0xFFFFFFFF: SOURCE_NONE, meaning the receiver is not requesting a video source.

0xFFFFFFFE: SOURCE_ANY, meaning the sender selects the video source.

Request Id (2 bytes): A 16-bit unsigned number that uniquely identifies the VSR. A VSR can be sent multiple times to the sender. All the retransmitted VSRs MUST carry the same **Request Id**. A different VSR MUST carry a different Request Id.

Reserve1 (2 bytes): Reserved field. The sender SHOULD set it to zero. The receiver MUST ignore it

Version (1 byte): The VSR version number. The sender SHOULD set it to zero. The receiver MUST ignore it

A - K (1 bit): The key frame request flag. Set to 1 when a video key frame is requested.

Reserved2 (7 bits): Reserved field. The sender SHOULD set it to zero. The receiver MUST ignore it

B - Number of Entries (1 byte): The number of VSR entries in this VSR message. It MUST be less than or equal to 20. A VSR message MAY have zero VSR entries when the **Requested MSI** field value is 0xFFFFFFFF.

C - Entry Length (1 byte): The size of each VSR entry. Set to 0x44.

Reserve3 (4 bytes): Reserved field. The sender SHOULD set it to zero. The receiver MUST ignore it.

The VSR entries follow the VSR header. Each VSR entry represents a set of video parameters that one or multiple receivers request. The VSR entry is defined 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
Payload Type								UCConfig Mode								Flags								Aspect Ratio / Preferred Resolution Bit Mask									
Maximum Width																Maximum Height															
Minimum bit rate																															
Reserve / Macroblock Processing Rate Bitmask																															
Bit rate per level																															
Bit rate histogram (20 bytes)																															

…	
…	
…	
…	
Frame rate bit mask	
Number of MUST instances	Number of MAY instances
Quality Report Histogram (16 bytes)	
…	
…	
…	
Maximum number of pixels	

Payload type (1 byte): The encoding type of the requested video. Encoding types are specified in section 2.2.1. Only RT video and H264 encoding types are allowed.

UCConfig Mode (1 byte): The maximum UCConfig Mode the receiver supports. The UCConfig Mode is defined in [MSFT-H264UCConfig].

The value is defined as follows:

0: MUST NOT be used.

1: UCConfig Mode 1.

2 or larger: MUST NOT be used.

Flags (1 byte): The video flags.

Bit 0 (least significant bit): 1 means the receiver supports CGS rewrite; for H264 encoding only.

Bit 1: 1 means the receiver only supports either constrained baseline profile or scalable constrained baseline profile; for H264 encoding only.

Bit 2: 1 means the receiver doesn't support SP frames; for RT Video encoding only.

Bit 3: 1 means the receiver does not support seamless resolution change; for H264 encoding only.

Bit 4-7: Reserved. The sender SHOULD set it to 0. The receiver MUST ignore it

Aspect Ratio Bit Mask/Preferred Resolution Bit Mask (1 byte):

For video modality

This byte represents the bit mask of aspect ratio, defined as follows:

Bit 0: 4 by 3.

Bit 1: 16 by 9.

Bit 2: 1 by 1.

Bit 3: 3 by 4.

Bit 4: 9 by 16.

Bit 5: 20 by 3.

Bit 6-7: not used. They MUST be ignored.

For application-sharing modality (a=label:applicationsharing-video)
This byte represents bit mask of the shorter dimension of the preferred resolution, defined as follows in pixels:

Bit 0: 270

Bit 1: 540

Bit 2: 810

Bit 3: 1080

Bit 4: 1350

Bit 5: 1620

Bit 6: 1890

Bit 7: 2160

Maximum Width (2 bytes): The maximum width of the video resolution in pixels.

Maximum Height (2 bytes): The maximum height of the video resolution in pixels.

Minimum bit rate (4 bytes): The minimum video bit rate. The unit is in bits per second.

Reserve / Macroblock Processing Rate Bit Mask (4 bytes):

For video modality
These bytes are reserved. The sender SHOULD set it to 0. The receiver MUST ignore it.
For application-sharing modality (a=label:applicationsharing-video)
This byte represents the bit mask of the maximum macroblock processing rate per second requested, defined as follows:

Bit 0: 2812

Bit 1: 4218

Bit 2: 8437

Bit 3: 16875

Bit 4: 33750

Bit 5: 67500

Bit 6: 108000

Bit 7: 135000

Bit 8: 198000

Bit 9: 244800

Bit 10: 270000

Bit 11-31: not used. The sender SHOULD set it to 0. The receiver MUST ignore it

Bit rate per level (4 bytes): The bit rate difference for one level in the bit rate histogram.

Bit rate Histogram (20 bytes): The requested bit rate histogram of the receivers represented in this VSR entry. It is an array of 10 16-bit values. The *i*-th element in the array counts the number of receivers requesting a bit rate ranging from [Minimum bit rate + (*i-1)* * Bit rate per level] to [Minimum bit rate + *i* * Bit rate per level].

Frame rate bit mask (4 bytes): The bit mask of frame rate requested, defined as follows:

Bit 0: 7.5 frames per second

Bit 1: 12.5 frames per second

Bit 2: 15 frames per second

Bit 3: 25 frames per second

Bit 4: 30 frames per second

Bit 5: 50 frames per second

Bit 6: 60 frames per second

For video modality

Bit 7-31: not used. The sender SHOULD set it to 0. The receiver MUST ignore it

For application-sharing modality (a=label:applicationsharing-video)

Bit 7: 1.875 frames per second

Bit 8: 3.75 frames per second

Bit 9-31: not used. The sender SHOULD set it to 0. The receiver MUST ignore it

Number of MUST instances (2 bytes): The number of receivers that only accept video payload type defined in this VSR.

Number of MAY instances (2 bytes): The number of receivers that can accept other video payload type.

Quality Report Histogram (16 bytes): The quality report histogram. This is an array of 8 16-bit values. The *i*-th element in the array counts the number of receivers reporting a quality level of value *i*, where level 1 is the best quality. Quality report is generated from the receiver, and the mixer aggregates the quality reports it receives from the receivers to the video source.

Maximum number of pixels (4 bytes): The maximum number of pixels the receivers can receive in one **video frame**.

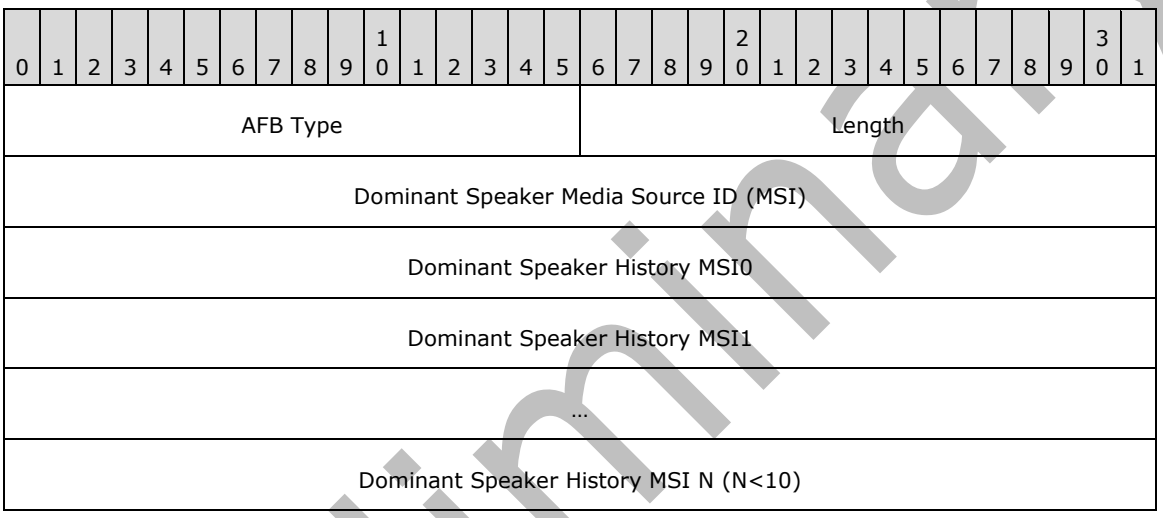
2.2.12.3 Dominant Speaker History Notification (DSH)

The **Dominant Speaker History Notification (DSH)** feedback message is an application layer feedback message. It is identified by PT=PSFB, FMT=15 and Type=3.

Application layer feedback message details are specified in [RFC4585] section 6.4.

The **DSH** message MAY be used to notify the **dominant speaker** history by the **mixer**. It contains one **Feedback Control Information (FCI)** field in addition to the application feedback message common field.

The **DSH FCI** field is defined as follows:



AFB Type (2 bytes): The application layer feedback message type. Set to 0x3.

