

AVT
Internet-Draft
Intended status: Standards Track
Expires: April 26, 2012

A. Begen
Cisco
C. Perkins
University of Glasgow
October 24, 2011

Duplicating RTP Streams
draft-begen-avtcore-rtp-duplication-00

Abstract

Packet loss is undesirable for real-time multimedia sessions, but it is not avoidable due to congestion or other unplanned network outages. This is especially the case for IP multicast networks. One technique to recover from packet loss without incurring unbounded delay for all the receivers is to duplicate the packets and send them in separate redundant streams. This document explains how RTP streams can be duplicated without breaking RTP and RTP Control Protocol (RTCP) rules.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 26, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology and Requirements Notation	3
3. Dual Streaming Use Cases	3
3.1. Temporal Redundancy	4
3.2. Spatial Redundancy	4
3.2.1. Using Separate Source Interfaces	4
3.2.2. Using Separate Destination Addresses and/or Ports	5
3.3. Dual Streaming over a Single Path or Multiple Paths	5
4. Use of RTP and RTCP with Temporal Redundancy	6
4.1. RTCP Considerations	6
4.2. Signaling Considerations	6
5. Use of RTP and RTCP with Spatial Redundancy	7
5.1. RTCP Considerations	7
5.2. Signaling Considerations	7
6. Use of RTP and RTCP with Temporal and Spatial Redundancy	8
7. Security Considerations	8
8. IANA Considerations	8
9. Acknowledgments	8
10. References	8
10.1. Normative References	8
10.2. Informative References	9
Authors' Addresses	9

1. Introduction

RTP [RFC3550] transport is widely used today for delivering real-time multimedia streams. Most of the applications also rely on IP multicast to reach many receivers efficiently.

While the combination proves successful, there does exist a weakness. As [RFC2354] noted, packet loss is not avoidable. This might be due to congestion, it might also be a result of an unplanned outage caused by a flapping link, link or interface failure, a software bug, or a maintenance person accidentally cutting the wrong fiber. Since UDP does not provide any means for detecting loss and retransmitting packets, it leaves up to the RTP or the applications to detect and recover from the loss. For retransmission-based recovery, one example is described in [RFC4588].

One technique to recover from packet loss without incurring unbounded delay for all the receivers is to duplicate the packets and send them in separate redundant streams. Variations of this technique have already been implemented and deployed today [IC2011]. However, duplication of RTP streams without breaking the RTP and RTCP functionality has not been documented properly. This document explains how duplication can be achieved for RTP streams.

2. Terminology and Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Dual Streaming Use Cases

Dual streaming refers to a technique that involves transmitting two redundant (often RTP) streams of the same content, with each stream itself capable of supporting the playback when there is no packet loss. Therefore, adding an additional stream provides a protection against packet loss. The level of protection depends on how the packets are sent and transmitted inside the network.

It is important to note that redundant streaming can easily be extended to support cases when more than two streams are desired. But triple, quadruple, or more, streaming is rarely used in practice.

3.1. Temporal Redundancy

From a routing perspective, two streams are considered identical if their following two fields are the same since they will be both routed over the same path:

- o IP Source Address
- o IP Destination Address

Two routing-plane identical RTP streams might carry the same payload but they could use different Synchronization Sources (SSRC) to differentiate the RTP packets belonging to each stream. In the context of dual streaming, we assume that the source duplicates the RTP packets and put them into separate RTP streams each with a unique SSRC identifier. All the redundant streams are transmitted in the same RTP session.

For example, two redundant RTP streams can be sent to the same IP destination address and UDP destination port with a certain delay between them [I-D.begen-mmusic-temporal-interleaving]. The streams carry the same payload in their respective RTP packets with identical sequence numbers. This allows the receiver (or any other node responsible for duplicate suppression) to identify and suppress the duplicate packets, and subsequently produce a hopefully loss-free and duplication-free output stream (called stream merging).

3.2. Spatial Redundancy

3.2.1. Using Separate Source Interfaces

An RTP source might have multiple network interfaces associated with it and it can send two redundant streams from two separate interfaces. Such streams can be routed over diverse or identical paths depending on the routing algorithm inside the network. At the receiving end, the node responsible for duplicate suppression can look into various RTP related fields to identify and suppress the duplicate packets.

If source-specific multicast (SSM) transport is used to carry such redundant streams, there will be a separate SSM session for each redundant stream since the streams are sourced from different interfaces (i.e., IP addresses). The receiving host has to join each SSM session separately.

3.2.2. Using Separate Destination Addresses and/or Ports

An RTP source might send the redundant streams to separate IP destination addresses and/or UDP ports.

3.3. Dual Streaming over a Single Path or Multiple Paths

Having described the characteristics of the streams, one can reach the following conclusions:

1. When two routing-plane identical streams are used, the two streams will have identical IP headers. This makes it impractical to forward the packets onto different paths. In order to minimize packet loss, the packets belonging to one stream are often interleaved with packets belonging to the other, and with a delay, so that if there is a packet loss, such a delay would allow the same packet from the other stream to reach the receiver because the chances that the same packet is lost in transit again is often small. This is what is also known as Time-shifted Redundancy, Temporal Redundancy or simply Delayed Duplication [I-D.begen-mmusic-temporal-interleaving] [IC2011]. This approach can be used with all three types of dual streaming described in Section 3.1, Section 3.2.1 and Section 3.2.2.
2. If the two streams have different IP headers, an additional opportunity arises in that one is able to build a network, with physically diverse paths, to deliver the two streams concurrently to the intended receivers. This reduces the delay when packet loss occurs and needs to be recovered. Additionally, it also further reduces chances for packet loss. An unrecoverable loss happens only when two network failures happen in such a way that the same packet is affected on both paths. This is referred to as Spatial Diversity or Spatial Redundancy [IC2011]. The techniques used to build diverse paths are beyond the scope of this document.

Note that spatial redundancy often offers less delay in recovering from packet loss provided that the forwarding delay of the network paths are more or less the same. For both temporal and spatial redundancy approaches, packet misordering might still happen and needs to be handled using the RTP sequence numbers.

To summarize, dual streaming allows an application and a network to work together to provide a near zero-loss transport with a bounded or minimum delay. The additional advantage includes a predictable bandwidth overhead that is proportional to the minimum bandwidth needed for the multimedia session, but independent of the number of receivers experiencing a packet loss and requesting a retransmission.

For a survey and comparison of similar approaches, refer to [IC2011].

4. Use of RTP and RTCP with Temporal Redundancy

To achieve temporal redundancy, the main and redundant RTP streams are sent using the same source and destination IP addresses and ports (that is the 5-tuple of transport protocol, source and destination IP addresses, and source and destination transport ports is the same for both main and redundant RTP streams). This is perhaps overly restrictive, but with the possible presence of network address and port translation (NAPT) devices, using anything other than an identical 5-tuple can also cause spatial redundancy.

Since main and redundant RTP streams follow an identical path, they are part of the same RTP session. Accordingly, the sender **MUST** choose a different SSRC for the redundant RTP stream than it chose for the main RTP stream, following the rules in [RFC3550] section 8.

4.1. RTCP Considerations

If RTCP is being sent for the main RTP stream, then the sender **MUST** also generate RTCP for the redundant RTP stream. The RTCP for the redundant RTP stream is generated exactly as-if the redundant RTP stream were a regular media stream; the sender **MUST NOT** duplicate the RTCP packets sent for the main RTP stream. The sender **MUST** use the same RTCP CNAME in the RTCP reports it sends for the main and redundant streams, so that the receiver can synchronize them.

Both main and redundant streams, and their corresponding RTCP, will be received. If RTCP is used, receivers **MUST** generate RTCP reports for both main and redundant streams in the usual way, treating them as entirely separate media streams.

Editor's note: The receiving node can also produce a new XR report to report on the (loss/delay/jitter/etc.) performance of the output stream after the stream merging process. This is TBD.

4.2. Signaling Considerations

Signaling is needed to allow the receiver to determine that an RTP stream is a redundant copy of another, rather than a separate stream that needs to be rendered in parallel. We need an SDP attribute to ensure that the receiver supports temporal redundancy, plus a new RTCP SDES item to indicate that this is a redundant stream that should not be directly rendered.

Editor's notes:

- o How should we correlate the duplicate streams? Grouping is straightforward when streams are SSRC-muxed but what if there are non-duplicated RTP streams in the same session? Maybe also use Magnus' srcname proposal?

The required SDP grouping semantics and SDP attribute have been defined in [I-D.begen-mmusic-redundancy-grouping] and [I-D.begen-mmusic-temporal-interleaving], respectively.

5. Use of RTP and RTCP with Spatial Redundancy

When using spatial redundancy, the redundant RTP stream is sent on using a different source and/or destination address/port pair. This will be a separate RTP session to the session conveying the main RTP stream.

SSRC for the redundant stream chosen randomly, following the rules in Section 8 of [RFC3550] and will almost certainly not match that of the main RTP stream. Sender MUST use the same RTCP CNAME for both main and redundant streams, in their separate sessions. Also the sender uses the new SDES item to indicate that this is a redundant stream. This is how the receiver can correlate the flows (can use srcname if appropriate).

5.1. RTCP Considerations

If RTCP is being sent for the main RTP stream, then the sender MUST also generate RTCP for the redundant RTP stream. The RTCP for the redundant RTP stream is generated exactly as-if the redundant RTP stream were a regular media stream; the sender MUST NOT duplicate the RTCP packets sent for the main RTP stream. The sender MUST use the same RTCP CNAME in the RTCP reports it sends for the main and redundant streams, so that the receiver can synchronize them.

Both main and redundant streams, and their corresponding RTCP, will be received. If RTCP is used, receivers MUST generate RTCP reports for both main and redundant streams in the usual way, treating them as entirely separate media streams.

Editor's note: The receiving node can also produce a new XR report to report on the (loss/delay/jitter/etc.) performance of the output stream after the stream merging process. This is TBD.

5.2. Signaling Considerations

The required SDP grouping semantics and SDP attribute have been defined in [I-D.begen-mmusic-redundancy-grouping] and

[I-D.begen-mmusic-temporal-interleaving], respectively.

6. Use of RTP and RTCP with Temporal and Spatial Redundancy

Editor's note: Nothing new here. This should use the same RTP/RTCP mechanisms, plus a combination of both sets of signaling.

7. Security Considerations

The security considerations of [RFC3550] apply to this memo.

Additional security considerations are TBC.

Editor's note: Email from csp. For the stream de-duplication device: it seems that this would work with SRTP encryption [RFC3711], since the headers are in the clear, but would break the authentication when the SSRC is rewritten. You could just re-authenticate the packets, and avoid re-encryption, with appropriate signaling of who authenticates the packets.

8. IANA Considerations

TBC.

9. Acknowledgments

Thanks to Magnus Westerlund for his suggestions.

10. References

10.1. Normative References

- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [I-D.begen-mmusic-temporal-interleaving]
Begen, A., Cai, Y., and H. Ou, "Delayed Duplication Attribute in the Session Description Protocol", draft-begen-mmusic-temporal-interleaving-03 (work in progress)

progress), October 2011.

[I-D.begen-mmusic-redundancy-grouping]

Begen, A., Cai, Y., and H. Ou, "Duplication Grouping Semantics in the Session Description Protocol", draft-begen-mmusic-redundancy-grouping-02 (work in progress), October 2011.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

10.2. Informative References

[RFC2354] Perkins, C. and O. Hodson, "Options for Repair of Streaming Media", RFC 2354, June 1998.

[RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.

[IC2011] Evans, J., Begen, A., Greengrass, J., and C. Filsfils, "Toward Lossless Video Transport (to appear in IEEE Internet Computing)", November 2011.

Authors' Addresses

Ali Begen
Cisco
181 Bay Street
Toronto, ON M5J 2T3
CANADA

Email: abegen@cisco.com

Colin Perkins
University of Glasgow
School of Computing Science
Glasgow, G12 8QQ
UK

Email: csp@cspcrkins.org

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2012

M. Westerlund
I. Johansson
Ericsson
C. Perkins
University of Glasgow
P. O'Hanlon
UCL
K. Carlberg
Gll
October 31, 2011

Explicit Congestion Notification (ECN) for RTP over UDP
draft-ietf-avtcore-ecn-for-rtp-05

Abstract

This memo specifies how Explicit Congestion Notification (ECN) can be used with the Real-time Transport Protocol (RTP) running over UDP, using RTP Control Protocol (RTCP) as a feedback mechanism. It defines a new RTCP Extended Report (XR) block for periodic ECN feedback, a new RTCP transport feedback message for timely reporting of congestion events, and a Session Traversal Utilities for NAT (STUN) extension used in the optional initialization method using Interactive Connectivity Establishment (ICE). Signalling and procedures for negotiation of capabilities and initialization methods are also defined.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 3, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the

document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
2. Conventions, Definitions and Acronyms	5
3. Discussion, Requirements, and Design Rationale	6
3.1. Requirements	7
3.2. Applicability	8
3.3. Interoperability	12
4. Overview of Use of ECN with RTP/UDP/IP	13
5. RTCP Extensions for ECN feedback	15
5.1. RTP/AVPF Transport Layer ECN Feedback packet	16
5.2. RTCP XR Report block for ECN summary information	19
6. SDP Signalling Extensions for ECN	21
6.1. Signalling ECN Capability using SDP	21
6.2. RTCP ECN Feedback SDP Parameter	25
6.3. XR Block ECN SDP Parameter	25
6.4. ICE Parameter to Signal ECN Capability	26
7. Use of ECN with RTP/UDP/IP	26
7.1. Negotiation of ECN Capability	26
7.2. Initiation of ECN Use in an RTP Session	27
7.3. Ongoing Use of ECN Within an RTP Session	34
7.4. Detecting Failures	37
8. Processing ECN in RTP Translators and Mixers	40
8.1. Transport Translators	40
8.2. Fragmentation and Reassembly in Translators	41
8.3. Generating RTCP ECN Feedback in Media Transcoders	43
8.4. Generating RTCP ECN Feedback in Mixers	44
9. Implementation considerations	44
10. IANA Considerations	45
10.1. SDP Attribute Registration	45
10.2. RTP/AVPF Transport Layer Feedback Message	45
10.3. RTCP Feedback SDP Parameter	46
10.4. RTCP XR Report blocks	46
10.5. RTCP XR SDP Parameter	46
10.6. STUN attribute	46
10.7. ICE Option	46
11. Security Considerations	46
12. Examples of SDP Signalling	49
12.1. Basic SDP Offer/Answer	49
12.2. Declarative Multicast SDP	51
13. Acknowledgments	52
14. References	52
14.1. Normative References	52
14.2. Informative References	53
Authors' Addresses	55

1. Introduction

This memo outlines how Explicit Congestion Notification (ECN) [RFC3168] can be used for Real-time Transport Protocol (RTP) [RFC3550] flows running over UDP/IP which use RTP Control Protocol (RTCP) as a feedback mechanism. The solution consists of feedback of ECN congestion experienced markings to the sender using RTCP, verification of ECN functionality end-to-end, and procedures for how to initiate ECN usage. Since the initiation process has some dependencies on the signalling mechanism used to establish the RTP session, a specification for signalling mechanisms using Session Description Protocol (SDP) [RFC4566] is included.

ECN is getting attention as a method to minimise the impact of congestion on real-time multimedia traffic. The use of ECN provides a way for the network to send a congestion control signal to a media transport without having to impair the media. Unlike packet loss, ECN signals unambiguously indicate congestion to the transport as quickly as feedback delays allow, and without confusing congestion with losses that might have occurred for other reasons such as transmission errors, packet-size errors, routing errors, badly implemented middleboxes, policy violations and so forth.

The introduction of ECN into the Internet requires changes to both the network and transport layers. At the network layer, IP forwarding has to be updated to allow routers to mark packets, rather than discarding them in times of congestion [RFC3168]. In addition, transport protocols have to be modified to inform the sender that ECN marked packets are being received, so it can respond to the congestion. The Transmission Control Protocol (TCP) [RFC3168], Stream Control Transmission Protocol (SCTP) [RFC4960] and Datagram Congestion Control Protocol (DCCP) [RFC4340] have been updated to support ECN, but to date there is no specification how UDP-based transports, such as RTP [RFC3550], can use ECN. This is due to the lack of feedback mechanisms directly in UDP. Instead the signaling control protocol on top of UDP needs to provide that feedback. For RTP that feedback is provided by RTCP.

The remainder of this memo is structured as follows. We start by describing the conventions, definitions and acronyms used in this memo in Section 2, and the design rationale and applicability in Section 3. Section 4 gives an overview of how ECN is used with RTP over UDP. RTCP extensions for ECN feedback are defined in Section 5, and SDP signalling extensions in Section 6. The details of how ECN is used with RTP over UDP are defined in Section 7. In Section 8 we describe how ECN is handled in RTP translators and mixers. Section 9 discusses some implementation considerations, Section 10 lists IANA considerations, and Section 11 discusses security considerations.

2. Conventions, Definitions and Acronyms

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Abbreviations and Definitions:

Sender: A sender of RTP packets carrying an encoded media stream. The sender can change how the media transmission is performed by varying the media coding or packetisation. It is one end-point of the ECN control loop.

Receiver: A receiver of RTP packets with the intention to consume the media stream. It sends RTCP feedback on the received stream. It is the other end-point of the ECN control loop.

ECN Capable Host: A sender or receiver of a media stream that is capable of setting and/or processing ECN marks.

ECN Capable Transport (ECT): A transport flow where both sender and receiver are ECN capable hosts. Packets sent by an ECN Capable Transport will be marked as ECT(0) or ECT(1) on transmission. See [RFC3168] for the definition of the ECT(0) and ECT(1) marks.

ECN-CE: ECN Congestion Experienced mark (see [RFC3168]).

ECN Capable Packets: Packets with ECN mark set to either ECT(0), ECT(1) or ECN-CE.

Not-ECT packets: Packets that are not sent by an ECN capable transport, and are not ECN-CE marked.

ECN Oblivious Relay: A router or middlebox that treats ECN Capable Packets no differently from Not-ECT packets.

ECN Capable Queue: A queue that supports ECN-CE marking of ECN-Capable Packets to indicate congestion.

ECN Blocking Middlebox: A middlebox that discards ECN-Capable Packets.

ECN Reverting Middlebox: A middlebox that changes ECN-Capable Packets to Not-ECT packets by removing the ECN mark.

Note that RTP mixers or translators that operate in such a manner that they terminate or split the ECN control loop will take on the

role of receivers or senders. This is further discussed in Section 3.2.

3. Discussion, Requirements, and Design Rationale

ECN has been specified for use with TCP [RFC3168], SCTP [RFC4960], and DCCP [RFC4340] transports. These are all unicast protocols which negotiate the use of ECN during the initial connection establishment handshake (supporting incremental deployment, and checking if ECN marked packets pass all middleboxes on the path). ECN-CE marks are immediately echoed back to the sender by the receiving end-point using an additional bit in feedback messages, and the sender then interprets the mark as equivalent to a packet loss for congestion control purposes.

If RTP is run over TCP, SCTP, or DCCP, it can use the native ECN support provided by those protocols. This memo does not concern itself further with these use cases. However, RTP is more commonly run over UDP. This combination does not currently support ECN, and we observe that it has significant differences from the other transport protocols for which ECN has been specified. These include:

Signalling: RTP relies on separate signalling protocols to negotiate parameters before a session can be created, and doesn't include an in-band handshake or negotiation at session set-up time (i.e., there is no equivalent to the TCP three-way handshake in RTP).

Feedback: RTP does not explicitly acknowledge receipt of datagrams. Instead, the RTP Control Protocol (RTCP) provides reception quality feedback, and other back channel communication, for RTP sessions. The feedback interval is generally on the order of seconds, rather than once per network RTT (although the RTP/AVPF profile [RFC4585] allows more rapid feedback in most cases). RTCP is also very much oriented around counting packets, which makes byte counting congestion algorithms difficult to utilize.

Congestion Response: While it is possible to adapt the transmission of many audio/visual streams in response to network congestion, and such adaptation is required by [RFC3550], the dynamics of the congestion response may be quite different to those of TCP or other transport protocols.

Middleboxes: The RTP framework explicitly supports the concept of mixers and translators, which are middleboxes that are involved in media transport functions.

Multicast: RTP is explicitly a group communication protocol, and was designed from the start to support IP multicast (primarily Any Source Multicast (ASM) [RFC1112], although a recent extension supports Source Specific Multicast (SSM) [RFC3569] with unicast feedback [RFC5760]).

Application Awareness: When ECN support is provided within the transport protocol, the ability of the application to react to congestion is limited, since it has little visibility into the transport layer. By adding support of ECN to RTP using RTCP feedback, the application is made aware of congestion, allowing a wider range of reactions in response to that loss.

Counting vs Detecting Congestion: TCP and the protocols derived from it are mainly designed to respond the same whether they experience a burst of congestion indications within one RTT or just one. Whereas real-time applications may be concerned with the amount of congestion experienced, whether it is distributed smoothly or in bursts. When feedback of ECN was added to TCP [RFC3168], the receiver was designed to flip the echo congestion experienced (ECE) flag to 1 for a whole RTT then flop it back to zero. Whereas ECN feedback in RTCP will need to report a count of how much congestion has been experienced within an RTCP reporting period, irrespective of round trip times.

These differences will significantly alter the shape of ECN support in RTP-over-UDP compared to ECN support in TCP, SCTP, and DCCP, but do not invalidate the need for ECN support.

ECN support is more important for RTP sessions than, for instance, is the case for TCP. This is because the impact of packet loss in real-time audio-visual media flows is highly visible to users. Effective ECN support for RTP flows running over UDP will allow real-time audio-visual applications to respond to the onset of congestion before routers are forced to drop packets, allowing those applications to control how they reduce their transmission rate, and hence media quality, rather than responding to, and trying to conceal the effects of unpredictable packet loss. Furthermore, widespread deployment for ECN and active queue management in routers, should it occur, can potentially reduce unnecessary queueing delays in routers, lowering the round-trip time and benefiting interactive applications of RTP, such as voice telephony.

3.1. Requirements

Considering ECN, transport protocols supporting ECN, and RTP based applications one can create a set of requirements that must be satisfied to at least some degree if ECN is to be used by RTP over UDP.

- o REQ 1: A mechanism MUST exist to negotiate and initiate the use of ECN for RTP/UDP/IP sessions so that an RTP sender will not send packets with ECT in the IP header unless it knows that all potential receivers will understand any ECN-CE indications they might receive.
- o REQ 2: A mechanism MUST exist to feed back the reception of any packets that are ECN-CE marked to the packet sender.
- o REQ 3: The provided mechanism SHOULD minimise the possibility of cheating (either by the sender or receiver).
- o REQ 4: Some detection and fallback mechanism SHOULD exist to avoid loss of communication due to the attempted usage of ECN in case an intermediate node clears ECT or drops packets that are ECT marked.
- o REQ 5: Negotiation of ECN SHOULD NOT significantly increase the time taken to negotiate and set-up the RTP session (an extra RTT before the media can flow is unlikely to be acceptable for some use cases).
- o REQ 6: Negotiation of ECN SHOULD NOT cause media clipping at the start of a session.

The following sections describes how these requirements can be met for RTP over UDP.

3.2. Applicability

The use of ECN with RTP over UDP is dependent on negotiation of ECN capability between the sender and receiver(s), and validation of ECN support in all elements of the network path(s) traversed. RTP is used in a heterogeneous range of network environments and topologies, with various different signalling protocols. The mechanisms defined here make it possible to verify support for ECN in each of these environments, and irrespective of the topology.

Due to the need for each RTP sender that intends to use ECN with RTP to track all participants in the RTP session, the sub-sampling of the group membership as specified by "Sampling of the Group Membership in RTP" [RFC2762] MUST NOT be used.

The use of ECN is further dependent on a capability of the RTP media flow to react to congestion signalled by ECN marked packets. Depending on the application, media codec, and network topology, this adaptation can occur in various forms and at various nodes. As an example, the sender can change the media encoding, or the receiver can change the subscription to a layered encoding, or either reaction

can be accomplished by a transcoding middlebox. RFC 5117 identifies seven topologies in which RTP sessions may be configured, and which may affect the ability to use ECN:

Topo-Point-to-Point: This utilises standard unicast flows. ECN may be used with RTP in this topology in an analogous manner to its use with other unicast transport protocols, with RTCP conveying ECN feedback messages.

Topo-Multicast: This is either an any source multicast (ASM) group [RFC3569] with potentially several active senders and multicast RTCP feedback, or a source specific multicast (SSM) group [RFC4607] with a single distribution source and unicast RTCP feedback from receivers. RTCP is designed to scale to large group sizes while avoiding feedback implosion (see Section 6.2 of [RFC3550], [RFC4585], and [RFC5760]), and can be used by a sender to determine if all its receivers, and the network paths to those receivers, support ECN (see Section 7.2). It is somewhat more difficult to determine if all network paths from all senders to all receivers support ECN. Accordingly, we allow ECN to be used by an RTP sender using multicast UDP provided the sender has verified that the paths to all its known receivers support ECN, and irrespective of whether the paths from other senders to their receivers support ECN ("all its known receivers" are all the SSRCs that the RTP sender has received RTP or RTCP from the last five reporting intervals, i.e., they have not timed out). Note that group membership may change during the lifetime of a multicast RTP session, potentially introducing new receivers that are not ECN capable or have a path that doesn't support ECN. Senders must use the mechanisms described in Section 7.4 to check that all receivers, and the network paths traversed to reach those receivers, continue to support ECN, and they need to fallback to non-ECN use if any receivers join that do not.

SSM groups that uses unicast RTCP feedback [RFC5760] do need a few extra considerations. This topology can have multiple media senders that provides traffic to the distribution source (DS) and are separated from the DS. There can also be multiple feedback targets. The requirement for using ECN for RTP in this topology is that the media sender must be provided the feedback from the receivers, it may be in aggregated form from the feedback targets. We will not mention this SSM use case in the below text specifically, but when actions are required by the media source, they do apply also to case of SSM where the RTCP feedback goes to the Feedback Target.

This specification do support multicast, but has primarily considered smaller multicast groups and is not optimized for larger groups and their needs.

Topo-Translator: An RTP translator is an RTP-level middlebox that is invisible to the other participants in the RTP session (although it is usually visible in the associated signalling session). There are two types of RTP translator: those that do not modify the media stream, and are concerned with transport parameters, for example a multicast to unicast gateway; and those that do modify the media stream, for example transcoding between different media codecs. A single RTP session traverses the translator, and the translator must rewrite RTCP messages passing through it to match the changes it makes to the RTP data packets. A legacy, ECN-unaware, RTP translator is expected to ignore the ECN bits on received packets, and to set the ECN bits to not-ECT when sending packets, so causing ECN negotiation on the path containing the translator to fail (any new RTP translator that does not wish to support ECN may do so similarly). An ECN aware RTP translator may act in one of three ways:

- * If the translator does not modify the media stream, it should copy the ECN bits unchanged from the incoming to the outgoing datagrams, unless it is overloaded and experiencing congestion, in which case it may mark the outgoing datagrams with an ECN-CE mark. Such a translator passes RTCP feedback unchanged. See Section 8.1.
- * If the translator modifies the media stream to combine or split RTP packets, but does not otherwise transcode the media, it must manage the ECN bits in a way analogous to that described in Section 5.3 of [RFC3168], see Section 8.2 for details.
- * If the translator is a media transcoder, or otherwise modifies the content of the media stream, the output RTP media stream may have radically different characteristics than the input RTP media stream. Each side of the translator must then be considered as a separate transport connection, with its own ECN processing. This requires the translator interpose itself into the ECN negotiation process, effectively splitting the connection into two parts with their own negotiation. Once negotiation has been completed, the translator must generate RTCP ECN feedback back to the source based on its own reception, and must respond to RTCP ECN feedback received from the receiver(s) (see Section 8.3).

It is recognised that ECN and RTCP processing in an RTP translator that modifies the media stream is non-trivial.

Topo-Mixer: A mixer is an RTP-level middlebox that aggregates multiple RTP streams, mixing them together to generate a new RTP stream. The mixer is visible to the other participants in the RTP session, and is also usually visible in the associated signalling session. The RTP flows on each side of the mixer are treated independently for ECN purposes, with the mixer generating its own RTCP ECN feedback, and responding to ECN feedback for data it sends. Since unicast transport between the mixer and any end-point are treated independently, it would seem reasonable to allow the transport on one side of the mixer to use ECN, while the transport on the other side of the mixer is not ECN capable, if this is desired. See Section 8.4 for details in how mixers should process ECN.

Topo-Video-switch-MCU: A video switching MCU receives several RTP flows, but forwards only one of those flows onwards to the other participants at a time. The flow that is forwarded changes during the session, often based on voice activity. Since only a subset of the RTP packets generated by a sender are forwarded to the receivers, a video switching MCU can break ECN negotiation (the success of the ECN negotiation may depend on the voice activity of the participant at the instant the negotiation takes place - shout if you want ECN). It also breaks congestion feedback and response, since RTP packets are dropped by the MCU depending on voice activity rather than network congestion. This topology is widely used in legacy products, but is NOT RECOMMENDED for new implementations and SHALL NOT be used with ECN.

Topo-RTCP-terminating-MCU: In this scenario, each participant runs an RTP point-to-point session between itself and the MCU. Each of these sessions is treated independently for the purposes of ECN and RTCP feedback, potentially with some using ECN and some not.

Topo-Asymmetric: It is theoretically possible to build a middlebox that is a combination of an RTP mixer in one direction and an RTP translator in the other. To quote RFC 5117 "This topology is so problematic and it is so easy to get the RTCP processing wrong, that it is NOT RECOMMENDED to implement this topology."

These topologies may be combined within a single RTP session.

The ECN mechanism defined in this memo is applicable to both sender and receiver controlled congestion algorithms. The mechanism ensures that both senders and receivers will know about ECN-CE markings and any packet losses. Thus the actual decision point for the congestion control is not relevant. This is a great benefit as the rate of an RTP session can be varied in a number of ways, for example a unicast media sender might use TFRC [RFC5348] or some other algorithm, while

a multicast session could use a sender based scheme adapting to the lowest common supported rate, or a receiver driven mechanism using layered coding to support more heterogeneous paths.

To ensure timely feedback of ECN-CE marked packets when needed, this mechanism requires support for the RTP/AVPF profile [RFC4585] or any of its derivatives, such as RTP/SAVPF [RFC5124]. The standard RTP/AVP profile [RFC3551] does not allow any early or immediate transmission of RTCP feedback, and has a minimal RTCP interval whose default value (5 seconds) is many times the normal RTT between sender and receiver.

3.3. Interoperability

The interoperability requirements for this specification are that there is at least one common interoperability point for all implementations. Since initialization using RTP and RTCP (Section 7.2.1) is the one method that works in all cases, although is not optimal for all uses, it is selected as mandatory to implement this initialisation method. This method requires both the RTCP XR extension and the ECN feedback format, which require the RTP/AVPF profile to ensure timely feedback.

When one considers all the uses of ECN for RTP it is clear that there exist congestion control mechanisms that are receiver driven only (Section 7.3.3). These congestion control mechanism do not require timely feedback of congestion events to the sender. If such a congestion control mechanism is combined with an initialization method that also doesn't require timely feedback using RTCP, like the leap of faith or the ICE based method then neither the ECN feedback format nor the RTP/AVPF profile would appear to be needed. However, fault detection can be greatly improved by using receiver side detection (Section 7.4.1) and early reporting of such cases using the ECN feedback mechanism.

For interoperability we mandate the implementation of the RTP/AVPF profile, with both RTCP extensions and the necessary signalling to support a common operations mode. This specification recommends the use of RTP/AVPF in all cases as negotiation of the common interoperability point requires RTP/AVPF, mixed negotiation of RTP/AVP and RTP/AVPF depending on other SDP attributes in the same media block is difficult, and the fact that fault detection can be improved when using RTP/AVPF.

The use of the ECN feedback format is also recommended, but cases exist where its use is not required due to no need for timely feedback. These will be explicitly noted using the term "no timely feedback required", and generally occur in combination with receiver

driven congestion control, and with the leap-of-faith and ICE-based initialization methods. We also note that any receiver driven congestion control solution that still requires RTCP for signalling of any adaptation information to the sender will still require RTP/AVPF for timeliness.

4. Overview of Use of ECN with RTP/UDP/IP

The solution for using ECN with RTP over UDP/IP consists of four different pieces that together make the solution work:

1. Negotiation of the capability to use ECN with RTP/UDP/IP
2. Initiation and initial verification of ECN capable transport
3. Ongoing use of ECN within an RTP session
4. Handling of dynamic behavior through failure detection, verification and fallback

Before an RTP session can be created, a signalling protocol is used to negotiate or at least configure session parameters (see Section 7.1). In some topologies the signalling protocol can also be used to discover the other participants. One of the parameters that must be agreed is the capability of a participant to support ECN. Note that all participants having the capability of supporting ECN does not necessarily imply that ECN is usable in an RTP session, since there may be middleboxes on the path between the participants which don't pass ECN-marked packets (for example, a firewall that blocks traffic with the ECN bits set). This document defines the information that needs to be negotiated, and provides a mapping to SDP for use in both declarative and offer/answer contexts.

When a sender joins a session for which all participants claim to support ECN, it must verify if that support is usable. There are three ways in which this verification can be done:

- o The sender may generate a (small) subset of its RTP data packets with the ECN field set to ECT(0) or ECT(1). Each receiver will then send an RTCP feedback packet indicating the reception of the ECT marked RTP packets. Upon reception of this feedback from each receiver it knows of, the sender can consider ECN functional for its traffic. Each sender does this verification independently. When a new receiver joins an existing RTP session, it will send RTCP reports in the usual manner. If those RTCP reports include ECN information, verification will have succeeded and sources can continue to send ECT packets. If not, verification fails and each

sender MUST stop using ECN (see Section 7.2.1 for details).

- o Alternatively, ECN support can be verified during an initial end-to-end STUN exchange (for example, as part of ICE connection establishment). After having verified connectivity without ECN capability an extra STUN exchange, this time with the ECN field set to ECT(0) or ECT(1), is performed on the candidate path that is about to be used. If successful the path's capability to convey ECN marked packets is verified. A new STUN attribute is defined to convey feedback that the ECT marked STUN request was received (see Section 7.2.2), along with an ICE signalling option (Section 6.4) to indicate that the check is to be performed.
- o Thirdly, the sender may make a leap of faith that ECN will work. This is only recommended for applications that know they are running in controlled environments where ECN functionality has been verified through other means. In this mode it is assumed that ECN works, and the system reacts to failure indicators if the assumption proved wrong. The use of this method relies on a high confidence that ECN operation will be successful, or an application where failure is not serious. The impact on the network and other users must be considered when making a leap of faith, so there are limitations on when this method is allowed (see Section 7.2.3).

The first mechanism, using RTP with RTCP feedback, has the advantage of working for all RTP sessions, but the disadvantages of potential clipping if ECN marked RTP packets are discarded by middleboxes, and slow verification of ECN support. The STUN-based mechanism is faster to verify ECN support, but only works in those scenarios supported by end-to-end STUN, such as within an ICE exchange. The third one, leap-of-faith, has the advantage of avoiding additional tests or complexities and enabling ECN usage from the first media packet. The downside is that if the end-to-end path contains middleboxes that do not pass ECN, the impact on the application can be severe: in the worst case, all media could be lost if a middlebox that discards ECN marked packets is present. A less severe effect, but still requiring reaction, is the presence of a middlebox that re-marks ECT marked packets to non-ECT, possibly marking packets with an ECN-CE mark as non-ECT. This could result in increased levels of congestion due to non-responsiveness, and impact media quality as applications end up relying on packet loss as an indication of congestion.

Once ECN support has been verified (or assumed) to work for all receivers, a sender marks all its RTP packets as ECT packets, while receivers rapidly feed back reports on any ECN-CE marks to the sender using RTCP in RTP/AVPF immediate or early feedback mode, unless no timely feedback is required. Each feedback report indicates the

receipt of new ECN-CE marks since the last ECN feedback packet, and also counts the total number of ECN-CE marked packets as a cumulative sum. This is the mechanism to provide the fastest possible feedback to senders about ECN-CE marks. On receipt of an ECN-CE marked packet, the system must react to congestion as-if packet loss has been reported. Section 7.3 describes the ongoing use of ECN within an RTP session.

This rapid feedback is not optimised for reliability, so another mechanism, RTCP XR ECN summary reports, is used to ensure more reliable, but less timely, reporting of the ECN information. The ECN summary report contains the same information as the ECN feedback format, only packed differently for better efficiency with reports for many sources. It is sent in a compound RTCP packet, along with regular RTCP reception reports. By using cumulative counters for observed ECN-CE, ECT, not-ECT, packet duplication, and packet loss the sender can determine what events have happened since the last report, independently of any RTCP packets having been lost.

RTCP reports MUST NOT be ECT marked, since ECT marked traffic may be dropped if the path is not ECN compliant. RTCP is used to provide feedback about what has been transmitted and what ECN markings that are received, so it is important that it is received in cases when ECT marked traffic is not getting through.

There are numerous reasons why the path the RTP packets take from the sender to the receiver may change, e.g., mobility, link failure followed by re-routing around it. Such an event may result in the packet being sent through a node that is ECN non-compliant, thus re-marking or dropping packets with ECT set. To prevent this from impacting the application for longer than necessary, the operation of ECN is constantly monitored by all senders (Section 7.4). Both the RTCP XR ECN summary reports and the ECN feedback packets allow the sender to compare the number of ECT(0), ECT(1), and non-ECT marked packets received with the number that were sent, while also reporting ECN-CE marked and lost packets. If these numbers do not agree, it can be inferred that the path does not reliably pass ECN-marked packets. A sender detecting a possible ECN non-compliance issue should then stop sending ECT marked packets to determine if that allows the packets to be correctly delivered. If the issues can be connected to ECN, then ECN usage is suspended.

5. RTCP Extensions for ECN feedback

This memo defines two new RTCP extensions: one RTP/AVPF [RFC4585] transport layer feedback format for urgent ECN information, and one RTCP XR [RFC3611] ECN summary report block type for regular reporting

of the ECN marking information.

5.1. RTP/AVPF Transport Layer ECN Feedback packet

This RTP/AVPF transport layer feedback format is intended for use in RTP/AVPF early or immediate feedback modes when information needs to urgently reach the sender. Thus its main use is to report reception of an ECN-CE marked RTP packet so that the sender may perform congestion control, or to speed up the initiation procedures by rapidly reporting that the path can support ECN-marked traffic. The feedback format is also defined with reduced size RTCP [RFC5506] in mind, where RTCP feedback packets may be sent without accompanying Sender or Receiver Reports that would contain the Extended Highest Sequence number and the accumulated number of packet losses. Both are important for ECN to verify functionality and keep track of when CE marking does occur.

The RTP/AVPF transport layer feedback packet starts with the common header defined by the RTP/AVPF profile [RFC4585] which is reproduced in Figure 1. The FMT field takes the value [TBA1] to indicate that the Feedback Control Information (FCI) contains ECN Feedback report, as defined in Figure 2.