Length (2 bytes): The **FCI** size, including the AFB Type and Length fields. **DSH** MUST have less than or equal to 10 **Dominant Speaker History** fields.

Dominant Speaker Media Source ID (MSI) (4 bytes): The **MSI** of the current dominant speaker's audio source. If there isn't a dominant speaker in the mixer at the moment of the notification, this field MUST be set to SOURCE_NONE (0xFFFFFFFF).

Dominant Speaker History MSI 0 to 9 (4 bytes each, variable length): The past dominant speakers' MSIs, sorted by the descending order of the most recent dominant speakers.

3 Protocol Details

3.1 RTP Details

This protocol MAY have multiple **RTP sessions** sharing the same transport as long as these RTP sessions have non-overlapping **SSRC** ranges. If the transport is shared, the SSRC range MUST be configured by the method specified in [MS-SDPEXT] section 3.1.5.31. When the SSRC range is set, the SSRC throttling is disabled and **RTP packets** with SSRC outside the specified range MUST be dropped.

This protocol MAY multiplex multiple related **streams** in one RTP session using different SSRCs.

This protocol differs from [RFC3550] in the **CSRC** values used in the CSRC list. If the sender is a **mixer**, this protocol uses the **MSI** of the contributing sources as CSRC values in the CSRC list<36>.

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.

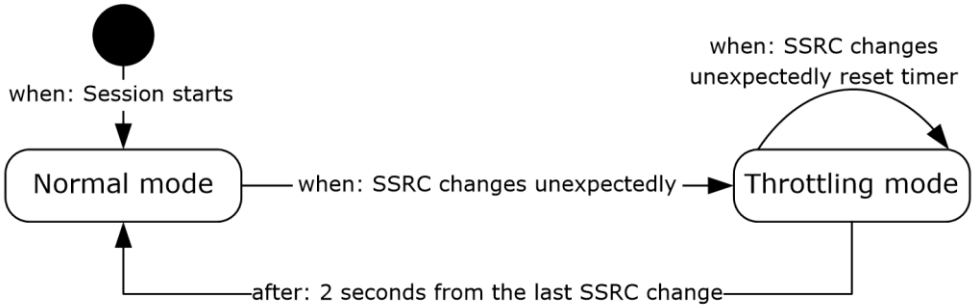


Figure 2: Synchronization source throttling

The sequence number throttling mechanism is analogous to the SSRC throttling mechanism, but is done using the values stored on the **RTP participant**, 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. Mixer 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 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**.
- **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**:

- **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 **DominantSpeaker** 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**. This timer SHOULD be set to 50 seconds. There MUST be one participant time-out timer per participant.

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** SHOULD NOT<37> 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] section 5.1, 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** time-out timer of the participant 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 <38>.

```
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:<39>

```
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** has no **dominant speaker**.

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

When there is more than one **stream** in the same RTP session, this protocol MUST only send RTCP packets using the lowest **SSRC** in the SSRC range. The packet and octet counts SHOULD be the sum of packet and octet counts of all streams in the RTP session.

The **RTCP** SDES PRIV extension for media quality, as specified in section 2.2.10.1, works as follows:<40>

- 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 2.2.11.1, 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.

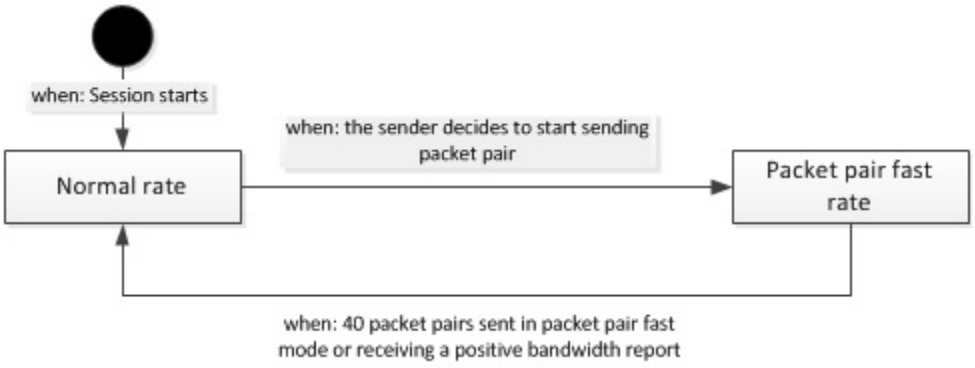


Figure 3: Behavior of packet pair bandwidth acceleration

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.

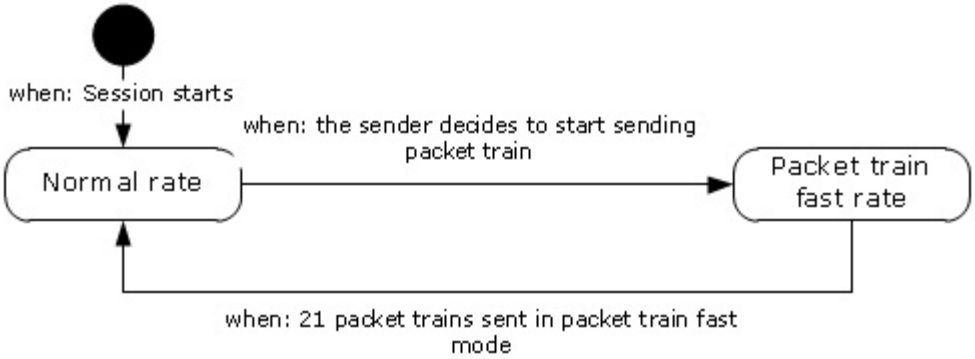


Figure 4: Behavior of packet train bandwidth acceleration

If RTCP packet pairs or packet trains are not sent as specified in this document, the receiver cannot send a bandwidth estimate back. Bandwidth estimates MUST 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.

A packet loss notification extension SHOULD be sent when video packet loss is detected, and MUST be received and parsed correctly, but the receiver MAY decide not to take any action in response to the notification.

A video source request extension SHOULD be sent when a receiver wants to watch a specific video source with the video parameters in the message. A video source request extension is configured by the method specified in [MS-SDPEXT] section 3.1.5.30.

When the video source request extension is not configured, a video preference extension, as specified in section 2.2.11.3, SHOULD<41> 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] section 3.1.5.25.

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.<42> 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<43>. 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.<44> 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<45>. 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 five times for a specific bandwidth to ensure that this profile extension is not lost on the wire.

The peer info exchange extension SHOULD be sent from the host to the remote host in every RTCP packet after the session starts 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 MUST be used to pad an RTCP packet pair packet or an RTCP packet train packet to a specific size.

Dominant Speaker History notification message SHOULD be sent through RTCP feedback message every time the **dominant speaker** changes in the mixer. When there isn't any dominant speaker, SOURCE_NONE (0xFFFFFFFF) SHOULD be sent as the dominant speaker **MSI**. When a new RTP session joins the mixer, a DSH notification MUST be sent to keep this new session's dominant speaker state up to date.<46>

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] section 6.

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: <47> This timer SHOULD be set to 20 seconds. It is started when an RTCP BYE is received. There MUST be one timer per **participant**.

Packet loss notification timer:<48> 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.

RTCP feedback message fast timer: This timer SHOULD be set to 190 milliseconds. It starts when a RTCP feedback message is sent out the first time. It fires 4 times for each message, then stops.

RTCP feedback message slow timer: This timer SHOULD be set to 3 seconds. It starts after the RTCP feedback message fast timer for a Video Source Request (VSR) message stops. It fires 5 times, then stops.

3.2.3 Initialization

This protocol has the following **RTCP**-related initialization requirements, in addition to those specified in [RFC3550] section 6:

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 zero (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** 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 SHOULD 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. The following conditions are tested:
    - 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

If the preceding conditions are met, 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 are 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**.

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.

RTCP feedback message fast timer expires: The RTCP feedback message is resent and the timer is reset. After resending the message 4 times, this timer stops. If the feedback message is a Source Request (SR) message, a RTCP feedback message slower timer SHOULD be started.

RTCP feedback message slow timer expires: The Source Request (SR) message is resent and the timer is reset. After resending the message 4 times, this timer stops.

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.

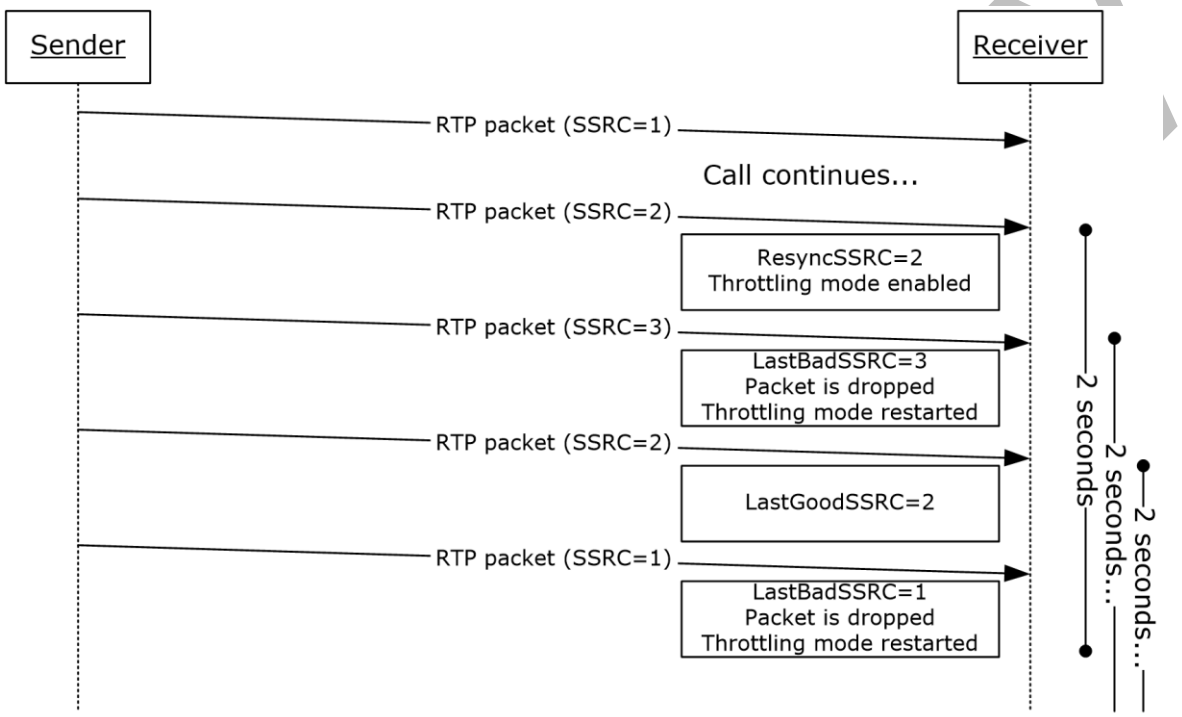


Figure 5: Synchronization source change throttling

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.

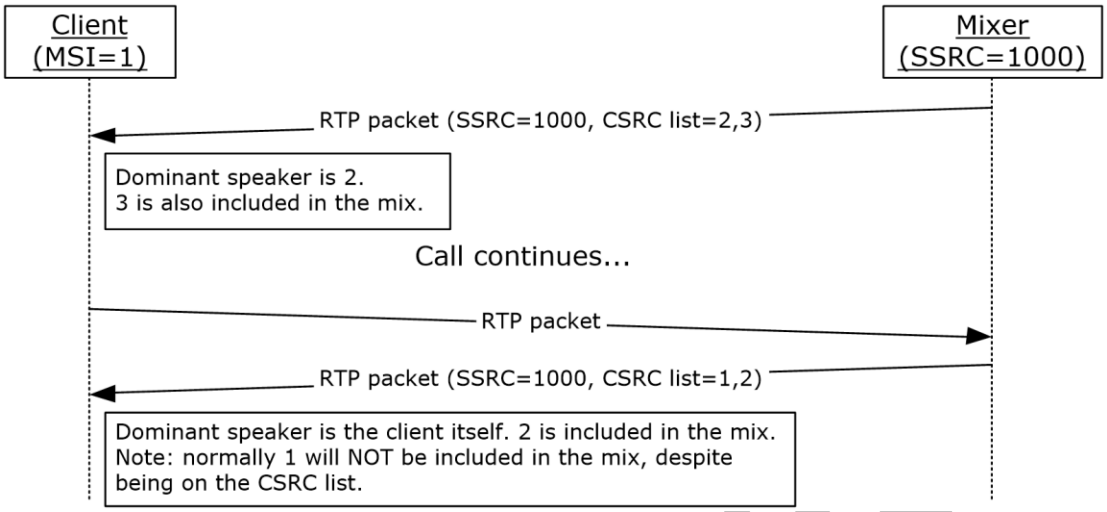


Figure 6: Message exchange for dominant speaker notifications

On the first message, the dominant speaker is notified to be the one with **MSI**=2. The protocol client can then use this information, perhaps in conjunction with the SDES data communicated through **RTCP** for MSI=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.

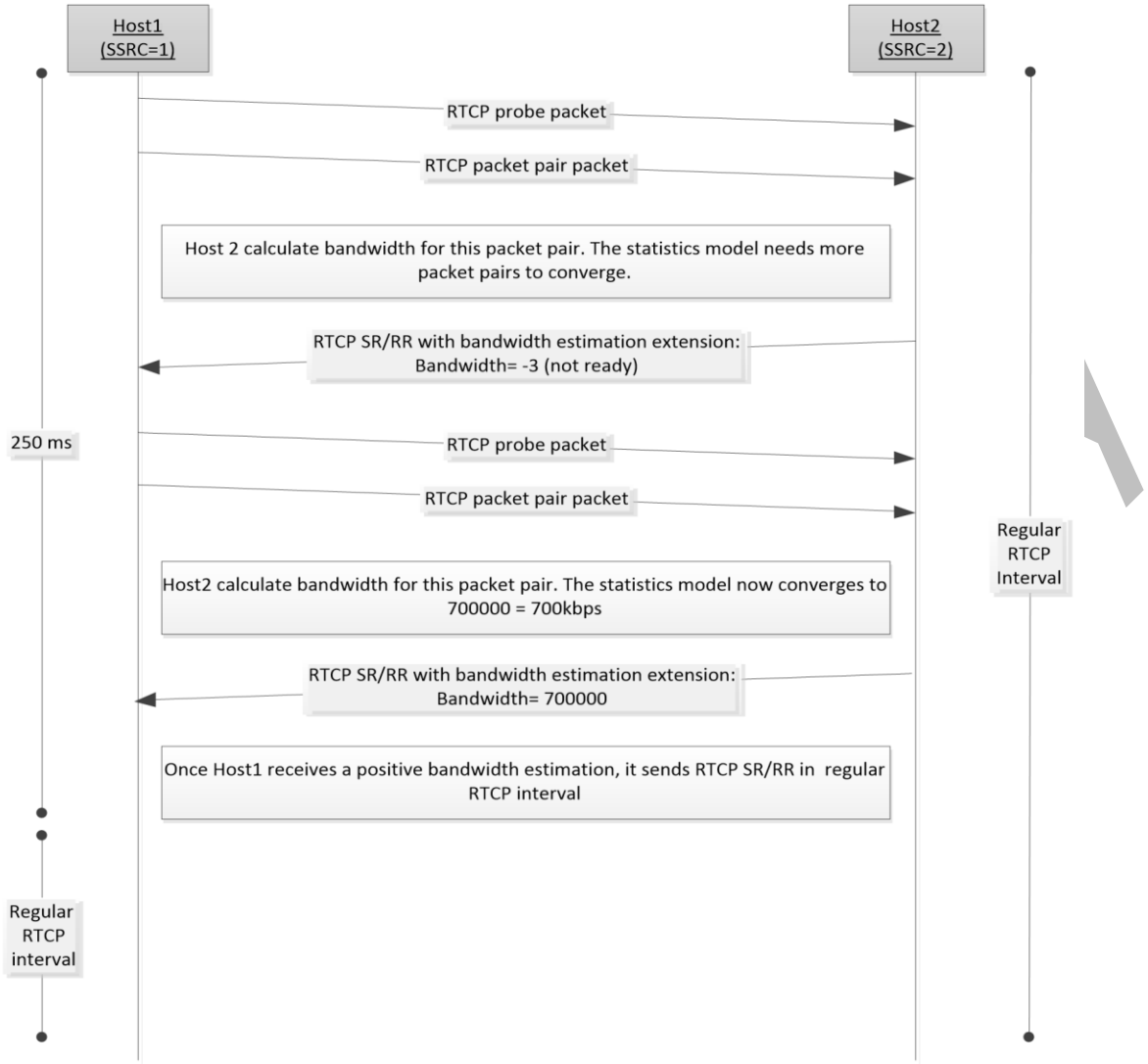


Figure 7: Message exchange for bandwidth estimation for packet pair

In the preceding figure, Host2 does not support packet train. On receipt of the first **RTCP packet** pair, Host2 calculates a bandwidth estimate. The calculation for the bandwidth estimate might need more packet pairs to converge to a valid bandwidth estimate. 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 700000. Host2 sends a bandwidth estimation report with 700000. Host1 switches to the regular **RTCP** interval after it receives the positive bandwidth estimation.