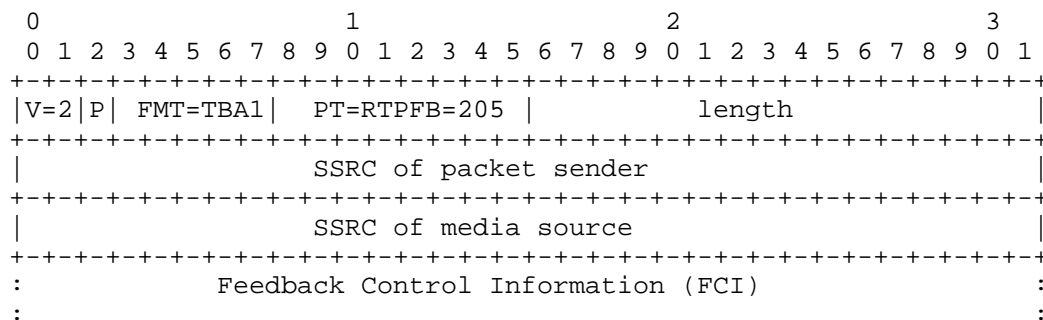


Figure 1: RTP/AVPF Common Packet Format for Feedback Messages

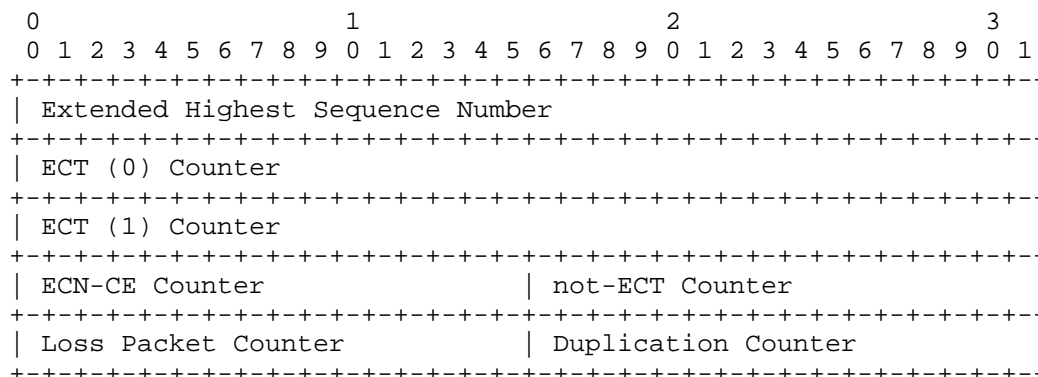


Figure 2: ECN Feedback Report Format

The ECN Feedback Report contains the following fields:

Extended Highest Sequence Number: The 32-bit Extended highest sequence number received, as defined by [RFC3550]. Indicates the highest RTP sequence number to which this report relates.

ECT(0) Counter: The 32-bit cumulative number of RTP packets with ECT(0) received from this SSRC.

ECT(1) Counter: The 32-bit cumulative number of RTP packets with ECT(1) received from this SSRC.

ECN-CE Counter: The cumulative number of RTP packets received from this SSRC since the receiver joined the RTP session that were ECN-CE marked, including ECN-CE marks in any duplicate packets. The receiver should keep track of this value using a local representation that is at least 32-bits, and only include the 16-bits with least significance. In other words, the field will wrap if more than 65535 ECN-CE marked packets have been received.

not-ECT Counter: The cumulative number of RTP packets received from this SSRC since the receiver joined the RTP session that had an ECN field value of not-ECT. The receiver should keep track of this value using a local representation that is at least 32-bits, and only include the 16-bits with least significance. In other words, the field will wrap if more than 65535 not-ECT packets have been received.

Lost Packets Counter: The cumulative number of RTP packets that the receiver expected to receive minus the number of packets it actually received that are not a duplicate of an already received packet, from this SSRC since the receiver joined the RTP session.

Note that packets that arrive late are not counted as lost. The receiver should keep track of this value using a local representation that is at least 32-bits, and only include the 16-bits with least significance. In other words, the field will wrap if more than 65535 packets are lost.

Duplication Counter: The cumulative number of RTP packets received that are a duplicate of an already received packet from this SSRC since the receiver joined the RTP session. The receiver should keep track of this value using a local representation that is at least 32-bits, and only include the 16-bits with least significance. In other words, the field will wrap if more than 65535 duplicate packets have been received.

All fields in the ECN Feedback Report are unsigned integers in network byte order. Each ECN Feedback Report corresponds to a single RTP source (SSRC). Multiple sources can be reported by including multiple ECN Feedback Reports packets in an compound RTCP packet.

The counters SHALL be initiated to 0 for each new SSRC received. This to enable detection of ECN-CE marks or Packet loss on the initial report from a specific participant.

The use of at least 32-bit counters allows even extremely high packet volume applications to not have wrapping of counters within any timescale close to the RTCP reporting intervals. However, 32-bits are not sufficiently large to disregard the fact that wrappings may happen during the life time of a long-lived RTP session. Thus handling of wrapping of these counters MUST be supported. It is recommended that implementations uses local representation of these counters that are longer than 32-bits to enable easy handling of wraps.

There is a difference in packet duplication reports between the packet loss counter that is defined in the Receiver Report Block [RFC3550] and that defined here. To avoid holding state for what RTP sequence numbers have been received, [RFC3550] specifies that one can count packet loss by counting the number of received packets and comparing it to the number of packets expected. As a result a packet duplication can hide a packet loss. However, when populating the ECN Feedback report, a receiver needs to track the sequence numbers actually received and count duplicates and packet loss separately to provide a more reliable indication. Reordering may however still result in that packet loss is reported in one report and then removed in the next.

The ECN-CE counter is robust for packet duplication. Adding each received ECN-CE marked packet to the counter is not an issue, in fact

it is required to ensure complete tracking of the ECN state. If one of the clones was ECN-CE marked that is still an indication of congestion. Packet duplication has potential impact on the ECN verification and thus there is a need to count the duplicates.

5.2. RTCP XR Report block for ECN summary information

This unilateral XR report block combined with RTCP SR or RR report blocks carries the same information as the ECN Feedback Report and is based on the same underlying information. However, the ECN Feedback Report is intended to report on an ECN-CE mark as soon as possible, while this extended report is for the regular RTCP reporting and continuous verification of the ECN functionality end-to-end.

The ECN Summary report block consists of one RTCP XR report block header, shown in Figure 3 followed by one or more ECN summary report data blocks, as defined in Figure 4.

```

      0                   1                   2                   3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   BT=[TBA2]   | Reserved           |      Block Length      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Figure 3: RTCP XR Report Header

```

      0                   1                   2                   3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| SSRC of Media Sender |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| ECT (0) Counter      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| ECT (1) Counter      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| ECN-CE Counter        | not-ECT Counter |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Loss Packet Counter   | Duplication Counter |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Figure 4: RTCP XR ECN Summary Report

The RTCP XR ECN Summary Report contains the following fields:

BT: Block Type identifying the ECN summary report block. Value is [TBA2].

Reserved: All bits SHALL be set to 0 on transmission and ignored on reception.

Block Length: The length of the report block. Used to indicate the number of report data blocks present in the ECN summary report. This length will be $5*n$, where n is the number of ECN summary report blocks, since blocks are a fixed size. The block length MAY be zero if there is nothing to report. Receivers MUST discard reports where the block length is not a multiple of five octets, since these cannot be valid.

SSRC of Media Sender: The SSRC identifying the media sender this report is for.

ECT(0) Counter: as in Section 5.1.

ECT(1) Counter: as in Section 5.1.

ECN-CE Counter: as in Section 5.1.

not-ECT Counter: as in Section 5.1.

Loss Packet Counter: as in Section 5.1.

Duplication Counter: as in Section 5.1.

The Extended Highest Sequence number counter for each SSRC is not present in RTCP XR report, in contrast to the feedback version. The reason is that this summary report will rely on the information sent in the Sender Report (SR) or Receiver Report (RR) blocks part of the same RTCP compound packet. The Extended Highest Sequence number is available from the SR or RR.

All the SSRCs that are present in the SR or RR SHOULD also be included in the RTCP XR ECN summary report. In cases where the number of senders are so large that the combination of SR/RR and the ECN summary for all the senders exceed the MTU, then only a subset of the senders SHOULD be included so that the reports for the subset fits within the MTU. The subsets SHOULD be selected round-robin across multiple intervals so that all sources are periodically reported. In case there are no SSRCs that currently are counted as senders in the session, the report block SHALL still be sent with no report block entry and a zero report block length to continuously indicate to the other participants the receiver capability to report ECN information.

6. SDP Signalling Extensions for ECN

This section defines a number of SDP signalling extensions used in the negotiation of the ECN for RTP support when using SDP. This includes one SDP attribute "ecn-capable-rtp" that negotiates the actual operation of ECN for RTP. Two SDP signalling parameters are defined to indicate the use of the RTCP XR ECN summary block and the RTP/AVPF feedback format for ECN. One ICE option SDP representation is also defined.

6.1. Signalling ECN Capability using SDP

One new SDP attribute, "a=ecn-capable-rtp", is defined. This is a media level attribute, and MUST NOT be used at the session level. It is not subject to the character set chosen. The aim of this signalling is to indicate the capability of the sender and receivers to support ECN, and to negotiate the method of ECN initiation to be used in the session. The attribute takes a list of initiation methods, ordered in decreasing preference. The defined values for the initiation method are:

rtp: Using RTP and RTCP as defined in Section 7.2.1.

ice: Using STUN within ICE as defined in Section 7.2.2.

leap: Using the leap of faith method as defined in Section 7.2.3.

Further methods may be specified in the future, so unknown methods MUST be ignored upon reception.

In addition, a number of OPTIONAL parameters may be included in the "a=ecn-capable-rtp" attribute as follows:

mode: This parameter signals the endpoint's capability to set and read ECN marks in UDP packets. An examination of various operating systems has shown that end-system support for ECN marking of UDP packets may be symmetric or asymmetric. By this we mean that some systems may allow end points to set the ECN bits in an outgoing UDP packet but not read them, while others may allow applications to read the ECN bits but not set them. This either/or case may produce an asymmetric support for ECN and thus should be conveyed in the SDP signalling. The "mode=setread" state is the ideal condition where an endpoint can both set and read ECN bits in UDP packets. The "mode=setonly" state indicates that an endpoint can set the ECT bit, but cannot read the ECN bits from received UDP packets to determine if upstream congestion occurred. The "mode=readonly" state indicates that the endpoint can read the ECN bits to determine if congestion has occurred for

incoming packets, but it cannot set the ECT bits in outgoing UDP packets. When the "mode=" parameter is omitted it is assumed that the node has "setread" capabilities. This option can provide for an early indication that ECN cannot be used in a session. This would be case when both the offerer and answerer set the "mode=" parameter to "setonly" or both set it to "readonly".

ect: This parameter makes it possible to express the preferred ECT marking. This is either "random", "0", or "1", with "0" being implied if not specified. The "ect" parameter describes a receiver preference, and is useful in the case where the receiver knows it is behind a link using IP header compression, the efficiency of which would be seriously disrupted if it were to receive packets with randomly chosen ECT marks. It is RECOMMENDED that ECT(0) marking be used.

The ABNF [RFC5234] grammar for the "a=ecn-capable-rtp" attribute is shown in Figure 5.

```

ecn-attribute = "a=ecn-capable-rtp:" SP init-list [SP parm-list]
init-list    = init-value *("," init-value)
init-value   = "rtp" / "ice" / "leap" / init-ext
init-ext     = token
parm-list    = parm-value *("; " SP parm-value)
parm-value   = mode / ect / parm-ext
mode         = "mode=" ("setonly" / "setread" / "readonly")
ect          = "ect=" ("0" / "1" / "random")
parm-ext     = parm-name "=" parm-value-ext
parm-name    = token
parm-value-ext = token / quoted-string
quoted-string = DQUOTE *qdtex DQUOTE
qdtex        = %x20-21 / %x23-7E / %x80-FF
              ; any 8-bit ASCII except <">

; external references:
; token: from RFC 4566
; SP and DQUOTE from RFC 5234

```

Figure 5: ABNF Grammar for the "a=ecn-capable-rtp" attribute

6.1.1. Use of "a=ecn-capable-rtp:" with the Offer/Answer Model

When SDP is used with the offer/answer model [RFC3264], the party generating the SDP offer MUST insert an "a=ecn-capable-rtp" attribute into the media section of the SDP offer of each RTP session for which it wishes to use ECN. The attribute includes one or more ECN initiation methods in a comma separated list in decreasing order of preference, with any number of optional parameters following. The answering party compares the list of initiation methods in the offer

with those it supports in order of preference. If there is a match, and if the receiver wishes to attempt to use ECN in the session, it includes an "a=ecn-capable-rtp" attribute containing its single preferred choice of initiation method, and any optional parameters, in the media sections of the answer. If there is no matching initiation method capability, or if the receiver does not wish to attempt to use ECN in the session, it does not include an "a=ecn-capable-rtp" attribute in its answer. If the attribute is removed in the answer then ECN MUST NOT be used in any direction for that media flow. If there are initialization methods that are unknown, they MUST be ignored on reception and MUST NOT be included in an answer.

The endpoints' capability to set and read ECN marks, as expressed by the optional "mode=" parameter, determines whether ECN support can be negotiated for flows in one or both directions:

- o If the "mode=setonly" parameter is present in the "a=ecn-capable-rtp" attribute of the offer and the answering party is also "mode=setonly", then there is no common ECN capability, and the answer MUST NOT include the "a=ecn-capable-rtp" attribute. Otherwise, if the offer is "mode=setonly" then ECN may only be initiated in the direction from the offering party to the answering party.
- o If the "mode=readonly" parameter is present in the "a=ecn-capable-rtp" attribute of the offer and the answering party is "mode=readonly", then there is no common ECN capability, and the answer MUST NOT include the "a=ecn-capable-rtp" attribute. Otherwise, if the offer is "mode=readonly" then ECN may only be initiated in the direction from the answering party to the offering party.
- o If the "mode=setread" parameter is present in the "a=ecn-capable-rtp" attribute of the offer and the answering party is "setonly", then ECN may only be initiated in the direction from the answering party to the offering party. If the offering party is "mode=setread" but the answering party is "mode=readonly", then ECN may only be initiated in the direction from the offering party to the answering party. If both offer and answer are "mode=setread", then ECN may be initiated in both directions. Note that "mode=setread" is implied by the absence of a "mode=" parameter in the offer or the answer.
- o An offer that does not include a "mode=" parameter MUST be treated as-if a "mode=setread" parameter had been included.

In an RTP session using multicast and ECN, participants that intend to send RTP packets SHOULD support setting ECT marks in RTP packets

(i.e., should be "mode=setonly" or "mode=setread"). Participants receiving data need the capability to read ECN marks on incoming packets. It is important that receivers can read ECN marks (are "mode=readonly" or "mode=setread"), since otherwise no sender in the multicast session will be able to enable ECN. Accordingly, receivers that are "mode=setonly" SHOULD NOT join multicast RTP sessions that use ECN. If session participants that are not aware of the ECN for RTP signalling are invited to a multicast session, and simply ignore the signalling attribute, the other party in the offer/answer exchange SHOULD terminate the SDP dialogue so that the participant leaves the session.

The "ect=" parameter in the "a=ecn-capable-rtp" attribute is set independently in the offer and the answer. Its value in the offer indicates a preference for the sending behaviour of the answering party, and its value in the answer indicates a sending preference for the behaviour of the offering party. It will be the senders choice to honour the receivers preference for what to receive or not. In multicast sessions, all senders SHOULD set the ECT marks using the value declared in the "ect=" parameter.

Unknown optional parameters MUST be ignored on reception, and MUST NOT be included in the answer. That way new parameters may be introduced and verified to be supported by the other end-point by having them include it in any answer.

6.1.2. Use of "a=ecn-capable-rtp:" with Declarative SDP

When SDP is used in a declarative manner, for example in a multicast session using the Session Announcement Protocol (SAP, [RFC2974]), negotiation of session description parameters is not possible. The "a=ecn-capable-rtp" attribute MAY be added to the session description to indicate that the sender will use ECN in the RTP session. The attribute MUST include a single method of initiation. Participants MUST NOT join such a session unless they have the capability to receive ECN-marked UDP packets, implement the method of initiation, and can generate RTCP ECN feedback. The mode parameter MAY also be included in declarative usage, to indicate the minimal capability is required by the consumer of the SDP. So for example in a SSM session the participants configured with a particular SDP will all be in a media receive only mode, thus mode=readonly will work as the capability of reporting on the ECN markings in the received is what is required. However, using "mode=readonly" also in ASM sessions is reasonable, unless all senders are required to attempt to use ECN for their outgoing RTP data traffic, in which case the mode needs to be set to "setread".

6.1.3. General Use of the "a=ecn-capable-rtp:" Attribute

The "a=ecn-capable-rtp" attribute MAY be used with RTP media sessions using UDP/IP transport. It MUST NOT be used for RTP sessions using TCP, SCTP, or DCCP transport, or for non-RTP sessions.

As described in Section 7.3.3, RTP sessions using ECN require rapid RTCP ECN feedback, unless timely feedback is not required due to a receiver driven congestion control. To ensure that the sender can react to ECN-CE marked packets timely feedback is usually required. Thus, the use of the Extended RTP Profile for RTCP-Based Feedback (RTP/AVPF) [RFC4585] or other profile that inherits RTP/AVPF's signalling rules, MUST be signalled unless timely feedback is not required. If timely feedback is not required it is still RECOMMENDED to use RTP/AVPF. The signalling of an RTP/AVPF based profile is likely to be required even if the preferred method of initialization and the congestion control does not require timely feedback, as the common interoperable method is likely to be signalled or the improved fault reaction is desired.

6.2. RTCP ECN Feedback SDP Parameter

A new "nack" feedback parameter "ecn" is defined to indicate the usage of the RTCP ECN feedback packet format (Section 5.1). The ABNF [RFC5234] definition of the SDP parameter extension is:

```
rtcp-fb-nack-param = <See section 4.2 of RFC 4585>
rtcp-fb-nack-param /= ecn-fb-par
ecn-fb-par          = SP "ecn"
```

The offer/answer rules for this SDP feedback parameters are specified in the RTP/AVPF profile [RFC4585].

6.3. XR Block ECN SDP Parameter

A new unilateral RTCP XR block for ECN summary information is specified, thus the XR block SDP signalling also needs to be extended with a parameter. This is done in the same way as for the other XR blocks. The XR block SDP attribute as defined in Section 5.1 of the RTCP XR specification [RFC3611] is defined to be extensible. As no parameter values are needed for this ECN summary block, this parameter extension consists of a simple parameter name used to indicate support and intent to use the XR block.

```
xr-format          = <See Section 5.1 of [RFC3611]>
xr-format          /= ecn-summary-par
ecn-summary-par    = "ecn-sum"
```

For SDP declarative and offer/answer usage, see the RTCP XR specification [RFC3611] and its description of how to handle

unilateral parameters.

6.4. ICE Parameter to Signal ECN Capability

One new ICE [RFC5245] option, "rtp+ecn", is defined. This is used with the SDP session level "a=ice-options" attribute in an SDP offer to indicate that the initiator of the ICE exchange has the capability to support ECN for RTP-over-UDP flows (via "a=ice-options: rtp+ecn"). The answering party includes this same attribute at the session level in the SDP answer if it also has the capability, and removes the attribute if it does not wish to use ECN, or doesn't have the capability to use ECN. If the ICE initiation method (Section 7.2.2) is actually going to be used, it is also needs to be explicitly negotiated using the "a=ecn-capable-rtp" attribute. This ICE option SHALL be included when the ICE initiation method is offered or declared in the SDP.

Note: This signalling mechanism is not strictly needed as long as the STUN ECN testing capability is used within the context of this document. It may however be useful if the ECN verification capability is used in additional contexts.

7. Use of ECN with RTP/UDP/IP

In the detailed specification of the behaviour below, the different functions in the general case will first be discussed. In case special considerations are needed for middleboxes, multicast usage etc, those will be specially discussed in related subsections.

7.1. Negotiation of ECN Capability

The first stage of ECN negotiation for RTP-over-UDP is to signal the capability to use ECN. An RTP system that supports ECN and uses SDP for its signalling MUST implement the SDP extension to signal ECN capability as described in Section 6.1, the RTCP ECN feedback SDP parameter defined in Section 6.2, and the XR Block ECN SDP parameter defined in Section 6.3. It MAY also implement alternative ECN capability negotiation schemes, such as the ICE extension described in Section 6.4. Other signalling systems will need to define signalling parameters corresponding to those defined for SDP.

The "ecn-capable-rtp" SDP attribute MUST be used when employing ECN for RTP according to this specification in systems using SDP. As the RTCP XR ECN summary report is required independently of the initialization method or congestion control scheme, the "rtcp-xr" attribute with the "ecn-sum" parameter MUST also be used. The "rtcp-fb" attribute with the "nack" parameter "ecn" MUST be used

whenever the initialization method or a congestion control algorithm requires timely sender side knowledge of received CE markings. If the congestion control scheme requires additional signalling, this should be indicated as appropriate.

7.2. Initiation of ECN Use in an RTP Session

Once the sender and the receiver(s) have agreed that they have the capability to use ECN within a session, they may attempt to initiate ECN use. All session participants connected over the same transport MUST use the same initiation method. RTP mixers or translators can use different initiation methods to different participants that are connected over different underlying transports. The mixer or translator will need to do individual signalling with each participant to ensure it is consistent with the ECN support in those cases where it does not function as one end-point for the ECN control loop.

At the start of the RTP session, when the first packets with ECT are sent, it is important to verify that IP packets with ECN field values of ECT or ECN-CE will reach their destination(s). There is some risk that the use of ECN will result in either reset of the ECN field, or loss of all packets with ECT or ECN-CE markings. If the path between the sender and the receivers exhibits either of these behaviours one needs to stop using ECN immediately to protect both the network and the application.

The RTP senders and receivers SHALL NOT ECT mark their RTCP traffic at any time. This is to ensure that packet loss due to ECN marking will not effect the RTCP traffic and the necessary feedback information it carries.

An RTP system that supports ECN MUST implement the initiation of ECN using in-band RTP and RTCP described in Section 7.2.1. It MAY also implement other mechanisms to initiate ECN support, for example the STUN-based mechanism described in Section 7.2.2, or use the leap of faith option if the session supports the limitations provided in Section 7.2.3. If support for both in-band and out-of-band mechanisms are signalled, the sender when negotiating SHOULD offer detection of ECT using STUN with ICE with higher priority than detection of ECT using RTP and RTCP.

No matter how ECN usage is initiated, the sender MUST continually monitor the ability of the network, and all its receivers, to support ECN, following the mechanisms described in Section 7.4. This is necessary because path changes or changes in the receiver population may invalidate the ability of the system to use ECN.

7.2.1. Detection of ECT using RTP and RTCP

The ECN initiation phase using RTP and RTCP to detect if the network path supports ECN comprises three stages. Firstly, the RTP sender generates some small fraction of its traffic with ECT marks to act as probe for ECN support. Then, on receipt of these ECT-marked packets, the receivers send RTCP ECN feedback packets and RTCP ECN summary reports to inform the sender that their path supports ECN. Finally, the RTP sender makes the decision to use ECN or not, based on whether the paths to all RTP receivers have been verified to support ECN.

Generating ECN Probe Packets: During the ECN initiation phase, an RTP sender SHALL mark a small fraction of its RTP traffic as ECT, while leaving the remainder of the packets unmarked. The main reason for only marking some packets is to maintain usable media delivery during the ECN initiation phase in those cases where ECN is not supported by the network path. A secondary reason to send some not-ECT packets are to ensure that the receivers will send RTCP reports on this sender, even if all ECT marked packets are lost in transit. The not-ECT packets also provide a base-line to compare performance parameters against. A fourth reason for only probing with a small number of packets is to reduce the risk that significant numbers of congestion markings might be lost if ECT is cleared to Not-ECT by an ECN-Reverting Middlebox. Then any resulting lack of congestion response is likely to have little damaging effect on others. An RTP sender is RECOMMENDED to send a minimum of two packets with ECT markings per RTCP reporting interval. In case a random ECT pattern is intended to be used, at least one packet with ECT(0) and one with ECT(1) should be sent per reporting interval; in case a single ECT marking is to be used, only that ECT value SHOULD be sent. The RTP sender SHALL continue to send some ECT marked traffic as long as the ECN initiation phase continues. The sender SHOULD NOT mark all RTP packets as ECT during the ECN initiation phase.

This memo does not mandate which RTP packets are marked with ECT during the ECN initiation phase. An implementation should insert ECT marks in RTP packets in a way that minimises the impact on media quality if those packets are lost. The choice of packets to mark is clearly very media dependent, but the use of RTP NO-OP payloads [I-D.ietf-avt-rtp-no-op], if supported, would be an appropriate choice. For audio formats, it would make sense for the sender to mark comfort noise packets or similar. For video formats, packets containing P- or B-frames (rather than I-frames) would be an appropriate choice. No matter which RTP packets are marked, those packets MUST NOT be sent in duplicate, with and without ECT, since the RTP sequence number is used to identify packets that are received with ECN markings.

Generating RTCP ECN Feedback: If ECN capability has been negotiated in an RTP session, the receivers in the session MUST listen for ECT or ECN-CE marked RTP packets, and generate RTCP ECN feedback packets (Section 5.1) to mark their receipt. An immediate or early (depending on the RTP/AVPF mode) ECN feedback packet SHOULD be generated on receipt of the first ECT or ECN-CE marked packet from a sender that has not previously sent any ECT traffic. Each regular RTCP report MUST also contain an ECN summary report (Section 5.2). Reception of subsequent ECN-CE marked packets MUST result in additional early or immediate ECN feedback packets being sent unless no timely feedback is required.

Determination of ECN Support: RTP is a group communication protocol, where members can join and leave the group at any time. This complicates the ECN initiation phase, since the sender must wait until it believes the group membership has stabilised before it can determine if the paths to all receivers support ECN (group membership changes after the ECN initiation phase has completed are discussed in Section 7.3).

An RTP sender shall consider the group membership to be stable after it has been in the session and sending ECT-marked probe packets for at least three RTCP reporting intervals (i.e., after sending its third regularly scheduled RTCP packet), and when a complete RTCP reporting interval has passed without changes to the group membership. ECN initiation is considered successful when the group membership is stable, and all known participants have sent one or more RTCP ECN feedback packets or RTCP XR ECN summary reports indicating correct receipt of the ECT-marked RTP packets generated by the sender.

As an optimisation, if an RTP sender is initiating ECN usage towards a unicast address, then it MAY treat the ECN initiation as provisionally successful if it receives an RTCP ECN feedback report or an RTCP XR ECN summary report indicating successful receipt of the ECT-marked packets, with no negative indications, from a single RTP receiver (where a single RTP receiver is considered as all SSRCs used by a single RTCP CNAME). After declaring provisional success, the sender MAY generate ECT-marked packets as described in Section 7.3, provided it continues to monitor the RTCP reports for a period of three RTCP reporting intervals from the time the ECN initiation started, to check if there are any other participants in the session. Thus as long as any additional SSRC that report on the ECN usage are using the same RTCP CNAME as the previous reports and they are all indicating functional ECN the sender may continue. If other participants are detected, i.e., other RTCP CNAMEs, the sender MUST fallback to only ECT-marking a small fraction of its RTP

packets, while it determines if ECN can be supported following the full procedure described above. Different RTCP CNAMEs received over an unicast transport may occur when using translators in a multi-party RTP session (e.g., when using a centralised conference bridge).

Note: The above optimization supports peer to peer unicast transport with several SSRCs multiplexed onto the same flow (e.g., a single participant with two video cameras, or SSRC multiplexed RTP retransmission [RFC4588]). It is desirable to be able to rapidly negotiate ECN support for such a session, but the optimisation above can fail if there are implementations that use the same CNAME for different parts of a distributed implementation that have different transport characteristics (e.g., if a single logical endpoint is split across multiple hosts).

ECN initiation is considered to have failed at the instant the initiating RTP sender received an RTCP packet that doesn't contain an RTCP ECN feedback report or ECN summary report from any RTP session participant that has an RTCP RR with an extended RTP sequence number field that indicates that it should have received multiple (>3) ECT marked RTP packets. This can be due to failure to support the ECN feedback format by the receiver or some middlebox, or the loss of all ECT marked packets. Both indicate a lack of ECN support.

If the ECN negotiation succeeds, this indicates that the path can pass some ECN-marked traffic, and that the receivers support ECN feedback. This does not necessarily imply that the path can robustly convey ECN feedback; Section 7.3 describes the ongoing monitoring that must be performed to ensure the path continues to robustly support ECN.

When a sender or receiver detects ECN failures on paths they should log these to enable follow up and statistics gathering regarding broken paths. The logging mechanism used is implementation dependent.

7.2.2. Detection of ECT using STUN with ICE

This section describes an OPTIONAL method that can be used to avoid media impact and also ensure an ECN capable path prior to media transmission. This method is considered in the context where the session participants are using ICE [RFC5245] to find working connectivity. We need to use ICE rather than STUN only, as the verification needs to happen from the media sender to the address and port on which the receiver is listening.

Note that this method is only applicable to sessions when the remote destinations are unicast addresses. In addition, transport translators that do not terminate the ECN control loop and may distribute received packets to more than one other receiver must either disallow this method (and use the RTP/RTCP method instead), or implement additional handling as discussed below. This is because the ICE initialization method verifies the underlying transport to one particular address and port. If the receiver at that address and port intends to use the received packets in a multi-point session then the tested capabilities and the actual session behavior are not matched.

To minimise the impact of set-up delay, and to prioritise the fact that one has a working connectivity rather than necessarily finding the best ECN capable network path, this procedure is applied after having performed a successful connectivity check for a candidate, which is nominated for usage. At that point an additional connectivity check is performed, sending the "ECN Check" attribute in a STUN packet that is ECT marked. On reception of the packet, a STUN server supporting this extension will note the received ECN field value, and send a STUN/UDP/IP packet in reply with the ECN field set to not-ECT and including an ECN check attribute. A STUN server that doesn't understand the extension, or is incapable of reading the ECN values on incoming STUN packets, should follow the rule in the STUN specification for unknown comprehension-optional attributes, and ignore the attribute, resulting in the sender receiving a STUN response without the ECN Check STUN attribute.

The ECN STUN checks can be lost on the path, for example due to the ECT marking, but also various other non ECN related reasons causing packet loss. The goal is to detect when the ECT markings are rewritten or if it is the ECT marking that causes packet loss so that the path can be determined as not ECT. Other reasons for packet loss should not result in a failure to verify the path as ECT. Therefore a number of retransmissions should be attempted. But, the sender of ECN STUN checks will also have to set a criteria for when it gives up testing for ECN capability on the path. As the ICE agent have successfully verified the path a RTT measurement for this path can be performed. To have high probability of successfully verifying the path it is RECOMMENDED that the client retransmitt the ECN STUN check 4 times. The interval between the retransmissions will be based on the Ta timer as defined in Section 16.1 for RTP Media Streams in ICE [RFC5245]. The number of ECN STUN checks needing to be sent will depend on the number of ECN capable flows (N) that is to be established. The interval between each transmission of an ECN check packet MUST be Ta. In other words for a given flow being verified for ECT the RTO is set to Ta*N. The transmission for that flow is stopped when an ECN Check STUN response has been received, which

don't indicate to retransmit the request due to temporary error, the maximum number of retransmissions has been sent. The ICE agent is recommended to give up on the ECN verification $\text{MAX}(1.5 \cdot \text{RTT}, 20 \text{ ms})$ after the last ECN STUN check was sent.

The STUN ECN check attribute contains one field and a flag, as shown in Figure 6. The flag indicates whether the echo field contains a valid value or not. The field is the ECN echo field, and when valid contains the two ECN bits from the packet it echoes back. The ECN check attribute is a comprehension optional attribute.

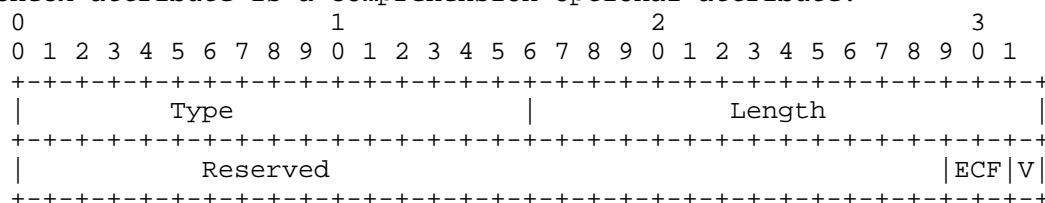


Figure 6: ECN Check STUN Attribute

V: Valid (1 bit) ECN Echo value field is valid when set to 1, and invalid when set 0.

ECF: ECN Echo value field (2 bits) contains the ECN field value of the STUN packet it echoes back when field is valid. If invalid the content is arbitrary.

Reserved: Reserved bits (29 bits) SHALL be set to 0 on transmission, and SHALL be ignored on reception.

This attribute MAY be included in any STUN request to request the ECN field to be echoed back. In STUN requests the V bit SHALL be set to 0. A compliant STUN server receiving a request with the ECN Check attribute SHALL read the ECN field value of the IP/UDP packet the request was received in. Upon forming the response the server SHALL include the ECN Check attribute setting the V bit to valid and include the read value of the ECN field into the ECF field. If the STUN responder was unable to ascertain, due to temporary errors, the ECN value of the STUN request, it SHALL set the V bit in the response to 0. The STUN client may retry immediately.

The ICE based initialization method does require some special consideration when used by a translator. This is especially for transport translators and translators that fragment or reassemble packets, since they do not separate the ECN control loops between the end-points and the translator. When using ICE-based initiation, such a translator must ensure that any participants joining an RTP session for which ECN has been negotiated are successfully verified in the

direction from the translator to the joining participant. Alternatively, it must correctly handle remarking of ECT RTP packets towards that participant. When a new participant joins the session, the translator will perform a check towards the new participant. If that is successfully completed the ECT properties of the session are maintained for the other senders in the session. If the check fails then the existing senders will now see a participant that fails to receive ECT. Thus the failure detection in those senders will eventually detect this. However to avoid misusing the network on the path from the translator to the new participant, the translator SHALL remark the traffic intended to be forwarded from ECT to non-ECT. Any packet intended to be forward that are ECN-CE marked SHALL be discard and not sent. In cases where the path from a new participant to the translator fails the ECT check then only that sender will not contribute any ECT marked traffic towards the translator.

7.2.3. Leap of Faith ECT initiation method

This method for initiating ECN usage is a leap of faith that assumes that ECN will work on the used path(s). The method is to go directly to "ongoing use of ECN" as defined in Section 7.3. Thus all RTP packets MAY be marked as ECT and the failure detection MUST be used to detect any case when the assumption that the path was ECT capable is wrong. This method is only recommended for controlled environments where the whole path(s) between sender and receiver(s) has been built and verified to be ECT.

If the sender marks all packets as ECT while transmitting on a path that contains an ECN-blocking middlebox, then receivers downstream of that middlebox will not receive any RTP data packets from the sender, and hence will not consider it to be an active RTP SSRC. The sender can detect this and revert to sending packets without ECT marks, since RTCP SR/RR packets from such receivers will either not include a report for sender's SSRC, or will report that no packets have been received, but this takes at least one RTCP reporting interval. It should be noted that a receiver might generate its first RTCP packet immediately on joining a unicast session, or very shortly after joining a RTP/AVPF session, before it has had chance to receive any data packets. A sender that receives RTCP SR/RR packet indicating lack of reception by a receiver SHOULD therefore wait for a second RTCP report from that receiver to be sure that the lack of reception is due to ECT-marking. Since this recovery process can take several tens of seconds, during which time the RTP session is unusable for media, it is NOT RECOMMENDED that the leap-of-faith ECT initiation method be used in environments where ECN-blocking middleboxes are likely to be present.

7.3. Ongoing Use of ECN Within an RTP Session

Once ECN has been successfully initiated for an RTP sender, that sender begins sending all RTP data packets as ECT-marked, and its receivers send ECN feedback information via RTCP packets. This section describes procedures for sending ECT-marked data, providing ECN feedback information via RTCP, and responding to ECN feedback information.

7.3.1. Transmission of ECT-marked RTP Packets

After a sender has successfully initiated ECN use, it SHOULD mark all the RTP data packets it sends as ECT. The sender SHOULD mark packets as ECT(0) unless the receiver expresses a preference for ECT(1) or random using the "ect" parameter in the "a=ecn-capable-rtp" attribute.

The sender SHALL NOT include ECT marks on outgoing RTCP packets, and SHOULD NOT include ECT marks on any other outgoing control messages (e.g., STUN [RFC5389] packets, DTLS [RFC4347] handshake packets, or ZRTP [RFC6189] control packets) that are multiplexed on the same UDP port. For control packets there might be exceptions, like the STUN based ECN check defined in Section 7.2.2.

7.3.2. Reporting ECN Feedback via RTCP

An RTP receiver that receives a packet with an ECN-CE mark, or that detects a packet loss, MUST schedule the transmission of an RTCP ECN feedback packet as soon as possible (subject to the constraints of [RFC4585] and [RFC3550]) to report this back to the sender unless no timely feedback is required. The feedback RTCP packet SHALL consist of at least one ECN feedback packet (Section 5.1) reporting on the packets received since the last ECN feedback packet, and will contain (at least) an RTCP SR/RR packet and an SDES packet, unless reduced size RTCP [RFC5506] is used. The RTP/AVPF profile in early or immediate feedback mode SHOULD be used where possible, to reduce the interval before feedback can be sent. To reduce the size of the feedback message, reduced size RTCP [RFC5506] MAY be used if supported by the end-points. Both RTP/AVPF and reduced size RTCP MUST be negotiated in the session set-up signalling before they can be used.

Every time a regular compound RTCP packet is to be transmitted, an ECN-capable RTP receiver MUST include an RTCP XR ECN summary report as described in Section 5.2 as part of the compound packet.

The multicast feedback implosion problem, that occurs when many receivers simultaneously send feedback to a single sender, must be

considered. The RTP/AVPF transmission rules will limit the amount of feedback that can be sent, avoiding the implosion problem but also delaying feedback by varying degrees from nothing up to a full RTCP reporting interval. As a result, the full extent of a congestion situation may take some time to reach the sender, although some feedback should arrive in a reasonably timely manner, allowing the sender to react on a single or a few reports.

7.3.3. Response to Congestion Notifications

The reception of RTP packets with ECN-CE marks in the IP header is a notification that congestion is being experienced. The default reaction on the reception of these ECN-CE marked packets MUST be to provide the congestion control algorithm with a congestion notification that triggers the algorithm to react as if packet loss had occurred. There should be no difference in congestion response if ECN-CE marks or packet drops are detected.

We note that there MAY be other reactions to ECN-CE specified in the future. Such an alternative reaction MUST be specified and considered to be safe for deployment under any restrictions specified. A potential example for an alternative reaction could be emergency communications (such as that generated by first responders, as opposed to the general public) in networks where the user has been authorized. A more detailed description of these other reactions, as well as the types of congestion control algorithms used by end-nodes, is outside of the scope of this document.

Depending on the media format, type of session, and RTP topology used, there are several different types of congestion control that can be used:

Sender-Driven Congestion Control: The sender is responsible for adapting the transmitted bit-rate in response to RTCP ECN feedback. When the sender receives the ECN feedback data it feeds this information into its congestion control or bit-rate adaptation mechanism so that it can react as if packet loss was reported. The congestion control algorithm to be used is not specified here, although TFRC [RFC5348] is one example that might be used.

Receiver-Driven Congestion Control: In a receiver driven congestion control mechanism, the receivers can react to the ECN-CE marks themselves without providing ECN-CE feedback to the sender. This may allow faster response than sender-driven congestion control in some circumstances and also scale to large number of receivers and multicast usage. One example of receiver-driven congestion control is implemented by providing the content in a layered way,

with each layer providing improved media quality but also increased bandwidth usage. The receiver locally monitors the ECN-CE marks on received packets to check if it experiences congestion with the current number of layers. If congestion is experienced, the receiver drops one layer, so reducing the resource consumption on the path towards itself. For example, if a layered media encoding scheme such as H.264 SVC is used, the receiver may change its layer subscription, and so reduce the bit rate it receives. The receiver MUST still send RTCP XR ECN Summary to the sender, even if it can adapt without contact with the sender, so that the sender can determine if ECN is supported on the network path. The timeliness of RTCP feedback is less of a concern with receiver driven congestion control, and regular RTCP reporting of ECN summary information is sufficient (without using RTP/AVPF immediate or early feedback).