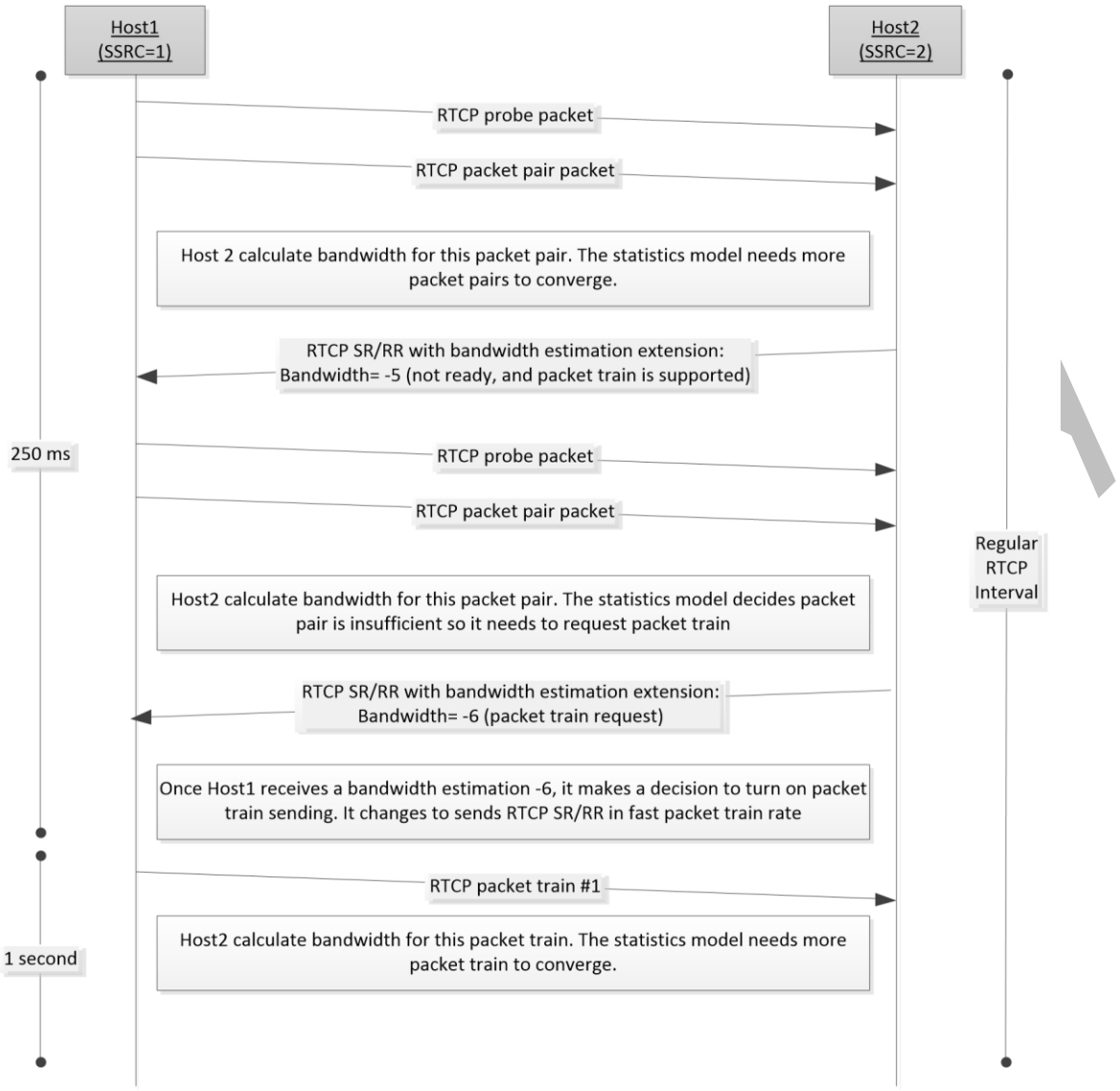


Figure 8: Message exchange for bandwidth estimation for packet train (Part 1)

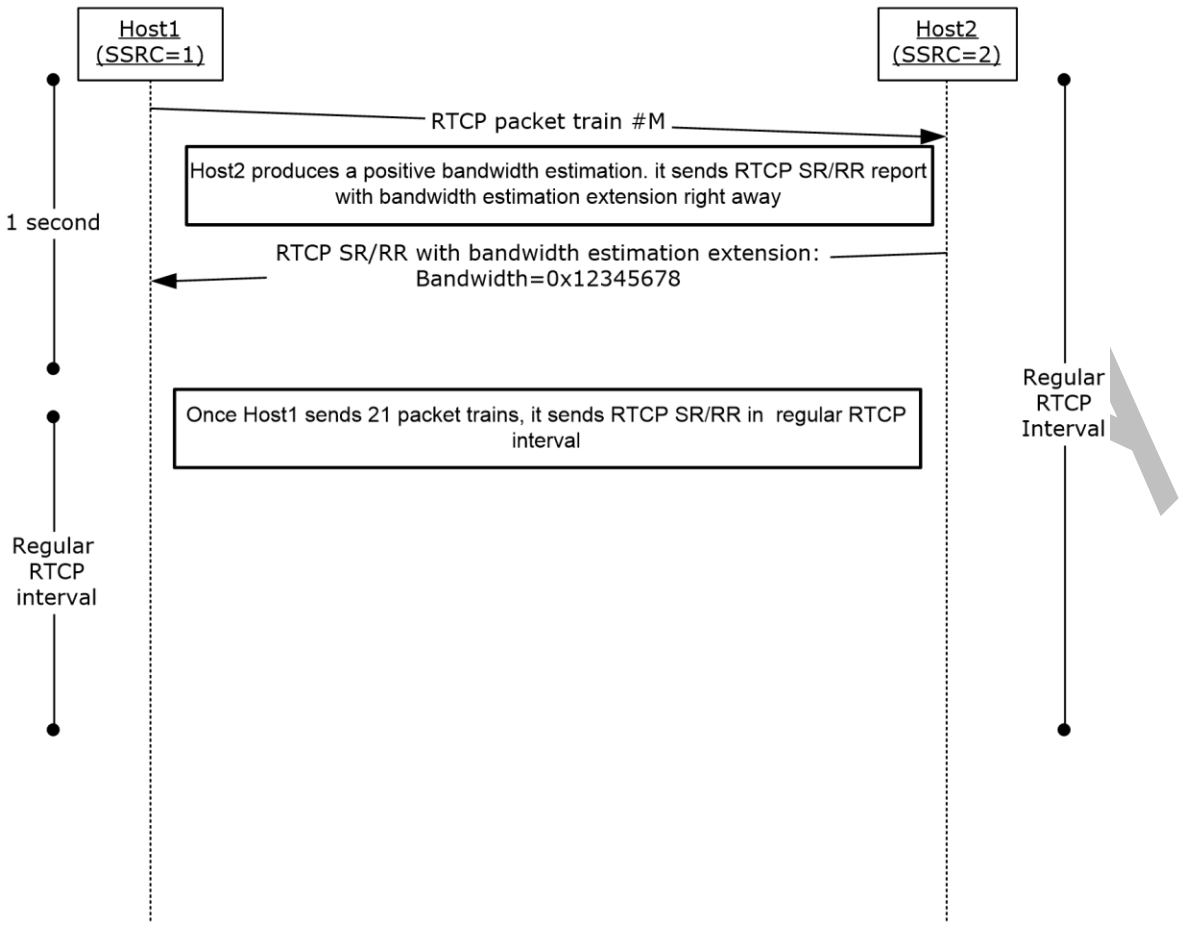


Figure 9: Message exchange for bandwidth estimation for packet train (Part 2)

On receipt of the first RTCP packet pair, Host2 calculates a bandwidth estimate. The calculation for the bandwidth estimate might need more packet pairs to converge to a valid bandwidth estimate. 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 requests the packet train, and sends 0xFFFFFFFA (-6) in the bandwidth report to indicate that Host1 can send the 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

If a **RTP packet** is lost a notification packet is generated as follows:

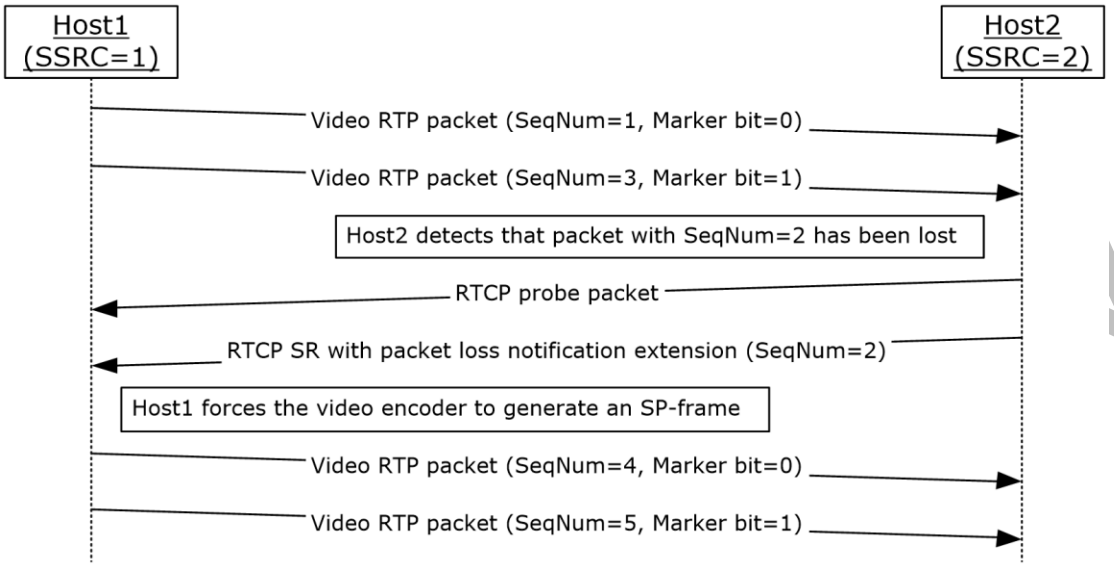


Figure 10: Message exchange for packet loss notifications

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

If a host decides that it would like to change the video resolution it is receiving, it does the following:

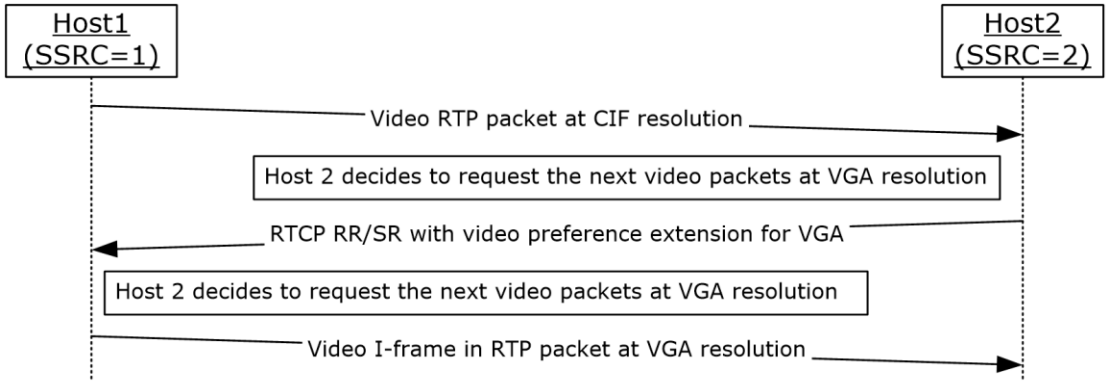


Figure 11: Message exchange for video preference

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

On receipt of the policy server bandwidth via the **TURN** protocol, the host informs the other host of the bandwidth information.

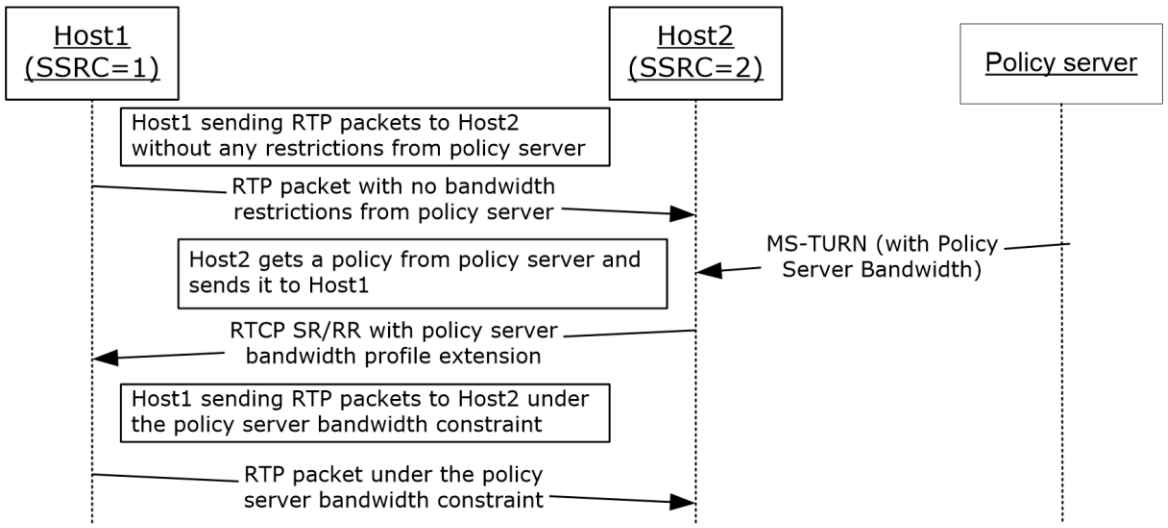


Figure 12: Message Exchange for Policy server bandwidth extension

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

On receipt of the **TURN server** bandwidth via the **TURN** protocol, the host informs the other host of the bandwidth information.

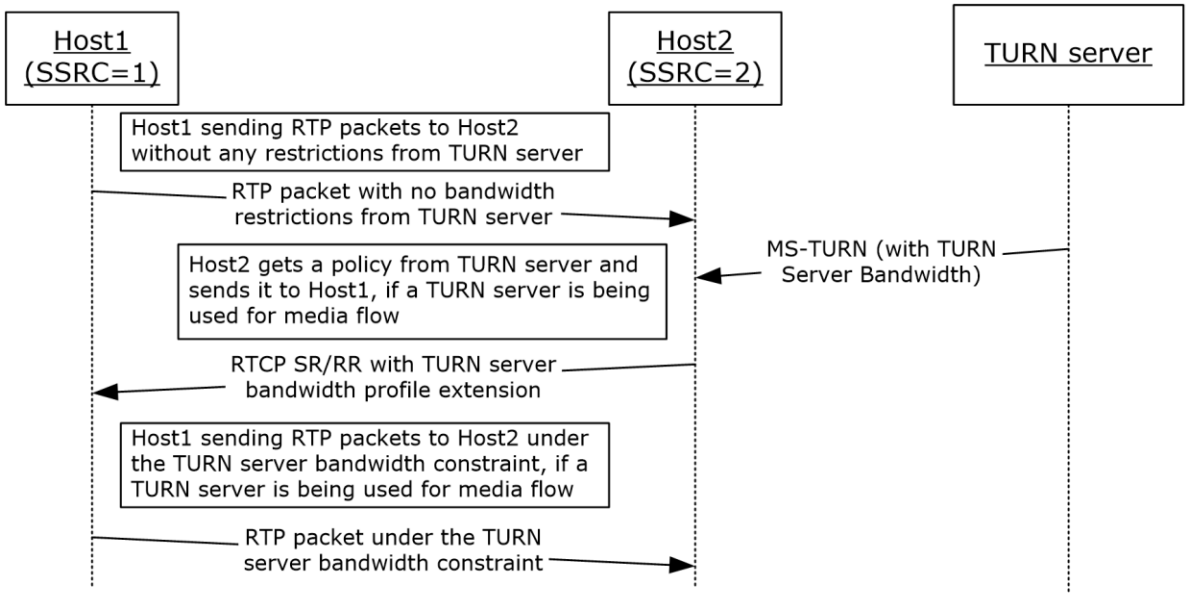


Figure 13: Message exchange for TURN Server bandwidth extension

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

A host adds metrics to the regular **RTCP** RR to inform the other host about the **audio healer** in the following way:

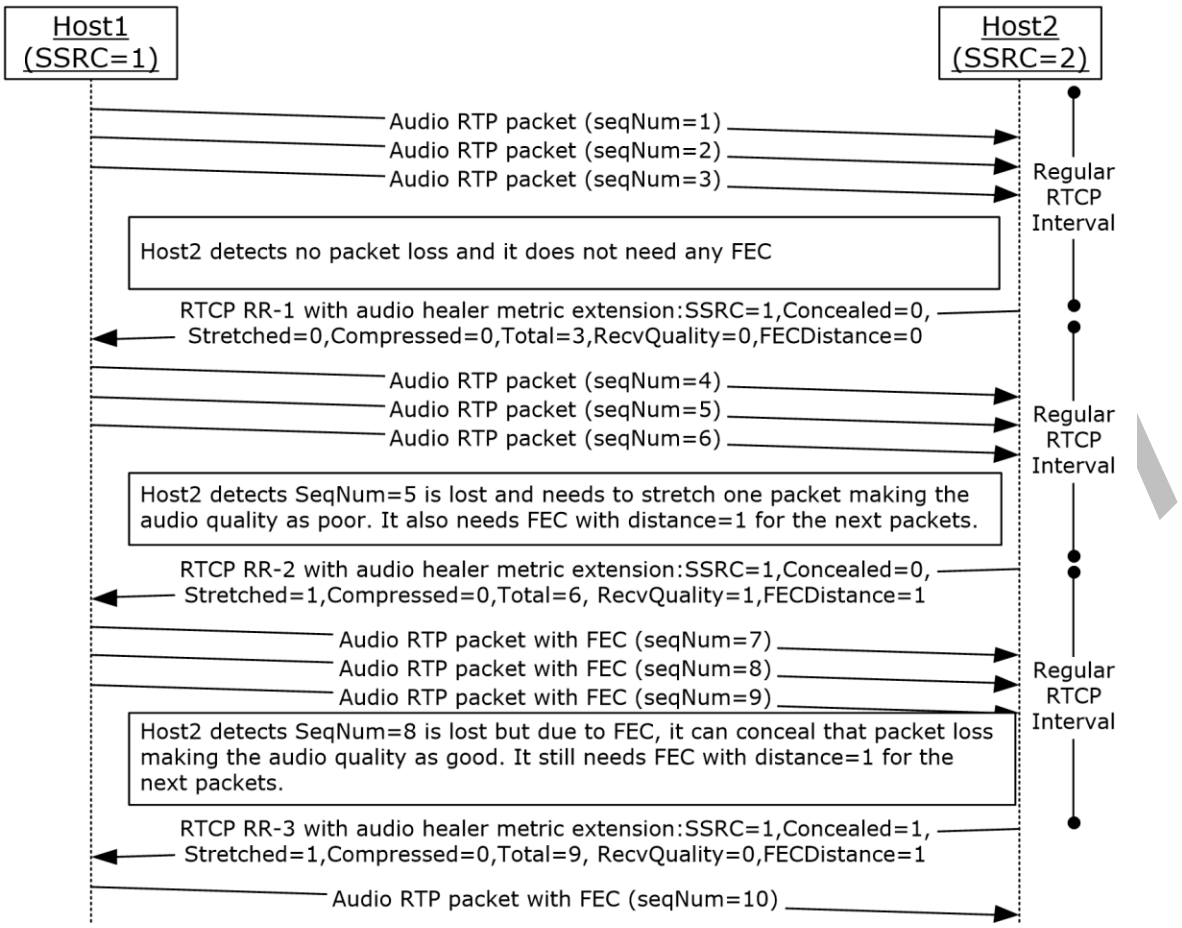


Figure 14: Message exchange for audio healer metrics extension

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

When a host is sending application sharing payload to a **mixer** the mixer can request that less payload is sent to the mixer. The following describes how the mixer does this.

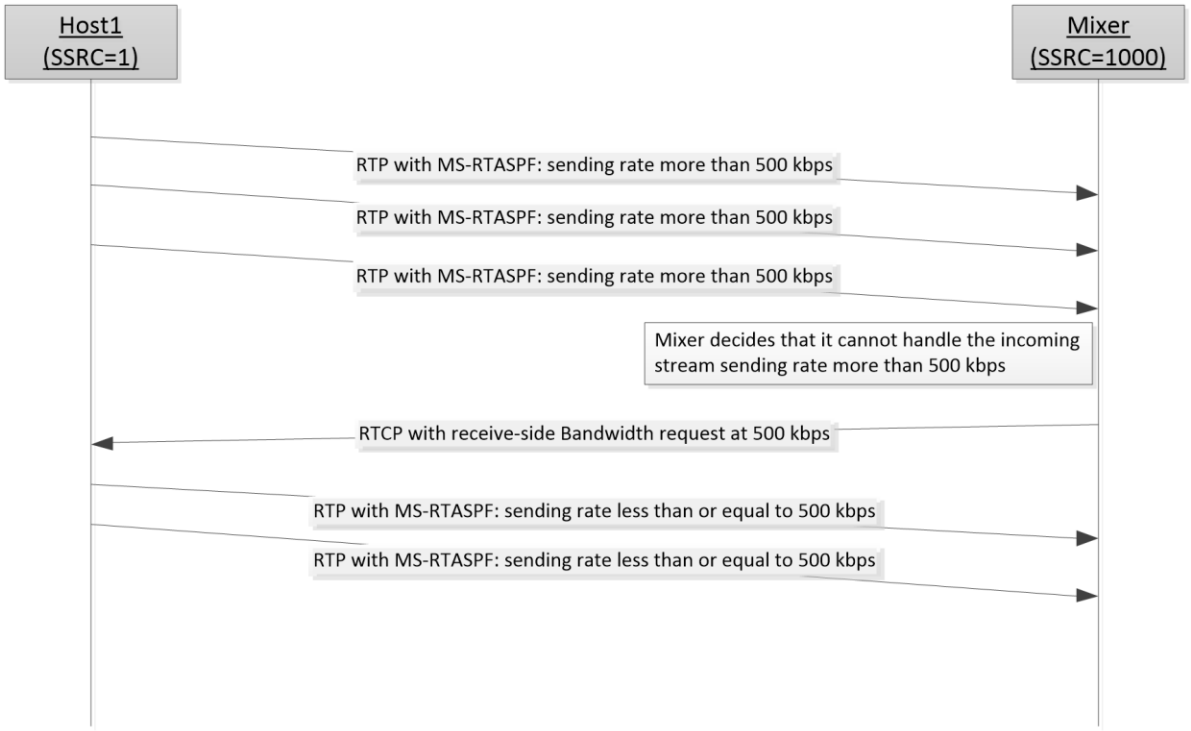


Figure 15: Message exchange for receiver-side bandwidth limit

Host 1 is sending an application sharing payload (as described in [MS-RTASPF] section 3.2.5) to the mixer at a bitrate more than 500 kbps. The mixer determines 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

A host and a **mixer** can send information to another host to inform it of the media quality that is being sent to it. Some of the information is sent from the host originating the media, and additional information can be added by the mixer.

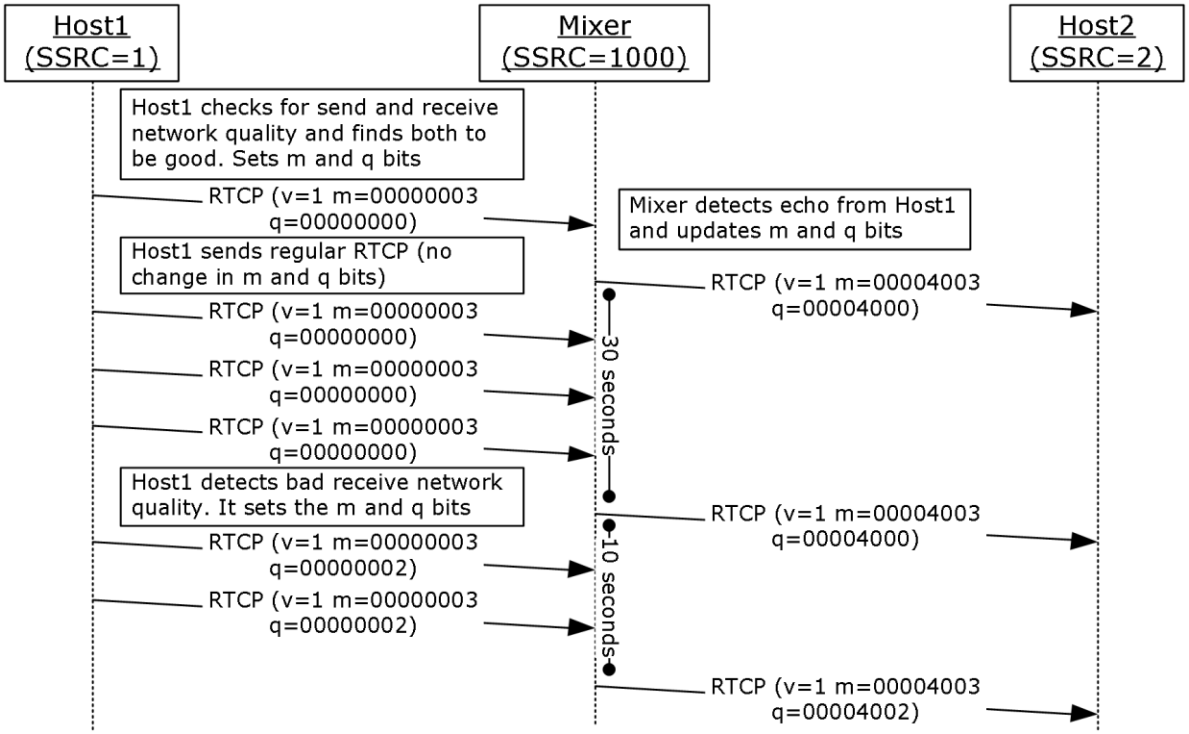


Figure 16: SDES PRIV extension for media quality

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.