Hybrid: There might be mechanisms that utilize both some receiver behaviors and some sender side monitoring, thus requiring both feedback of congestion events to the sender and taking receiver decisions and possible signalling to the sender. In this case the congestion control algorithm needs to use the signalling to indicate which features of ECN for RTP are required.

Responding to congestion indication in the case of multicast traffic is a more complex problem than for unicast traffic. The fundamental problem is diverse paths, i.e., when different receivers don't see the same path, and thus have different bottlenecks, so the receivers may get ECN-CE marked packets due to congestion at different points in the network. This is problematic for sender driven congestion control, since when receivers are heterogeneous in regards to capacity, the sender is limited to transmitting at the rate the slowest receiver can support. This often becomes a significant limitation as group size grows. Also, as group size increases the frequency of reports from each receiver decreases, which further reduces the responsiveness of the mechanism. Receiver-driven congestion control has the advantage that each receiver can choose the appropriate rate for its network path, rather than all receivers having to settle for the lowest common rate.

We note that ECN support is not a silver bullet to improving performance. The use of ECN gives the chance to respond to congestion before packets are dropped in the network, improving the user experience by allowing the RTP application to control how the quality is reduced. An application which ignores ECN Congestion Experienced feedback is not immune to congestion: the network will eventually begin to discard packets if traffic doesn't respond. It is in the best interest of an application to respond to ECN congestion feedback promptly, to avoid packet loss.

7.4. Detecting Failures

Senders and receivers can deliberately ignore ECN-CE and thus get a benefit over behaving flows (cheating). The ECN Nonce [RFC3540] is an addition to TCP that attempts to solve this issue as long as the sender acts on behalf of the network. The assumption about the senders acting on the behalf of the network may be reduced due to the nature of peer-to-peer use of RTP. Still a significant portion of RTP senders are infrastructure devices (for example, streaming media servers) that do have an interest in protecting both service quality and the network. Even though there may be cases where nonce can be applicable also for RTP, it is not included in this specification. This as a receiver interested in cheating would simply claim to not support nonce, or even ECN itself. It is, however, worth mentioning that, as real-time media is commonly sensitive to increased delay and packet loss, it will be in both the media sender and receivers interest to minimise the number and duration of any congestion events as they will adversely affect media quality.

RTP sessions can also suffer from path changes resulting in a non-ECN compliant node becoming part of the path. That node may perform either of two actions that has effect on the ECN and application functionality. The gravest is if the node drops packets with the ECN field set to ECT(0), ECT(1), or ECN-CE. This can be detected by the receiver when it receives an RTCP SR packet indicating that a sender has sent a number of packets that it has not received. The sender may also detect such a middlebox based on the receiver's RTCP RR packet, when the extended sequence number is not advanced due to the failure to receive packets. If the packet loss is less than 100%, then packet loss reporting in either the ECN feedback information or RTCP RR will indicate the situation. The other action is to re-mark a packet from ECT or ECN-CE to not-ECT. That has less dire results, however it should be detected so that ECN usage can be suspended to prevent misusing the network.

The RTCP XR ECN summary packet and the ECN feedback packet allow the sender to compare the number of ECT marked packets of different types received with the number it actually sent. The number of ECT packets received, plus the number of ECN-CE marked and lost packets, should correspond to the number of sent ECT marked packets plus the number of received duplicates. If these numbers doesn't agree there are two likely reasons, a translator changing the stream or not carrying the ECN markings forward, or that some node re-marks the packets. In both cases the usage of ECN is broken on the path. By tracking all the different possible ECN field values a sender can quickly detect if some non-compliant behavior is happening on the path.

Thus packet losses and non-matching ECN field value statistics are

possible indication of issues with using ECN over the path. The next section defines both sender and receiver reactions to these cases.

7.4.1. Fallback mechanisms

Upon the detection of a potential failure, both the sender and the receiver can react to mitigate the situation.

A receiver that detects a packet loss burst MAY schedule an early feedback packet that includes at least the RTCP RR and the ECN feedback message to report this to the sender. This will speed up the detection of the loss at the sender, thus triggering sender side mitigation.

A sender that detects high packet loss rates for ECT-marked packets SHOULD immediately switch to sending packets as not-ECT to determine if the losses are potentially due to the ECT markings. If the losses disappear when the ECT-marking is discontinued, the RTP sender should go back to initiation procedures to attempt to verify the apparent loss of ECN capability of the used path. If a re-initiation fails then the two possible actions exist:

1. Periodically retry the ECN initiation to detect if a path change occurs to a path that is ECN capable.
2. Renegotiate the session to disable ECN support. This is a choice that is suitable if the impact of ECT probing on the media quality is noticeable. If multiple initiations have been successful, but the following full usage of ECN has resulted in the fallback procedures, then disabling of the ECN support is RECOMMENDED.

We foresee the possibility of flapping ECN capability due to several reasons: video switching MCU or similar middleboxes that selects to deliver media from the sender only intermittently; load balancing devices may in worst case result in that some packets take a different network path than the others; mobility solutions that switch underlying network path in a transparent way for the sender or receiver; and membership changes in a multicast group. It is however appropriate to mention that there are also issues such as re-routing of traffic due to a flappy route table or excessive reordering and other issues that are not directly ECN related but nevertheless may cause problems for ECN.

7.4.2. Interpretation of ECN Summary information

This section contains discussion on how the ECN summary report information can be used to detect various types of ECN path issues.

We first review the information the RTCP reports provide on a per source (SSRC) basis:

ECN-CE Counter: The number of RTP packets received so far in the session with an ECN field set to CE.

ECT (0/1) Counters: The number of RTP packets received so far in the session with an ECN field set to ECT (0) and ECT (1) respectively.

not-ECT Counter: The number of RTP packets received so far in the session with an ECN field set to not-ECT.

Lost Packets counter: The number of RTP packets that were expected based on sequence numbers but never received.

Duplication Counter: The number of received RTP packets that are duplicates of already received ones.

Extended Highest Sequence number: The highest sequence number seen when sending this report, but with additional bits, to handle disambiguation when wrapping the RTP sequence number field.

The counters will be initialised to zero to provide value for the RTP stream sender from the first report. After the first report, the changes between the last received report and the previous report are determined by simply taking the values of the latest minus the previous, taking wrapping into account. This definition is also robust to packet losses, since if one report is missing, the reporting interval becomes longer, but is otherwise equally valid.

In a perfect world, the number of not-ECT packets received should be equal to the number sent minus the lost packets counter, and the sum of the ECT(0), ECT(1), and ECN-CE counters should be equal to the number of ECT marked packet sent. Two issues may cause a mismatch in these statistics: severe network congestion or unresponsive congestion control might cause some ECT-marked packets to be lost, and packet duplication might result in some packets being received, and counted in the statistics, multiple times (potentially with a different ECN-mark on each copy of the duplicate).

The rate of packet duplication is tracked, allowing one to take the duplication into account. The value of the ECN field for duplicates will also be counted and when comparing the figures one needs to take some fraction of packet duplicates that are non-ECT and some fraction of packet duplicates being ECT into account into the calculation. Thus when only sending non-ECT then the number of sent packets plus reported duplicates equals the number of received non-ECT. When sending only ECT then number of sent ECT packets plus duplicates will

equal ECT(0), ECT(1), ECN-CE and packet loss. When sending a mix of non-ECT and ECT then there is an uncertainty if any duplicate or packet loss was an non-ECT or ECT. If the packet duplication is completely independent of the usage of ECN, then the fraction of packet duplicates should be in relation to the number of non-ECT vs ECT packet sent during the period of comparison. This relation does not hold for packet loss, where higher rates of packet loss for non-ECT is expected than for ECT traffic.

Detecting clearing of ECN field: If the ratio between ECT and not-ECT transmitted in the reports has become all not-ECT, or has substantially changed towards not-ECT, then this is clearly an indication that the path results in clearing of the ECT field.

Dropping of ECT packets: To determine if the packet drop ratio is different between not-ECT and ECT marked transmission requires a mix of transmitted traffic. The sender should compare if the delivery percentage (delivered / transmitted) between ECT and not-ECT is significantly different. Care must be taken if the number of packets are low in either of the categories. One must also take into account the level of CE marking. A CE marked packet would have been dropped unless it was ECT marked. Thus, the packet loss level for not-ECT should be approximately equal to the loss rate for ECT when counting the CE marked packets as lost ones. A sender performing this calculation needs to ensure that the difference is statistically significant.

If erroneous behavior is detected, it should be logged to enable follow up and statistics gathering.

8. Processing ECN in RTP Translators and Mixers

RTP translators and mixers that support ECN for RTP are required to process, and potentially modify or generate ECN marking in RTP packets. They also need to process, and potentially modify or generate RTCP ECN feedback packets for the translated and/or mixed streams. This includes both downstream RTCP reports generated by the media sender, and also reports generated by the receivers, flowing upstream back towards the sender.

8.1. Transport Translators

Some translators only perform transport level translations, like copying packets from one address domain, like unicast to multicast. It may also perform relaying like copying an incoming packet to a number of unicast receivers. This section details the ECN related actions for RTP and RTCP.

For the RTP data packets the translator, which does not modify the media stream, SHOULD copy the ECN bits unchanged from the incoming to the outgoing datagrams, unless the translator itself is overloaded and experiencing congestion, in which case it may mark the outgoing datagrams with an ECN-CE mark.

A Transport translator does not modify RTCP packets. It however MUST perform the corresponding transport translation of the RTCP packets as it does with RTP packets being sent from the same source/end-point.

8.2. Fragmentation and Reassembly in Translators

An RTP translator may fragment or reassemble RTP data packets without changing the media encoding, and without reference to the congestion state of the networks it bridges. An example of this might be to combine packets of a voice-over-IP stream coded with one 20ms frame per RTP packet into new RTP packets with two 20ms frames per packet, thereby reducing the header overheads and so stream bandwidth, at the expense of an increase in latency. If multiple data packets are re-encoded into one, or vice versa, the RTP translator MUST assign new sequence numbers to the outgoing packets. Losses in the incoming RTP packet stream may also induce corresponding gaps in the outgoing RTP sequence numbers. An RTP translator MUST rewrite RTCP packets to make the corresponding changes to their sequence numbers, and to reflect the impact of the fragmentation or reassembly. This section describes how that rewriting is to be done for RTCP ECN feedback packets. Section 7.2 of [RFC3550] describes general procedures for other RTCP packet types.

The processing of arriving RTP packets for this case is as follows. If an ECN marked packet is split into two, then both the outgoing packets MUST be ECN marked identically to the original; if several ECN marked packets are combined into one, the outgoing packet MUST be either ECN-CE marked or dropped if any of the incoming packets are ECN-CE marked. If the outgoing combined packet is not ECN-CE marked, then it MUST be ECT marked if any of the incoming packets were ECT marked.

RTCP ECN feedback packets (Section 5.1) contain seven fields that are rewritten in an RTP translator that fragments or reassembles packets: the extended highest sequence number, the duplication counter, the lost packets counter, the ECN-CE counter, and not-ECT counter, the ECT(0) counter, and the ECT(1) counter. The RTCP XR report block for ECN summary information (Section 5.2) includes all of these fields except the extended highest sequence number which is present in the report block in an SR or RR packet. The procedures for rewriting these fields are the same for both RTCP ECN feedback packet and the

RTCP XR ECN summary packet.

When receiving an RTCP ECN feedback packet for the translated stream, an RTP translator first determines the range of packets to which the report corresponds. The extended highest sequence number in the RTCP ECN feedback packet (or in the RTCP SR/RR packet contained within the compound packet, in the case of RTCP XR ECN summary reports) specifies the end sequence number of the range. For the first RTCP ECN feedback packet received, the initial extended sequence number of the range may be determined by subtracting the sum of the lost packets counter, the ECN-CE counter, the not-ECT counter, the ECT(0) counter and the ECT(1) counter minus the duplication counter, from the extended highest sequence number. For subsequent RTCP ECN feedback packets, the starting sequence number may be determined as being one after the extended highest sequence number of the previous RTCP ECN feedback packet received from the same SSRC. These values are in the sequence number space of the translated packets.

Based on its knowledge of the translation process, the translator determines the sequence number range for the corresponding original, pre-translation, packets. The extended highest sequence number in the RTCP ECN feedback packet is rewritten to match the final sequence number in the pre-translation sequence number range.

The translator then determines the ratio, R , of the number of packets in the translated sequence number space (numTrans) to the number of packets in the pre-translation sequence number space (numOrig) such that $R = \text{numTrans} / \text{numOrig}$. The counter values in the RTCP ECN feedback report are then scaled by dividing each of them by R . For example, if the translation process combines two RTP packets into one, then numOrig will be twice numTrans , giving $R=0.5$, and the counters in the translated RTCP ECN feedback packet will be twice those in the original.

The ratio, R , may have a value that leads to non-integer multiples of the counters when translating the RTCP packet. For example, a VoIP translator that combines two adjacent RTP packets into one if they contain active speech data, but passes comfort noise packets unchanged, would have an R values of between 0.5 and 1.0 depending on the amount of active speech. Since the counter values in the translated RTCP report are integer values, rounding will be necessary in this case.

When rounding counter values in the translated RTCP packet, the translator should try to ensure that they sum to the number of RTP packets in the pre-translation sequence number space (numOrig). The translator should also try to ensure that no non-zero counter is rounded to a zero value, unless the pre-translated values are zero,

since that will lose information that a particular type of event has occurred. It is recognised that it may be impossible to satisfy both of these constraints; in such cases, it is better to ensure that no non-zero counter is mapped to a zero value, since this preserves congestion adaptation and helps the RTCP-based ECN initiation process.

One should be aware of the impact this type of translators have on the measurement of packet duplication. A translator performing aggregation and most likely also an fragmenting translator will suppress any duplication happening prior to itself. Thus the reports and what is being scaled will only represent packet duplication happening from the translator to the receiver reporting on the flow.

It should be noted that scaling the RTCP counter values in this way is meaningful only on the assumption that the level of congestion in the network is related to the number of packets being sent. This is likely to be a reasonable assumption in the type of environment where RTP translators that fragment or reassemble packets are deployed, as their entire purpose is to change the number of packets being sent to adapt to known limitations of the network, but is not necessarily valid in general.

The rewritten RTCP ECN feedback report is sent from the other side of the translator to that which it arrived (as part of a compound RTCP packet containing other translated RTCP packets, where appropriate).

8.3. Generating RTCP ECN Feedback in Media Transcoders

An RTP translator that acts as a media transcoder cannot directly forward RTCP packets corresponding to the transcoded stream, since those packets will relate to the non-transcoded stream, and will not be useful in relation to the transcoded RTP flow. Such a transcoder will need to interpose itself into the RTCP flow, acting as a proxy for the receiver to generate RTCP feedback in the direction of the sender relating to the pre-transcoded stream, and acting in place of the sender to generate RTCP relating to the transcoded stream, to be sent towards the receiver. This section describes how this proxying is to be done for RTCP ECN feedback packets. Section 7.2 of [RFC3550] describes general procedures for other RTCP packet types.

An RTP translator acting as a media transcoder in this manner does not have its own SSRC, and hence is not visible to other entities at the RTP layer. RTCP ECN feedback packets and RTCP XR report blocks for ECN summary information that are received from downstream relate to the translated stream, and so must be processed by the translator as if it were the original media source. These reports drive the congestion control loop and media adaptation between the translator

and the downstream receiver. If there are multiple downstream receivers, a logically separate transcoder instance must be used for each receiver, and must process RTCP ECN feedback and summary reports independently to the other transcoder instances. An RTP translator acting as a media transcoder in this manner MUST NOT forward RTCP ECN feedback packets or RTCP XR ECN summary reports from downstream receivers in the upstream direction.

An RTP translator acting as a media transcoder will generate RTCP reports upstream towards the original media sender, based on the reception quality of the original media stream at the translator. The translator will run a separate congestion control loop and media adaptation between itself and the media sender for each of its downstream receivers, and must generate RTCP ECN feedback packets and RTCP XR ECN summary reports for that congestion control loop using the SSRC of that downstream receiver.

8.4. Generating RTCP ECN Feedback in Mixers

An RTP mixer terminates one-or-more RTP flows, combines them into a single outgoing media stream, and transmits that new stream as a separate RTP flow. A mixer has its own SSRC, and is visible to other participants in the session at the RTP layer.

An ECN-aware RTP mixer must generate RTCP ECN feedback packets and RTCP XR report blocks for ECN summary information relating to the RTP flows it terminates, in exactly the same way it would if it were an RTP receiver. These reports form part of the congestion control loop between the mixer and the media senders generating the streams it is mixing. A separate control loop runs between each sender and the mixer.

An ECN-aware RTP mixer will negotiate and initiate the use of ECN on the mixed RTP flows it generates, and will accept and process RTCP ECN feedback reports and RTCP XR report blocks for ECN relating to those mixed flows as if it were a standard media sender. A congestion control loop runs between the mixer and its receivers, driven in part by the ECN reports received.

An RTP mixer MUST NOT forward RTCP ECN feedback packets or RTCP XR ECN summary reports from downstream receivers in the upstream direction.

9. Implementation considerations

To allow the use of ECN with RTP over UDP, an RTP implementation desiring to support receiving ECN controlled media streams must

support reading the value of the ECT bits on received UDP datagrams, and an RTP implementation desiring to support sending ECN controlled media streams must support setting the ECT bits in outgoing UDP datagrams. The standard Berkeley sockets API pre-dates the specification of ECN, and does not provide the functionality which is required for this mechanism to be used with UDP flows, making this specification difficult to implement portably.

10. IANA Considerations

Note to RFC Editor: please replace "RFC XXXX" below with the RFC number of this memo, and remove this note.

10.1. SDP Attribute Registration

Following the guidelines in [RFC4566], the IANA is requested to register one new SDP attribute:

- o Contact name, email address and telephone number: Authors of RFCXXXX
- o Attribute-name: ecn-capable-rtp
- o Type of attribute: media-level
- o Subject to charset: no

This attribute defines the ability to negotiate the use of ECT (ECN capable transport) for RTP flows running over UDP/IP. This attribute should be put in the SDP offer if the offering party wishes to receive an ECT flow. The answering party should include the attribute in the answer if it wish to receive an ECT flow. If the answerer does not include the attribute then ECT MUST be disabled in both directions.

10.2. RTP/AVPF Transport Layer Feedback Message

The IANA is requested to register one new RTP/AVPF Transport Layer Feedback Message in the table of FMT values for RTPFB Payload Types [RFC4585] as defined in Section 5.1:

Name:	RTCP-ECN-FB
Long name:	RTCP ECN Feedback
Value:	TBA1
Reference:	RFC XXXX

10.3. RTCP Feedback SDP Parameter

The IANA is requested to register one new SDP "rtcp-fb" attribute "nack" parameter "ecn" in the SDP ("ack" and "nack" Attribute Values) registry.

Value name:	ecn
Long name:	Explicit Congestion Notification
Usable with:	nack
Reference:	RFC XXXX

10.4. RTCP XR Report blocks

The IANA is requested to register one new RTCP XR Block Type as defined in Section 5.2:

Block Type:	TBA2
Name:	ECN Summary Report
Reference:	RFC XXXX

10.5. RTCP XR SDP Parameter

The IANA is requested to register one new RTCP XR SDP Parameter "ecn-sum" in the "RTCP XR SDP Parameters" registry.

Parameter name	XR block (block type and name)
-----	-----
ecn-sum	TBA2 ECN Summary Report Block

10.6. STUN attribute

A new STUN [RFC5389] attribute in the Comprehension-optional range under IETF Review (0x8000-0xFFFF) is request to be assigned to the STUN attribute defined in Section 7.2.2. The STUN attribute registry can currently be found at: <http://www.iana.org/assignments/stun-parameters/stun-parameters.xhtml>.

10.7. ICE Option

A new ICE option "rtp+ecn" is registered in the registry that "IANA Registry for Interactive Connectivity Establishment (ICE) Options" [RFC6336] creates.

11. Security Considerations

The use of ECN with RTP over UDP as specified in this document has the following known security issues that need to be considered.

External threats to the RTP and RTCP traffic:

Denial of Service affecting RTCP: An attacker that can modify the traffic between the media sender and a receiver can achieve either of two things: 1) Report a lot of packets as being Congestion Experience marked, thus forcing the sender into a congestion response; or 2) Ensure that the sender disable the usage of ECN by reporting failures to receive ECN by changing the counter fields. This can also be accomplished by injecting false RTCP packets to the media sender. Reporting a lot of ECN-CE marked traffic is likely the more efficient denial of service tool as that may likely force the application to use lowest possible bit-rates. The prevention against an external threat is to integrity protect the RTCP feedback information and authenticate the sender.

Information leakage: The ECN feedback mechanism exposes the receivers perceived packet loss, what packets it considers to be ECN-CE marked and its calculation of the ECN-none. This is mostly not considered as sensitive information. If it is considered sensitive the RTCP feedback should be encrypted.

Changing the ECN bits: An on-path attacker that sees the RTP packet flow from sender to receiver and who has the capability to change the packets can rewrite ECT into ECN-CE thus forcing the sender or receiver to take congestion control response. This denial of service against the media quality in the RTP session is impossible for an end-point to protect itself against. Only network infrastructure nodes can detect this illicit re-marking. It will be mitigated by turning off ECN, however, if the attacker can modify its response to drop packets the same vulnerability exist.

Denial of Service affecting the session set-up signalling: If an attacker can modify the session signalling it can prevent the usage of ECN by removing the signalling attributes used to indicate that the initiator is capable and willing to use ECN with RTP/UDP. This attack can be prevented by authentication and integrity protection of the signalling. We do note that any attacker that can modify the signalling has more interesting attacks they can perform than prevent the usage of ECN, like inserting itself as a middleman in the media flows enabling wire-tapping also for an off-path attacker.

The following are threats that exist from misbehaving senders or receivers:

Receivers cheating: A receiver may attempt to cheat and fail to report reception of ECN-CE marked packets. The benefit for a receiver cheating in its reporting would be to get an unfair bit-rate share across the resource bottleneck. It is far from certain that a receiver would be able to get a significant larger share of

the resources. That assumes a high enough level of aggregation that there are flows to acquire shares from. The risk of cheating is that failure to react to congestion results in packet loss and increased path delay.

Receivers misbehaving: A receiver may prevent the usage of ECN in an RTP session by reporting itself as non ECN capable, forcing the sender to turn off usage of ECN. In a point-to-point scenario there is little incentive to do this as it will only affect the receiver. Thus failing to utilise an optimisation. For multi-party session there exist some motivation why a receiver would misbehave as it can prevent also the other receivers from using ECN. As an insider into the session it is difficult to determine if a receiver is misbehaving or simply incapable, making it basically impossible in the incremental deployment phase of ECN for RTP usage to determine this. If additional information about the receivers and the network is known it might be possible to deduce that a receiver is misbehaving. If it can be determined that a receiver is misbehaving, the only response is to exclude it from the RTP session and ensure that it does not any longer have any valid security context to affect the session.

Misbehaving Senders: The enabling of ECN gives the media packets a higher degree of probability to reach the receiver compared to not-ECT marked ones on a ECN capable path. However, this is no magic bullet and failure to react to congestion will most likely only slightly delay a network buffer over-run, in which its session also will experience packet loss and increased delay. There is some possibility that the media senders traffic will push other traffic out of the way without being affected too negatively. However, we do note that a media sender still needs to implement congestion control functions to prevent the media from being badly affected by congestion events. Thus the misbehaving sender is getting a unfair share. This can only be detected and potentially prevented by network monitoring and administrative entities. See Section 7 of [RFC3168] for more discussion of this issue.

We note that the end-point security functions needed to prevent an external attacker from inferring with the signalling are source authentication and integrity protection. To prevent information leakage from the feedback packets encryption of the RTCP is also needed. For RTP there exist multiple solutions possible depending on the application context. Secure RTP (SRTP) [RFC3711] does satisfy the requirement to protect this mechanism despite only providing authentication if a entity is within the security context or not. IPsec [RFC4301] and DTLS [RFC4347] can also provide the necessary security functions.

The signalling protocols used to initiate an RTP session also need to be source authenticated and integrity protected to prevent an external attacker from modifying any signalling. Here an appropriate mechanism to protect the used signalling needs to be used. For SIP/SDP ideally S/MIME [RFC5751] would be used. However, with the limited deployment a minimal mitigation strategy is to require use of SIPS (SIP over TLS) [RFC3261] [RFC5630] to at least accomplish hop-by-hop protection.

We do note that certain mitigation methods will require network functions.

12. Examples of SDP Signalling

This section contain a few different examples of the signalling mechanism defined in this specification in an SDP context. If there are discrepancies between these examples and the specification text, the specification text is definitive.

12.1. Basic SDP Offer/Answer

This example is a basic offer/answer SDP exchange, assumed done by SIP (not shown). The intention is to establish a basic audio session point to point between two users.

The Offer:

```
v=0
o=jdoe 3502844782 3502844782 IN IP4 10.0.1.4
s=VoIP call
i=SDP offer for VoIP call with ICE and ECN for RTP
b=AS:128
b=RR:2000
b=RS:2500
a=ice-pwd:YH75Fviy6338Vbrhrlp8Yh
a=ice-ufrag:9uB6
a=ice-options:rtp+ecn
t=0 0
m=audio 45664 RTP/AVPF 97 98 99
c=IN IP4 192.0.2.3
a=rtpmap:97 G719/48000/1
a=fmtp:97 maxred=160
a=rtpmap:98 AMR-WB/16000/1
a=fmtp:98 octet-align=1; mode-change-capability=2
a=rtpmap:99 PCMA/8000/1
a=maxptime:160
a=ptime:20
a=ecn-capable-rtp: ice rtp ect=0 mode=setread
a=rtcp-fb:* nack ecn
a=rtcp-fb:* trr-int 1000
a=rtcp-xr:ecn-sum
a=rtcp-rsize
a=candidate:1 1 UDP 2130706431 10.0.1.4 8998 typ host
a=candidate:2 1 UDP 1694498815 192.0.2.3 45664 typ srflx raddr
10.0.1.4 rport 8998
```

This SDP offer offers a single media stream with 3 media payload types. It proposes to use ECN with RTP, with the ICE based initialization as being preferred over the RTP/RTCP one. Leap of faith is not suggested to be used. The offerer is capable of both setting and reading the ECN bits. In addition the use of both the RTCP ECN feedback packet and the RTCP XR ECN summary report are supported. ICE is also proposed with two candidates. It also supports reduced size RTCP and can to use it.

The Answer:

```
v=0
o=jdoe 3502844783 3502844783 IN IP4 198.51.100.235
s=VoIP call
i=SDP offer for VoIP call with ICE and ECN for RTP
b=AS:128
b=RR:2000
b=RS:2500
a=ice-pwd:asd88fgpdd777uzjYhagZg
a=ice-ufrag:8hhY
a=ice-options:rtp+ecn
t=0 0
m=audio 53879 RTP/AVPF 97 99
c=IN IP4 198.51.100.235
a=rtpmap:97 G719/48000/1
a=fmtp:97 maxred=160
a=rtpmap:99 PCMA/8000/1
a=maxptime:160
a=ptime:20
a=ecn-capable-rtp: ice ect=0 mode=readonly
a=rtcp-fb:* nack ecn
a=rtcp-fb:* trr-int 1000
a=rtcp-xr:ecn-sum
a=candidate:1 1 UDP 2130706431 198.51.100.235 53879 typ host
```

The answer confirms that only one media stream will be used. One RTP Payload type was removed. ECN capability was confirmed, and the initialization method will be ICE. However, the answerer is only capable of reading the ECN bits, which means that ECN can only be used for RTP flowing from the offerer to the answerer. ECT always set to 0 will be used in both directions. Both the RTCP ECN feedback packet and the RTCP XR ECN summary report will be used. Reduced size RTCP will not be used as the answerer has not indicated support for it in the answer.

12.2. Declarative Multicast SDP

The below session describes an any source multicast using session with a single media stream.

```
v=0
o=jdoe 3502844782 3502844782 IN IP4 198.51.100.235
s=Multicast SDP session using ECN for RTP
i=Multicasted audio chat using ECN for RTP
b=AS:128
t=3502892703 3502910700
m=audio 56144 RTP/AVPF 97
c=IN IP4 233.252.0.212/127
a=rtpmap:97 g719/48000/1
a=fmtp:97 maxred=160
a=maxptime:160
a=ptime:20
a=ecn-capable-rtp: rtp mode=readonly; ect=0
a=rtcp-fb:* nack ecn
a=rtcp-fb:* trr-int 1500
a=rtcp-xr:ecn-sum
```

In the above example, as this is declarative we need to require certain functionality. As it is ASM the initialization method that can work here is the RTP/RTCP based one. So that is indicated. The ECN setting and reading capability to take part of this session is at least read. If one is capable of setting that is good, but not required as one can skip using ECN for anything one sends oneself. The ECT value is recommended to be set to 0 always. The ECN usage in this session requires both ECN feedback and the XR ECN summary report, so their use is also indicated.

13. Acknowledgments

The authors wish to thank the following persons for their reviews and comments: Thomas Belling, Bob Briscoe, Roni Even, Kevin P. Flemming, Thomas Frankkila, Christian Groves, Christer Holmgren, Cullen Jennings Tom Van Caenegem, Simo Veikkolainen, Bill Ver Steeg, Dan Wing, Qin Wu, and Lei Zhu.

14. References

14.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC3611] Friedman, T., Caceres, R., and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, November 2003.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.
- [RFC5348] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 5348, September 2008.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.
- [RFC6336] Westerlund, M. and C. Perkins, "IANA Registry for Interactive Connectivity Establishment (ICE) Options", RFC 6336, July 2011.

14.2. Informative References

- [I-D.ietf-avt-rtp-no-op] Andreasen, F., "A No-Op Payload Format for RTP", draft-ietf-avt-rtp-no-op-04 (work in progress), May 2007.
- [RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, August 1989.
- [RFC2762] Rosenberg, J. and H. Schulzrinne, "Sampling of the Group Membership in RTP", RFC 2762, February 2000.
- [RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, October 2000.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.
- [RFC3540] Spring, N., Wetherall, D., and D. Ely, "Robust Explicit Congestion Notification (ECN) Signaling with Nonces", RFC 3540, June 2003.
- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.
- [RFC3569] Bhattacharyya, S., "An Overview of Source-Specific Multicast (SSM)", RFC 3569, July 2003.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.
- [RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, March 2006.
- [RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, April 2006.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.
- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.
- [RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, August 2006.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.
- [RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback

(RTP/SAVPF)", RFC 5124, February 2008.

- [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.
- [RFC5630] Audet, F., "The Use of the SIPS URI Scheme in the Session Initiation Protocol (SIP)", RFC 5630, October 2009.
- [RFC5751] Ramsdell, B. and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification", RFC 5751, January 2010.
- [RFC5760] Ott, J., Chesterfield, J., and E. Schooler, "RTP Control Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback", RFC 5760, February 2010.
- [RFC6189] Zimmermann, P., Johnston, A., and J. Callas, "ZRTP: Media Path Key Agreement for Unicast Secure RTP", RFC 6189, April 2011.

Authors' Addresses

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

Phone: +46 10 714 82 87
Email: magnus.westerlund@ericsson.com

Ingemar Johansson
Ericsson
Laboratoriegatan 11
SE-971 28 Lulea
SWEDEN

Phone: +46 73 0783289
Email: ingemar.s.johansson@ericsson.com

Colin Perkins
University of Glasgow
School of Computing Science
Glasgow G12 8QQ
United Kingdom

Email: csp@csp Perkins.org

Piers O'Hanlon
University College London
Computer Science Department
Gower Street
London WC1E 6BT
United Kingdom

Email: p.ohanlon@cs.ucl.ac.uk

Ken Carlberg
G11
1600 Clarendon Blvd
Arlington VA
USA

Email: carlberg@g11.org.uk

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 27, 2012

Q. Wu
F. Xia
R. Even
Huawei
October 25, 2011

RTCP Extension for Third-party Loss Report
draft-ietf-avtcore-feedback-supression-rtp-08

Abstract

In a large RTP session using the RTCP feedback mechanism defined in RFC 4585, a feedback target may experience transient overload if some event causes a large number of receivers to send feedback at once. This overload is usually avoided by ensuring that feedback reports are forwarded to all receivers, allowing them to avoid sending duplicate feedback reports. However, there are cases where it is not recommended to forward feedback reports, and this may allow feedback implosion. This memo discusses these cases and defines a new RTCP third-party loss report that can be used to inform receivers that the feedback target is aware of some loss event, allowing them to suppress feedback. Associated SDP signalling is also defined.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 27, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	4
3. Protocol Overview	4
4. Format of RTCP Feedback Messages	5
4.1. Transport Layer Feedback: Third-party Loss Report	6
4.2. Payload Specific Feedback: Third-party Loss Report	7
5. SDP Signaling	7
6. Example Use Cases	8
6.1. Source Specific Multicast (SSM) use case	8
6.2. Unicast based Rapid Acquisition of Multicast Stream (RAMS) use case	9
6.3. RTP Transport Translator use case	9
6.4. Multipoint Control Unit (MCU) use case	10
6.5. Mixer use case	10
7. Security Considerations	10
8. IANA Consideration	11
9. Acknowledgement	12
10. References	12
10.1. Normative References	12
10.2. Informative References	13
Appendix A. Change Log	13
A.1. draft-ietf-avtcore-feedback-suppression-rtp-01	14
A.2. draft-ietf-avtcore-feedback-suppression-rtp-02	14
A.3. draft-ietf-avtcore-feedback-suppression-rtp-03	14
A.4. draft-ietf-avtcore-feedback-suppression-rtp-04	15
A.5. draft-ietf-avtcore-feedback-suppression-rtp-05	15
A.6. draft-ietf-avtcore-feedback-suppression-rtp-06	15
A.7. draft-ietf-avtcore-feedback-suppression-rtp-07	16
A.8. draft-ietf-avtcore-feedback-suppression-rtp-08	16
Authors' Addresses	16

1. Introduction

RTCP feedback messages [RFC4585] allow the receivers in an RTP session to report events and ask for action from the media source (or a delegated feedback target when using unicast RTCP feedback with SSM [RFC5760]). There are cases where multiple receivers may initiate the same, or an equivalent message towards the same media source. When the receiver count is large, this behavior may cause transient overload of the media source, the network or both. This is known as a "feedback storm" or a "NACK storm". One common cause of such a feedback storm is receivers utilizing RTP retransmission [RFC4588] as a packet loss recovery technique, sending feedback using RTCP NACK messages [RFC4585] without proper dithering of the retransmission requests (e.g., not implementing the RFC 4585 dithering rules or sending NACKs to a middlebox that doesn't redistribute them to other receivers).

Another use case involves video Fast Update requests. A storm of these feedback messages can occur in conversational multimedia scenarios like multipoint video switching conference [RFC4587]. In this scenario, the receiver may lose synchronization with the video stream when speaker is changed in the middle of session. Poorly designed receivers that blindly issue fast update requests (i.e., Full Intra Request (FIR) described in [RFC5104]), can cause an implosion of FIR requests from receivers to the same media source.

RTCP feedback storms may cause short term overload, and in extreme cases to pose a possible risk of increasing network congestion on the control channel (e.g. RTCP feedback), the data channel, or both. It is therefore desirable to provide a way of suppressing unneeded feedback.

One approach to this, suggested in [DVB-IPTV], involves sending a NACK message to the other clients (or receiver) in the same group as the sender of NACK. However NACK is defined as a receiver report sent from a receiver observing a packet loss, therefore it only inform others that sender of NACK detected loss while the case where the sender of the feedback has received reports that the indicated packets were lost is not covered. This document specifies a new third-party loss report for this function. It supplements the existing the use of RTCP NACK packet and further is more precise in the uses where the network is active to suppress feedback. It tells receivers explicitly that feedback for a particular packet or frame loss is not needed for a period of time and can provide an early indication before the receiver reacts to the loss and invokes its packet loss repair machinery. Section 6 provides some examples of when to send the Third Party Loss Report message.

2. Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Protocol Overview

This document extends the RTCP feedback messages defined in the Audio-Visual Profile with feedback (RTP/AVPF) [RFC4585] defining a Third Party Loss Report message. The Third Party Loss Report message can be used by the intermediaries to inform the receiver that the sender of the Third Party Loss Report has received reports that the indicated packets were lost, and asks the receiver not to send feedback to it regarding these packets.

RTCP Third Party Loss Report follows the similar format of message type as RTCP NACK or Full Intra Request Command. However, the Third Party Loss Report is defined as an indication that the sender of the feedback has received reports that the indicated packets were lost, while NACK [RFC4585] just indicates that the sender of the NACK observed that these packets were lost. The Third Party Loss Report (TPLR) message is generated by a RTP system that may not see the actual packet loss. It is sent following the same timing rule as sending NACK defined in [RFC4585]. The TPLR feedback message may be sent in a regular full compound RTCP packet or in an early RTCP packet, as per the RTP/AVPF rules. RTP Systems in the network that receive a Third Party Loss Report SHOULD NOT send their own additional Third Party Loss Report messages for the same packet sequence numbers. They should simply forward the Third Party Loss Report message received from upstream direction, additionally, they may generate their own Third Party Loss Report that reports a set of the losses they see, which are different from ones reported in the Third Party Loss report they received. The Third Party Loss Report does not have the retransmission request [RFC4588] semantics.

When a receiver gets a Third Party Loss Report message, it should start a timer for the retransmitted data packet and this message and refrain from sending a feedback request (e.g., NACK or FIR) for the missing packets reported in the message during the lifetime of the timer. If the sender of retransmitted packet is the media source, the timer value shall be based on the observed time difference between the round-trip time from the receiver to the original media source and the round-trip time from the receiver to the sender of the TPLR. A receiver should compute an estimate of the round-trip time (RTT) to the original media source or the sender of retransmitted data packet from Sender Report (SR) packets for the original stream,

or any other means. The round-trip time from the receiver to the sender of the TPLR can be calculated from RTCP report round-trip time if available, or any other means.

To increase the robustness to the loss of a TPLR or of a transmission RTP data packet, TPLR for the same Packet may be generated and sent out. The receiver should view the TPLR as a retransmission if this TPLR is received from the same media source after the timer set previously expires. In the case where the first TPLR is lost and the additional TPLR arrives at the receiver, the receiver should immediately refresh the timer to the same value as the previous timer it set for the the retransmitted data packet. When the timer expires and there is no retransmitted packet or a new Third Party Loss Report message, the receiver should take its normal behavior as if there is no current feedback suppression.

A receiver may still have sent a Feedback message according to the RTP/AVPF scheduling algorithm of [RFC4585] before receiving a Third Party Loss Report message, but further feedback messages for those sequence numbers SHOULD be suppressed for a period of time after receiving the TPLR. Nodes that do not understand the Third Party Loss Report message will ignore it, and might therefore still send feedback according to the AVPF scheduling algorithm of [RFC4585]. The media source or intermediate nodes cannot be certain that the use of a Third Party Loss Report message actually reduces the amount of feedback it receives.

Since Third Party Loss Report interacts strongly with repair timing, it has to work together with feedback to not adversely impact the repair of lost source packets. In order not to incur a lot of NACK requests due to additional TPLR described above, it is recommended that the RTP system sending TPLR should be implemented more closer to the media source. When the loss was detected and repair initiated much closer to the media source, the delay for the receiver to recover from packet loss can be reduced through the combination of intermediary feedback to the source and Third Party Loss Report downstream.

4. Format of RTCP Feedback Messages

This document registers two new RTCP Feedback messages for Third Party Loss Report. Applications that are employing one or more loss-repair methods MAY use the Third Party Loss Report together with their existing loss-repair methods either for every packet they expect to receive, or for an application-specific subset of the RTP packets in a session. In other words, receivers MAY ignore Third Party Loss Report messages, but SHOULD react to them unless they have

good reason to still send feedback messages despite having been requested to suppress them.