4.11 Network Congestion Notification Extension

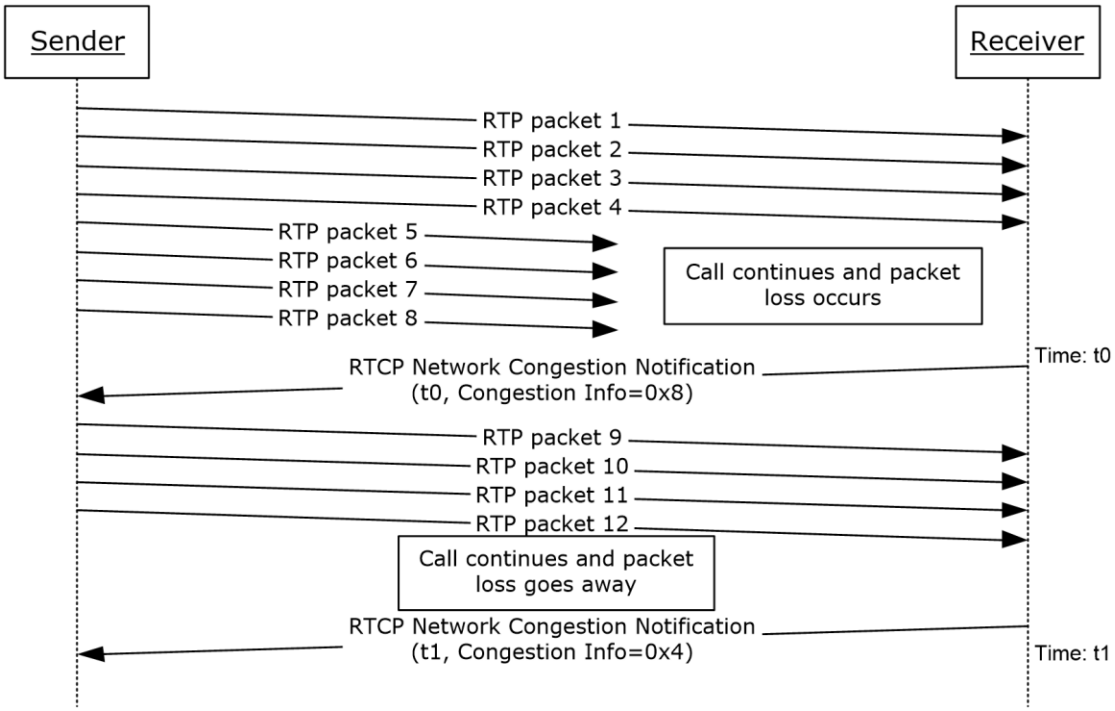


Figure 17: Network Congestion Notification extension

Receiver monitors the network **jitter** and loss in the received **RTP packets**. Receiver notifies the sender of the network congestion condition changes.

4.12 Picture Loss Indication Extension

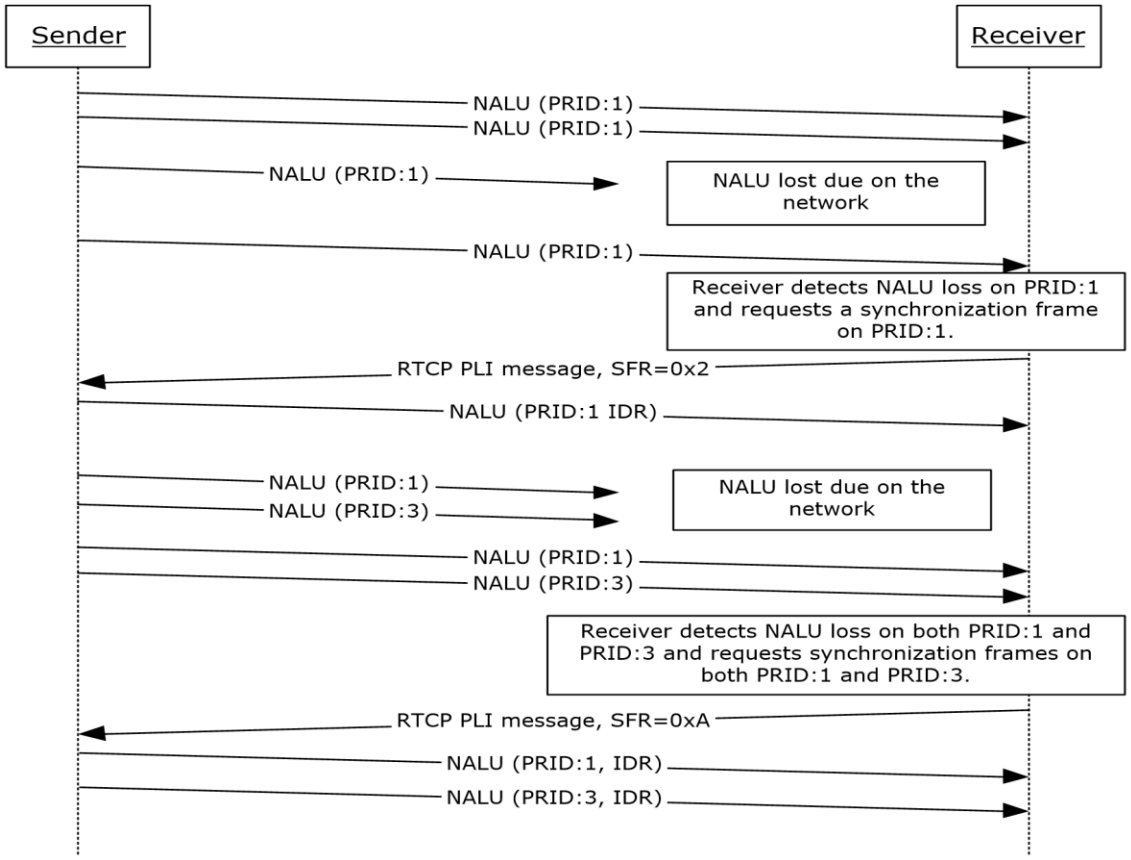


Figure 18: Picture Loss Indication Extension

The sender sends H.264 NAL units to the receiver, and some of them get lost on the network.

The receiver detects the loss and decides to ask for synchronization frames, so it sends out PLI messages.

4.13 Video Source Request Extension

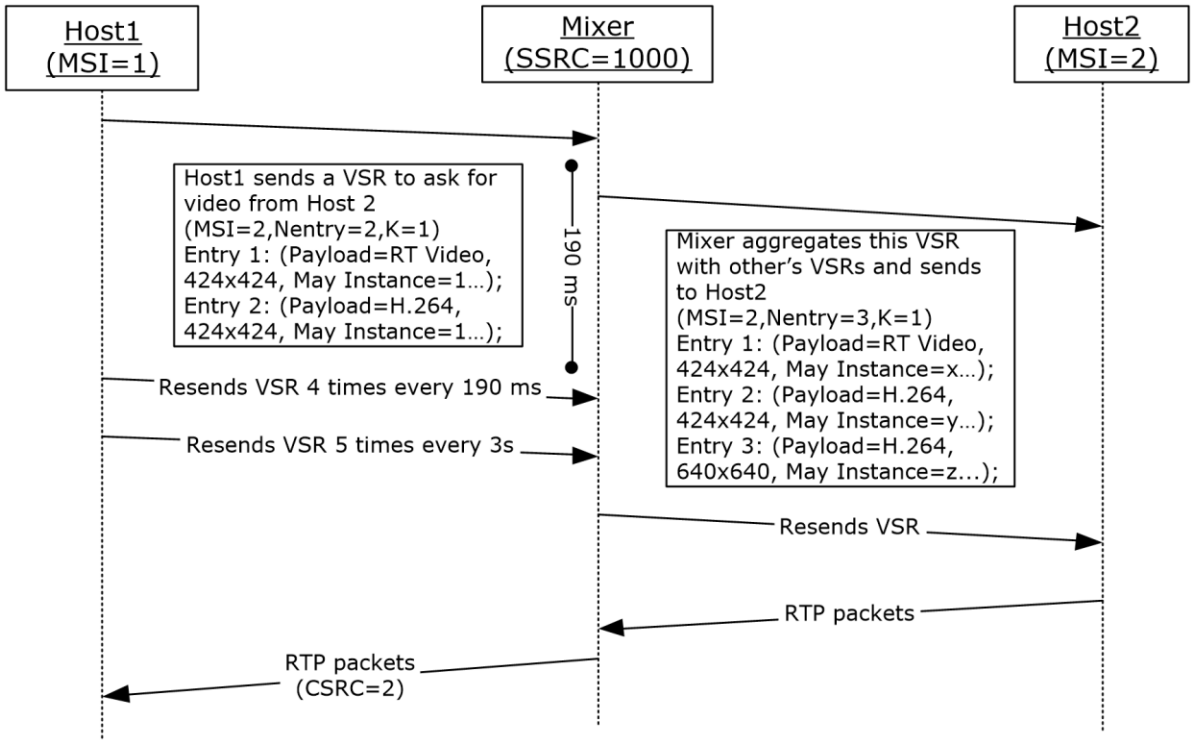


Figure 19: Video Source Request Extension

Host1 sends a VSR to **mixer** to ask for video from Host2. The VSR contains Host2's **MSI** and the video parameters. Mixer receives this VSR and aggregates it with VSRs from other hosts asking for Host2's video. All VSRs are resent 4 times every 190ms, then resent 5 more times every 3s. Host2 receives the aggregated VSR and sends back the video packets. Mixer forwards the video packets to Host1 and others.

4.14 Dominant Speaker History Notification extension

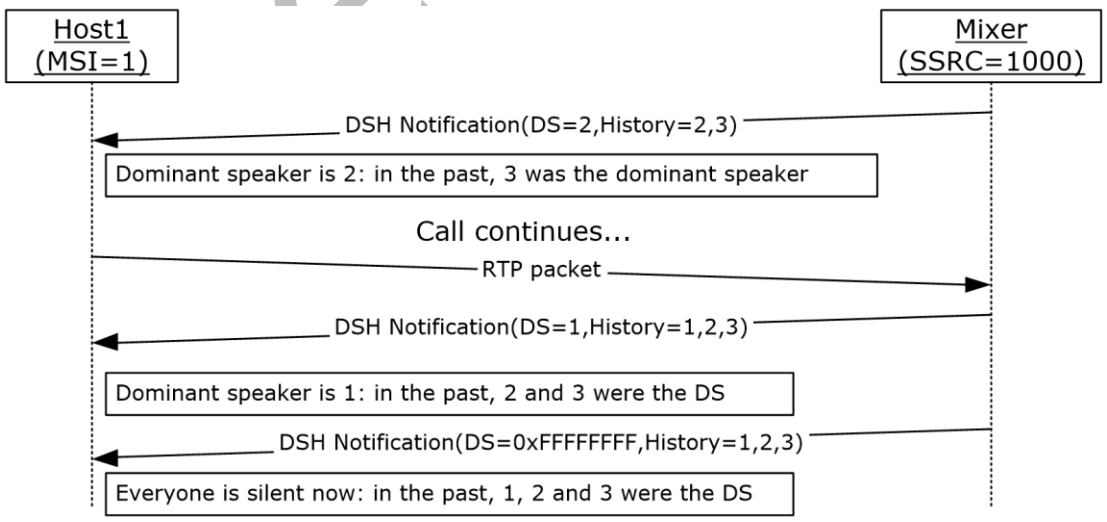


Figure 20: Dominant Speaker History Notification

On receipt of the first **dominant speaker** notification, Host1 knows the current dominant speaker is Host2 and that Host3 was the dominant speaker before. Host1 starts to talk and becomes the dominant speaker. When Host1 stops talking, there is no dominant speaker in the **mixer**.

4.15 Modality Send Bandwidth Limit

The sender sends an **RTCP packet** with modality send bandwidth limit profile extension to notify the receiver of the outbound bandwidth available for video on the sender.

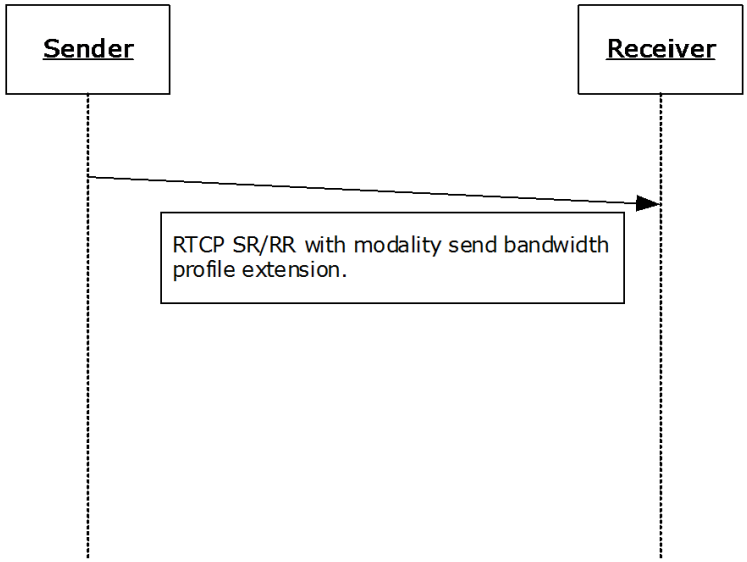


Figure 21: Message Exchange for Modality Send Bandwidth Limit Extension

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 updates to those products.

- 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
- Microsoft Lync Server 2013
- Microsoft Lync Client 2013/Skype for Business
- Microsoft Skype for Business 2016
- Microsoft Skype for Business Server 2015
- Windows 10 v1511 operating system
- Windows Server 2016 operating system
- Windows Server operating system

- Microsoft Skype for Business 2019 Preview

Exceptions, if any, are noted in this section. If an update version, service pack or Knowledge Base (KB) number appears with a product name, the behavior changed in that update. The new behavior also applies to subsequent updates 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, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: **IPv6** 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: Silk and OPUS codecs are not supported for Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010, Lync Server 2013, Lync Client 2013/Skype for Business, Skype for Business Server 2015 and Skype for Business 2016.

<11> Section 2.2.1: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: This behavior is not supported.

<12> Section 2.2.1: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<13> Section 2.2.1: This behavior is not supported in Lync Server 2013, Lync Client 2013/Skype for Business.

<14> Section 2.2.1: This behavior is only supported in Lync Server 2013, Lync Client 2013/Skype for Business.

<15> Section 2.2.1: This behavior is only supported in Lync Server 2013, Lync Client 2013/Skype for Business.

<16> Section 2.2.1: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<17> Section 2.2.1: This behavior is only supported in Lync Server 2013, Lync Client 2013/Skype for Business. All other releases use **SSRCs** of the contributing source as **CSRC** list as described in [RFC3550].

<18> Section 2.2.1.2: This behavior is supported for Microsoft Edge only.

<19> Section 2.2.1.2: This behavior is supported for Microsoft Edge only.

<20> 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.

<21> Section 2.2.11: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<22> Section 2.2.11.1: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<23> Section 2.2.11.1: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<24> Section 2.2.11.1: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<25> Section 2.2.11.3: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<26> Section 2.2.11.4: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<27> Section 2.2.11.5: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<28> Section 2.2.11.6: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<29> Section 2.2.11.7: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<30> Section 2.2.11.8: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<31> Section 2.2.11.9: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<32> Section 2.2.11.10: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<33> Section 2.2.11.11: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: This behavior is not supported.

<34> Section 2.2.11.12: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync Server 2010, Lync 2010: This behavior is not supported.

<35> Section 2.2.12: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync 2010, Lync Server 2010: This behavior is not supported.

<36> Section 3.1: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync 2010, Lync Server 2010: this behavior is not supported. SSRC of the contributing source is used instead.

<37> 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** is required to trigger the throttling mode as if it were an SSRC change.

<38> Section 3.1.5: Office Communications Server 2007, Office Communicator 2007: This behavior is not supported.

<39> 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.

<40> Section 3.2: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<41> Section 3.2: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<42> Section 3.2: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<43> Section 3.2: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<44> Section 3.2: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<45> Section 3.2: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: This behavior is not supported.

<46> Section 3.2: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2, Lync 2010, Lync Server 2010: DSH message is not supported.

<47> Section 3.2.2: Office Communications Server 2007, Office Communicator 2007, Office Communications Server 2007 R2, Office Communicator 2007 R2: RTCP Bye timer:This timer is set to 2 seconds. It is started when an **RTCP** BYE is received. There MUST be one timer per **participant**.

<48> 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

This section identifies changes that were made to this document since the last release. Changes are classified as Major, Minor, or None.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements.
- A document revision that captures changes to protocol functionality.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **None** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the relevant technical content is identical to the last released version.

The changes made to this document are listed in the following table. For more information, please contact dochelp@microsoft.com.

Section	Description	Revision class
6 Appendix A: Product Behavior	Updated list of supported products.	major

8 Index