4.1. Transport Layer Feedback: Third-party Loss Report

This Third Party Loss Report message is an extension to the RTCP Transport Layer Feedback Report and identified by RTCP packet type value PT=RTPFB and FMT=TBD.

Within the common packet header for feedback messages (as defined in section 6.1 of [RFC4585]), the "SSRC of packet sender" field indicates the source of the request, and the "SSRC of media source" denotes the media sender of the flow for which the indicated losses are being suppressed .

The FCI field MUST contain one or more entries of transport layer third party loss Early Indication (TLLEI). Each entry applies to the same media source identified by the SSRC contained in the SSRC of media source field of Feedback header. The length of the TLLEI feedback message MUST be set to $2+1*N$, where N is the number of FCI entries.

The Feedback Control Information (FCI) for TLLEI uses the similar format of message Types defined in the section 6.2.1 of [RFC4585]. The format is shown in Figure 1.

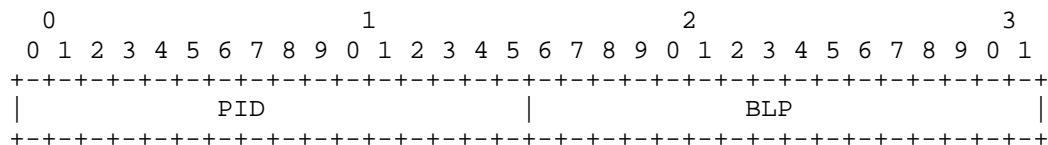


Figure 1: Syntax of an FCI Entry in the TLLEI Feedback Message

Packet ID (PID): 16 bits

The PID field is used to specify a lost packet. The PID field refers to the RTP sequence number of the lost packet.

bitmask of proceeding lost packets (BLP): 16 bits

The BLP allows for reporting losses of any of the 16 RTP packets immediately following the RTP packet indicated by the PID. The BLP's definition is identical to that given in [RFC4585].

4.2. Payload Specific Feedback: Third-party Loss Report

This message is an extension to the RTCP Payload Specific Feedback report and identified by RTCP packet type value PT=PSFB and FMT=TBD, which is used to suppress FIR [RFC5104] or PLI [RFC4585].

Within the common packet header for feedback messages (as defined in section 6.1 of [RFC4585]), the "SSRC of packet sender" field indicates the source of the request, and the "SSRC of media source" is not used and SHALL be set to 0. The SSRCS of the media senders to which this message applies are in the corresponding FCI entries.

The Feedback Control Information (FCI) for a Payload Specific Third Party Loss Early Indication (PSLEI) consists one or more FCI entries. Each entry applies to a different media Source, identified by its SSRC. the content of which is depicted in Figure 2. The length of the PSLEI feedback message MUST be set to $2+1*N$, where N is the number of FCI entries.

The format is shown in Figure 2.

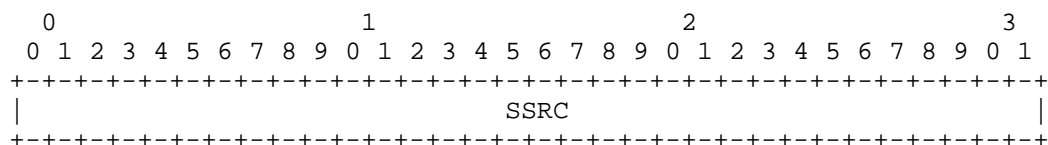


Figure 2: Syntax of an FCI Entry in the PSLEI Feedback Message

Synchronization source (SSRC):32 bits

The SSRC value of the media source that is requested to send a decoder refresh point or that is indicated that it lost synchronization with the video stream.

5. SDP Signaling

A new feedback value "tplr" needs to be defined for the Third Party Loss Report message to be used with Session Description Protocol (SDP) [RFC4566] using the Augmented Backus-Naur Form (ABNF) [RFC4585].

The "tplr" feedback value SHOULD be used with parameters that indicate the third party loss supported. In this document, we define two such parameter, namely:

- o "tllei" denotes support of transport layer third party loss early indication.
- o "pslei" denotes support of payload specific third party loss early indication.

In the ABNF [RFC5234] for rtcp-fb-val defined in [RFC4585], there is a placeholder called rtcp-fb-id to define new feedback types. "tplr" is defined as a new feedback type in this document, and the ABNF for the parameters for tplr is defined here (please refer to section 4.2 of [RFC4585] for complete ABNF syntax).

```
rtcp-fb-val          =/ "tplr" rtcp-fb-tplr-param
rtcp-fb-tplr-param = SP "tllei"
                    ;transport layer third party
                    ; loss early indication
                    / SP "pslei"
                    ;payload specific third party
                    ; loss early indication
                    / SP token [SP byte-string]
                    ; for future commands/indications
token =              <as defined in section 9 of [RFC4566]>
byte-string = <as defined in section 4.2 of [RFC4585] >
```

Refer to Section 4.2 of [RFC4585] for a detailed description and the full syntax of the "rtcp-fb" attribute.

6. Example Use Cases

The operation of feedback suppression is similar for all types of RTP sessions and topologies [RFC5117], however the exact messages used and the scenarios in which suppression is employed differ for various use cases. The following sections outline some of the intended use cases for using the Third Party Loss Report for feedback suppression and give an overview of the particular mechanisms.

6.1. Source Specific Multicast (SSM) use case

In SSM RTP sessions as described in [RFC5760], one or more Media Sources send RTP packets to a Distribution Source. The Distribution Source relays the RTP packets to the receivers using a source-specific multicast group.

As outlined in the [RFC5760], there are two Unicast Feedback models that may be used for reporting, the Simple Feedback model and the Distribution Source Feedback Summary Model. In the simple Feedback Model, there's no need for distribution source to create the Third

Party Loss Report, instead, NACKs are reflected by the distribution source to the other Receivers. However in the Distribution Source Feedback Summary model, the distribution source will not redistribute the NACK for some reason(e.g., to prevent revealing the identity or existence of a system sending NACK)and may send a Third Party Loss Report to the systems that were unable to receive the NACK, and won't receive the NACK via other means. since the summary feedback does not mandate the forwarding of NACK downstream. The Third Party Loss Report can be generated at the distribution source when downstream loss is told (e.g., downstream loss report is received), which indicates to the receivers that they should not transmit feedback messages for the same loss event for a certain time. Therefore the distribution source in the feedback summary model can be reasonably certain that it will help the situation by sending this Third Party Loss Report message to all the relevant receivers impacted by the packet loss.

6.2. Unicast based Rapid Acquisition of Multicast Stream (RAMS) use case

The typical RAMS architecture [RFC6285] may have several Burst/Retransmission Sources(BRS) behind the multicast source (MS) placed at the same level. These BRSSes will receive the primary multicast RTP stream from the media source and cache most recent packets after joining multicast session. If packet loss happens at the upstream of all the BRSSes or the downstream of BRSSes. One of the BRSSes or all the BRSSes may send a NACK or TPLR message to the DS, where the SSRC in this NACK or TPLR message is the BRS that is sending the message. The DS forwards/reflects this message down on the primary SSM. The details on how DS deal with this message is specified in [RETRANSMISSION-FOR-SSM].

6.3. RTP Transport Translator use case

A Transport Translator (Topo-Trn-Translator), as defined in [RFC5117] is typically forwarding the RTP and RTCP traffic between RTP clients, for example converting from multicast to unicast for domains that do not support multicast. The translator may suffer a loss of important video packets. In this case, the translator may forward TPLR message received from upstream in the same way as forwarding other RTCP traffic. If the translator acting as quality monitor [Monarch]is aware of packet loss, it may use the SSRC of monitor as packet sender SSRC to create NACK message and send it to the receivers that is not aware of packet loss.

6.4. Multipoint Control Unit (MCU) use case

When the speaker is changed in a voice-activated multipoint video switching conference [RFC4587], an RTP mixer can be used to select the available input streams and forward them to each participants. If the MCU is doing a blind switch without waiting for a synchronization point on the new stream it can send a FIR to the new video source. In this case the MCU should send a FIR suppression message to the new receivers. e.g., when the RTP Mixer starts to receive FIR from some participants it can suppress the remaining session participants from sending FIR by sending out a Third party Loss report message.

6.5. Mixer use case

A Mixer, in accordance with [RFC5117], aggregates multiple RTP streams from other session participants and generates a new RTP stream sent to the session participants. In some cases, the video frames may get badly screwed up between media source and the mixer. In such case, the mixer need to check if the packet loss will result in PLI or FIR transmissions from most of the group by analyzing the received video. If so the mixer may initiate FIR or PLI towards the media source on behalf of all the session participants and send out a Third party Loss report message to these session participants that may or are expected to send a PLI or FIR. Alternatively, when the mixer starts to receive FIR or PLI from some participants and like to suppress the remaining session participants from sending FIR or PLI by forwarding the FIR/PLI from one session participant to others.

7. Security Considerations

The defined messages have certain properties that have security implications. These must be addressed and taken into account by users of this protocol.

Spoofed or maliciously created feedback messages of the type defined in this specification can have the following implications:

Sending the spurious Third Party Loss Report (e.g., the Third Party Loss Report with the wrong sequence number of lost packet) that causes missing RTP packets to not be repaired in a timely fashion.

To prevent these attacks, there is a need to apply authentication and integrity protection of the feedback messages. This can be accomplished against threats external to the current RTP session using the RTP profile that combines Secure RTP [RFC3711] and AVPF into SAVPF [RFC5124].

Note that middleboxes that are not visible at the RTP layer that wish to send the Third Party Loss Reports on behalf of the media source can only do so if they spoof the SSRC of the media source. This is difficult in case SRTP is in use. If the middlebox is visible at the RTP layer, this is not an issue, provided the middlebox is part of the security context for the session.

Also note that endpoints that receive a Third Party Loss Report would be well-advised to ignore it, unless the security is in place to authenticate the sender of the Third Party Loss Report. Accepting Third Party Loss Report from un-authenticated sender can lead to a denial of service attack, where the endpoint accepts poor quality media that could be repaired.

8. IANA Consideration

The new value "TPLR" has been registered with IANA in the "rtcp-fb" Attribute Values registry located at the time of publication at:
<http://www.iana.org/assignments/sdp-parameters>

Value name:	tplr
Long Name:	Third Party Loss Reports
Reference:	This document

A new registry "Third Party Loss Report Messages" has been created to hold "tplr" parameters located at time of publication at:
<http://www.iana.org/assignments/sdp-parameters>

New registration in this registry follows the "Specification required" policy as defined by [RFC5226]. In addition, they are required to indicate any additional RTCP feedback types, such as "nack" and "ack".

The following value have been registered as one FMT value in the "FMT Values for RTPFB Payload Types" registry located at the time of publication at: <http://www.iana.org/assignments/rtp-parameters>

RTPFB range			
Name	Long Name	Value	Reference
-----	-----	-----	-----
TLLEI	Transport Layer Third Party Loss Early Indication	TBA1	[RFCXXXX]

The following value have been registered as one FMT value in the "FMT Values for PSFB Payload Types" registry located at the time of publication at: <http://www.iana.org/assignments/rtp-parameters>

PSFB range Name	Long Name	Value	Reference
PSLEI	Payload Specific Third Party Loss Early Indication	TBA2	[RFCXXXX]

9. Acknowledgement

The authors would like to thank David R Oran, Magnus Westerlund, Colin Perkins, Ali C. Begen, Tom VAN CAENEGEM, Ingemar Johansson S, Bill Ver Steeg, Jonathan Lennox, WeeSan Lee for their valuable comments and suggestions on this document.

10. References

10.1. Normative References

- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.
- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, February 2008.

10.2. Informative References

- [RFC5740] Adamson, B., Bormann, C., Handley, M., and J. Macker, "NACK-Oriented Reliable Multicast (NORM) Transport Protocol", November 2009.
- [DVB-IPTV] ETSI Standard, "Digital Video Broadcasting(DVB); Transport of MPEG-2 TS Based DVB Services over IP Based Networks", ETSI TS 102 034, V1.4.1 , August 2009.
- [RFC6285] Steeg, B., Begen, A., Caenagem, T., and Z. Vax, "Unicast-Based Rapid Acquisition of Multicast RTP Sessions", June 2011.
- [Monarch] Wu, Q., Hunt, G., and P. Arden, "Monitoring Architectures for RTP", June 2011.
- [RETRANSMISSION-FOR-SSM] Caenagem, T., Steeg, B., and A. Begen, "Retransmission for Source-Specific Multicast (SSM) Sessions", May 2011.
- [RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, January 2008.
- [RFC4587] Even, R., "RTP Payload Format for H.261 Video Streams", RFC 4587, August 2006.
- [RFC5760] Ott, J., Chesterfield, J., and E. Schooler, "RTP Control Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback", RFC 5760, February 2010.
- [RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, February 2008.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

Appendix A. Change Log

Note to the RFC-Editor: please remove this section prior to publication as an RFC.

A.1. draft-ietf-avtcore-feedback-suppression-rtp-01

The following are the major changes compared to previous version:

- o Remove the merge report from SSM use case and additional text to address report merging issue.
- o Revise section 3 and section 6 to address FEC packet dealing issue and Leave how to repair packet loss beyond the scope.
- o Modify the SSM use case and RAMS use case to focus on uses.
- o Other Editorial changes.

A.2. draft-ietf-avtcore-feedback-suppression-rtp-02

The following are the major changes compared to previous version:

- o In Section 4.1, fix typo: Section 4.3.1.1 of section [RFC5104]-> section 6.2.1 of [RFC4585].
- o In Section 3: Clarify how to deal with downstream loss using Third party loss report and upstream loss using NACK.
- o Update title and abstract to focus on third party loss report.
- o In Section 6.1: Update this section to explain how third party loss report is used to deal with downstream loss.
- o In section 6.1.2: Update this section to explain how third party loss report is used to deal with downstream loss.
- o In section 6.2: Rephrase the text to discuss how BRS deal with the third party loss report.

A.3. draft-ietf-avtcore-feedback-suppression-rtp-03

The following are the major changes compared to previous version:

- o In Appendix A, fix typo: Appendix A. Appendix A. -> Appendix A.
- o Update abstract to clarify when third-party loss reports should be sent instead of NACKs.
- o Update section 3 Paragraph 2 to differentiate when a third-party loss report should be used compared to a NACK.

- o Update section 3 Paragraph 3 to explain when media source to send a third-party loss.
- o Move specific rules for section 6.1.1 and section 6.1.2 to section 6.1 as generic rules and delete section 6.1.1.

A.4. draft-ietf-avtcore-feedback-suppression-rtp-04

The following are the major changes compared to previous version:

- o Reference Update.
- o Clarify the use of the third party loss report in section 3 and section 6.1.1.

A.5. draft-ietf-avtcore-feedback-suppression-rtp-05

The following are the major changes compared to previous version:

- o Remove 3rd and 4th paragraphs of section 6.1 and replaced them with 2nd and 3rd paragraphs of section 3.
- o Remove section 6.1.1.1.
- o Revise the last paragraph of section 1 to clarify the rationale of using new message.
- o Update RTP transport translator case in section 6.3 to correct the use of the third party loss report.
- o Update MCU case in section 6.4 to correct the use of the third party loss report.
- o Revise SSM use case to address multiple DS issue.
- o References Update.
- o Move one rationale on preventing sending unicast NACK in introduction section to SSM case section.
- o Other Editorial changes to section 6.1, 6.1.1, 6.2.

A.6. draft-ietf-avtcore-feedback-suppression-rtp-06

The following are the major changes compared to previous version:

- o A few Editorial changes to the whole document.

A.7. draft-ietf-avtcore-feedback-suppression-rtp-07

The following are the major changes compared to previous version:

- o Restructuring the protocol overview section to clarify the round trip time calculation and receiver behavior to the additional TPLR.
- o Restructuring the SSM use case section to focus on the use of TPLR.
- o Editorial changes to the abstract, introduction, message format, use cases and IANA sections.
- o References update

A.8. draft-ietf-avtcore-feedback-suppression-rtp-08

The following are the major changes compared to previous version:

- o Clarify which RTT is used and how timer is refreshed in the section 3.
- o Editorial changes to the Introduction, Protocol Overview, SDP Signaling, Message Format, Use case, Security Consideration and IANA sections.
- o Remove Seq Nr field in the figure 2 for payload specific feedback.
- o References reorganizing.

Authors' Addresses

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: sunseawq@huawei.com

Frank Xia
Huawei
1700 Alma Dr. Suite 500
Plano, TX 75075
USA

Phone: +1 972-509-5599
Email: xiayangsong@huawei.com

Roni Even
Huawei
14 David Hamelech
Tel Aviv 64953
Israel

Email: even.roni@huawei.com

AVTCore
Internet Draft
Intended status: Standards Track
Expires: May 3, 2012

R. van Brandenburg
H. Stokking
M. van Deventer
TNO Netherlands
F. Boronat
M. Montagud
Universidad Politecnica de Valencia
Kevin Gross
AVA Networks
October 31, 2011

RTCP for inter-destination media synchronization
draft-ietf-avtcare-idms-02.txt

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

This Internet-Draft will expire on May 3, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Abstract

This document gives information on an RTCP Packet Type and RTCP XR Block Type including associated SDP parameters for inter-destination media synchronization (IDMS). The RTCP XR Block Type, registered with IANA based on an ETSI specification, is used to collect media play-out information from participants in a group playing-out (watching, listening, etc.) a specific RTP media stream. The RTCP packet type specified by this document is used to distribute a summary of the collected information so that the participants can synchronize play-out.

Typical applications for IDMS are social TV, shared service control (i.e. applications where two or more geographically separated users are watching a media stream together), distance learning, networked video walls, networked speakers, etc.

Table of Contents

1. Introduction	3
1.1. Inter-destination Media Synchronization	3
1.2. Applicability of RTCP to IDMS	3
1.3. Applicability of SDP to IDMS	4
1.4. This document and ETSI TISPAN	4
2. Terminology	4
3. Overview of IDMS operation	4
4. Inter-destination media synchronization use cases	6
5. Architecture for inter-destination media synchronization	7
5.1. Media Synchronization Application Server (MSAS)	7
5.2. Synchronization Client (SC)	8
5.3. Communication between MSAS and SCs	8
6. RTCP XR Block Type for IDMS	9
7. RTCP Packet Type for IDMS (IDMS report)	11
8. Timing and NTP Considerations	13
8.1. Leap Seconds	14
9. SDP Parameter for RTCP XR IDMS Block Type	15
10. SDP Parameter for RTCP IDMS Packet Type	15
11. SDP parameter for clock source	16
12. Compatibility with ETSI TISPAN	18
13. On the use of presentation timestamps	19
14. Security Considerations	19
15. IANA Considerations	20

16. Contributors	21
17. Conclusions	21
18. References	22
18.1. Normative References	22
18.2. Informative References	22

1. Introduction

1.1. Inter-destination Media Synchronization

Inter-destination media synchronization (IDMS) refers to the play-out of media streams at two or more geographically distributed locations in a temporally synchronized manner. It can be applied to both unicast and multicast media streams and can be applied to any type and/or combination of streaming media, such as audio, video and text (subtitles). [Ishibashi2006] and [Boronat2009] provide an overview of technologies and algorithms for IDMS.

IDMS requires the exchange of information on media receipt and playout times. It may also require signaling for the initiation and maintenance of IDMS sessions and groups.

The presented RTCP specification for IDMS is independent of the used synchronization algorithm, which is out-of-scope of this document.

1.2. Applicability of RTCP to IDMS

Currently, most multimedia applications make use of RTP and RTCP [RFC3550]. RTP (Real-time Transport Protocol) provides end-to-end network transport functions suitable for applications requiring real-time data transport, such as audio, video or data, over multicast or unicast network services. The timestamps and sequence number mechanisms provided by RTP are very useful to reconstruct the original media timing, reorder and detect some packet loss at the receiver side.

The data transport is augmented by a control protocol (RTCP) to allow monitoring of the data delivery in a manner that is scalable to large multicast networks, and to provide minimal control and identification functionality.

RTP receivers and senders provide reception quality feedback by sending out RTCP Receiver Report (RR) and Sender Report (SR) packets [RFC3550] respectively, which may be augmented by eXtended Reports (XR) [RFC3611]. Thus, the feedback reporting features provided by RTCP make QoS monitoring possible and can be used for troubleshooting

and fault tolerance management in multicast distribution services such as IPTV.

These protocols are intended to be tailored through modification and/or additions in order to include profile-specific information required by particular applications, and the guidelines on doing so are specified in [RFC5968].

IDMS involves the collection, summarizing and distribution of RTP packet arrival and play-out times. As information on RTP packet arrival times and play-out times can be considered reception quality feedback information, RTCP becomes a promising candidate for carrying out IDMS, which may facilitate implementation in typical multimedia applications.

1.3. Applicability of SDP to IDMS

RTCP XR [RFC3611] defines the Extended Report (XR) packet type for the RTP Control Protocol (RTCP), and defines how the use of XR packets can be signaled by an application using the Session Description Protocol (SDP) [RFC4566].

SDP signaling is used to set up and maintain a synchronization group between Synchronization Clients (SCs). This document describes two SDP parameters for doing this, one for the RTCP XR block type and one for the new RTCP packet type.

This document also allows for a receiver to indicate a used clock source for synchronizing the receiver clock used in the IDMS session. This is also done using an SDP parameter, which is described in this document.

1.4. This document and ETSI TISPAN

ETSI TISPAN [TS 183 063] has specified architecture and protocol for IDMS using RTCP XR exchange and SDP signaling.

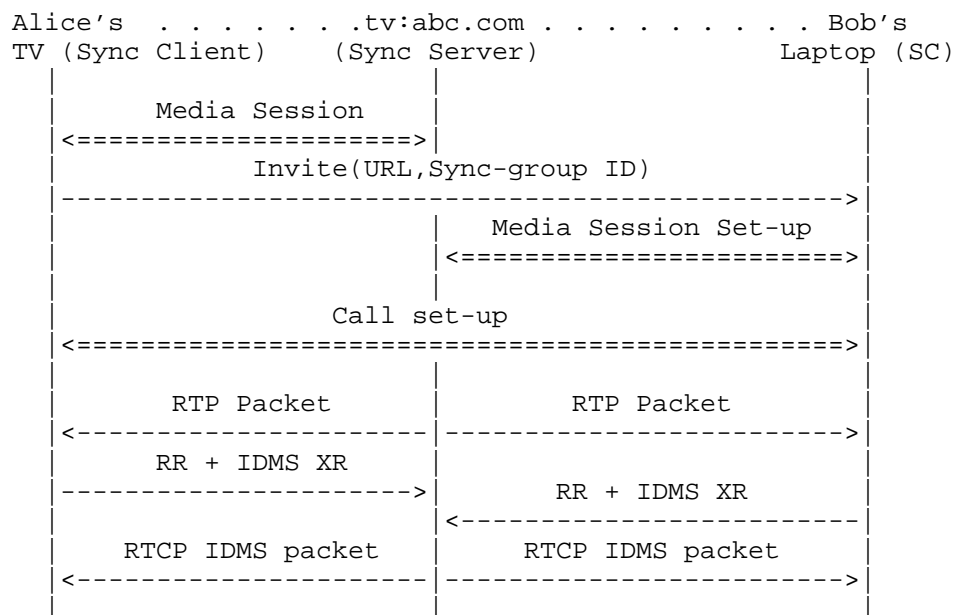
2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119] and indicate requirement levels for compliant implementations.

3. Overview of IDMS operation

This section provides a brief example of how the IDMS RTCP functionality is used. The section is tutorial in nature and does not

contain any normative statements.



Alice is watching TV in her living room. At some point she sees that a football game of Bob's favorite team is on. She sends him an invite to watch the program together. Embedded in the invitation is the link to the media server and a unique sync-group identifier.

Bob, who is also at home, receives the invite on his laptop. He accepts Alice's invitation and the RTP client on his laptop sets up a session to the media server. A VoIP connection to Alice's TV is also set up, so that Alice and Bob can talk while watching the program.

As is common with RTP, both the RTP client in Alice's TV as well as the one in Bob's laptop send periodic RTCP Receiver Reports (RR) to the media server. However, in order to make sure Alice and Bob see the events in the football game at the same time, their clients also periodically send an IDMS XR block to the sync server function of the media server. Included in the XR blocks are timestamps on when both Alice and Bob have received (or played out) a particular RTP packet.

The sync server function in the media server calculates a reference client from the received IDMS XR blocks (e.g. by selecting whichever client received the packet the latest as the reference client). It then sends an RTCP IDMS packet containing the play-out information of

this reference client to both Alice and Bob.

In this case Bob has the slowest connection and the reference client therefore includes a delay similar to the one experienced by Bob. Upon reception of this information, Alice's RTP client can choose what to do with this information. In this case it decreases its play-out rate temporarily until it matches with the reference client play-out (and thus matches Bob's play-out). Another option for Alice's TV would be to simply pause playback until it catches up. The exact implementation of the synchronization algorithm is up to the client.

Upon reception of the reference client RTCP IDMS packet, Bob's client does not have to do anything since it is already synchronized to the reference client (since it is based on Bob's delay). Note that other synchronization algorithms may introduce even more delay than the one experienced by the most delayed client, e.g. to account for delay variations, for new clients joining an existing synchronization group, etc.

4. Inter-destination media synchronization use cases

There are a large number of use cases imaginable in which IDMS might be useful. This section will highlight some of them. It should be noted that this section is in no way meant to be exhaustive

A first usage scenario for IDMS is Social TV. Social TV is the combination of media content consumption by two or more users at different devices and locations and real-time communication between those users. An example of Social TV, is when two or more users are watching the same television broadcast at different devices and locations, while communicating with each other using text, audio and/or video. A skew in the media play-out of the two or more users can have adverse effects on their experience. A well-known use case here is one friend experiencing a goal in a football match well before or after other friend(s). Thus IDMS is required to provide play-out synchronization.

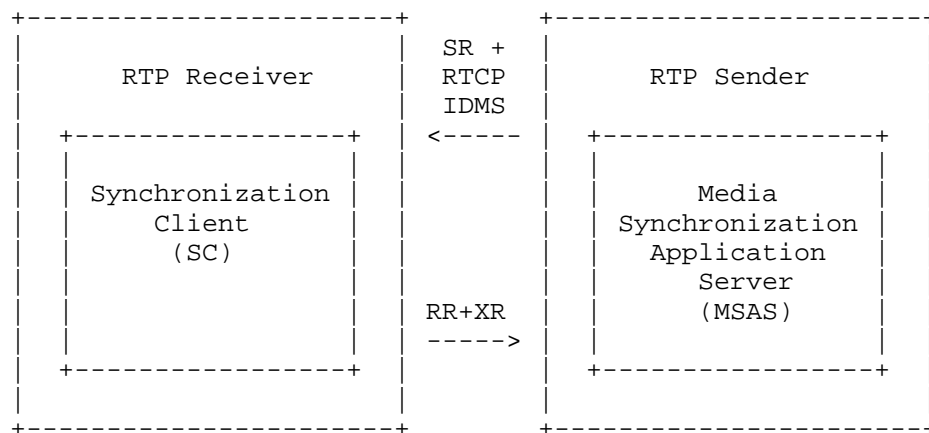
Another potential use case for IDMS is the video wall. A video wall consists of multiple computer monitors, video projectors, or television sets tiled together contiguously or overlapped in order to form one large screen. Each of the screens reproduces a portion of the larger picture. In some implementations, each screen may be individually connected to the network and receive its portion of the overall image from a network-connected video server or video scaler. Screens are refreshed at 60 hertz (every 16-2/3 milliseconds) or potentially faster. If the refresh is not synchronized, the effect of multiple screens acting as one is broken.

A third usage scenario is that of the networked loudspeakers, in which two or more speakers are connected to the network individually. Such situations can for example be found in large conference rooms, legislative chambers, classrooms (especially those supporting distance learning) and other large-scale environments such as stadiums. Since humans are more susceptible to differences in audio delay, this use case needs even more accuracy than the video wall use case. Depending on the exact application, the need for accuracy can then be in the range of microseconds.

5. Architecture for inter-destination media synchronization

The architecture for IDMS, which is based on a sync-maestro architecture [Boronat2009], is sketched below. The Synchronization Client (SC) and Media Synchronization Application Server (MSAS) entities are shown as additional functionality for the RTP receiver and sender respectively.

It should be noted that a master/slave type of architecture is also supported by having one of the SC devices also act as an MSAS. In this case the MSAS functionality is thus embedded in an RTP receiver instead of an RTP sender.



5.1. Media Synchronization Application Server (MSAS)

An MSAS collects RTP packet arrival times and play-out times from one or more SC(s) in a synchronization group. The MSAS summarizes and distributes this information to the SCs in the synchronization group as synchronization settings, e.g. by determining the SC with the most lagged play-out and using its reported RTP packet arrival time and play-out time as a summary.

5.2. Synchronization Client (SC)

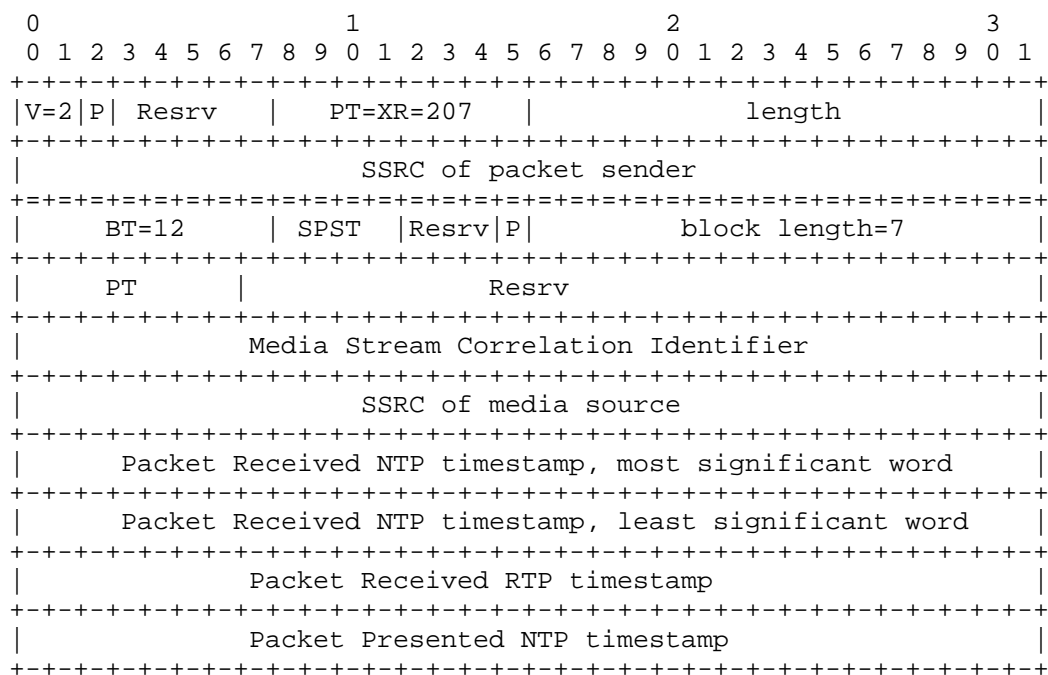
An SC reports RTP packet arrival times and play-out times of a media stream. It can receive summaries of such information, and use that to adjust its play-out buffer.

5.3. Communication between MSAS and SCs

Two different message types are used for the communication between MSAS and SCs. For the SC->MSAS message containing the play-out information of a particular client, an RTCP XR Block Type is used (see Section 6). For the MSAS->SC message containing the synchronization settings instructions, a new RTCP Packet Type is defined in Section 7.

6. RTCP XR Block Type for IDMS

This section describes the RTCP XR Block Type for reporting IDMS information on an RTP media stream. Its definition is based on [RFC3611]. The RTCP XR is used to provide feedback information on receipt times and presentation times of RTP packets to e.g. a Sender [RFC3611], a Feedback Target [RFC5576] or a Third Party Monitor [RFC3611].



The first 64 bits form the header of the RTCP XR, as defined in [RFC3611]. The SSRC of packet sender identifies the sender of the specific RTCP packet.

The IDMS report block consists of 7 32-bit words, with the following fields:

Block Type (BT): 8 bits. It identifies the block format. Its value SHALL be set to 12.

Synchronization Packet Sender Type (SPST): 4 bits. This field identifies the role of the packet sender for this specific eXtended Report. It can have the following values:

SPST=0 Reserved For future use.

SPST=1 The packet sender is an SC. It uses this XR to report synchronization status information. Timestamps relate to the SC input.

SPST=2 This setting is reserved in order to preserve compatibility with ETSI TISPAN [TS 183 063]. See section 12. for more information.

SPST=3-15 Reserved For future use.

Reserved bits (Resrv): 3 bits. These bits are reserved for future definition. In the absence of such a definition, the bits in this field MUST be set to zero and MUST be ignored by the receiver.

Packet Presented NTP timestamp flag (P): 1 bit. Bit set to 1 if the Packet Presented NTP timestamp field contains a value, 0 if it is empty. If this flag is set to zero, then the Packet Presented NTP timestamp shall not be inspected.

Block Length: 16 bits. This field indicates the length of the block in 32 bit words and shall be set to 7, as this RTCP Block Type has a fixed length.

Payload Type (PT): 7 bits. This field identifies the format of the media payload, according to [RFC3551]. The media payload is associated with an RTP timestamp clock rate. This clock rate provides the time base for the RTP timestamp counter.

Reserved bits (Resrv): 25 bits. These bits are reserved for future use and shall be set to 0.

Media Stream Correlation Identifier: 32 bits. This identifier is used to correlate synchronized media streams. The value 0 (all bits are set "0") indicates that this field is empty. The value $2^{32}-1$ (all bits are set "1") is reserved for future use. If the RTCP Packet Sender is an SC (SPST=1), then the Media Stream Correlation Identifier maps on the SyncGroupId to which the SC belongs.

SSRC: 32 bits. The SSRC of the media source shall be set to the value of the SSRC identifier carried in the RTP header [RFC3550] of the RTP packet to which the XR relates.

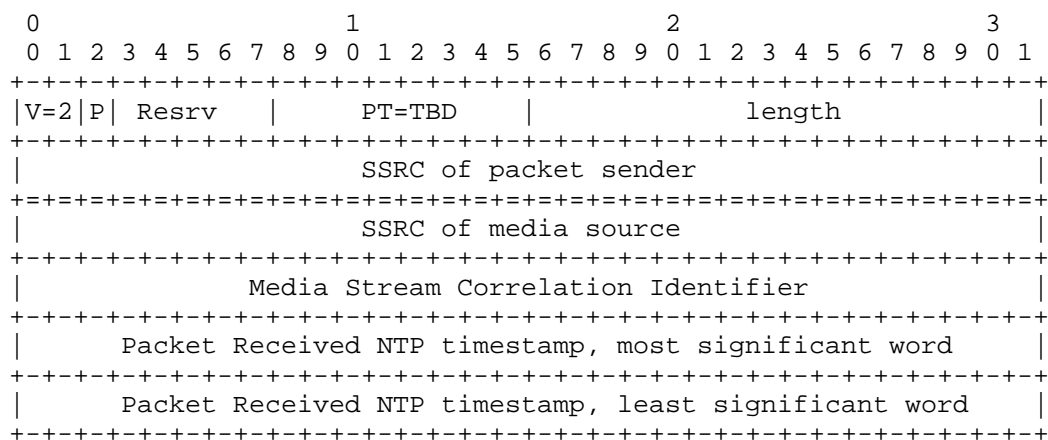
Packet Received NTP timestamp: 64 bits. This timestamp reflects the wall clock time at the moment of arrival of the first octet of the RTP packet to which the XR relates. It is formatted based on the NTP timestamp format as specified in [RFC5905]. See section 8 for more information on how this field is set.

Packet Received RTP timestamp: 32 bits. This timestamp has the value of the RTP time stamp carried in the RTP header [RFC3550] of the RTP packet to which the XR relates. Several consecutive RTP packets will have equal timestamps if they are (logically) generated at once, e.g., belong to the same video frame. It may well be the case that one receiver reports on the first RTP packet having a certain RTP timestamp and a second receiver reports on the last RTP packet having that same RTP timestamp. This would lead to an error in the synchronization algorithm due to the faulty interpretation of considering both reports to be on the same RTP packet. To solve this, an SC SHOULD report on RTP packets in which a certain RTP timestamp shows up for the first time.

Packet Presented NTP timestamp: 32 bits. This timestamp reflects the wall clock time at the moment the data contained in the first octet of the associated RTP packet is presented to the user. It is based on the time format used by NTP and consists of the least significant 16 bits of the NTP seconds part and the most significant 16 bits of the NTP fractional second part. If this field is empty, then it SHALL be set to 0 and the Packet Presented NTP timestamp flag (P) SHALL be set to 0. Presented here means the moment the data is presented to the user of the system, i.e. sound played out through speakers, video images being displayed on some display, etc. The accuracy resulting from the synchronization algorithm will only be as good as the accuracy with which the receivers can determine the delay between receiving packets and presenting them to the end-user.

7. RTCP Packet Type for IDMS (IDMS report)

This section specifies the RTCP Packet Type for indicating synchronization settings instructions to a receiver of the RTP media stream. Its definition is based on [RFC3550].




```

|          Packet Received RTP timestamp          |
+-----+
|          Packet Presented NTP timestamp, most significant word          |
+-----+
|          Packet Presented NTP timestamp, least significant word          |
+-----+

```

The first 64 bits form the header of the RTCP Packet Type, as defined in [RFC3550]. The SSRC of packet sender identifies the sender of the specific RTCP packet.

The RTCP IDMS packet consists of 7 32-bit words, with the following fields:

SSRC: 32 bits. The SSRC of the media source shall be set to the value of the SSRC identifier carried in the RTP header [RFC3550] of the RTP packet to which the RTCP IDMS packet relates.

Media Stream Correlation Identifier: 32 bits. This identifier is used to correlate synchronized media streams. The value 0 (all bits are set "0") indicates that this field is empty. The value $2^{32}-1$ (all bits are set "1") is reserved for future use. The Media Stream Correlation Identifier maps on the SyncGroupId of the group to which this packet is sent.

Packet Received NTP timestamp: 64 bits. This timestamp reflects the wall clock time at the reference client at the moment it received the first octet of the RTP packet to which this packet relates. It can be used by the synchronization algorithm on the receiving SC to set the required playout delay. The timestamp is formatted based on the NTP timestamp format as specified in [RFC5905]. See section 8 for more information on how this field is set.

Packet Received RTP timestamp: 32 bits. This timestamp has the value of the RTP time stamp carried in the RTP header [RFC3550] of the RTP packet to which the XR relates. This SHOULD relate to the first RTP packet containing this particular RTP timestamp, in case multiple RTP packets contain the same RTP timestamp.

Packet Presented NTP timestamp: 64 bits. This timestamp reflects the wall clock time at the reference client at the moment it presented the data contained in the first octet of the associated RTP packet to the user. The timestamp is formatted based on the NTP timestamp format as specified in [RFC5905]. If this field is empty, then it SHALL be set to 0. This field MAY be left empty if none or only one of the receivers reported on presentation timestamps. Presented here means the moment the data is presented to the user of the system.

In some use cases (e.g. phased array transducers), the level of control a MSAS might need to have over the exact moment of playout is so precise that a 32bit Presentation Timestamp might not suffice. For this reason, this RTCP Packet Type for IDMS includes a 64bit Presentation Timestamp field. Since an MSAS will in practice always add some extra delay to the delay reported by the most lagged receiver (to account for packet jitter), it suffices for the IDMS XR Block Type with which the SCs report on their playout to have a 32bit Presentation Timestamp field.

8. Timing and NTP Considerations

To achieve IDMS, the different receivers involved need synchronized clocks as a common timeline for synchronization. Depending on the synchronization accuracy required, different clock synchronization methods can be used. For social TV, synchronization accuracy should be achieved in order of hundreds of milliseconds. In that case, correct use of NTP on receivers will in most situations achieve the required accuracy. As a guideline, to deal with clock drift of receivers, receivers should synchronize their clocks at the beginning of a synchronized session. In case of high required accuracy, the synchronized clocks of different receivers should not drift beyond the accuracy required for the synchronization mechanism. In practice this can mean that receivers need to synchronize their clocks repeatedly during a synchronization session.

Because of the stringent synchronization requirements for achieving good audio, a high accuracy will be needed. In this case, NTP usage may not be sufficient. Either a local NTP server could be setup, or some other more accurate clock synchronization mechanism could be used, such as using GPS time or the Precision Time Protocol [IEEE-1588].

In this document, a new SDP parameter is introduced to signal the clock synchronization source or sources used or able to be used (see section 10). An SC can indicate which synchronization source is being used at the moment and the last time the SC synchronized with this source. An SC can also indicate any other synchronization sources available to it. This allows multiple SCs in an IDMS session to use the same or a similar clock synchronization source for their session.

Applications performing IDMS may or may not be able to choose a synchronization method for the system clock. How applications deal with this is up to the implementation. The application might control the system clock, or it might use a separate application clock or even a separate IDMS session clock. It might also report on the system clock and the synchronization method used, without being able to change it.

8.1. Leap Seconds

IDMS implementation is simplified by using a clock reference with a timescale which does not include leap seconds. IEEE 1588, GPS and other TAI (International Atomic Time) references do not include leap seconds. NTP time, operating system clocks and other UTC (Coordinated Universal Time) references include leap seconds (though the ITU is studying a proposal which could eventually eliminate leap seconds from UTC).

Leap seconds are potentially scheduled at the end of the last day of December and June each year. NTP inserts a leap second at the beginning of the last second of the day. This results in the clock freezing for one second immediately prior to the last second of the affected day. Most system clocks insert the leap second at the end of the last second. This results in repetition of the last second of the day. Generating or using timestamps during the entire last second of a day on which a leap second has been scheduled should therefore be avoided. Note that the period to be avoided has a real-time duration of two seconds.

It is also important that all participants correctly implement leap seconds and have a working communications channel to receive notification of leap second scheduling. Without prior knowledge of leap second schedule, NTP servers and clients may be offset by exactly one second with respect to their UTC reference. This potential discrepancy begins when a leap second occurs and ends when all participants receive a time update from a server or peer (which, depending on the operating system and/or implementation, could be anywhere from a few minutes to a week). Such a long-lived discrepancy can be particularly disruptive to RTP and IDMS operation.

Apart from the long-lived discrepancy due to dependence on both timing (e.g. NTP) updates and leap seconds scheduling updates, there is also the potential for a short-lived timing discontinuity having an effect on RTP and IDMS playout (even though leap seconds are quite rare).

If a timescale with leap seconds is used for IDMS:

- SCs must take care not to generate any IDMS XR reports in the immediate vicinity of the leap second. An MSAS must ignore any such reports that may be inadvertently generated.
- RTP Senders using a leap-second-bearing reference must not generate sender reports (SR) containing an originating NTP timestamp in the vicinity of a leap second. Receivers should ignore timestamps in any such reports inadvertently generated.

- Receivers working to a leap-second-bearing reference must be careful to take leap seconds into account if a leap second occurs between the time a RTP packet is originated and when it is to be presented.

9. SDP Parameter for RTCP XR IDMS Block Type

The SDP parameter `sync-group` is used to signal the use of the RTCP XR block for inter-destination media synchronization. It is also used to carry an identifier for the synchronization group to which clients belong or will belong. This SDP parameter extends `rtcp-xr-attr` as follows, using Augmented Backus-Naur Form [RFC5234].

```
rtcp-xr-attr = "a=" "rtcp-xr" ":" [xr-format *(SP xr-format)] CRLF
; Original definition from [RFC3611], section 5.1
```

```
xr-format = / grp-sync ; Extending xr-format for inter-destination
media synchronization
```

```
grp-sync = "grp-sync" [",sync-group=" SyncGroupId]
```

```
SyncGroupId = 1*DIGIT ; Numerical value from 0 till 4294967295
```

```
DIGIT = %x30-39
```

`SyncGroupId` is a 32-bit unsigned integer in network byte order and represented in decimal. `SyncGroupId` identifies a group of SCs for IDMS. It maps on the Media Stream Correlation Identifier as described in sections 6 and 7. The value `SyncGroupId=0` represents an empty `SyncGroupId`. The value 4294967295 ($2^{32}-1$) is reserved for future use.

The following is an example of the SDP attribute for IDMS

```
a=rtcp-xr:grp-sync,sync-group=42
```

10. SDP Parameter for RTCP IDMS Packet Type

The SDP parameter `rtcp-idms` is used to signal the use of the RTCP IDMS Packet Type for IDMS. It is also used to carry an identifier for the synchronization group to which clients belong or will belong. The SDP parameter is used as a media-level attribute during session setup. This SDP parameter is defined as follows, using Augmented Backus-Naur Form [RFC5234].

```
rtcp-idms = "a=" "rtcp-idms" ":" [sync-grp] CRLF
```

```
sync-grp = "sync-group=" SyncGroupId
```

SyncGroupId = 1*DIGIT ; Numerical value from 0 till 4294967295

DIGIT = %x30-39

SyncGroupId is a 32-bit unsigned integer in network byte order and represented in decimal. SyncGroupId identifies a group of SCs for IDMS. The value SyncGroupId=0 represents an empty SyncGroupId. The value 4294967295 ($2^{32}-1$) is reserved for future use.

The following is an example of the SDP attribute for IDMS.

a=rtcp-idms:sync-group=42

11. SDP parameter for clock source

The SDP parameter clocksource is used to signal the source for clock synchronization. This SDP parameter is specified as follows, using Augmented Backus-Naur Form [RFC5234].

clocksource = "a=" "clocksource" ":" source SP [last-synced] CRLF

source = local / ntp / gps / gal / ptp

local = "local"

ntp = "ntp" ["=" ntp-server]

ntp-server = host [":" port]

host = hostname / IPv4address / IPv6reference

hostname = *(domainlabel ".") toplabel ["."]

domainlabel = alphanum

/ alphanum *(alphanum / "-") alphanum

toplabel = ALPHA / ALPHA *(alphanum / "-") alphanum

IPv4address = 1*3DIGIT "." 1*3DIGIT "." 1*3DIGIT "." 1*3DIGIT

IPv6reference = "[" IPv6address "]"

IPv6address = hexpart [":" IPv4address]

hexpart = hexseq / hexseq "::" [hexseq] / "::" [hexseq]

hexseq = hex4 *(":" hex4)

hex4 = 1*4HEXDIG

port = 1*DIGIT

gps = "gps"

gal = "gal"

ptp = "ptp" SP ptp-version [":" ptp-id]

ptp-version = "IEEE 1588-2002" / "IEEE 1588-2008" / "IEEE 802.1AS-2011"

ptp-id = 1*alphanum

last-synced = date SP time UTCOffset

date = 2DIGIT "-" 2DIGIT "-" 4DIGIT
; day month year (e.g., 02-06-1982)

time = 2DIGIT ":" 2DIGIT ":" 2DIGIT "." 3DIGIT
; 00:00:00.000 - 23:59:59.999

UTCOffset = plusoffset / minusoffset

plusoffset = + 2DIGIT ":" 2DIGIT
; +01:00 (+HH:MM)

minusoffset = - 2DIGIT ":" 2DIGIT
; -00:00 (-HH:MM)

alphanum = ALPHA / DIGIT

EXAMPLE

a=clocksource:ntp=139.63.192.5:123 19-02-2011 21:03:20.345+01:00

A client MAY include this attribute multiple times. If multiple time synchronization sources were used in the past, the client MUST only report the 'last synced' parameter on the latest synchronization performed. If a client supports a specific synchronization method, but does not know any sources to use for synchronization, it SHOULD indicate the method without specifying the source. A client MAY indicate itself as source if it is a clock synchronization source,

but it SHOULD do so using a publicly reachable address.

The parameter can be used as both a session or media level attribute. It will normally be a session level parameter, since it is not directly media-related. In case of IDMS however, it can be used in conjunction with the rtcp-idms SDP parameter, and then it SHOULD be used as a media-level parameter as well.

The meaning of 'local' is that no clock synchronization is performed. Although not re-used, the meaning of 'gps' (Global Positioning System), 'gal' (Galileo Positioning System) and ptp (Precision Time Protocol) follows the use of reference identifiers in NTP [RFC5905]. When using ptp, the ptp-id MUST contain the proper identifier from the used IEEE specification.

The 'last synced' parameter is used as an indication for the receiver of the parameter on the accuracy of the clock. If the indicated last synchronization time is very recent, this is an indication that the clock can be trusted to be accurate, given the method of clock synchronization used. If the indicated last synchronization time is longer ago or in the future, either the clock synchronization has been performed long ago, or the clock is synchronized to an incorrect synchronization source. Either way, this shows that the clock used can not be trusted to be accurate.

12. Compatibility with ETSI TISpan

As described in section 1.4, ETSI TISpan has also described a mechanism for IDMS in [TS 183 063]. One of the main differences between the TISpan document and this document is the fact that the TISpan solution uses an RTPC XR block for both the SC->MSAS message as well as for the MSAS->SC message (by selecting different SPST-types), while this document specifies a new RTCP Packet Type for the MSAS->SC message. The message from MSAS to SC is not in any way a report on how a receiver sees a session, and therefore a separate RTCP packet type is more appropriate than the XR block solution chosen in ETSI TISpan.

In order to maintain backward-compatibility, the RTCP XR block used for SC->MSAS signaling specified in this document is fully compatible with the TISpan defined XR block.

For the MSAS->SC signaling, it is recommended to use the RTCP IDMS Packet Type defined in this document. The TISpan XR block with SPST=2 MAY be used for purposes of compatibility with the TISpan solution, but MUST NOT be used if all nodes involved support the new RTCP IDMS Packet Type.

The above means that the IANA registry contains two SDP parameters for the MSAS->SC signaling; one for the ETSI TISPAN solution and one for the IETF solution. This also means that if all elements in the SDP negotiation support the IETF solution they SHOULD use the new RTCP IDMS Packet Type.

13. On the use of presentation timestamps

A receiver can report on different timing events, i.e. on packet arrival times and on playout times. A receiver SHALL report on arrival times and a receiver MAY report on playout times. RTP packet arrival times are relatively easy to report on. Normally, the processing and play-out of the same media stream by different receivers will take roughly the same amount of time. By synchronizing on packet arrival times, you may lose some accuracy, but it will be adequate for many applications, such as social TV. Also, if the receivers are in some way controlled, e.g. having the same buffer settings and decoding times, high accuracy can be achieved. However, if all receivers in a synchronization session have the ability to report on, and thus synchronize on, actual playout times, or packet presentation times, this may be more accurate. It is up to applications and implementations of this RTCP extension whether to implement and use this.

14. Security Considerations

The specified RTCP XR Block Type in this document is used to collect, summarize and distribute information on packet reception- and playout-times of streaming media. The information may be used to orchestrate the media play-out at multiple devices.

Errors in the information, either accidental or malicious, may lead to undesired behavior. For example, if one device erroneously reports a two-hour delayed play-out, then another device in the same synchronization group could decide to delay its play-out by two hours as well, in order to keep its play-out synchronized. A user would likely interpret this two hour delay as a malfunctioning service.

Therefore, the application logic of both Synchronization Clients and Media Synchronization Application Servers should check for inconsistent information. Differences in play-out time exceeding configured limits (e.g. more than ten seconds) could be an indication of such inconsistent information.

No new mechanisms are introduced in this document to ensure confidentiality. Encryption procedures, such as those being suggested for a Secure RTP (SRTP) at the time that this document was written, can be used when confidentiality is a concern to end hosts.

15. IANA Considerations

New RTCP Packet Types and RTCP XR Block Types are subject to IANA registration. For general guidelines on IANA considerations for RTCP XR, refer to [RFC3611].

[TS 183 063] assigns the block type value 12 in the RTCP XR Block Type Registry to "Inter-destination Media Synchronization Block". [TS 183 063] also registers the SDP [RFC4566] parameter "grp-sync" for the "rtcp-xr" attribute in the RTCP XR SDP Parameters Registry.

Further, this document defines a new RTCP packet type called IDMS report. This new packet type is registered with the IANA registry of RTP parameters, based on the specification in section 7.

Further, this document defines a new SDP parameter "rtcp-idms" within the existing IANA registry of SDP Parameters.

The SDP attribute "rtcp-idms" defined by this document is registered with the IANA registry of SDP Parameters as follows:

SDP Attribute ("att-field"):

Attribute name:	rtcp-idms
Long form:	RTCP report block for IDMS
Type of name:	att-field
Type of attribute:	media level
Subject to charset:	no
Purpose:	see sections 7 and 10 of this document
Reference:	this document
Values:	see this document

Further, this document defines a new SDP attribute, "clocksource", within the existing IANA registry of SDP Parameters.

The SDP attribute "clocksource" defined by this document is registered with the IANA registry of SDP Parameters as follows:

SDP Attribute ("att-field"):

Attribute name:	clocksource
-----------------	-------------

Long form: clock synchronization source

Type of name: att-field

Type of attribute: session level

Subject to charset: no

Purpose: see sections 8 and 11 of this document

Reference: this document

Values: see this document and registrations below

The attribute has an extensible parameter field and therefore a registry for these parameters is required. This document creates an IANA registry called the Clocksource Source Parameters Registry. It contains the five parameters defined in Section 11: "local", "ntp", "gps", "gal" and "ptp".

16. Contributors

The following people have participated as co-authors or provided substantial contributions to this document: Omar Niamut, Fabian Walraven, Ishan Vaishnavi, Rufael Mekuria

17. Conclusions

This document describes the RTCP XR block type for IDMS, the RTCP IDMS report and the associated SDP parameters for inter-destination media synchronization. It also describes an SDP parameter for indicating which source is used for synchronizing a (systems) (wall) clock.

18. References

18.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC5234] Crocker, D., Ed., and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.
- [RFC3611] Friedman, T., Ed., Caceres, R., Ed., and A. Clark, Ed., "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, November 2003.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.
- [RFC5760] Ott, J., Chesterfield, J., and E. Schooler, "RTP Control Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback", RFC 5760, February 2010.
- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.
- [TS 183 063] ETSI TISPAN, "IMS-based IPTV stage 3 specification", TS 183 063 v3.4.1, June 2010.

18.2. Informative References

- [RFC5968] Ott, J. and C. Perkins, "Guidelines for Extending the RTP Control Protocol (RTCP)", RFC 5968, September 2010.

- [Boronat2009] Boronat, F., et al, "Multimedia group and inter-stream synchronization techniques: A comparative study", Elsevier Information Systems 34 (2009), pp. 108-131
- [Ishibashi2006] Ishibashi Y. et al, "Subjective Assesment of Fairness among users in multipoint communications". Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology, 2006.
- [IEEE-1588] IEEE Standards Association, "1588-2008 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", 2008

Authors' Addresses

Ray van Brandenburg
TNO
Brassersplein 2, Delft, the Netherlands

Phone: +31 88 86 63609
Email: ray.vanbrandenburg@tno.nl

Hans M. Stokking
TNO
Brassersplein 2, Delft, the Netherlands

Phone: +31 88 86 67278
Email: hans.stokking@tno.nl

M. Oskar van Deventer
TNO
Brassersplein 2, Delft, the Netherlands

Phone: +31 88 86 67078
Email: oskar.vandeventer@tno.nl

Fernando Boronat
IGIC Institute, Universidad Politecnica de Valencia-Campus de Gandia
C/ Paraninfo, 1, Grao de Gandia, 46730, Valencia, Spain

Phone: +34 962 849 341
Email: fboronat@dcom.upv.es

Mario Montagud
IGIC Institute, Universidad Politecnica de Valencia-Campus de Gandia
C/ Paraninfo, 1, Grao de Gandia, 46730, Valencia, Spain

Phone: +34 962 849 341
Email: mamontor@posgrado.upv.es

Kevin Gross
AVA Networks

Audio/Video Transport Working Group
Internet-Draft
Intended status: Informational
Expires: April 27, 2012

Q. Wu, Ed.
Huawei
G. Hunt
Unaffiliated
P. Arden
BT
October 25, 2011

Monitoring Architectures for RTP
draft-ietf-avtcore-monarch-05.txt

Abstract

This memo proposes an architecture for extending RTCP with a new RTCP XR (RFC3611) block type to report new metrics regarding media transmission or reception quality, following RTCP guideline established in RFC5968. This memo suggests that a new block should contain a single metric or a small number of metrics relevant to a single parameter of interest or concern, rather than containing a number of metrics which attempt to provide full coverage of all those parameters of concern to a specific application. Applications may then "mix and match" to create a set of blocks which covers their set of concerns. Where possible, a specific block should be designed to be re-usable across more than one application, for example, for all of voice, streaming audio and video.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 27, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Requirements notation	4
3. RTP monitoring architecture	5
3.1. RTCP Metric Block Report and associated parameters	7
4. Issues with reporting metric block using RTCP XR extension	9
5. Guideline for reporting metric block using RTCP XR	11
5.1. Using single metric blocks	11
5.2. Correlating RTCP XR with the non-RTP data	11
5.3. Reducing Measurement information repetition	12
6. An example of a metric block	13
7. Application to RFC 5117 topologies	14
7.1. Applicability to MCU	14
7.2. Applicability to Translators	15
8. IANA Considerations	16
9. Security Considerations	17
10. Acknowledgement	18
11. Informative References	19
Appendix A. Change Log	21
A.1. draft-ietf-avtc core-monarch-00	21
A.2. draft-ietf-avtc core-monarch-01	21
A.3. draft-ietf-avtc core-monarch-02	21
A.4. draft-ietf-avtc core-monarch-03	22
A.5. draft-ietf-avtc core-monarch-04	22
A.6. draft-ietf-avtc core-monarch-05	22
Authors' Addresses	24

1. Introduction

As more users and subscribers rely on real time application services, uncertainties in the performance and availability of these services are driving the need to support new standard methods for gathering performance metrics from RTP applications. These rapidly emerging standards, such as RTCP XR [RFC3611] and other RTCP extension to Sender Reports (SR), Receiver Reports (RR) [RFC3550] are being developed for the purpose of collecting and reporting performance metrics from endpoint devices that can be used to correlate the metrics, provide end to end service visibility and measure and monitor Quality of Experience (QoE).

However the proliferation of RTP/RTCP specific metrics for transport and application quality monitoring has been identified as a potential problem for RTP/RTCP interoperability, which attempt to provide full coverage of all those parameters of concern to a specific application. Since different applications layered on RTP may have some monitoring requirements in common, therefore these metrics should be satisfied by a common design.

The objective of this document is to define an extensible RTP monitoring framework to provide a small number of re-usable Quality of Service (QoS)/QoE metrics which facilitate reduced implementation costs and help maximize inter-operability. RTCP Guideline [RFC5968] has stated that, where RTCP is to be extended with a new metric, the preferred mechanism is by the addition of a new RTCP XR [RFC3611] block. This memo assumes that any requirement for a new metric to be transported in RTCP will use a new RTCP XR block.

2. Requirements notation

This memo is informative and as such contains no normative requirements.

In addition, the following terms are defined:

Transport level metrics

A set of metrics which characterise the three transport impairments of packet loss, packet delay, and packet delay variation. These metrics should be usable by any application which uses RTP transport.

Application level metrics

Metrics relating to QoE related parameters. These metrics are measured at the application level and focus on quality of content rather than network parameters. One example of such metrics is the QoE Metric specified in QoE metric reporting Block [MQ].

End System metrics

Metrics relating to the way a terminal deals with transport impairments affecting the incident RTP stream. These may include de-jitter buffering, packet loss concealment, and the use of redundant streams (if any) for correction of error or loss.

3. RTP monitoring architecture

The RTP monitoring architecture comprises the following two key functional components shown below:

- o Monitor
- o Metric Block Structure

Monitor is the functional component defined in the Real-time Transport Protocol (RTP) [RFC3550] that acts as a source of information gathered for monitoring purposes. It may gather such information reported by RTCP XR or other RTCP extension and calculate statistics from multiple source. According to the definition of monitor in the RTP Protocol [RFC3550], the end system that source RTP streams, an intermediate-system that forwards RTP packets to End-devices or a third party that does not participate in the RTP session (i.e., the third party monitor depicted in figure 1) can be envisioned to act as the Monitor within the RTP monitoring architecture.

The Metric Block exposes real time Application Quality information in the appropriate report block format to the management system within the RTP monitoring architecture. Such information can be formulated as:

- o The basic metric that is directly measured.
- o or the composed metric that is derived from one or more basic metrics.

Both the RTCP or RTCP XR can be extended to convey these metrics. The details on transport protocols for metric blocks are described in Section 3.1.

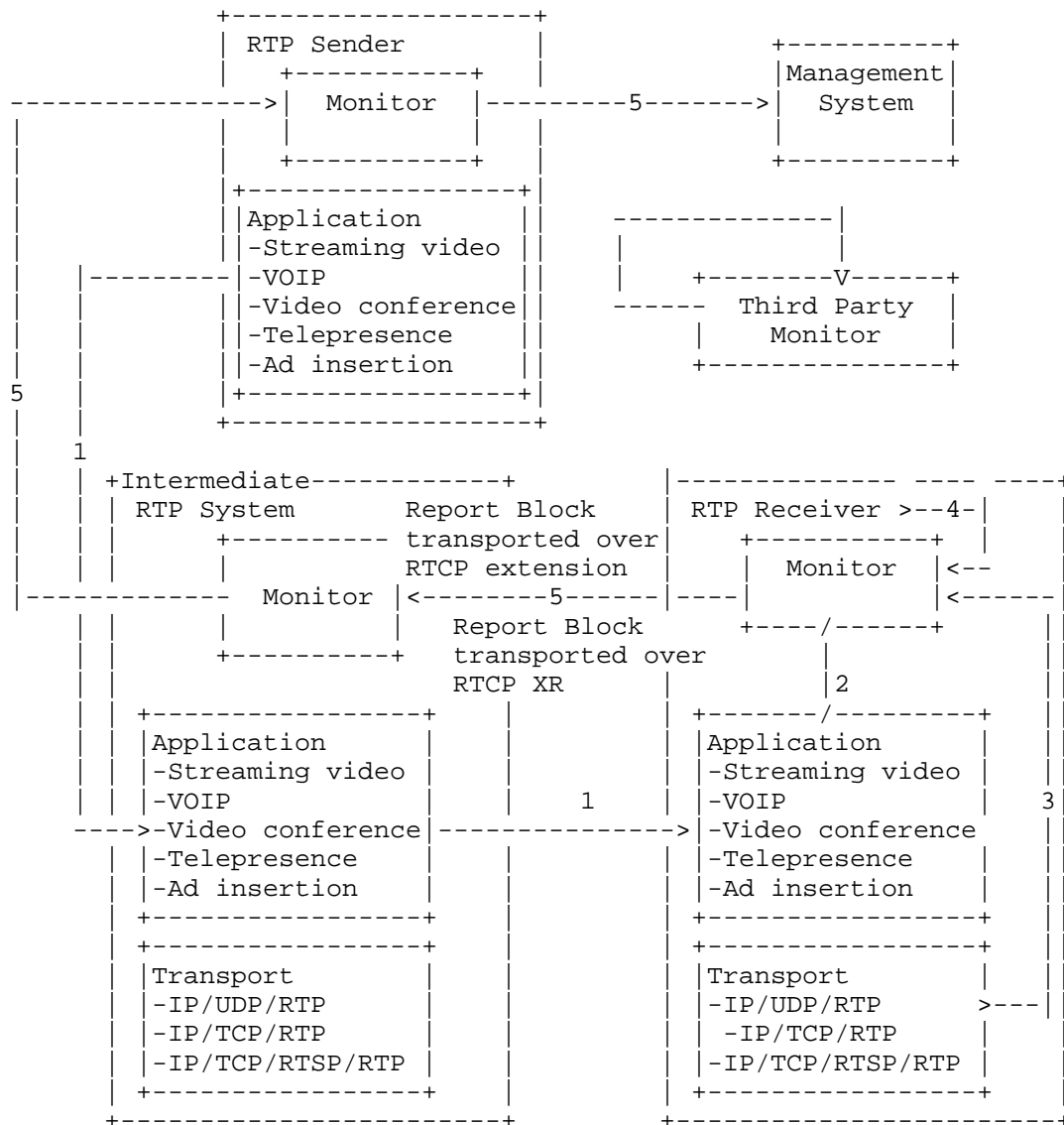


Figure 1: RTP Monitoring Architecture

1. RTP communication between real time applications.
2. Application level metrics collection.
3. Transport level metrics collection.

4. End System metrics collection.

5. Reporting Session- metrics transmitted over specified interfaces.

3.1. RTCP Metric Block Report and associated parameters

The basic RTCP Reception Report (RR) [RFC3550] conveys reception statistics (i.e., transport level statistics) in metric block report format for multiple RTP media streams including

- o the fraction of packet lost since the last report
- o the cumulative number of packets lost
- o the highest sequence number received
- o an estimate of the inter-arrival jitter
- o and information to allow senders to calculate the network round trip time.

The RTCP XRs [RFC3611] supplement the existing RTCP packets and provide more detailed feedback on reception quality in several categories:

- o Loss and duplicate RLE reports
- o Packet-receipt times reports
- o Round-trip time reports
- o Statistics Summary Reports

There are also various other scenarios in which it is desirable to send RTCP Metric reports more frequently. For example, the Audio/Video Profile with Feedback [RFC4585] extends the standard A/V Profile [RFC3551] to allow RTCP reports to be sent early provided RTCP bandwidth allocation is respected. The following are four use cases but are not limited to:

- o RTCP NACK is used to provide feedback on the RTP sequence number of the lost packets [RFC4585].
- o RTCP is extended to convey requests for full intra-coded frames or select the reference picture, and signal changes in the desired temporal/spatial trade-off and maximum media bit rate [RFC5104].

- o RTCP or RTCP XR is extended to provide feedback on ECN statistics information [ECN].
- o RTCP XR is extended to provide feedback on multicast acquisition statistics information and parameters [RFC6332].

4. Issues with reporting metric block using RTCP XR extension

Issues that have come up in the past with reporting metric block using RTCP XR extensions include (but are probably not limited to) the following:

- o Using large block. A single report block or metric is designed to contain a large number of parameters in different classes for a specific application. For example, the RTCP Extended Reports (XRs) [RFC3611] defines seven report block formats for network management and quality monitoring. Some of these block types defined in the RTCP XRs [RFC3611] are only specifically designed for conveying multicast inference of network characteristics (MINC) or voice over IP (VoIP) monitoring. However different applications layered on RTP may have different monitoring requirements, design large block only for specific applications may increase implementation cost and minimize interoperability.
- o Correlating RTCP XR with the non-RTP data. CNAME defined in the RTP Protocol [RFC3550] is an example of existing tool that allows to bind an SSRC that may change to a fixed source name in one RTP session. It may be also fixed across multiple RTP sessions from the same source. However there may be situations where RTCP reports are sent to other participating endpoints using non-RTP protocol in a session. For example, as described in the SIP RTCP Summary Report Protocol [RFC6035], the data contained in RTCP XR VoIP metrics reports [RFC3611] are forwarded to a central collection server systems using SIP. In such case, there is a large portfolio of quality parameters that can be associated with real time application, e.g., VOIP application, but only a minimal number of parameters are included on the RTCP-XR reports. Therefore correlation between RTCP XR and non-RTP data should be concerned if administration or management systems need to rely on the mapping RTCP statistics to non-RTCP measurements to conduct data analysis and creates alerts to the users. Without such correlation, it is hardly to provide accurate measures of real time application quality with a minimal number of parameters included on the RTCP-XR reports in such case.
- o Measurement Information duplication. Measurement information provides information relevant to a measurement reported in one or more other block types. For example, we may set a metric interval for the session and monitor RTP packets within one or several consecutive metric interval. In such case, the extra measurement information (e.g., extended sequence number of 1st packet, measurement period) may be expected. However if we put such extra measurement information into each metric block, there may be situations where an RTCP XR packet containing multiple metric

blocks, reports on the same streams from the same source. In other words, duplicated data for the measurement is provided multiple times, once in every metric block. Though this design ensures immunity to packet loss, it may bring more packetization complexity and the processing overhead is not completely trivial in some cases. Therefore compromise between processing overhead and reliability should be taken into account.

5. Guideline for reporting metric block using RTCP XR

5.1. Using single metric blocks

Different applications using RTP for media transport certainly have differing requirements for metrics transported in RTCP to support their operation. For many applications, the basic metrics for transport impairments provided in RTCP SR and RR packets [RFC3550] (together with source identification provided in RTCP SDES packets) are sufficient. For other applications additional metrics may be required or at least sufficiently useful to justify the overheads, both of processing in endpoints and of increased session bandwidth. For example an IPTV application using Forward Error Correction (FEC) might use either a metric of post-repair loss or a metric giving detailed information about pre-repair loss bursts to optimise payload bandwidth and the strength of FEC required for changing network conditions. However there are many metrics available. It is likely that different applications or classes of applications will wish to use different metrics. Any one application is likely to require metrics for more than one parameter but if this is the case, different applications will almost certainly require different combinations of metrics. If larger blocks are defined containing multiple metrics to address the needs of each application, it becomes likely that many different such larger blocks are defined, which becomes a danger to interoperability.

To avoid this pitfall, this memo proposes the use of small RTCP XR metrics blocks each containing a very small number of individual metrics characterizing only one parameter of interest to an application running over RTP. For example, at the RTP transport layer, the parameter of interest might be packet delay variation, and specifically the metric "IPDV" defined by [Y1540]. See Section 6 for architectural considerations for a metrics block, using as an example a metrics block to report packet delay variation.

5.2. Correlating RTCP XR with the non-RTP data

There may be situation where more than one media transport protocols are used by one application to interconnect to the same session in the gateway. For example, one RTCP XR Packet is sent to the participating endpoints using non- RTP-based media transport (e.g., using SIP) in a VOIP session, one crucial factor lies in how to handle their different identities that are corresponding to different media transport.

This memo proposes an approach to facilitate the correlation of the RTCP Session with other session-related non-RTP data. That is to say if there is a need to correlate RTP sessions with non-RTP sessions,

then the correlation information needed should be conveyed in a new RTCP Source Description (SDES) item ,since such correlation information describes the source, rather than providing a quality report. An example use case is for a participant endpoint may convey a call identifier or a global call identifier associated with the SSRC of measured RTP stream . In such case, the participant endpoint uses the SSRC of source to bind the call identifier using SDES item in the SDES RTCP packet and send such correlation to the network management system. A flow measurement tool that is configured with the 5-tuple and not call-aware then forward the RTCP XR reports along with the SSRC of the measured RTP stream which is included in the XR Block header and 5-tuple to the network management system. Network management system can then correlate this report using SSRC with other diagnostic information such as call detail records.

5.3. Reducing Measurement information repetition

When multiple metric blocks are carried in one RTCP XR packet, reporting on the same streams from the same source, RTCP should use the SSRC to identify and correlate and group participants between metric blocks. If extra measurement information (e.g., measurement period) is expected and this information relates measurement data in the different metric blocks to the same stream, and measurement period, this memo propose to define new XR Block to convey it. In order to reduce measurement information repetition in one RTCP XR packet containing multiple metric blocks, the measurement information should be sent together with the related metric block if measurement information changes.

6. An example of a metric block

This section uses the example of an existing proposed metrics block to illustrate the application of the principles set out in Section 5.1.

The example [PDV] is a block to convey information about packet delay variation (PDV) only, consistent with the principle that a metrics block should address only one parameter of interest. One simple metric of PDV is available in the RTCP RR packet as the "interarrival jitter" field. There are other PDV metrics which may be more useful to certain applications. Two such metrics are the IPDV metric ([Y1540], [RFC3393]) and the MAPDV2 metric [G1020]. Use of these metrics is consistent with the principle in Section 5 of RTCP guideline [RFC5968] that metrics should usually be defined elsewhere, so that RTCP standards define only the transport of the metric rather than its nature. The purpose of this section is to illustrate the architecture using the example of [PDV] rather than to document the design of the PDV metrics block or to provide a tutorial on PDV in general.

Given the availability of at least three metrics for PDV, there are design options for the allocation of metrics to RTCP XR blocks:

- o provide an RTCP XR block per metric
- o provide a single RTCP XR block which contains all three metrics
- o provide a single RTCP block to convey any one of the three metrics, together with a identifier to inform the receiving RTP system of the specific metric being conveyed

In choosing between these options, extensibility is important, because additional metrics of PDV may well be standardized and require inclusion in this framework. The first option is extensible but only by use of additional RTCP XR blocks, which may consume the limited namespace for RTCP XR blocks at an unacceptable rate. The second option is not extensible, so could be rejected on that basis, but in any case a single application is quite unlikely to require transport of more than one metric for PDV. Hence the third option was chosen. This implies the creation of a subsidiary namespace to enumerate the PDV metrics which may be transported by this block, as discussed further in [PDV] .

7. Application to RFC 5117 topologies

The topologies specified in [RFC5117] fall into two categories. The first category relates to the RTP system model utilizing multicast and/or unicast. The topologies in this category are specifically Topo-Point-to-Point, Topo- Multicast, Topo-Translator (both variants, Topo-Trn-Translator and Topo-Media-Translator, and combinations of the two), and Topo-Mixer. These topologies use RTP end systems, RTP mixers and RTP translators defined in the RTP protocol [RFC3550]. For purposes of reporting connection quality to other RTP systems, RTP mixers and RTP end systems are very similar. Mixers resynchronize packets and do not relay RTCP reports received from one cloud towards other cloud(s). Translators do not resynchronize packets and SHOULD forward certain RTCP reports between clouds. In this category, the RTP system (end system, mixer or translator) which originates, terminates or forwards RTCP XR blocks is expected to handle RTCP, including RTCP XR, according to the RTP protocol [RFC3550]. Provided this expectation is met, an RTP system using RTCP XR is architecturally no different from an RTP system of the same class (end system, mixer, or translator) which does not use RTCP XR. The second category relates to deployed system models used in many H.323 [H323] video conferences. The topologies in this category are Topo-Video-Switch-MCU and Topo-RTCP-terminating-MCU. Such topologies based on systems do not behave according to the RTP protocol [RFC3550].

Considering the MCU and translator are two typical topologies in the two categories mentioned above, this document will take them as two typical examples to explain how RTCP XR report works in different RFC5117 topologies.

7.1. Applicability to MCU

Topo-Video-Switch-MCU and Topo-RTCP-terminating-MCU, suffer from the difficulties described in [RFC5117]. These difficulties apply to systems sending, and expecting to receive, RTCP XR blocks as much as to systems using other RTCP packet types. For example, a participant RTP end system may send media to a video switch MCU. If the media stream is not selected for forwarding by the switch, neither RTCP RR packets nor RTCP XR blocks referring to the end system's generated stream will be received at the RTP end system. Strictly the RTP end system can only conclude that its RTP has been lost in the network, though an RTP end system complying with the robustness principle of [RFC1122] should survive with essential functions (i.e., media distribution) unimpaired.

7.2. Applicability to Translators

Section 7.2 of the RTP protocol [RFC3550] describes processing of RTCP by translators. RTCP XR is within the scope of the recommendations of the RTP protocol [RFC3550]. Some RTCP XR metrics blocks may usefully be measured at, and reported by, translators. As described in the RTP protocol [RFC3550] this creates a requirement for the translator to allocate an SSRC for the monitor collocated with itself so that the monitor may populate the SSRC in the RTCP XR packet header as packet sender SSRC and send it out (although the translator is not a Synchronisation Source in the sense of originating RTP media packets). It must also supply this SSRC and the corresponding CNAME in RTCP SDES packets.

In RTP sessions where one or more translators generate any RTCP traffic towards their next-neighbour RTP system, other translators in the session have a choice as to whether they forward a translator's RTCP packets. Forwarding may provide additional information to other RTP systems in the connection but increases RTCP bandwidth and may in some cases present a security risk. RTP translators may have forwarding behaviour based on local policy, which might differ between different interfaces of the same translator.

For bidirectional unicast, an RTP system may check the presence of a translator by receiving RTCP XR and noting that the sending SSRC in the XR packet is not present in any RTP media packet. However the RTP system can't determine that an SSRC in the XR packet is associated with a translator rather than a receiving only end-point unless the necessary session information on RTP node type is expected. This Checking may fail if a source sends RTCP XR before it has sent any RTP media (leading to transient mis-categorisation of an RTP end system or RTP mixer as a translator). Another case is, for multicast sessions - or unidirectional/streaming unicast, there is also a possibility of a receive-only end system being permanently mis-categorised as a translator sending XR report, i.e., the monitor sending XR report within the translator. Hence it is desirable for a translator that sends XR report to have a way to announce its SSRC and associated RTP node type explicitly to the management system.

8. IANA Considerations

There is no IANA action in this document.

9. Security Considerations

This document focuses on the RTCP reporting extension using RTCP XR and should not give rise to any new security vulnerabilities beyond those described in RTCP XRs [RFC3611]. However it also describes the architectural framework to be used for monitoring at RTP layer. The security issues with monitoring needs to be considered.

In RTP sessions, a translator may use its own SSRC to send its monitoring reports towards its next-neighbour RTP system. Other translators in the session may have a choice as to whether they forward the translator's RTCP packets. This presents a security issue since the information in the report may be exposed by the other translator to any malicious node. Therefore if the information is considered as sensitive, the monitoring report should be encrypted.

Also note that the third party monitors are not visible at the RTP layer since they do not send any RTCP packets. In order to prevent any sensitive information leakage, the monitoring from the third party monitors should be prohibited unless the security is in place to authenticate them.

10. Acknowledgement

The authors would also like to thank Colin Perkins, Graeme Gibbs, Debbie Greenstreet, Keith Drage, Dan Romascanu, Ali C. Begen, Roni Even, Magnus Westerlund for their valuable comments and suggestions on the early version of this document.

11. Informative References

- [ECN] Westerlund, M., Johansson, I., Perkins, C., O'Hanlon, P., and K. Carlberg, "Explicit Congestion Notification (ECN) for RTP over UDP", ID draft-ietf-avtcore-ecn-for-rtp-04, July 2011.
- [G1020] ITU-T, "ITU-T Rec. G.1020, Performance parameter definitions for quality of speech and other voiceband applications utilizing IP networks", July 2006.
- [H323] ITU-T, "ITU-T Rec. H.323, Packet-based multimedia communications systems", June 2006.
- [MQ] Wu, Q., Zorn, G., Schott, R., and K. Lee, "RTCP XR Blocks for multimedia quality metric reporting", ID draft-wu-xrblock-rtcp-xr-quality-monitoring-02, May 2011.
- [PDV] Hunt, G., "RTCP XR Report Block for Packet Delay Variation Metric Reporting", ID draft-ietf-xrblock-rtcp-xr-pdv-00, September 2011.
- [RFC1122] Braden, R., "Requirements for Internet Hosts -- Communication Layers", RFC 1122, October 1989.
- [RFC3393] Demichelis, C., "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)", RFC 3393, November 2002.
- [RFC3550] Schulzrinne, H., "RTP: A Transport Protocol for Real-Time Applications", RFC 3550, July 2003.
- [RFC3551] Schulzrinne, H. and S. Casner, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 3551, July 2003.
- [RFC3611] Friedman, T., "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, November 2003.
- [RFC4585] Ott, J. and S. Wenger, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.
- [RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Session Initiation Protocol Event Package for Voice Quality Reporting", RFC 5104, February 2008.
- [RFC5117] Westerlund, M., "RTP Topologies", RFC 5117, January 2008.

- [RFC5968] Ott, J. and C. Perkins, "Guidelines for Extending the RTP Control Protocol (RTCP)", RFC 5968, September 2010.
- [RFC6035] Pendleton, A., Clark, A., Johnston, A., and H. Sinnreich, "Session Initiation Protocol Event Package for Voice Quality Reporting", RFC 6035, November 2010.
- [RFC6332] Begen, A. and E. Friedrich, "Multicast Acquisition Report Block Type for RTP Control Protocol (RTCP) Extended Reports (XRs)", RFC 6332, July 2011.
- [Y1540] ITU-T, "ITU-T Rec. Y.1540, IP packet transfer and availability performance parameters", November 2007.

Appendix A. Change Log

Note to the RFC-Editor: please remove this section prior to publication as an RFC.

A.1. draft-ietf-avtc core-monarch-00

The following are the major changes compared to draft-hunt-avtc core-monarch-02:

- o Move Geoff Hunt and Philip Arden to acknowledgement section.

A.2. draft-ietf-avtc core-monarch-01

The following are the major changes compared to 00:

- o Restructure the document by merging section 4 into section 3.
- o Remove section 4.1, section 5 that is out of scope of this document.
- o Remove the last bullet in section 6 and section 7.3 based on conclusion of last meeting.
- o Update figure 1 and related text in section 3 according to the monitor definition in RFC3550.
- o Revise section 9 to address monitor declaration issue.
- o Merge the first two bullet in section 6.
- o Add one new bullet to discuss metric block association in section 6.

A.3. draft-ietf-avtc core-monarch-02

The following are the major changes compared to 01:

- o Deleting first paragraph of Section 1.
- o Deleting Section 3.1, since the interaction with the management application is out of scope of this draft.
- o Separate identity information correlation from section 5.2 as new section 5.3.
- o Remove figure 2 and related text from section 5.2.

- o Editorial changes in the section 4 and the first paragraph of section 7.

A.4. draft-ietf-avtcore-monarch-03

The following are the major changes compared to 02:

- o Update bullet 2 in section 4 to explain the ill-effect of Identity Information duplication.
- o Update bullet 3 in section 4 to explain why Correlating RTCP XR with the non-RTP data is needed.
- o Update section 5.2 to focus on how to reduce the identity information repetition
- o Update section 5.3 to explain how to correlate identity information with the non-RTP data

A.5. draft-ietf-avtcore-monarch-04

The following are the major changes compared to 03:

- o Update section 5.2 to clarify using SDES packet to carry correlation information.
- o Remove section 5.3 since additional identity information goes to SDES packet and using SSRC to identify each block is standard RTP feature.
- o Swap the last two paragraphs in the section 4 since identity information duplication can not been 100% avoided.
- o Other editorial changes.

A.6. draft-ietf-avtcore-monarch-05

The following are the major changes compared to 04:

- o Replace "chunk" with "new SDES item".
- o Add texts in security section to discussion potential security issues.
- o Add new sub-section 5.3 to discuss Reducing Measurement information repetition.

- o Other editorial changes.

Authors' Addresses

Qin Wu (editor)
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: sunseawq@huawei.com

Geoff Hunt
Unaffiliated

Email: r.geoff.hunt@gmail.com

Philip Arden
BT
Orion 3/7 PP4
Adastral Park
Martlesham Heath
Ipswich, Suffolk IP5 3RE
United Kingdom

Phone: +44 1473 644192
Email: philip.arden@bt.com

RTCWEB
Internet-Draft
Intended status: Standards Track
Expires: April 26, 2012

J. Lennox
Vidyo
J. Rosenberg
Skype
October 24, 2011

Multiplexing Multiple Media Types In a Single Real-Time Transport
Protocol (RTP) Session
draft-lennox-rtcweb-rtp-media-type-mux-00

Abstract

This document describes mechanisms and recommended practice for transmitting media streams of multiple media types (e.g., audio and video) over a single Real-Time Transport Protocol (RTP) session, primarily for the use of Real-Time Communication for the Web (rtcweb).

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 26, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. Transmitting multiple types of media in a single RTP session	4
3.1. Optimizations	5
4. Backward compatibility	6
5. Signaling	7
6. Protocols with SSRC semantics	8
7. Security Considerations	8
8. IANA Considerations	9
9. References	9
9.1. Normative References	9
9.2. Informative References	9
Authors' Addresses	10

1. Introduction

Classically, multimedia sessions using the Real-Time Transport Protocol (RTP) [RFC3550] have transported different media types (most commonly, audio and video) in different RTP sessions, each with its own transport flow. At the time RTP was designed, this was a reasonable design decision, reducing system variability and adding flexibility ([RFC3550] discusses the motivation for this design decision in section 5.2).

However, the de facto architecture of the Internet has changed substantially since RTP was originally designed, nearly twenty years ago. In particular, Network Address Translators (NATs) and firewalls are now ubiquitous, and IPv4 address space scarcity is becoming more severe. As a consequence, the network resources used up by an application, and its probability of failure, are directly proportional to the number of distinct transport flows it uses.

Furthermore, applications have developed mechanisms (notably Interactive Connectivity Establishment (ICE) [RFC5245]) to traverse NATs and firewalls. The time such mechanisms need to perform the traversal process is proportional to the number of distinct transport flows in use.

As a result, in the modern Internet, it is advisable and useful to revisit the transport-layer separation of media in a multimedia session. Fortunately, the architecture of RTP allows this to be done in a straightforward and natural way: by placing multiple sources of different media types in the same RTP session.

Since this is architecturally somewhat different from existing RTP deployments, however, this decision has some consequences that may be non-obvious. Furthermore, it is somewhat complex to negotiate such flows in signaling protocols that assumed the older architecture, most notably the Session Description Protocol (SDP) [RFC4566]. The rest of this document discusses these issues.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119] and indicate requirement levels for compliant implementations.

3. Transmitting multiple types of media in a single RTP session

RTP [RFC3550] supports the notion of multiple sources within a session. Historically, this was typically used for distinct users within a group to send media of the same type. Each source has its own synchronization source (SSRC) value and has a distinct sequence number and timestamp space. This document specifies that this same mechanism is used to allow sources of multiple media types in the same RTP session, even if they come from the same user. For example, in a call containing audio and video between two users, each sending a single audio and a single video source, there would be a single RTP session containing two sources (one audio, one video) from each user, for a total of four sources (and thus four SSRC values) within the RTP session.

Transmitting multiple types of media in a single RTP [RFC3550] session is done using the same RTP mechanisms as are used to transmit multiple sources of the same media type on a session. Notably:

- o Each stream (of every media type) is a distinct source (distinct stream of consecutive packets to be sent to a decoder) and is given a distinct synchronization source ID (SSRC), and has its own distinct timestamp and sequence number space.
- o Every media type (full media type and subtype, e.g. video/h264 or audio/pcmu) has a distinct payload type value. The same payload type value mappings apply across all sources in the session.
- o RTP SSRCs, initial sequence numbers, and initial timestamps are chosen at random, independently for each source (of each media type).
- o RTCP bandwidth is five percent of the total RTP session bandwidth.
- o RTP session bandwidth and RTCP bandwidth are divided among all the sources in the session.
- o RTCP sender report (SR) or receiver report (RR) packets, and source description (SDES) packets, are sent periodically for every source in the session.

In other words, no special RTP mechanisms are specifically needed for senders of multiplexed media. The only constraint is that senders sources MUST NOT change the top-level media type (e.g. audio or video) of a given source. (It remains valid to change a source's subtype, e.g. switching between audio/pcmu and audio/g729.)

For a receiver, the primary complexity of multiplexing is knowing how to process a received source. Without multiplexing, all sources in an RTP session can (in theory) be processed the same manner; e.g., all audio sources can be fed to an audio mixer, and all video sources displayed on a screen. With multiplexing, however, receivers must apply additional knowledge.

If the streams being multiplexed are simply audio and video, this processing can decision can be made based simply on a source's payload type. For more complex situations (for example, simultaneous live-video and shared-application sources, both sent as video), signaling-level descriptions of sources would be needed, using a mechanism such as SDP Source Descriptions [RFC5576].

Additionally, due to the large difference in typical bitrate between different media (video can easily use a bit rate an order of magnitude or more larger than audio), some complications arise with RTCP timing. Because RTCP bandwidth is shared evenly among all sources in a session, the RTCP for an audio source can end up being sent significantly more frequently than it would in a non-multiplexed session. (The RTCP for video will, correspondingly, be sent slightly less frequently; this is not nearly as serious an issue.)

For RTP sessions that use RTP's recommended minimum fixed timing interval of 5 seconds, this problem is not likely to arise, as most sessions' bandwidth is not so low that RTCP timing exceeds this limit. The RTP/AVP [RFC3551] or RTP/SAVP [RFC3711] profiles use this minimum interval by default, and do not have a mechanism in SDP to negotiate an alternate interval.

For sessions using the RTP/AVPF [RFC4585] and RTP/SAVPF [RFC5124] profiles, however, endpoints SHOULD set the minimum RTCP regular reporting interval trr-int to 5000 (5 seconds), unless they explicitly need it to be lower. This minimizes the excessive RTCP bandwidth consumption, as well as aiding compatibility with AVP endpoints. Since this value only affects regular RTCP reports, not RTCP feedback, this does not prevent AVPF feedback messages from being sent as needed.

3.1. Optimizations

For multiple sources in the same session, several optimizations are possible. (Most of these optimizations also apply to multiple sources of the same type in a session.) In all cases, endpoints MUST be prepared for their peers to be using these optimizations.

An endpoint sending multiple sources MAY, as needed, reallocate media bandwidth among the RTP sources it is sending. This includes adding or removing sources as more or less bandwidth becomes available.

An endpoint MAY choose to send multiple sources' RTCP messages in a single compound RTCP packet (though such compound packets SHOULD NOT exceed the path MTU, if avoidable and if it is known). This will reduce the average compound RTCP packet size, and thus increase the frequency with which RTCP messages can be sent. Regular (non-

feedback) RTCP compound packets MUST still begin with an SR or RR packet, but otherwise may contain RTCP packets in any order. Receivers MUST be prepared to receive such compound packets.

An endpoint SHOULD NOT send reception reports from one of its own sources about another one ("cross-reports"). Such reports are useless (they would always indicate zero loss and jitter) and use up bandwidth that could more profitably be used to send information about remote sources. Endpoints receiving reception reports MUST be prepared that their peers might not be sending reception reports about their own sources. (A naive RTCP monitor might think that there is a network disconnection between these sources; however, architecturally it is very unclear if such monitors actually exist, or would care about a disconnection of this sort.)

Similarly, an endpoint sending multiple sources SHOULD NOT send reception reports about a remote source from more than one of its local sources. Instead, it SHOULD pick one of its local sources as the "reporting" source for each remote source, which sends full report blocks; all its other sources SHOULD be treated as if they were disconnected, and never saw that remote source. An endpoint MAY choose different local sources as the reporting source for different remote sources (for example, it could choose to send reports about remote audio sources from its local audio source, and reports about remote video sources from its local video source), or it MAY choose a single local source for all its reports. If the reporting source leaves the session (sends BYE), another reporting source MUST be chosen. This "reporting" source SHOULD also be the source for any AVPF feedback messages about its remote sources, as well. Endpoints interpreting reception reports MUST be prepared to receive RTCP SR or RR messages where only one remote source is reporting about its sources.

4. Backward compatibility

In some circumstances, the offerer in an offer/answer exchange [RFC3264] will not know whether the peer which will receive its offer supports media type multiplexing.

In scenarios where endpoints can rely on their peers supporting Interactive Connectivity Establishment (ICE) [RFC5245], even if they might not support multiplexing, this should not be a problem. An endpoint could construct a list of ICE candidates for its single session, and then offer that list, for backward compatibility, toward each of the peers; it would disambiguate the flows based on the ufrag fields in the received ICE connectivity checks. (This would result in the chosen ICE candidates participating in multiple RTP sessions,

in much the same manner as following a forked SIP offer.) For RTCWeb, it is currently anticipated that ICE will be required in all cases, for consent verification.

The more difficult case is if an offerer cannot reply on its potential peers supporting any features beyond baseline RTP (i.e., neither ICE nor multiplexing). In this case, it would either need to be prepared to use only a single media type (e.g., audio) with such a peer, or else will need to do the pre-offer steps to set up all the non-multiplexed sessions. Notably, this would include opening local ports, and doing ICE address gathering (collecting candidate addresses from STUN and/or TURN servers) for each session, even if it is anticipated that in most cases backward compatibility is not going to be necessary.

If the signaling protocol in use supports sending additional ICE candidates for an ongoing ICE exchange, or updating the destination of a non-ICE RTP session, it is instead possible for an offerer to do such gathering lazily, e.g. opening only local host candidates for the non-default RTP sessions, and gathering and offering additional candidates or public relay addresses once it becomes clear that they are needed. (With SIP, sending updated candidates or RTP destinations prior to the call being answered is possible only if both peers support the SIP 100rel feature [RFC3262], i.e. PRACK and UPDATE; otherwise, the initial offer cannot be updated until after the 200 OK response to the initial INVITE.)

5. Signaling

There is a need to signal multiplexed media in the Session Description Protocol (SDP) [RFC4566] -- for inter-domain federation in the case of RTCWeb, as well as for "pure" SIP endpoints that also want to use media-multiplexed sessions.

To signal multiplexed sessions, two approaches seem to present themselves: either using the SDP grouping framework [RFC5888], as in [I-D.holmberg-mmusic-sdp-bundle-negotiation], or directly representing the multiplexed sessions in SDP.

Directly encoded multiplexed sessions would have some grammar issues in SDP, as the syntax of SDP mixes together top-level media types and transport information in the m= line, splitting media types to be partially described in the m= line and partially in the a=rtpmap attribute. New SDP attributes would need to be invented to describe the top-level media types for each source.

```
m=multiplex 49170 RTP/AVP 96 97
a=mediamap:96 video
a=rtpmap:96 H264/90000
a=mediamap:97 audio
a=rtpmap:97 pcmu/8000
```

Figure 1: Hypothetical syntax for describing multiplexed media lines in SDP

If single-pass backward compatibility is (ever) a goal, directly encoding multiplexed sessions in SDP m= lines becomes much more complex, as it would require SDP Capability Negotiation [RFC5939] in order to offer both the legacy and the multiplexed streams.

Using SDP grouping seems to rule out the possibility of non-backward-compatible multiplexed streams. Other than that, however, it seems that it would be the easier path to signal multiplexed sessions.

6. Protocols with SSRC semantics

There are some RTP protocols that impose semantics on SSRC values. Most notably, there are several protocols (for instance, FEC [RFC5109], layered codecs [RFC5583], or RTP retransmission [RFC4588]) have modes that require that sources in multiple RTP sessions have the same SSRC value.

When multiplexing, this is impossible. Fortunately, in each case, there are alternative ways to do this, by explicitly signaling RTP SSRC values [RFC5576]. Thus, when multiplexing, these modes need to be used instead.

It is unclear how to signal this in a backward-compatible way (falling back to session-multiplexed modes) if SDP grouping semantics are used to described multiplexed sources in SDP.

7. Security Considerations

The security considerations of a muxed stream appear to be similar to those of multiple sources of the same media type in an RTP session.

Notably, it is crucial that SSRC values are never used more than once with the same SRTP keys.

8. IANA Considerations

The IANA actions required depend on the decision about how muxed streams are signaled.

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.

9.2. Informative References

- [I-D.holmberg-mmusic-sdp-bundle-negotiation] Holmberg, C. and H. Alvestrand, "Multiplexing Negotiation Using Session Description Protocol (SDP) Port Numbers", draft-holmberg-mmusic-sdp-bundle-negotiation-00 (work in progress), October 2011.
- [RFC3262] Rosenberg, J. and H. Schulzrinne, "Reliability of Provisional Responses in Session Initiation Protocol (SIP)", RFC 3262, June 2002.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.
- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.
- [RFC5109] Li, A., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, December 2007.
- [RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, February 2008.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.
- [RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.
- [RFC5583] Schierl, T. and S. Wenger, "Signaling Media Decoding Dependency in the Session Description Protocol (SDP)", RFC 5583, July 2009.
- [RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.
- [RFC5939] Andreasen, F., "Session Description Protocol (SDP) Capability Negotiation", RFC 5939, September 2010.

Authors' Addresses

Jonathan Lennox
Vidyo, Inc.
433 Hackensack Avenue
Seventh Floor
Hackensack, NJ 07601
US

Email: jonathan@vidyo.com

Jonathan Rosenberg
Skype

Email: jdrosen@skype.net
URI: <http://www.jdrosen.net>

This Internet-Draft, draft-singh-avt-mprtp-02.txt, has been replaced by another document, draft-singh-avtcore-mprtp, and has been deleted from the Internet-Drafts directory.

Internet-Drafts are not archival documents, and copies of Internet-Drafts that have been deleted from the directory are not available. The Secretariat does not have any information regarding the future plans of the author(s) or working group, if applicable, with respect to this deleted Internet-Draft. For more information, or to request a copy of the document, please contact the author(s) directly.

Draft Author(s):

V. Singh <>

T. Karkkainen <>

J. Ott <>

S. Ahsan <>

L. Eggert <lars.eggert@nokia.com>

Network Working Group
Internet-Draft
Intended status: BCP
Expires: April 26, 2012

M. Westerlund
B. Burman
Ericsson
C. Perkins
University of Glasgow
October 24, 2011

RTP Multiplexing Architecture
draft-westerlund-avtcore-multiplex-architecture-00

Abstract

RTP has always been a protocol that supports multiple participants each sending their own media streams in an RTP session. Thus relying on the three main multiplexing points in RTP; RTP session, SSRC and Payload Type for their various needs. However, most usages of RTP have been less complex often with a single SSRC in each direction, with a single RTP session per media type. But the more complex usages start to be more common and thus guidance on how to use RTP in various complex cases are needed. This document analyzes a number of cases and discusses the usage of the various multiplexing points and the need for functionality when defining RTP/RTCP extensions that utilize multiple RTP streams and multiple RTP sessions.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 26, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal

Provisions Relating to IETF Documents
(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
2. Definitions	5
2.1. Requirements Language	5
2.2. Terminology	5
3. RTP Multiplex Points	6
3.1. Session	6
3.2. SSRC	7
3.3. CSRC	8
3.4. Payload Type	8
4. Multiple Streams Alternatives	9
5. RTP Topologies and Issues	10
5.1. Point to Point	11
5.1.1. RTCP Reporting	11
5.1.2. Compound RTCP Packets	12
5.2. Point to Multipoint Using Multicast	12
5.3. Point to Multipoint Using an RTP Translator	14
5.4. Point to Multipoint Using an RTP Mixer	15
5.5. Point to Multipoint using Multiple Unicast flows	16
5.6. Decomposited End-Point	16
6. Dismissing Payload Type Multiplexing	18
7. Multiple Streams Discussion	20
7.1. Introduction	20
7.2. RTP/RTCP Aspects	20
7.2.1. The RTP Specification	20
7.2.2. Multiple SSRC Legacy Considerations	22
7.2.3. RTP Specification Clarifications Needed	23
7.2.4. Handling Varying sets of Senders	23
7.2.5. Cross Session RTCP requests	23
7.2.6. Binding Related Sources	23
7.2.7. Forward Error Correction	25
7.2.8. Transport Translator Sessions	26
7.2.9. Multiple Media Types in one RTP session	26
7.3. Signalling Aspects	28
7.3.1. Session Oriented Properties	28
7.3.2. SDP Prevents Multiple Media Types	29
7.4. Network Aspects	29

7.4.1.	Quality of Service	29
7.4.2.	NAT and Firewall Traversal	29
7.4.3.	Multicast	31
7.4.4.	Multiplexing multiple RTP Session on a Single Transport	32
7.5.	Security Aspects	32
7.5.1.	Security Context Scope	32
7.5.2.	Key-Management for Multi-party session	33
8.	Guidelines	33
9.	RTP Specification Clarifications	35
9.1.	RTCP Reporting from all SSRCs	35
9.2.	RTCP Self-reporting	35
9.3.	Combined RTCP Packets	35
10.	IANA Considerations	35
11.	Security Considerations	36
12.	Acknowledgements	36
13.	References	36
13.1.	Normative References	36
13.2.	Informative References	36
	Authors' Addresses	39

1. Introduction

This document focuses at issues of non-basic usage of RTP [RFC3550] where multiple media sources of the same media type are sent over RTP. Separation of different media types is another issue that will be discussed in this document. The intended uses include for example multiple sources from the same end-point, multiple streams from a single media source, multiple end-points each having a source, or an application that needs multiple representations (encodings) of a particular source. It will be shown that these uses are inter-related and need a common discussion to ensure consistency. In general, usage of the RTP session and media streams will be discussed in detail.

RTP is already designed for multiple participants in a communication session. This is not restricted to multicast, as many believe, but also provides functionality over unicast, using either multiple transport flows below RTP or a network node that re-distributes the RTP packets. The node can for example be a transport translator (relay) that forwards the packets unchanged, a translator performing media translation in addition to forwarding, or an RTP mixer that creates new conceptual sources from the received streams. In addition, multiple streams may occur when a single end-point have multiple media sources of the same media type, like multiple cameras or microphones that need to be sent simultaneously.

Historically, the most common RTP use cases have been point to point Voice over IP (VoIP) or streaming applications, commonly with no more than one media source per end-point and media type (typically audio and video). Even in conferencing applications, especially voice only, the conference focus or bridge has provided a single stream with a mix of the other participants to each participant. It is also common to have individual RTP sessions between each end-point and the RTP mixer.

SSRC is the RTP media stream identifier that helps to uniquely identify media sources in RTP sessions. Even though available SSRC space can theoretically handle more than 4 billion simultaneous sources, the perceived need for handling multiple SSRCs in implementations has been small. This has resulted in an installed legacy base that isn't fully RTP specification compliant and will have different issues if they receive multiple SSRCs of media, either simultaneously or in sequence. These issues will manifest themselves in various ways, either by software crashes or simply in limited functionality, like only decoding and playing back the first or latest received SSRC and discarding media related to any other SSRCs.

There have also arisen various cases where multiple SSRCs are used to

represent different aspects of what is in fact a single underlying media source. A very basic case is RTP retransmission [RFC4588] which have one SSRC for the original stream, and a second SSRC either in the same session or in a different session to represent the retransmitted packets to ensure that the monitoring functions still function. Another use case is scalable encoding, such as the RTP payload format for Scalable Video Coding (SVC) [RFC6190], which has an operation mode named Multiple Session Transmission (MST) that uses one SSRC in each RTP session to send one or more scalability layers. A third example is simulcast where a single media source is encoded in different versions and transmitted to an RTP mixer that picks which version to actually distribute to a given receiver part of the RTP session.

This situation has created a need for a document that discusses the existing possibilities in the RTP protocol and how these can and should be used in applications. A new set of applications needing more advanced functionalities from RTP is also emerging on the market, such as telepresence and advanced video conferencing. Thus furthering the need for a more common understanding of how multiple streams are handled in RTP to ensure media plane interoperability.

The document starts with some definitions and then goes into the existing RTP functionalities around multiplexing. Both the desired behavior and the implications of a particular behavior depend on which topologies are used, which requires some consideration. This is followed by a discussion of some choices in multiplexing behavior and their impacts. Finally, some recommendations and examples are provided.

2. Definitions

2.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2.2. Terminology

The following terms and abbreviations are used in this document:

End-point: A single entity sending or receiving RTP packets. It may be decomposed into several functional blocks, but as long as it behaves a single RTP stack entity it is classified as a single end-point.

Media Stream: A sequence of RTP packets using a single SSRC that together carry part or all of the content of a specific Media Type from a specific sender source within a given RTP session.

Media Aggregate: All Media Streams related to a particular Source.

Media Type: Audio, video, text or data whose form and meaning are defined by a specific real-time application.

Source: The source of a particular media stream. Either a single media capturing device such as a video camera, or a microphone, or a specific output of a media production function, such as an audio mixer, or some video editing function.

3. RTP Multiplex Points

This section describes the existing RTP tools that enable multiplexing of different media streams and RTP functionalities.

3.1. Session

The RTP Session is the highest semantic level in RTP and contains all of the RTP functionality.

RTP in itself does not contain any Session identifier, but relies on the underlying transport. For example, when running RTP on top of UDP, an RTP endpoint can identify and delimit an RTP Session from other RTP Sessions through the UDP source and destination transport address, consisting of network address and port number(s). Most commonly only the destination address, i.e. all traffic received on a particular port, is defined as belonging to a specific RTP Session. It is worth noting that in practice a more narrow definition of the transport flows that are related to a give RTP session is possible. An RTP session can for example be defined as one or more 5-tuples (Transport Protocol, Source Address, Source Port, Destination Address, Destination Port). Any set of identifiers of RTP and RTCP packet flows are sufficient to determine if the flow belongs to a particular session or not.

Commonly, RTP and RTCP use separate ports and the destination transport address is in fact an address pair, but in the case of RTP/RTCP multiplex [RFC5761] there is only a single port.

A source that changes its source transport address during a session must also choose a new SSRC identifier to avoid being interpreted as a looped source.

The set of participants considered part of the same RTP Session is defined by[RFC3550] as those that share a single SSRC space. That is, those participants that can see an SSRC identifier transmitted by any one of the other participants. A participant can receive an SSRC either as SSRC or CSRC in RTP and RTCP packets. Thus, the RTP Session scope is decided by the participants' network interconnection topology, in combination with RTP and RTCP forwarding strategies deployed by end-points and any interconnecting middle nodes.

3.2. SSRC

The Synchronization Source (SSRC) identifier is used to identify individual sources within an RTP Session. The SSRC number is globally unique within an RTP Session and all RTP implementations must be prepared to use procedures for SSRC collision handling, which results in an SSRC number change. The SSRC number is randomly chosen, carried in every RTP packet header and is not dependent on network address. SSRC is also used as identifier to refer to separate media streams in RTCP.

A media source having an SSRC identifier can be of different types:

Real: Connected to a "physical" media source, for example a camera or microphone.

Conceptual: A source with some attributed property generated by some network node, for example a filtering function in an RTP mixer that provides the most active speaker based on some criteria, or a mix representing a set of other sources.

Virtual: A source that does not generate any RTP media stream in itself (e.g. an end-point only receiving in an RTP session), but anyway need a sender SSRC for use as source in RTCP reports.

Note that a "multimedia source" that generates more than one media type, e.g. a conference participant sending both audio and video, need not (and commonly should not) use the same SSRC value across RTP sessions. RTCP Compound packets containing the CNAME SDES item is the designated method to bind an SSRC to a CNAME, effectively cross-correlating SSRCs within and between RTP Sessions as coming from the same end-point. The main property attributed to SSRCs associated with the same CNAME is that they are from a particular synchronization context and may be synchronized at playback. There exist a few other methods to relate different SSRC where use of CNAME is inappropriate, such as session-based RTP retransmission [RFC4588].

Note also that RTP sequence number and RTP timestamp are scoped by SSRC and thus independent between different SSRCs.

An RTP receiver receiving a previously unseen SSRC value must interpret it as a new source. It may in fact be a previously existing source that had to change SSRC number due to an SSRC conflict. However, the originator of the previous SSRC should have ended the conflicting source by sending an RTCP BYE for it prior to starting to send with the new SSRC, so the new SSRC is anyway effectively a new source.

Some RTP extension mechanisms already require the RTP stacks to handle additional SSRCs, like SSRC multiplexed RTP retransmission [RFC4588]. However, that still only requires handling a single media decoding chain per pair of SSRCs.

3.3. CSRC

The Contributing Source (CSRC) can arguably be seen as a sub-part of a specific SSRC and thus a multiplexing point. It is optionally included in the RTP header, shares the SSRC number space and specifies which set of SSRCs that has contributed to the RTP payload. However, even though each RTP packet and SSRC can be tagged with the contained CSRCs, the media representation of an individual CSRC is in general not possible to extract from the RTP payload since it is typically the result of a media mixing (merge) operation (by an RTP mixer) on the individual media streams corresponding to the CSRC identifiers. Due to these restrictions, CSRC will not be considered a fully qualified multiplex point and will be disregarded in the rest of this document.

3.4. Payload Type

The Payload Type number is also carried in every RTP packet header and identifies what format the RTP payload has. The term "format" here includes whatever can be described by out-of-band signaling means for dynamic payload types, as well as the statically allocated payload types in [RFC3551]. In SDP the term "format" includes media type, RTP timestamp sampling rate, codec, codec configuration, payload format configurations, and various robustness mechanisms such as redundant encodings [RFC2198].

The meaning of a Payload Type definition (the number) is re-used between all media streams within an RTP session, when the definition is either static or signaled through SDP. There however do exist cases where each end-point have different sets of payload types due to SDP offer/answer.

Although Payload Type definitions are commonly local to an RTP Session, there are some uses where Payload Type numbers need be unique across RTP Sessions. This is for example the case in Media

Decoding Dependency [RFC5583] where Payload Types are used to describe media dependency across RTP Sessions.

Given that multiple Payload Types are defined in an RTP Session, a media sender is free to change the Payload Type on a per packet basis. One example of designed per-packet change of Payload Type is a speech codec that makes use of generic Comfort Noise [RFC3389].

The RTP Payload Type in RTP is designed such that only a single Payload Type is valid at any time instant in the SSRC's timestamp time line, effectively time-multiplexing different Payload Types if any switch occurs. Even when this constraint is met, having different rates on the RTP timestamp clock for the RTP Payload Types in use in the same RTP Session have issues such as loss of synchronization. Payload Type clock rate switching requires some special consideration that is described in the multiple clock rates specification [I-D.ietf-avtext-multiple-clock-rates].

If there is a true need to send multiple Payload Types for the same SSRC that are valid for the same RTP Timestamps, then redundant encodings [RFC2198] can be used. Several additional constraints than the ones mentioned above need to be met to enable this use, one of which are that the combined payload sizes of the different Payload Types must not exceed the transport MTU.

Other aspects of RTP payload format use are described in RTP Payload HowTo [I-D.ietf-payload-rtp-howto].

4. Multiple Streams Alternatives

This section reviews the alternatives to enable multi-stream handling. Let's start with describing mechanisms that could enable multiple media streams, independent of the purpose for having multiple streams.

SSRC Multiplexing: Each additional Media Stream gets its own SSRC within a RTP Session.

Session Multiplexing: Using additional RTP Sessions to handle additional Media Streams

Payload Type Multiplexing: Using different RTP payload types for different additional streams.

Independent of the reason to use additional media streams, achieving it using payload type multiplexing is not a good choice as can be seen in the below section (Section 6). The RTP payload type alone is

not suitable for cases where additional media streams are required. Streams need their own SSRCs, so that they get their own sequence number space. The SSRC itself is also important so that the media stream can be referenced and reported on.

This leaves us with two choices, either using SSRC multiplexing to have multiple SSRCs from one end-point in one RTP session, or create additional RTP sessions to hold that additional SSRC. As the below discussion will show, in reality we cannot choose a single one of the two solutions. To utilize RTP well and as efficiently as possible, both are needed. The real issue is finding the right guidance on when to create RTP sessions and when additional SSRCs in an RTP session is the right choice.

In the below discussion, please keep in mind that the reasons for having multiple media streams vary and include but are not limited to the following:

- o Multiple Media Sources of the same media type
- o Retransmission streams
- o FEC stream
- o Alternative Encoding
- o Scalability layer

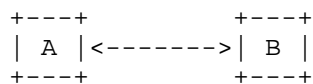
Thus the choice made due to one reason may not be the choice suitable for another reason. In the above list, the different items have different levels of maturity in the discussion on how to solve them. The clearest understanding is associated with multiple media sources of the same media type. However, all warrant discussion and clarification on how to deal with them.

5. RTP Topologies and Issues

The impact of how RTP Multiplex is performed will in general vary with how the RTP Session participants are interconnected; the RTP Topology [RFC5117]. This section describes the topologies and attempts to highlight the important behaviors concerning RTP multiplexing and multi-stream handling. It lists any identified issues regarding RTP and RTCP handling, and introduces additional topologies that are supported by RTP beyond those included in RTP Topologies [RFC5117]. The RTP Topologies that do not follow the RTP specification or do not attempt to utilize the facilities of RTP are ignored in this document.

5.1. Point to Point

This is the most basic use case with an RTP session containing of two end-points. Each end-point has one or more SSRCs.



Point to Point

5.1.1. RTCP Reporting

In cases when an end-point uses multiple SSRCs, we have found two closely related issues. The first is if every SSRC shall report on all other SSRC, even the ones originating from the same end-point. The reason for this would be ensure that no monitoring function should suspect a breakage in the RTP session.

The second issue around RTCP reporting arise when an end-point receives one or more media streams, and when the receiving end-point itself sends multiple SSRC in the same RTP session. As transport statistics are gathered per end-point and shared between the nodes, all the end-point's SSRC will report based on the same received data, the only difference will be which SSRCs sends the report. This could be considered unnecessary overhead, but for consistency it might be simplest to always have all sending SSRCs send RTCP reports on all media streams the end-point receives.

The current RTP text is silent about sending RTCP Receiver Reports for an endpoint's own sources, but does not preclude either sending or omitting them. The uncertainty in the expected behavior in those cases have likely caused variations in the implementation strategy. This could cause an interoperability issue where it is not possible to determine if the lack of reports are a true transport issue, or simply a result of implementation.

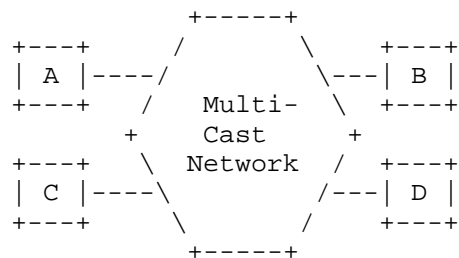
Although this issue is valid already for the simple point to point case, it needs to be considered in all topologies. From the perspective of an end-point, any solution needs to take into account what a particular end-point can determine without explicit information of the topology. For example, a Transport Translator (Relay) topology will look quite similar as point to point on an RTP level but is different. The main difference between a point to point with two SSRC being sent from the remote end-point and a Transport Translator with two single SSRC remote clients are that the RTT may vary between the SSRCs (but it is not guaranteed), and that the SSRCs may have different CNAMEs.

5.1.2. Compound RTCP Packets

When an end-point has multiple SSRCs and it needs to send RTCP packets on behalf of these SSRCs, the question arises if and how RTCP packets with different source SSRCs can be sent in the same compound packet. If it is allowed, then some consideration of the transmission scheduling is needed.

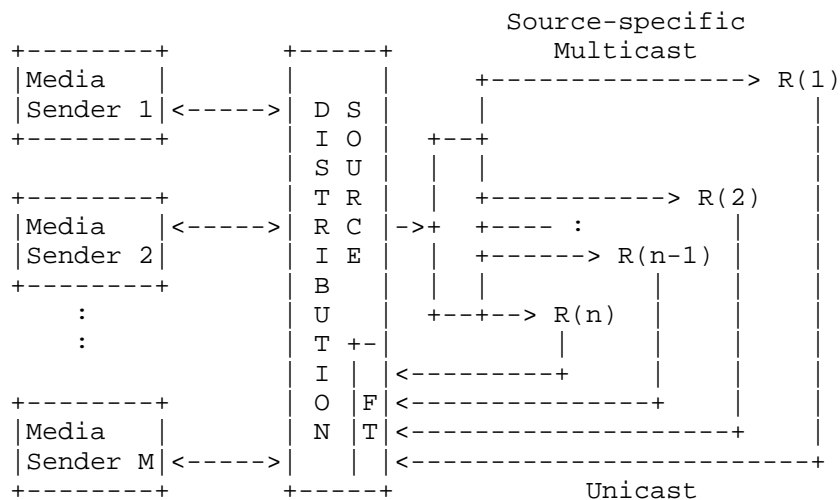
5.2. Point to Multipoint Using Multicast

This section discusses the Point to Multi-point using Multicast to interconnect the session participants. This needs to consider both Any Source Multicast (ASM) and Source-Specific Multicast (SSM).



Point to Multipoint Using Any Source Multicast

In Any Source Multicast, any of the participants can send to all the other participants, simply by sending a packet to the multicast group. That is not possible in Source Specific Multicast [RFC4607] where only a single source (Distribution Source) can send to the multicast group, creating a topology that looks like the one below:



FT = Feedback Target

Transport from the Feedback Target to the Distribution Source is via unicast or multicast RTCP if they are not co-located.

Point to Multipoint using Source Specific Multicast

In this topology a number of Media Senders (1 to M) are allowed to send media to the SSM group, sends media to the distribution source which then forwards the media streams to the multicast group. The media streams reach the Receivers (R(1) to R(n)). The Receiver's RTCP cannot be sent to the multicast group. To support RTCP, an RTP extension for SSM [RFC5760] was defined that use unicast transmission to send RTCP from the receivers to one or more Feedback Targets (FT).

As multicast is a one to many distribution system this must be taken into consideration. For example, the only practical method for adapting the bit-rate sent towards a given receiver is to use a set of multicast groups, where each multicast group represents a particular bit-rate. The media encoding is either scalable, where multiple layers can be combined, or simulcast where a single version is selected. By either selecting or combining multicast groups, the receiver can control the bit-rate sent on the path to itself. It is also common that transport robustification is sent in its own multicast group to allow for interworking with legacy or to support different levels of protection.

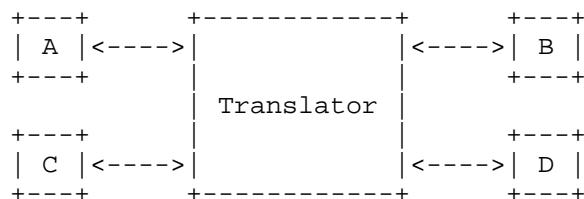
The result of this is three common behaviors for RTP multicast:

1. Use of multiple RTP sessions for the same media type.
2. The need for identifying RTP sessions that are related in one of several ways.
3. The need for binding related SSRCs in different RTP sessions together.

This indicates that Multicast is an important consideration when working with the RTP multiplexing and multi stream architecture questions. It is also important to note that so far there is no special mode for basic behavior between multicast and unicast usages of RTP. Yes, there are extensions targeted to deal with multicast specific cases but the general applicability does need to be considered.

5.3. Point to Multipoint Using an RTP Translator

Transport Translators (Relays) are a very important consideration for this document as they result in an RTP session situation that is very similar to how an ASM group RTP session would behave.



Transport Translator (Relay)

One of the most important aspects with the simple relay is that it is both easy to implement and require minimal amount of resources as only transport headers are rewritten, no RTP modifications nor media transcoding occur. Thus it is most likely the cheapest and most generally deployable method for multi-point sessions. The most obvious downside of this basic relaying is that the translator has no control over how many streams needs to be delivered to a receiver. Nor can it simply select to deliver only certain streams, at least not without new RTCP extensions to coherently handle the fact that some middlebox temporarily stops a stream, preventing some receivers from reporting on it. This consistency problem in RTCP reporting needs to be handled.

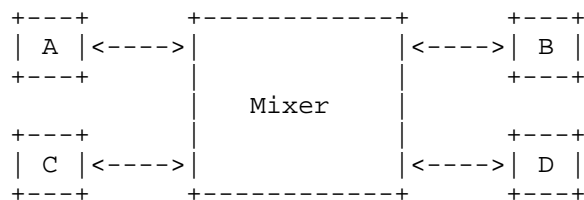
The Transport Translator does not need to have an SSRC of itself, nor need it send any RTCP reports on the flows that passes it, but it may choose to do that.

Use of a transport translator results in that any of the end-points will receive multiple SSRCs over a single unicast transport flow from the translator. That is independent of the other end-points having only a single or several SSRCs. End-points that have multiple SSRCs put further requirements on how SSRCs can be related or bound within and across RTP sessions and how they can be identified on an application level.

A Media Translator can perform a large variety of media functions affecting the media stream passing the translator, coming from one source and destined to a particular end-point. The media stream can be transcoded to a different bit-rate, change to another encoder, change the packetization of the media stream, add FEC streams, or terminate RTP retransmissions. The latter behaviors require the translator to use SSRCs that only exist in a particular sub-domain of the RTP session, and it may also create additional sessions when the mechanism applied on one side so requires.

5.4. Point to Multipoint Using an RTP Mixer

The most commonly used topology in centralized conferencing is based on the RTP Mixer. The main reason for this is that it provides a very consistent view of the RTP session towards each participant. That is accomplished through the mixer having its own SSRCs and any media sent to the participants will be sent using those SSRCs. If the mixer wants to identify the underlying media sources for its conceptual streams, it can identify them using CSRC. The media stream the mixer provides can be an actual media mixing of multiple media sources. It might also be as simple as selecting one of the underlying sources based on some mixer policy or control signalling.

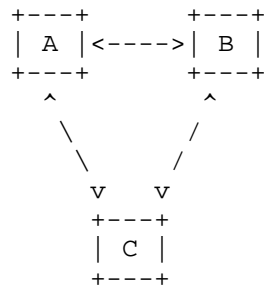


RTP Mixer

In the case where the mixer does stream selection, an application may in fact desire multiple simultaneous streams but only as many as the mixer can handle. As long as the mixer and an end-point can agree on the maximum number of streams and how the streams that are delivered are selected, this provides very good functionality. As these streams are forwarded using the mixer's SSRCs, there are no inconsistencies within the session.

5.5. Point to Multipoint using Multiple Unicast flows

Based on the RTP session definition, it is clearly possible to have a joint RTP session over multiple transport flows like the below three end-point joint session. In this case, A needs to send its' media streams and RTCP packets to both B and C over their respective transport flows. As long as all participants do the same, everyone will have a joint view of the RTP session.

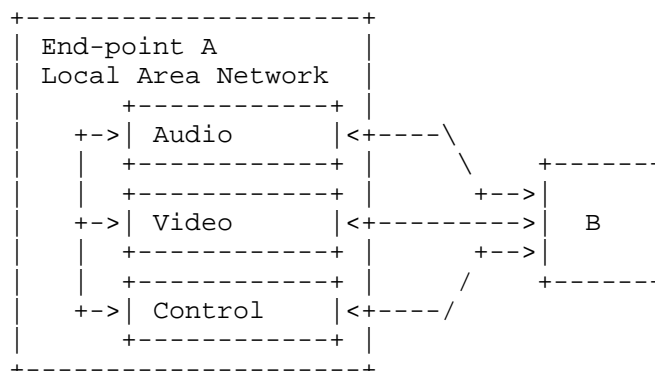


Point to Multi-Point using Multiple Unicast Transports

This doesn't create any additional requirements beyond the need to have multiple transport flows associated with a single RTP session. Note that an end-point may use a single local port to receive all these transport flows, or it might have separate local reception ports for each of the end-points.

5.6. Decomposited End-Point

There is some possibility that an RTP end-point implementation in fact reside on multiple devices, each with their own network address. A very basic use case for this would be to separate audio and video processing for a particular end-point, like a conference room, into one device handling the audio and another handling the video being interconnected by some control functions allowing them to behave as a single end-point.

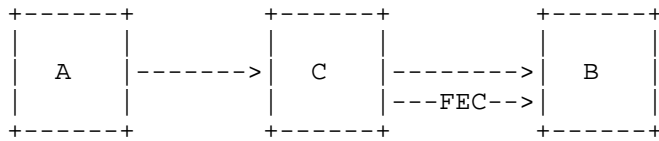


Decomposited End-Point

In the above usage, let us assume that the RTP sessions are different for audio and video. The audio and video parts will use a common CNAME and also have a common clock to ensure that synchronization and clock drift handling works despite the decomposition. However, if the audio and video were in a single RTP session then this use case becomes problematic. This as all transport flow receivers will need to receive all the other media streams that are part of the session. Thus the audio component will receive also all the video media streams, while the video component will receive all the audio ones, thus doubling the site's bandwidth requirements from all other session participants. With a joint RTP session it also becomes evident that a given end-point, as interpreted from a CNAME perspective, has two sets of transport flows for receiving the streams and the decomposition isn't hidden.

The requirements that can be derived from the above usage is that the transport flows for each RTP session might be under common control but still go to what looks like different end-points based on addresses and ports. A conclusion can also be reached that decomposition without using separate RTP sessions has downsides and potential for RTP/RTCP issues.

There exist another use case which might be considered as a decomposited end-point. However, as will be shown this should be considered a translator instead. An example of this is when an end-point A sends a media flow to B. On the path there is a device C that on A's behalf does something with the media streams, for example adds an RTP session with FEC information for A's media streams. C will in this case need to bind the new FEC streams to A's media stream by using the same CNAME as A.



When Decomposition is a Translator

This type of functionality where C does something with the media stream on behalf of A is clearly covered under the media translator definition (Section 5.3).

6. Dismissing Payload Type Multiplexing

Before starting a discussion on when to use what alternative, we will first document a number of reasons why using the payload type as a multiplexing point for anything related to multiple streams is unsuitable and will not be considered further.

If one attempts to use Payload type multiplexing beyond it's defined usage, that has well known negative effects on RTP. To use Payload type as the single discriminator for multiple streams implies that all the different media streams are being sent with the same SSRC, thus using the same timestamp and sequence number space. This has many effects:

1. Putting restraint on RTP timestamp rate for the multiplexed media. For example, media streams that use different RTP timestamp rates cannot be combined, as the timestamp values need to be consistent across all multiplexed media frames. Thus streams are forced to use the same rate. When this is not possible, Payload Type multiplexing cannot be used.
2. Many RTP payload formats may fragment a media object over multiple packets, like parts of a video frame. These payload formats need to determine the order of the fragments to correctly decode them. Thus it is important to ensure that all fragments related to a frame or a similar media object are transmitted in sequence and without interruptions within the object. This can relatively simple be solved on the sender side by ensuring that the fragments of each media stream are sent in sequence.
3. Some media formats require uninterrupted sequence number space between media parts. These are media formats where any missing RTP sequence number will result in decoding failure or invoking of a repair mechanism within a single media context. The text/

T140 payload format [RFC4103] is an example of such a format. These formats will need a sequence numbering abstraction function between RTP and the individual media stream before being used with Payload Type multiplexing.

4. Sending multiple streams in the same sequence number space makes it impossible to determine which Payload Type and thus which stream a packet loss relates to.
5. If RTP Retransmission [RFC4588] is used and there is a loss, it is possible to ask for the missing packet(s) by SSRC and sequence number, not by Payload Type. If only some of the Payload Type multiplexed streams are of interest, there is no way of telling which missing packet(s) belong to the interesting stream(s) and all lost packets must be requested, wasting bandwidth.
6. The current RTCP feedback mechanisms are built around providing feedback on media streams based on stream ID (SSRC), packet (sequence numbers) and time interval (RTP Timestamps). There is almost never a field to indicate which Payload Type is reported, so sending feedback for a specific media stream is difficult without extending existing RTCP reporting.
7. The current RTCP media control messages [RFC5104] specification is oriented around controlling particular media flows, i.e. requests are done addressing a particular SSRC. Such mechanisms would need to be redefined to support Payload Type multiplexing.
8. The number of payload types are inherently limited. Accordingly, using Payload Type multiplexing limits the number of streams that can be multiplexed and does not scale. This limitation is exacerbated if one uses solutions like RTP and RTCP multiplexing [RFC5761] where a number of payload types are blocked due to the overlap between RTP and RTCP.
9. At times, there is a need to group multiplexed streams and this is currently possible for RTP Sessions and for SSRC, but there is no defined way to group Payload Types.
10. It is currently not possible to signal bandwidth requirements per media stream when using Payload Type Multiplexing.
11. Most existing SDP media level attributes cannot be applied on a per Payload Type level and would require re-definition in that context.

12. A legacy end-point that doesn't understand the indication that different RTP payload types are different media streams may be slightly confused by the large amount of possibly overlapping or identically defined RTP Payload Types.

7. Multiple Streams Discussion

7.1. Introduction

Using multiple media streams is a well supported feature of RTP. However, what can be unclear for most implementors or people writing RTP/RTCP extensions attempting to apply multiple streams, is when it is most appropriate to add an additional SSRC in an existing RTP session and when it is better to use multiple RTP sessions. This section tries to discuss the various considerations needed. The next section then concludes with some guidelines.

7.2. RTP/RTCP Aspects

This section discusses RTP and RTCP aspects worth considering when selecting between SSRC multiplexing and Session multiplexing.

7.2.1. The RTP Specification

RFC 3550 contains some recommendations and a bullet list with 5 arguments for different aspects of RTP multiplexing. Let's review Section 5.2 of [RFC3550], reproduced below:

"For efficient protocol processing, the number of multiplexing points should be minimized, as described in the integrated layer processing design principle [ALF]. In RTP, multiplexing is provided by the destination transport address (network address and port number) which is different for each RTP session. For example, in a teleconference composed of audio and video media encoded separately, each medium SHOULD be carried in a separate RTP session with its own destination transport address.

Separate audio and video streams SHOULD NOT be carried in a single RTP session and demultiplexed based on the payload type or SSRC fields. Interleaving packets with different RTP media types but using the same SSRC would introduce several problems:

1. If, say, two audio streams shared the same RTP session and the same SSRC value, and one were to change encodings and thus acquire a different RTP payload type, there would be no general way of identifying which stream had changed encodings.

2. An SSRC is defined to identify a single timing and sequence number space. Interleaving multiple payload types would require different timing spaces if the media clock rates differ and would require different sequence number spaces to tell which payload type suffered packet loss.
3. The RTCP sender and receiver reports (see Section 6.4) can only describe one timing and sequence number space per SSRC and do not carry a payload type field.
4. An RTP mixer would not be able to combine interleaved streams of incompatible media into one stream.
5. Carrying multiple media in one RTP session precludes: the use of different network paths or network resource allocations if appropriate; reception of a subset of the media if desired, for example just audio if video would exceed the available bandwidth; and receiver implementations that use separate processes for the different media, whereas using separate RTP sessions permits either single- or multiple-process implementations.

Using a different SSRC for each medium but sending them in the same RTP session would avoid the first three problems but not the last two.

On the other hand, multiplexing multiple related sources of the same medium in one RTP session using different SSRC values is the norm for multicast sessions. The problems listed above don't apply: an RTP mixer can combine multiple audio sources, for example, and the same treatment is applicable for all of them. It may also be appropriate to multiplex streams of the same medium using different SSRC values in other scenarios where the last two problems do not apply."

Let's consider one argument at a time. The first is an argument for using different SSRC for each individual media stream, which still is very applicable.

The second argument is advocating against using payload type multiplexing, which still stands as can be seen by the extensive list of issues found in Section 6.

The third argument is yet another argument against payload type multiplexing.

The fourth is an argument against multiplexing media streams that require different handling into the same session. This is to simplify the processing at any receiver of the media stream. If all media streams that exist in an RTP session is of one media type and

one particular purpose, there is no need for deeper inspection of the packets before processing them in both end-points and RTP aware middle nodes.

The fifth argument discusses network aspects that we will discuss more below in Section 7.4. It also goes into aspects of implementation, like decomposed end-points where different processes or inter-connected devices handle different aspects of the whole multi-media session.

A summary of RFC 3550's view on multiplexing is to use unique SSRCs for anything that is its' own media/packet stream, and secondly use different RTP sessions for media streams that don't share media type and purpose, to maximize flexibility when it comes to processing and handling of the media streams.

This mostly agrees with the discussion and recommendations in this document. However, there has been an evolution of RTP since that text was written which needs to be reflected in the discussion. Additional clarifications for specific cases are also needed.

7.2.2. Multiple SSRC Legacy Considerations

When establishing RTP sessions that may contain end-points that aren't updated to handle multiple streams following these recommendations, a particular application can have issues with multiple SSRCs within a single session. These issues include:

1. Need to handle more than one stream simultaneously rather than replacing an already existing stream with a new one.
2. Be capable of decoding multiple streams simultaneously.
3. Be capable of rendering multiple streams simultaneously.

RTP Session multiplexing could potentially avoid these issues if there is only a single SSRC at each end-point, and in topologies which appears like point to point as seen the end-point. However, forcing the usage of session multiplexing due to this reason would be a great mistake, as it is likely that a significant set of applications will need a combination of SSRC multiplexing of several media sources and session multiplexing for other aspects such as encoding alternatives, robustification or simply to support legacy. However, this issue does need consideration when deploying multiple media streams within an RTP session where legacy end-points may occur.

7.2.3. RTP Specification Clarifications Needed

The RTP specification contains a few things that are potential interoperability issues when using multiple SSRCs within a session. These issues are described and discussed in Section 9. These should not be considered strong arguments against using SSRC multiplexing when otherwise appropriate, and there are some issues we expect to be solved in the near future.

7.2.4. Handling Varying sets of Senders

Another potential issue that needs to be considered is where a limited set of simultaneously active sources varies within a larger set of session members. As each media decoding chain may contain state, it is important that this type of usage ensures that a receiver can flush a decoding state for an inactive source and if that source becomes active again, it does not assume that this previous state exists.

This behavior might in certain applications be possible to limit to a particular RTP Session and instead use multiple RTP sessions. But in some cases it is likely unavoidable and the most appropriate thing is to SSRC multiplex.

7.2.5. Cross Session RTCP requests

There currently exist no functionality to make truly synchronized and atomic RTCP requests across multiple RTP Sessions. Instead separate RTCP messages will have to be sent in each session. This gives SSRC multiplexed streams a slight advantage as RTCP requests for different streams in the same session can be sent in a compound RTCP packet. Thus providing an atomic operation if different modifications of different streams are requested at the same time.

In Session multiplexed cases, the RTCP timing rules in the sessions and the transport aspects, such as packet loss and jitter, prevents a receiver from relying on atomic operations, instead more robust and forgiving mechanisms need to be used.

7.2.6. Binding Related Sources

A common problem in a number of various RTP extensions has been how to bind together related sources. This issue is common independent of SSRC multiplexing and Session Multiplexing, and any solution and recommendation to the problem should work equally well for both to avoid creating barriers between using session multiplexing and SSRC multiplexing.

The current solutions don't have these properties. There exist one solution for grouping RTP session together in SDP [RFC5888] to know which RTP session contains for example the FEC data for the source data in another session. However, this mechanism does not work on individual media flows and is thus not directly applicable to the problem. The other solution is also SDP based and can group SSRCs within a single RTP session [RFC5576]. Thus this mechanism can bind media streams in SSRC multiplexed cases. Both solutions have the shortcoming of being restricted to SDP based signalling and also do not work in cases where the session's dynamic properties are such that it is difficult or resource consuming to keep the list of related SSRCs up to date.

One possible solution could be to mandate the same SSRC being used in all RTP session in case of session multiplexing. We do note that Section 8.3 of the RTP Specification [RFC3550] recommends using a single SSRC space across all RTP sessions for layered coding. However this recommendation has some downsides and is less applicable beyond the field of layered coding. To use the same sender SSRC in all RTP sessions from a particular end-point can cause issues if an SSRC collision occurs. If the same SSRC is used as the required binding between the streams, then all streams in the related RTP sessions must change their SSRC. This is extra likely to cause problems if the participant populations are different in the different sessions. For example, in case of large number of receivers having selected totally random SSRC values in each RTP session as RFC 3550 specifies, a change due to a SSRC collision in one session can then cause a new collision in another session. This cascading effect is not severe but there is an increased risk that this occurs for well populated sessions. In addition, being forced to change the SSRC affects all the related media streams; instead of having to re-synchronize only the originally conflicting stream, all streams will suddenly need to be re-synchronized with each other. This will prevent also the media streams not having an actual collision from being usable during the re-synchronization and also increases the time until synchronization is finalized. In addition, it requires exception handling in the SSRC generation.

The above collision issue does not occur in case of having only one SSRC space across all sessions and all participants will be part of at least one session, like the base layer in layered encoding. In that case the only downside is the special behavior that needs to be well defined by anyone using this. But, having an exception behavior where the SSRC space is common across all session and that doesn't fit all the RTP extensions or payload formats present in the sessions is a issue. It is possible to create a situation where the different mechanisms can't be combined due to the non standard SSRC allocation behavior.

Existing mechanisms with known issues:

RTP Retransmission (RFC4588): Has two modes, one for SSRC multiplexing and one for Session multiplexing. The session multiplexing requires the same CNAME and mandates that the same SSRC is used in both sessions. Using the same SSRC does work but will potentially have issues in certain cases. In SSRC multiplexed mode the CNAME is used, and when the first retransmission request is sent, one must not have another retransmission request outstanding for an SSRC which don't have a the binding between the original SSRC and the retransmission stream's SSRC. This works but creates some limitations that can be avoided by a more explicit mechanism. The SDP based ssrc-group mechanism is sufficient in this case as long as the application can rely on the signalling based solution.

Scalable Video Coding (RFC6190): As an example of scalable coding, SVC [RFC6190] has various modes. The Multi Session Transmission (MST) uses Session multiplexing to separate scalability layers. However, this specification has failed to explicit how these layers are bound together in cases where CNAME isn't sufficient. CNAME is no longer sufficient when more than one media source occur within a session that have the same CNAME, for example due to multiple video cameras capturing the same lecture hall. This likely implies that a single SSRC space as recommend by Section 8.3 of RTP [RFC3550] is to be used.

Forward Error Correction: If some type of FEC or redundancy stream is being sent, it needs it's own SSRC, with the exception of constructions like redundancy encoding [RFC2198]. Thus in case of transmitting the FEC in the same session as the source data, the inter SSRC relation within a session is needed. In case of sending the redundant data in a separate session from the source, the SSRC in each session needs to be related. This occurs for example in RFC5109 when using session separation of original and FEC data. SSRC multiplexing is not supported, only using redundant encoding is supported.

This issue appears to need action to harmonize and avoid future shortcomings in extension specifications. A proposed solution for handling this issue is [I-D.westerlund-avtext-rtcp-sdes-srcname].

7.2.7. Forward Error Correction

There exist a number of Forward Error Correction (FEC) based schemes for how to reduce the packet loss of the original streams. Most of the FEC schemes will protect a single source flow. The protection is achieved by transmitting a certain amount of redundant information

that is encoded such that it can repair one or more packet loss over the set of packets they protect. This sequence of redundant information also needs to be transmitted as its own media stream, or in some cases instead of the original media stream. Thus many of these schemes creates a need for binding the related flows as discussed above. They also create additional flows that need to be transported. Looking at the history of these schemes, there is both SSRC multiplexed and Session multiplexed solutions and some schemes that support both.

Using a Session multiplexed solution provides good support for legacy when deploying FEC or changing the scheme used so that some set of receivers may not be able to utilize the FEC information. By placing it in a separate RTP session, it can easily be ignored.

In usages involving multicast, having the FEC information on its own multicast group and RTP session allows for flexibility, for example when using Rapid Acquisition of Multicast Groups (RAMS) [RFC6285]. During the RAMS burst where data is received over unicast and where it is possible to combine with unicast based retransmission [RFC4588], there is no need to burst the FEC data related to the burst of the source media streams needed to catch up with the multicast group. This saves bandwidth to the receiver during the burst, enabling quicker catch up. When the receiver has caught up and joins the multicast group(s) for the source, it can at the same time join the multicast group with the FEC information. Having the source stream and the FEC in separate groups allow for easy separation in the Burst/Retransmission Source (BRS) without having to individually classify packets.

7.2.8. Transport Translator Sessions

A basic Transport Translator relays any incoming RTP and RTCP packets to the other participants. The main difference between SSRC multiplexing and Session multiplexing resulting from this use case is that for SSRC multiplexing it is not possible for a particular session participant to decide to receive a subset of media streams. When using separate RTP sessions for the different sets of media streams, a single participant can choose to leave one of the sessions but not the other.

7.2.9. Multiple Media Types in one RTP session

Having different media types, like audio and video, in the same RTP sessions is not forbidden, only recommended against as can be seen in Section 7.2.1. When using multiple media types, there are a number of considerations:

Payload Type gives Media Type: This solution is dependent on getting the media type from the Payload Type. Thus overloading this demultiplexing point in a receiver for two purposes. First for the main media type and determining the processing chain, then later for the exact configuration of the encoder and packetization.

Payload Type field limitations: The total number of Payload Types available to use in an RTP session is fairly limited, especially if Multiplexing RTP Data and Control Packets on a Single Port [RFC5761] is used. For certain applications negotiating a large set of codes and configuration may become an issue.

Don't switch media types for an SSRC: The primary reasons to avoid switching from sending for example audio to sending video using the same SSRC is the implications on a receiver. When this happens, the processing chain in the receiver will have to switch from one media type to another. As the different media type's entire processing chains are different and are connected to different outputs it is difficult to reuse the decoding chain, which a normal codec change likely can. Instead the entire processing chain has to be torn down and replaced. In addition, there is likely a clock rate switching problem, possibly resulting in synchronization loss at the point of switching media type if some packet loss occurs.

RTCP Bit-rate Issues: If the media types are significantly different in bit-rate, the RTCP bandwidth rates assigned to each source in a session can result in interesting effects, like that the RTCP bit-rate share for an audio stream is larger than the actual audio bit-rate. In itself this doesn't cause any conflicts, only potentially unnecessary overhead. It is possible to avoid this using AVPF or SAVPF and setting trr-int parameter, which can bring down unnecessary regular reporting while still allowing for rapid feedback.

Decomposited end-points: Decomposited nodes that rely on the regular network to separate audio and video to different devices do not work well with this session setup. If they are forced to work, all media receiver parts of a decomposited end-point will receive all media, thus doubling the bit-rate consumption for the end-point.

RTP Mixers and Translators: An RTP mixer or Media Translator will also have to support this particular session setup, where it before could rely on the RTP session to determine what processing options should be applied to the incoming packets.

As can be seen, there is nothing in here that prevents using a single

RTP session for multiple media types, however it does create a number of limitations and special case implementation requirements. So anyone considering to use this setup should carefully review if the reasons for using a single RTP session is sufficient to motivate this special case.

7.3. Signalling Aspects

There exist various signalling solutions for establishing RTP sessions. Many are SDP [RFC4566] based, however SDP functionality is also dependent on the signalling protocols carrying the SDP. Where RTSP [RFC2326] and SAP [RFC2974] both use SDP in a declarative fashion, SIP [RFC3261] uses SDP with the additional definition of Offer/Answer [RFC3264]. The impact on signalling and especially SDP needs to be considered as it can greatly affect how to deploy a certain multiplexing point choice.

7.3.1. Session Oriented Properties

One aspect of the existing signalling is that it is focused around sessions, or at least in the case of SDP the media description. There are a number of things that are signalled on a session level/media description but that are not necessarily strictly bound to an RTP session and could be of interest to signal specifically for a particular media stream within the session. The following properties have been identified as being potentially useful to signal not only on RTP session level:

- o Bitrate/Bandwidth exist today only at aggregate or a common any media stream limit
- o Which SSRC that will use which RTP Payload Types

Some of these issues are clearly SDP's problem rather than RTP limitations. However, if the aim is to deploy an SSRC multiplexed solution that contains several sets of media streams with different properties (encoding/packetization parameter, bit-rate, etc), putting each set in a different RTP session would directly enable negotiation of the parameters for each set. If insisting on SSRC multiplexing, a number of signalling extensions are needed to clarify that there are multiple sets of media streams with different properties and that they shall in fact be kept different, since a single set will not satisfy the applications requirements.

This does in fact create a strong driver to use RTP session multiplexing for any case where different sets of media streams with different requirements exist.

7.3.2. SDP Prevents Multiple Media Types

SDP encoded in its structure a prevention against using multiple media types in the same RTP session. A media description in SDP can only have a single media type; audio, video, text, image, application. This media type is used as the top-level media type for identifying the actual payload format bound to a particular payload type using the rtpmap attribute. Thus a high fence against using multiple media types in the same session was created.

There is a proposal in the MMUSIC WG for how one could allow multiple media lines describe a single underlying transport [I-D.holmberg-mmusic-sdp-bundle-negotiation] and thus support either one RTP session with multiple media types. There is also a solution for multiplexing multiple RTP sessions onto the same transport [I-D.westerlund-avtcore-single-transport-multiplexing].

7.4. Network Aspects

The multiplexing choice has impact on network level mechanisms that need to be considered by the implementor.

7.4.1. Quality of Service

When it comes to Quality of Service mechanisms, they are either flow based or marking based. RSVP [RFC2205] is an example of a flow based mechanism, while Diff-Serv [RFC2474] is an example of a Marking based one. For a marking based scheme, the method of multiplexing will not affect the possibility to use QoS.

However, for a flow based scheme there is a clear difference between the methods. SSRC multiplexing will result in all media streams being part of the same 5-tuple (protocol, source address, destination address, source port, destination port) which is the most common selector for flow based QoS. Thus, separation of the level of QoS between media streams is not possible. That is however possible for session based multiplexing, where each different version can be in a different RTP session that can be sent over different 5-tuples.

7.4.2. NAT and Firewall Traversal

In today's network there exist a large number of middleboxes. The ones that normally have most impact on RTP are Network Address Translators (NAT) and Firewalls (FW).

Below we analyze and comment on the impact of requiring more underlying transport flows in the presence of NATs and Firewalls:

End-Point Port Consumption: A given IP address only has 65536 available local ports per transport protocol for all consumers of ports that exist on the machine. This is normally never an issue for an end-user machine. It can become an issue for servers that handle large number of simultaneous streams. However, if the application uses ICE to authenticate STUN requests, a server can serve multiple end-points from the same local port, and use the whole 5-tuple (source and destination address, source and destination port, protocol) as identifier of flows after having securely bound them to the remote end-point address using the STUN request. In theory the minimum number of media server ports needed are the maximum number of simultaneous RTP Sessions a single end-point may use. In practice, implementation will probably benefit from using more server ports to simplify implementation or avoid performance bottlenecks.

NAT State: If an end-point sits behind a NAT, each flow it generates to an external address will result in a state that has to be kept in the NAT. That state is a limited resource. In home or Small Office/Home Office (SOHO) NATs, memory or processing are usually the most limited resources. For large scale NATs serving many internal end-points, available external ports are typically the scarce resource. Port limitations is primarily a problem for larger centralized NATs where end-point independent mapping requires each flow to use one port for the external IP address. This affects the maximum number of internal users per external IP address. However, it is worth pointing out that a real-time video conference session with audio and video is likely using less than 10 UDP flows, compared to certain web applications that can use 100+ TCP flows to various servers from a single browser instance.

NAT Traversal Excess Time: Making the NAT/FW traversal takes a certain amount of time for each flow. It also takes time in a phase of communication between accepting to communicate and the media path being established which is fairly critical. The best case scenario for how much extra time it can take following the specified ICE procedures are: $1.5 * RTT + T_a * (Additional_Flows - 1)$, where T_a is the pacing timer, which ICE specifies to be no smaller than 20 ms. That assumes a message in one direction, and then an immediate triggered check back. This as ICE first finds one candidate pair that works prior to establish multiple flows. Thus, there are no extra time until one has found a working candidate pair. Based on that working pair the extra time is to in parallel establish the, in most cases 2-3, additional flows.

NAT Traversal Failure Rate: Due to the need to establish more than a single flow through the NAT, there is some risk that establishing the first flow succeeds but that one or more of the additional flows fail. The risk that this happens is hard to quantify, but it should be fairly low as one flow from the same interfaces has just been successfully established. Thus only rare events such as NAT resource overload, or selecting particular port numbers that are filtered etc, should be reasons for failure.

SSRC multiplexing keeps additional media streams within one RTP Session and does not introduce any additional NAT traversal complexities per media stream. In contrast, the session multiplexing is using one RTP session per media stream. Thus additional lower layer transport flows will be required, unless an explicit de-multiplexing layer is added between RTP and the transport protocol. A proposal for how to multiplex multiple RTP sessions over the same single lower layer transport exist in [I-D.westerlund-avtcore-single-transport-multiplexing].

7.4.3. Multicast

Multicast groups provides a powerful semantics for a number of real-time applications, especially the ones that desire broadcast-like behaviors with one end-point transmitting to a large number of receivers, like in IPTV. But that same semantics do result in a certain number of limitations.

One limitation is that for any group, sender side adaptation to the actual receiver properties causes a degradation for all participants to what is supported by the receiver with the worst conditions among the group participants. In most cases this is not acceptable. Instead various receiver based solutions are employed to ensure that the receivers achieve best possible performance. By using scalable encoding and placing each scalability layer in a different multicast group, the receiver can control the amount of traffic it receives. To have each scalability layer on a different multicast group, one RTP session per multicast group is used.

If instead a single RTP session over multiple transports were to be deployed, i.e. multicast groups with each layer as it's own SSRC, then very different views of the RTP session would exist. That as one receiver may see only a single layer (SSRC), while another may see three SSRCs if it joined three multicast groups. This would cause disjoint RTCP reports where a management system would not be able to determine if a receiver isn't reporting on a particular SSRC due to that it is not a member of that multicast group, or because it doesn't receive it as a result of a transport failure.

Thus it appears easiest and most straightforward to use multiple RTP sessions. In addition, the transport flow considerations in multicast are a bit different from unicast. First of all there is no shortage of port space, as each multicast group has its own port space.

7.4.4. Multiplexing multiple RTP Session on a Single Transport

For applications that doesn't need flow based QoS and like to save ports and NAT/FW traversal costs, there is a proposal for how to achieve multiplexing of multiple RTP sessions over the same lower layer transport [I-D.westerlund-avtcore-single-transport-multiplexing]. Using such a solution would allow session multiplexing without most of the perceived downsides of additional RTP sessions creating a need for additional transport flows.

7.5. Security Aspects

On the basic level there is no significant difference in security when having one RTP session and having multiple. However, there are a few more detailed considerations that might need to be considered in certain usages.

7.5.1. Security Context Scope

When using SRTP [RFC3711] the security context scope is important and can be a necessary differentiation in some applications. As SRTP's crypto suites (so far) is built around symmetric keys, the receiver will need to have the same key as the sender. This results in that none in a multi-party session can be certain that a received packet really was sent by the claimed sender or by another party having access to the key. In most cases this is a sufficient security property, but there are a few cases where this does create situations.

The first case is when someone leaves a multi-party session and one wants to ensure that the party that left can no longer access the media streams. This requires that everyone re-keys without disclosing the keys to the excluded party.

A second case is when using security as an enforcing mechanism for differentiation. Take for example a scalable layer or a high quality simulcast version which only premium users are allowed to access. The mechanism preventing a receiver from getting the high quality stream can be based on the stream being encrypted with a key that user can't access without paying premium, having the key-management limit access to the key.

In the latter case it is likely easiest from signalling, transport (if done over multicast) and security to use a different RTP session. That way the user(s) not intended to receive a particular stream can easily be excluded. There is no need to have SSRC specific keys, which many of the key-management systems cannot handle.

7.5.2. Key-Management for Multi-party session

Performing key-management for Multi-party session can be a challenge. This section considers some of the issues.

Transport translator based session cannot use Security Description [RFC4568] nor DTLS-SRTP [RFC5764] without an extension as each end-point provides it's set of keys. In centralized conference, the signalling counterpart is a conference server and the media plane unicast counterpart (to which DTLS messages would be sent) is the translator. Thus an extension like Encrypted Key Transport [I-D.ietf-avt-srtp-ekt] are needed or a MIKEY [RFC3830] based solution that allows for keying all session participants with the same master key.

Keying of multicast transported SRTP face similar challenges as the transport translator case.

8. Guidelines

This section contains a number of recommendations for implementors or specification writers when it comes to handling multi-stream.

Don't Require the same SSRC across Sessions: As discussed in Section 7.2.6 there exist drawbacks in using the same SSRC in multiple RTP sessions as a mechanism to bind related media streams together. Instead a mechanism to explicitly signal the relation SHOULD be used, either in RTP/RTCP or in the used signalling mechanism that establish the RTP session(s).

Use SSRC multiplexing for additional Media Sources: In the cases an RTP end-point needs to transmit additional media source(s) of the same media type and purpose in the application it is RECOMMENDED to send them as additional SSRCs in the same RTP session. For example a telepresence room where there are three cameras, and each camera captures 2 persons sitting at the table, sending each camera as its own SSRC within a single RTP session is recommended.

Use additional RTP sessions for streams with different purposes:
When media streams have different purpose or processing requirements it is RECOMMENDED that the different types of streams are put in different RTP sessions.

When using Session Multiplexing use grouping: When using Session Multiplexing solutions it is RECOMMENDED to be explicitly group the involved RTP sessions using the signalling mechanism, for example The Session Description Protocol (SDP) Grouping Framework. [RFC5888]

RTP/RTCP Extensions May Support SSRC and Session Multiplexing: When defining an RTP or RTCP extension, the creator needs to consider if this extension is applicable in both SSRC multiplexed and Session multiplexed usages. If it is, then any generic extensions are RECOMMENDED to support both. Applications that are not as generally applicable will have to consider if interoperability is better served by defining a single solution or providing both options.

Transport Support Extensions: When defining new RTP/RTCP extensions intended for transport support, like the retransmission or FEC mechanisms, they are RECOMMENDED to include support for both SSRC and Session multiplexing so that application developers can choose freely from the set of mechanisms without concerning themselves with if a particular solution only supports one of the multiplexing choices.

This discussion and guidelines points out that a small set of extension mechanisms could greatly improve the situation when it comes to using multiple streams independently of Session multiplexing or SSRC multiplexing. These extensions are:

Media Source Identification: A Media source identification that can be used to bind together media streams that are related to the same media source. A proposal [I-D.westerlund-avtext-rtcp-sdes-srcname] exist for a new SDES item SRCNAME that also can be used with the a=ssrc SDP attribute to provide signalling layer binding information.

SSRC limitations within RTP sessions: By providing a signalling solution that allows the signalling peers to explicitly express both support and limitations on how many simultaneous media streams an end-point can handle within a given RTP Session. That ensures that usage of SSRC multiplexing occurs when supported and without overloading an end-point. This extension is proposed in [I-D.westerlund-avtcore-max-ssrc].

9. RTP Specification Clarifications

This section describes a number of clarifications to the RTP specifications that are likely necessary for aligned behavior when RTP sessions contains more SSRCS than one local and one remote.

9.1. RTCP Reporting from all SSRCS

When one have multiple SSRC in an RTP node, then all these SSRC must send RTCP SR or RR as long as the SSRC exist. It is not sufficient that only one SSRC in the node sends report blocks on the incoming RTP streams. The reason for this is that a third party monitor may not necessarily be able to determine that all these SSRC are in fact co-located and originate from the same stack instance that gather report data.

9.2. RTCP Self-reporting

For any RTP node that sends more than one SSRC, there exist the question if SSRC1 needs to report its reception of SSRC2 and vice versa. The reason that they in fact need to report on all other local streams as being received is report consistency. A third party monitor that considers the full matrix of media streams and all known SSRC reports on these media streams would detect a gap in the reports which could be a transport issue unless identified as in fact being sources from same node.

9.3. Combined RTCP Packets

When a node contains multiple SSRCS, it is questionable if an RTCP compound packet can only contain RTCP packets from a single SSRC or if multiple SSRCS can include their packets in a joint compound packet. The high level question is a matter for any receiver processing on what to expect. In addition to that question there is the issue of how to use the RTCP timer rules in these cases, as the existing rules are focused on determining when a single SSRC can send.

10. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

11. Security Considerations

12. Acknowledgements

13. References

13.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

13.2. Informative References

[ALF] Clark, D. and D. Tennenhouse, "Architectural Considerations for a New Generation of Protocols", SIGCOMM Symposium on Communications Architectures and Protocols (Philadelphia, Pennsylvania), pp. 200--208, IEEE Computer Communications Review, Vol. 20(4), September 1990.

[I-D.holmberg-mmusic-sdp-bundle-negotiation]
Holmberg, C. and H. Alvestrand, "Multiplexing Negotiation Using Session Description Protocol (SDP) Port Numbers", draft-holmberg-mmusic-sdp-bundle-negotiation-00 (work in progress), October 2011.

[I-D.ietf-avt-srtp-ekt]
McGrew, D., Andreasen, F., Wing, D., and K. Fischer, "Encrypted Key Transport for Secure RTP", draft-ietf-avt-srtp-ekt-02 (work in progress), March 2011.

[I-D.ietf-avtext-multiple-clock-rates]
Petit-Huguenin, M., "Support for multiple clock rates in an RTP session", draft-ietf-avtext-multiple-clock-rates-01 (work in progress), July 2011.

[I-D.ietf-payload-rtp-howto]
Westerlund, M., "How to Write an RTP Payload Format", draft-ietf-payload-rtp-howto-01 (work in progress), July 2011.

[I-D.westerlund-avtcore-max-ssrc]

Westerlund, M., Burman, B., and F. Jansson, "Multiple Synchronization sources (SSRC) in RTP Session Signaling", draft-westerlund-avtcore-max-ssrc (work in progress), October 2011.

[I-D.westerlund-avtcore-single-transport-multiplexing]
Westerlund, M., "Multiple RTP Session on a Single Lower-Layer Transport", draft-westerlund-avtcore-transport-multiplexing (work in progress), October 2011.

[I-D.westerlund-avtext-rtcp-sdes-srcname]
Westerlund, M., Burman, B., and P. Sandgren, "RTCP SDES Item SRCNAME to Label Individual Sources", draft-westerlund-avtext-rtcp-sdes-srcname (work in progress), October 2011.

[RFC2198] Perkins, C., Kouvelas, I., Hodson, O., Hardman, V., Handley, M., Bolot, J., Vega-Garcia, A., and S. Fosse-Parisis, "RTP Payload for Redundant Audio Data", RFC 2198, September 1997.

[RFC2205] Braden, B., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

[RFC2326] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, October 2000.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[RFC3389] Zopf, R., "Real-time Transport Protocol (RTP) Payload for Comfort Noise (CN)", RFC 3389, September 2002.

- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.
- [RFC3830] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and K. Norrman, "MIKEY: Multimedia Internet KEYing", RFC 3830, August 2004.
- [RFC4103] Hellstrom, G. and P. Jones, "RTP Payload for Text Conversation", RFC 4103, June 2005.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC4568] Andreassen, F., Baugher, M., and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, July 2006.
- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.
- [RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, August 2006.
- [RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, February 2008.
- [RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, January 2008.
- [RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.
- [RFC5583] Schierl, T. and S. Wenger, "Signaling Media Decoding Dependency in the Session Description Protocol (SDP)", RFC 5583, July 2009.
- [RFC5760] Ott, J., Chesterfield, J., and E. Schooler, "RTP Control Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback", RFC 5760, February 2010.

- [RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.
- [RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, May 2010.
- [RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.
- [RFC6190] Wenger, S., Wang, Y., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, May 2011.
- [RFC6285] Ver Steeg, B., Begen, A., Van Caenegem, T., and Z. Vax, "Unicast-Based Rapid Acquisition of Multicast RTP Sessions", RFC 6285, June 2011.

Authors' Addresses

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

Phone: +46 10 714 82 87
Email: magnus.westerlund@ericsson.com

Bo Burman
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

Phone: +46 10 714 13 11
Email: bo.burman@ericsson.com

Colin Perkins
University of Glasgow
School of Computing Science
Glasgow G12 8QQ
United Kingdom

Email: csp@csp Perkins.org

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2012

M. Westerlund
Ericsson
C. Perkins
University of Glasgow
October 31, 2011

Multiple RTP Session on a Single Lower-Layer Transport
draft-westerlund-avtcore-transport-multiplexing-01

Abstract

This document specifies how multiple RTP sessions are to be multiplexed on the same lower-layer transport, e.g. a UDP flow. It discusses various requirements that have been raised and their feasibility, which results in a solution with a certain applicability. A solution is recommended and that solution is provided in more detail, including signalling and examples.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 3, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Conventions	3
2.1. Terminology	3
2.2. Requirements Language	3
3. Requirements	3
3.1. Support Use of Multiple RTP Sessions	4
3.2. Same SSRC Value in Multiple RTP Sessions	4
3.3. SRTP	5
3.4. Don't Redefine Used Bits	6
3.5. Firewall Friendly	6
3.6. Monitoring and Reporting	6
3.7. Usable Also Over Multicast	6
3.8. Incremental Deployment	7
4. Possible Solutions	7
4.1. Header Extension	7
4.2. Multiplexing Shim	8
4.3. Single Session	9
4.4. Use the SRTP MKI field	10
4.5. Use an Octet in the Padding	11
4.6. Redefine the SSRC field	11
5. Recommendation	12
6. Specification	12
6.1. Shim Layer	12
6.2. Signalling	16
6.3. SRTP Key Management	17
6.3.1. Security Description	17
6.3.2. DTLS-SRTP	18
6.3.3. MIKEY	18
6.4. Examples	18
6.4.1. RTP Packet with Transport Header	18
6.4.2. SDP Offer/Answer example	19
7. Open Issues	21
8. IANA Considerations	22
9. Security Considerations	22
10. Acknowledgements	22
11. References	22
11.1. Normative References	22
11.2. Informational References	23
Authors' Addresses	24

1. Introduction

There has been renewed interest for having a solution that allows multiple RTP sessions [RFC3550] to use a single lower layer transport, such as a bi-directional UDP flow. The main reason is the cost of doing NAT/FW traversal for each individual flow. ICE and other NAT/FW traversal solutions are clearly capable of attempting to open multiple flows. However, there is both increased risk for failure and an increased cost in the creation of multiple flows. The increased cost comes as slightly higher delay in establishing the traversal, and the amount of consumed NAT/FW resources. The latter might be an increasing problem in the IPv4 to IPv6 transition period.

This document draws up some requirements for consideration on how to transport multiple RTP sessions over a single lower-layer transport. These requirements will have to be weighted as the combined set of requirements result in that no known solution exist that can fulfill them completely.

A number of possible solutions are then considered and discussed with respect to their properties. Based on that, the authors recommends a shim layer variant as single solution, which is described in more detail including signalling solution and examples.

2. Conventions

2.1. Terminology

Some terminology used in this document.

Multiplexing: Unless specifically noted, all mentioning of multiplexing in this document refer to the multiplexing of multiple RTP Sessions on the same lower layer transport. It is important to make this distinction as RTP does contain a number of multiplexing points for various purposes, such as media formats (Payload Type), media sources (SSRC), and RTP sessions.

2.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Requirements

This section lists and discusses a number of potential requirements.

However, it is not difficult to realize that it is in fact possible to put requirements that makes the set of feasible solutions an empty set. It is thus necessary to consider which requirements that are essential to fulfill and which can be compromised on to arrive at a solution.

3.1. Support Use of Multiple RTP Sessions

This may at first glance appear to be an obvious requirement. Although the authors are convinced it is a mandatory requirement for a solution, it warrants some discussion around the implications of not having multiple RTP sessions and instead use a single RTP session.

The main purpose of RTP sessions is to allow separation of streams that have different purposes, for example different media types. A big reason for establishing this is the knowledge that any SSRC within the session is supposed to be processed in a similar way.

For simpler cases, where the streams within each media type need the same processing, it is clearly possible to find other multiplex solutions, for example based on the Payload Type and the differences in encoding that the payload type allows to describe. This may anyhow be insufficient when you get into more advanced usages where you have multiple sources of the same media type, but for different purposes or as alternatives. For example when you have one set of video sources that shows session participants and another set of video sources that shares an application or slides, you likely want to separate those streams for various reasons such as control, prioritization, QoS, methods for robustification, etc. In those cases, using the RTP session for separation of properties is a powerful tool. A tool with properties that need to be preserved when providing a solution for how to use only a single lower-layer transport.

For more discussion of the usage of RTP sessions verses other multiplexing we recommend RTP Multiplexing Architecture [I-D.westerlund-avtcore-multiplex-architecture].

3.2. Same SSRC Value in Multiple RTP Sessions

Two different RTP sessions being multiplexed on the same lower layer transport need to be able to use the same SSRC value. This is a strong requirement, for two reasons:

1. To avoid mandating SSRC assignment rules that are coordinated between the sessions. If the RTP sessions multiplexed together must have unique SSRC values, then additional code that works

between RTP Sessions is needed in the implementations. Thus raising the bar for implementing this solution. In addition, if one gateways between parts of a system using this multiplexing and parts that aren't multiplexing, the part that isn't multiplexing must also fulfil the requirements on how SSRC is assigned or force the gateway to translate SSRCs. Translating SSRC is actually hard as it requires one to understand the semantics of all current and future RTP and RTCP extensions. Otherwise a barrier for deploying new extensions is created.

2. There are some few RTP extensions that currently rely on being able to use the same SSRC in different RTP sessions:

- * XOR FEC (RFC5109)
- * RTP Retransmission in session mode (RFC4588)
- * Certain Layered Coding

3.3. SRTP

SRTP [RFC3711] is one of the most commonly used security solutions for RTP. In addition, it is the only one recommended by IETF that is integrated into RTP. This integration has several aspects that needs to be considered when designing a solution for multiplexing RTP sessions on the same lower layer transport.

Determining Crypto Context: SRTP first of all needs to know which session context a received or to-be-sent packet relates to. It also normally relies on the lower layer transport to identify the session. It uses the MKI, if present, to determine which key set is to be used. Then the SSRC and sequence number are used by most crypto suites, including the most common use of AES Counter Mode, to actually generate the correct cipher stream.

Unencrypted Headers: SRTP has chosen to leave the RTP headers and the first two 32-bit words of the first RTCP header unencrypted, to allow for both header compression and monitoring to work also in the presence of encryption. As these fields are in clear text they are used in most crypto suites for SRTP to determine how to protect or recover the plain text.

It is here important to contrast SRTP against a set of other possible protection mechanisms. DTLS, TLS, and IPsec are all protecting and encapsulating the entire RTP and RTCP packets. They don't perform any partial operations on the RTP and RTCP packets. Any change that is considered to be part of the RTP and RTCP packet is transparent to them, but possibly not to SRTP. Thus the impact on SRTP operations

must be considered when defining a mechanism.

3.4. Don't Redefine Used Bits

As the core of RTP is in use in many systems and has a really large deployment story and numerous implementations, changing any of the field definitions is highly problematic. First of all, the implementations need to change to support this new semantics. Secondly, you get a large transition issue when you have some session participants that support the new semantics and some that don't. Combining the two behaviors in the same session can force the deployment of costly and less than perfect translation devices.

3.5. Firewall Friendly

It is desirable that current firewalls will accept the solutions as normal RTP packets. However, in the authors' opinion we can't let the firewall stifle invention and evolution of the protocol. It is also necessary to be aware that a change that will make most deep inspecting firewall consider the packet as not valid RTP/RTCP will have more difficult deployment story.

3.6. Monitoring and Reporting

It is desirable that a third party monitor can still operate on the multiplexed RTP Sessions. It is however likely that they will require an update to correctly monitor and report on multiplexed RTP Sessions.

Another type of function to consider is packet sniffers and their selector filters. These may be impacted by a change of the fields. An observation is that many such systems are usually quite rapidly updated to consider new types of standardized or simply common packet formats.

3.7. Usable Also Over Multicast

It is desirable that a solution should be possible to use also when RTP and RTCP packets are sent over multicast, both Any Source Multicast (ASM) and Single Source Multicast (SSM). The reason for this requirement is to allow a system using RTP to use the same configuration regardless of the transport being done over unicast or multicast. In addition, multicast can't be claimed to have an issue with using multiple ports, as each multicast group has a complete port space scoped by address.

3.8. Incremental Deployment

A good solution has the property that in topologies that contains RTP mixers or Translators, a single session participant can enable multiplexing without having any impact on any other session participants. Thus a node should be able to take a multiplexed packet and then easily send it out with minimal or no modification on another leg of the session, where each RTP session is transported over its own lower-layer transport. It should also be as easy to do the reverse forwarding operation.

4. Possible Solutions

This section looks at a few possible solutions and discusses their feasibility.

4.1. Header Extension

One proposal is to define an RTP header extension [RFC5285] that explicitly enumerates the session identifier in each packet. This proposal has some merits regarding RTP, since it uses an existing extension mechanism; it explicitly enumerates the session allowing for third parties to associate the packet to a given RTP session; and it works with SRTP as currently defined since a header extension is by default not encrypted, and is thus readable by the receiving stack without needing to guess which session it belongs to and attempt to decrypt it. This approach does, however, conflict with the requirement from [RFC5285] that "header extensions using this specification MUST only be used for data that can be safely ignored by the recipient", since correct processing of the received packet depends on using the header extension to demultiplex it to the correct RTP session.

Using a header extension also result in the session ID is in the integrity protected part of the packet. Thus a translator between multiplexed and non-multiplexed has the options:

1. to be part of the security context to verify the field
2. to be part of the security context to verify the field and remove it before forwarding the packet
3. to be outside of the security context and leave the header extension in the packet. However, that requires successful negotiation of the header extension, but not of the functionality, with the receiving end-points.

The biggest existing hurdle for this solution is that there exist no header extension field in the RTCP packets. This requires defining a solution for RTCP that allows carrying the explicit indicator, preferably in a position that isn't encrypted by SRTCP. However, the current SRTCP definition does not offer such a position in the packet.

Modifying the RR or SR packets is possible using profile specific extensions. However, that has issues when it comes to deployability and in addition any information placed there would end up in the encrypted part.

Another alternative could be to define another RTCP packet type that only contains the common header, using the 5 bits in the first byte of the common header to carry a session id. That would allow SRTCP to work correctly as long it accepts this new packet type being the first in the packet. Allowing a non-SR/RR packet as the first packet in a compound RTCP packet is also needed if an implementation is to support Reduced Size RTCP packets [RFC5506]. The remaining downside with this is that all stack implementations supporting multiplexing would need to modify its RTCP compound packet rules to include this packet type first. Thus a translator box between supporting nodes and non-supporting nodes needs to be in the crypto context.

This solution's per packet overhead is expected to be 64-bits for RTCP. For RTP it is 64-bits if no header extension was otherwise used, and an additional 16 bits (short header), or 24 bits plus (if needed) padding to next 32-bits boundary if other header extensions are used.

4.2. Multiplexing Shim

This proposal is to prefix or postfix all RTP and RTCP packets with a session ID field. This field would be outside of the normal RTP and RTCP packets, thus having no impact on the RTP and RTCP packets and their processing. An additional step of demultiplexing processing would be added prior to RTP stack processing to determine in which RTP session context the packet shall be included. This has also no impact on SRTCP/SRTCP as the shim layer would be outside of its protection context. The shim layer's session ID is however implicitly integrity protected as any error in the field will result in the packet being placed in the wrong or non-existing context, thus resulting in a integrity failure if processed by SRTCP/SRTCP.

This proposal is quite simple to implement in any gateway or translating device that goes from a multiplexed to a non-multiplexed domain or vice versa, as only an additional field needs to be added to or removed from the packet.

The main downside of this proposal is that it is very likely to trigger a firewall response from any deep packet inspection device. If the field is prefixed, the RTP fields are not matching the heuristics field (unless the shim is designed to look like an RTP header, in which case the payload length is unlikely to match the expected value) and thus are likely preventing classification of the packet as an RTP packet. If it is postfixed, it is likely classified as an RTP packet but may not correctly validate if the content validation is such that the payload length is expected to match certain values. It is expected that a postfixed shim will be less problematic than a prefixed shim in this regard, but we are lacking hard data on this.

This solution's per packet overhead is 1 byte.

4.3. Single Session

Given the difficulty of multiplexing several RTP sessions onto a single lower-layer transport, it's tempting to send multiple media streams in a single RTP session. Doing this avoids the need to demultiplex several sessions on a single transport, but at the cost of losing the RTP session as a separator for different type of streams. Lacking different RTP sessions to demultiplex incoming packets, a receiver will have to dig deeper into the packet before determining what to do with it. Care must be taken in that inspection. For example, you must be careful to ensure that each real media source uses its own SSRC in the session and that this SSRC doesn't change media type.

The loss of the RTP session as a purpose separator is likely not a big issue if the only difference between RTP Sessions is the media type. In this case, you can use the Payload Type field to identify the media type. The loss of the RTP Session functionality is more severe, however, if you actually use the RTP Session for separating different treatments, contexts etc. Then you would need additional signalling to bind the different sources to groups which can help make the necessary distinctions.

This approach has been proposed in the RTCWeb context in [I-D.lennox-rtcweb-rtp-media-type-mux] and [I-D.holmberg-mmusic-sdp-bundle-negotiation]. These drafts describe how to signal multiple media streams multiplexed into a single RTP session, and address some of the issues raised here and in Section 7.2.9 of the RTP Multiplexing Architecture [I-D.westerlund-avtcore-multiplex-architecture] draft. However, they fail to discuss maybe the largest issue with this solution: how to do incremental deployment and transition.

Many transition scenarios use an RTP translator as a gateway between a single RTP session containing multiple media types multiplexed together, and several separate RTP sessions each using a single media type. In this scenario, it is possible that a legacy device that uses one RTP session for each media type will use the same SSRC in each session. When translating these into a single RTP session, it will be necessary to rewrite one of the SSRCs, so that each stream has a unique SSRC. This SSRC translation process is straight-forward for RTP packets, but is very complex for RTCP packets. It also hinders innovation, since such a gateway will not be able to translate new RTCP extensions that it is unaware of, even if they are supported by devices on both sides of the gateway.

This method has several limitations that makes it unsuitable as general mechanism to provide multiple RTP sessions on the same lower layer transport. However, we acknowledge that there are some uses for which this method may be sufficient and which can accept the method limitations and other downsides. The RTCWEB WG has a working assumption to support this method. For more details of this method, see the relevant drafts under development.

This solution has no per packet overhead. The signalling overhead will be a different question.

4.4. Use the SRTP MKI field

This proposal is to overload the MKI SRTP/SRTCP identifier to not only identify a particular crypto context, but also identify the actual RTP Session. This clearly is a miss use of the MKI field, however it appears to be with little negative implications. SRTP already supports handling of multiple crypto contexts.

The two major downsides with this proposal is first the fact that it requires using SRTP/SRTCP to multiplex multiple sessions on a single lower layer transport. The second issue is that the session ID parameter needs to be put into the various key-management schemes and to make them understand that the reason to establish multiple crypto contexts is because they are connected to various RTP Sessions. Considering that SRTP have at least 3 used keying mechanisms, DTLS-SRTP [RFC5764], Security Descriptions [RFC4568], and MIKEY [RFC3830], this is not an insignificant amount of work.

This solution has 32-bit per packet overhead, but only if the MKI was not already used.

4.5. Use an Octet in the Padding

The basics of this proposal is to have the RTP packet and the last (required by RFC3550) RTCP packet in a compound to include padding, at least 2 bytes. One byte for the padding count (last byte) and one byte just before the padding count containing the session ID.

This proposal uses bytes to carry the session ID that have no defined value and is intended to be ignored by the receiver. From that perspective it only causes packet expansion that is supported and handled by all existing equipment. If an implementation fails to understand that it is required to interpret this padding byte to learn the session ID, it will see a mostly coherent RTP session except where SSRCs overlap or where the payload types overlap. However, reporting on the individual sources or forwarding the RTCP RR are not completely without merit.

There is one downside of this proposal and that has to do with SRTP. To be able to determine the crypto context, it is necessary to access to the encrypted payload of the packet. Thus, the only mechanism available for a receiver to solve this issue is to try the existing crypto contexts for any session on the same lower layer transport and then use the one where the packet decrypts and verifies correctly. Thus for transport flows with many crypto contexts, an attacker could simply generate packets that don't validate to force the receiver to try all crypto contexts they have rather than immediately discard it as not matching a context. A receiver can mitigate this somewhat by using hueristics based on the RTP header fields to determine which context applies for a received packet, but this is not a complete solution.

This solution has a 16-bit per packet overhead.

4.6. Redefine the SSRC field

The Rosenberg et. al. Internet draft "Multiplexing of Real-Time Transport Protocol (RTP) Traffic for Browser based Real-Time Communications (RTC)" [I-D.rosenberg-rtcweb-rtpmux] proposed to redefine the SSRC field. This has the advantage of no packet expansion. It also looks like regular RTP. However, it has a number of implications. First of all it prevents any RTP functionality that require the same SSRC in multiple RTP sessions.

Secondly its interoperability with normal RTP is problematic. Such interoperability requires an SSRC translator function in the gateway to ensure that the SSRCs fulfill the requirements of the different domains. That translator is actually far from easy as it needs to understand the semantics of all RTP and RTCP extensions that include

SSRC/CSRC. This as it is necessary to know when a particular matching 32-bit pattern is an SSRC field and when the field is just a combination of other fields that create the same matching 32-bit pattern. Thus any future RTCP extension might not work through the translator, causing a barrier for deployment of future extensions.

This solution has no per packet overhead.

5. Recommendation

Considering these options, the authors would recommend that AVTCORE standardize a solution based on a postfixed multiplexing field, i.e. a shim approach combined with the appropriate signalling as described in Section 4.2.

6. Specification

This section contains the specification of the solution based on a SHIM, with the explicit session identifier at the end of the encapsulated payload.

6.1. Shim Layer

This solution is based on a shim layer that is inserted in the stack between the regular RTP and RTCP packets and the transport layer being used by the RTP sessions. Thus the layering looks like the following:

```
+-----+
| RTP / RTCP Packet |
+-----+
| Session ID Layer   |
+-----+
| Transport layer    |
+-----+
```

Stack View with Session ID SHIM

The above stack is in fact a layered one as it does allow multiple RTP Sessions to be multiplexed on top of the Session ID shim layer. This enables the example presented in Figure 1 where four sessions, S1-S4 is sent over the same Transport layer and where the Session ID layer will combine and encapsulate them with the session ID on transmission and separate and decapsulate them on reception.

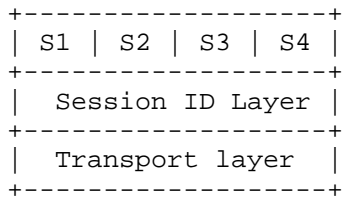
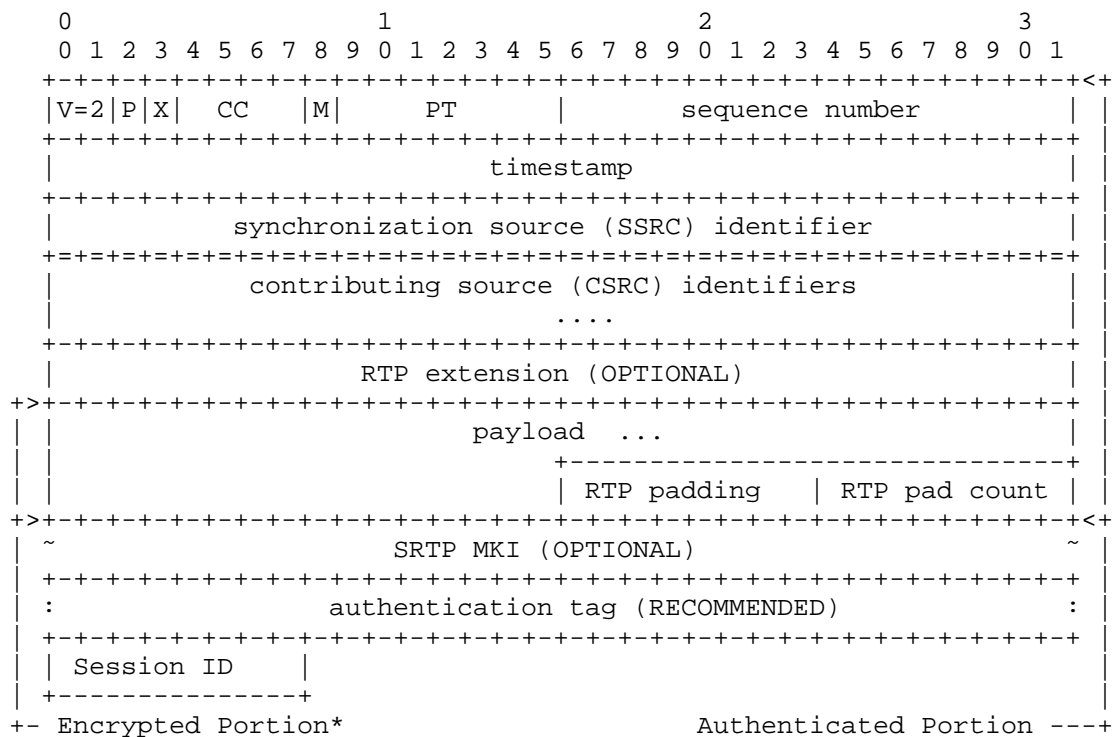


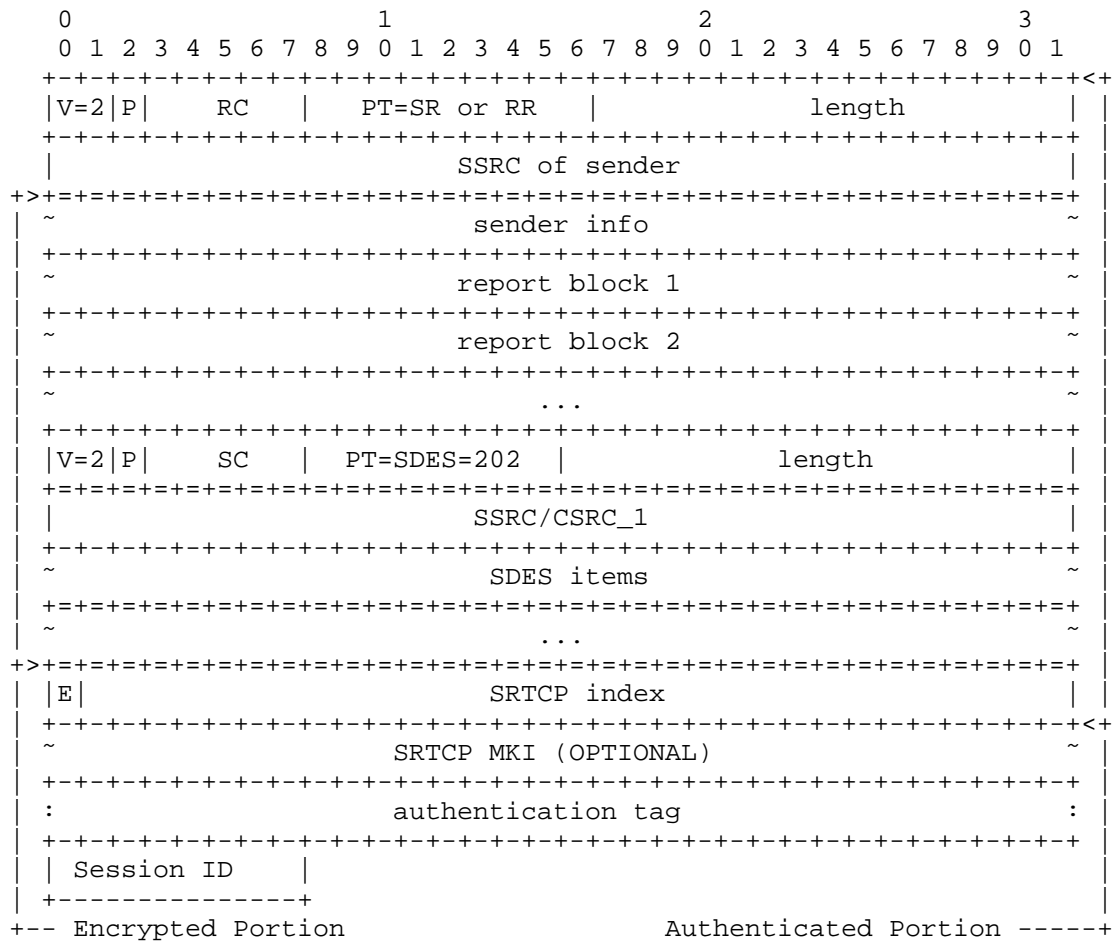
Figure 1: Multiple RTP Session On Top of Session ID Layer

The Session ID layer encapsulates one RTP or RTCP packet from a given RTP session and postfixes a one byte Session ID (SID) field to the packet. Each RTP session being multiplexed on top of a given transport layer is assigned either a single or a pair of unique SID in the range 0-255. The reason for assigning a pair of SIDs to a given RTP session are for RTP Sessions that doesn't support "Multiplexing RTP Data and Control Packets on a Single Port" [RFC5761] to still be able to use a single 5-tuple. The reasons for supporting this extra functionality is that RTP and RTCP multiplexing based on the payload type/packet type fields enforces certain restrictions on the RTP sessions. These restrictions may not be acceptable. As this solution does not have these restrictions, performing RTP and RTCP multiplexing in this way has benefits.

Each Session ID value space is scoped by the underlying transport protocol. Common transport protocols like UDP, DCCP, TCP, and SCTP can all be scoped by one or more 5-tuple (Transport protocol, source address and port, destination address and port). The case of multiple 5-tuples occur in the case of multi-unicast topologies, also called meshed multiparty RTP sessions.



SRTP Packet encapsulated by Session ID Layer



SRTCP packet encapsulated by Session ID layer

The processing in a receiver when the Session ID layer is present will be to

1. Pick up the packet from the lower layer transport
2. Inspect the SID field value
3. Strip the SID field from the packet
4. Forward it to the (S)RTP Session context identified by the SID value

6.2. Signalling

The use of the Session ID layer needs to be explicitly agreed on between the communicating parties. Each RTP Session the application uses must in addition to the regular configuration such as payload types, RTCP extension etc, have both the underlying 5-tuple (source address and port, destination address and port, and transport protocol) and the Session ID used for the particular RTP session. The signalling requirement is to assign unique Session ID values to all RTP Sessions being sent over the same 5-tuple. The same Session ID shall be used for an RTP session independently of the traffic direction. Note that nothing prevents a multi-media application from using multiple 5-tuples if desired for some reason, in which case each 5-tuple has its own session ID value space.

This section defines how to negotiate the use of the Session ID layer, using the Session Description Protocol (SDP) Offer/Answer mechanism [RFC3264]. A new media-level SDP attribute, 'session-mux-id', is defined, in order to be used with the media BUNDLE mechanism defined in [I-D.holmberg-mmusic-sdp-bundle-negotiation]. The attribute allows each media description ("m=" line) associated with a 'BUNDLE' group to form a separate RTP session.

The 'session-mux-id' attribute is included for a media description, in order to indicate the Session ID for that particular media description. Every media description that shares a common attribute value is assumed to be part of a single RTP session. An SDP Offerer MUST include the 'session-mux-id' attribute for every media description associated with a 'BUNDLE' group. If the SDP Answer does not contain 'session-mux-id' attributes, the SDP Offerer MUST NOT assume that separate RTP sessions will be used. If the SDP Answer still describes a 'BUNDLE' group, the procedures in [I-D.holmberg-mmusic-sdp-bundle-negotiation] apply.

An SDP Answerer MUST NOT include the 'session-mux-id' attribute in an SDP Answer, unless included in the SDP Offer.

The attribute has the following ABNF [RFC5234] definition.

```
Session-mux-id-attr = "a=session-mux-id:" SID *SID-prop
SID                  = SID-value / SID-pairs
SID-value            = 1*3DIGIT / "NoN"
SID-pairs            = SID-value "/" SID-value ; RTP/RTCP SIDs
SID-prop            = SP assignment-policy / prop-ext
prop-ext            = token "=" value
assignment-policy    = "policy=" ("tentative" / "fixed")
```

The following parameters MUST be configured as specified:

- o RTP Profile SHOULD be the same, but MUST be compatible, like AVP and AVPF.
- o RTCP bandwidth parameters are the same
- o RTP Payload type values are not overlapping

In declarative SDP usage, there is clearly no method for fallback unless some other negotiation protocol is used.

The SID property "policy" is used in negotiation by an end-point to indicate if the session ID values are merely a tentative suggestion or if they must have these values. This is used when negotiating SID for multi-party RTP sessions to support shared transports such as multicast or RTP translators that are unable to produce renumbered SIDs on a per end-point basis. The normal behavior is that the offer suggest a tentative set of values, indicated by "policy=tentative". These SHOULD be accepted by the peer unless that peer negotiate session IDs on behalf of a centralized policy, in which case it MAY change the value(s) in the answer. If the offer represents a policy that does not allow changing the session ID values, it can indicate that to the answerer by setting the policy to "fixed". This enables the answering peer to either accept the value or indicate that there is a conflict in who is performing the assignment by setting the SID value to NoN (Not a Number). Offerer and answerer SHOULD always include the policy they are operating under. Thus, in case of no centralized behaviors, both offerer and answerer will indicate the tentative policy.

6.3. SRTP Key Management

Key management for SRTP do needs discussion as we do cause multiple SRTP sessions to exist on the same underlying transport flow. Thus we need to ensure that the key management mechanism still are properly associated with the SRTP session context it intends to key. To ensure that we do look at the three SRTP key management mechanism that IETF has specified, one after another.

6.3.1. Security Description

Session Description Protocol (SDP) Security Descriptions for Media Streams [RFC4568] as being based on SDP has no issue with the RTP session multiplexing on lower layer specified here. The reason is that the actual keying is done using a media level SDP attribute. Thus the attribute is already associated with a particular media description. A media description that also will have an instance of

the "a=session-mux-id" attribute carrying the SID value/pair used with this particular crypto parameters.

6.3.2. DTLS-SRTP

Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP) [RFC5764] is a keying mechanism that works on the media plane on the same lower layer transport that SRTP/SRTCP will be transported over. Thus each DTLS message must be associated with the SRTP and/or SRTCP flow it is keying.

The most direct solution is to use the SHIM and the SID context identifier to be applied also on DTLS packets. Thus using the same SID that is used with RTP and/or RTCP also for the DTLS message intended to key that particular SRTP and/or SRTCP flow(s).

6.3.3. MIKEY

MIKEY: Multimedia Internet KEYing [RFC3830] is a key management protocol that has several transports. In some cases it is used directly on a transport protocol such as UDP, but there is also a specification for how MIKEY is used with SDP "Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP)" [RFC4567].

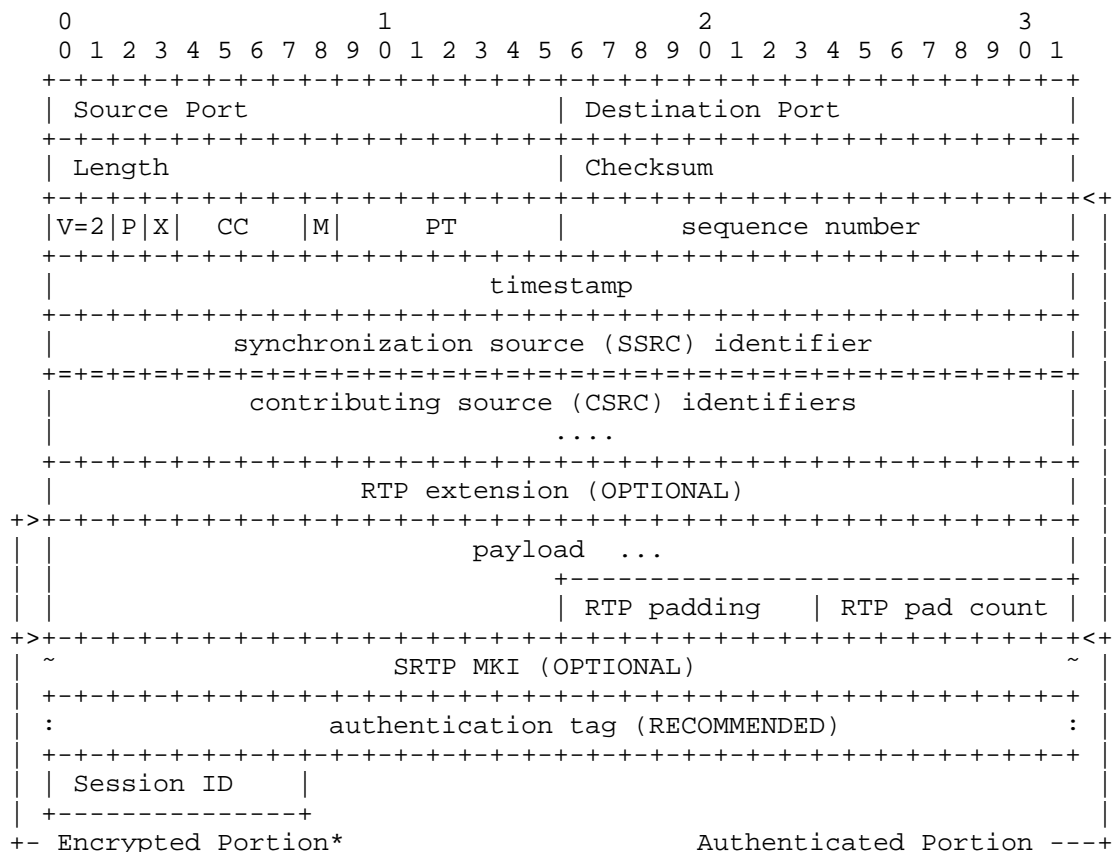
Lets start with the later, i.e. the SDP transport, which shares the properties with Security Description in that it can be associated with a particular media description in a SDP. As long as one avoids using the session level attribute one can be certain to correctly associate the key exchange with a given SRTP/SRTCP context.

It does appear that MIKEY directly over a lower layer transport protocol will have similar issues as DTLS.

6.4. Examples

6.4.1. RTP Packet with Transport Header

The below figure contains an RTP packet with SID field encapsulated by a UDP packet (added UDP header).



SRTP Packet Encapsulated by Session ID Layer

6.4.2. SDP Offer/Answer example

This section contains SDP offer/answer examples. First one example of successful BUNDLEing, and then two where fallback occurs.

In the below SDP offer, one audio and one video is being offered. The audio is using SID 0, and the video is using SID 1 to indicate that they are different RTP sessions despite being offered over the same 5-tuple.

```
v=0
o=alice 2890844526 2890844526 IN IP4 atlanta.example.com
s=
c=IN IP4 atlanta.example.com
t=0 0
a=group:BUNDLE foo bar
m=audio 10000 RTP/AVP 0 8 97
b=AS:200
a=mid:foo
a=session-mxu-id:0 policy=suggest
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
m=video 10000 RTP/AVP 31 32
b=AS:1000
a=mid:bar
a=session-mxu-id:1 policy=suggest
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
```

The SDP answer from an end-point that supports this BUNDLEing:

```
v=0
o=bob 2808844564 2808844564 IN IP4 biloxi.example.com
s=
c=IN IP4 biloxi.example.com
t=0 0
a=group:BUNDLE foo bar
m=audio 20000 RTP/AVP 0
b=AS:200
a=mid:foo
a=session-mux-id:0 policy=suggest
a=rtpmap:0 PCMU/8000
m=video 20000 RTP/AVP 32
b=AS:1000
a=mid:bar
a=session-mux-id:1 policy=suggest
a=rtpmap:32 MPV/90000
```

The SDP answer from an end-point that does not support this BUNDLEing or the general signalling of [I-D.holmberg-mmusic-sdp-bundle-negotiation].


```
v=0
o=bob 2808844564 2808844564 IN IP4 biloxi.example.com
s=
c=IN IP4 biloxi.example.com
t=0 0
m=audio 20000 RTP/AVP 0
b=AS:200
a=rtpmap:0 PCMU/8000
m=video 30000 RTP/AVP 32
b=AS:1000
a=rtpmap:32 MPV/90000
```

The SDP answer of a client supporting [I-D.holmberg-mmusic-sdp-bundle-negotiation] but not this BUNDLEing would look like this:

```
v=0
o=bob 2808844564 2808844564 IN IP4 biloxi.example.com
s=
c=IN IP4 biloxi.example.com
t=0 0
a=group:BUNDLE foo bar
m=audio 20000 RTP/AVP 0
a=mid:foo
b=AS:200
a=rtpmap:0 PCMU/8000
m=video 20000 RTP/AVP 32
a=mid:bar
b=AS:1000
a=rtpmap:32 MPV/90000
```

In this last case, the result is a sing RTP session with both media types being established. If that isn't supported or desired, the offerer will have to either re-invite without the BUNDLE grouping to force different 5-tuples, or simply terminate the session.

7. Open Issues

This is the first version of this draft. It will obviously have a number of open issues. This section contains a list of open issues where the author desires some input.

1. Should RTP and RTCP multiplexing without RFC 5761 support be included?

8. IANA Considerations

This document request the registration of one SDP attribute. Details of the registration to be filled in.

9. Security Considerations

The security properties of the Session ID layer is depending on what mechanism is used to protect the RTP and RTCP packets of a given RTP session. If IPsec or transport layer security solutions such as DTLS or TLS are being used then both the encapsulated RTP/RTCP packets and the session ID layer will be protected by that security mechanism. Thus potentially providing both confidentiality, integrity and source authentication. If SRTP is used, the session ID layer will not be directly protected by SRTP. However, it will be implicitly integrity protected (assuming the RTP/RTCP packet is integrity protected) as the only function of the field is to identify the session context. Thus any modification of the SID field will attempt to retrieve the wrong SRTP crypto context. If that retrieval fails, the packet will be anyway be discarded. If it is successful, the context will not lead to successful verification of the packet.

10. Acknowledgements

This document is based on the input from various people, especially in the context of the RTCWEB discussion of how to use only a single lower layer transport. The RTP and RTCP packet figures are borrowed from RFC3711. The SDP example is extended from the one present in [I-D.holmberg-mmusic-sdp-bundle-negotiation]. The authors would like to thank Christer Holmberg for assistance in utilizing the BUNDLE grouping mechanism.

The proposal in Section 4.5 is original suggested by Colin Perkins. The idea in Section 4.6 is from an Internet Draft [I-D.rosenberg-rtcweb-rtpmux] written by Jonathan Rosenberg et. al. The proposal in Section 4.3 is a result of discussion by a group of people at IETF meeting #81 in Quebec.

11. References

11.1. Normative References

[I-D.holmberg-mmusic-sdp-bundle-negotiation]
Holmberg, C. and H. Alvestrand, "Multiplexing Negotiation Using Session Description Protocol (SDP) Port Numbers",

draft-holmberg-mmusic-sdp-bundle-negotiation-00 (work in progress), October 2011.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

11.2. Informational References

- [I-D.lennox-rtcweb-rtp-media-type-mux]
Lennox, J. and J. Rosenberg, "Multiplexing Multiple Media Types In a Single Real-Time Transport Protocol (RTP) Session", draft-lennox-rtcweb-rtp-media-type-mux-00 (work in progress), October 2011.
- [I-D.rosenberg-rtcweb-rtpmux]
Rosenberg, J., Jennings, C., Peterson, J., Kaufman, M., Rescorla, E., and T. Terriberry, "Multiplexing of Real-Time Transport Protocol (RTP) Traffic for Browser based Real-Time Communications (RTC)", draft-rosenberg-rtcweb-rtpmux-00 (work in progress), July 2011.
- [I-D.westerlund-avtcore-multiplex-architecture]
Westerlund, M., Burman, B., and C. Perkins, "RTP Multiplexing Architecture", draft-westerlund-avtcore-multiplex-architecture-00 (work in progress), October 2011.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.
- [RFC3830] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and K. Norrman, "MIKEY: Multimedia Internet KEYing", RFC 3830, August 2004.
- [RFC4567] Arkko, J., Lindholm, F., Naslund, M., Norrman, K., and E.

Carrara, "Key Management Extensions for Session Description Protocol (SDP) and Real Time Streaming Protocol (RTSP)", RFC 4567, July 2006.

- [RFC4568] Andreassen, F., Baugher, M., and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, July 2006.
- [RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 2008.
- [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.
- [RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.
- [RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, May 2010.

Authors' Addresses

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

Phone: +46 10 714 82 87
Email: magnus.westerlund@ericsson.com

Colin Perkins
University of Glasgow
School of Computing Science
Glasgow G12 8QQ
United Kingdom

Email: csp@csp Perkins.org

