

AVTCORE Working Group  
Internet-Draft  
Updates: 3550 (if approved)  
Intended status: Standards Track  
Expires: April 30, 2015

C. S. Perkins  
University of Glasgow  
V. Singh  
Aalto University  
October 27, 2014

Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions  
draft-ietf-avtccore-rtp-circuit-breakers-07

Abstract

The Real-time Transport Protocol (RTP) is widely used in telephony, video conferencing, and telepresence applications. Such applications are often run on best-effort UDP/IP networks. If congestion control is not implemented in the applications, then network congestion will deteriorate the user's multimedia experience. This document does not propose a congestion control algorithm; instead, it defines a minimal set of RTP "circuit-breakers". Circuit-breakers are conditions under which an RTP sender needs to stop transmitting media data in order to protect the network from excessive congestion. It is expected that, in the absence of severe congestion, all RTP applications running on best-effort IP networks will be able to run without triggering these circuit breakers. Any future RTP congestion control specification will be expected to operate within the constraints defined by these circuit breakers.

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 30, 2015.

Copyright Notice

Copyright (c) 2014 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 . . . . .	2
2. Terminology . . . . .	3
3. Background . . . . .	3
4. RTP Circuit Breakers for Systems Using the RTP/AVP Profile . . . . .	6
4.1. RTP/AVP Circuit Breaker #1: Media Timeout . . . . .	8
4.2. RTP/AVP Circuit Breaker #2: RTCP Timeout . . . . .	8
4.3. RTP/AVP Circuit Breaker #3: Congestion . . . . .	9
4.4. RTP/AVP Circuit Breaker #4: Media Usability . . . . .	13
4.5. Ceasing Transmission . . . . .	14
5. RTP Circuit Breakers for Systems Using the RTP/AVPF Profile . . . . .	14
6. Impact of RTCP Extended Reports (XR) . . . . .	15
7. Impact of RTCP Reporting Groups . . . . .	15
8. Impact of Explicit Congestion Notification (ECN) . . . . .	16
9. Impact of Layered Coding . . . . .	16
10. Security Considerations . . . . .	17
11. IANA Considerations . . . . .	17
12. Open Issues . . . . .	17
13. Acknowledgements . . . . .	18
14. References . . . . .	18
14.1. Normative References . . . . .	18
14.2. Informative References . . . . .	18
Authors' Addresses . . . . .	20

## 1. Introduction

The Real-time Transport Protocol (RTP) [RFC3550] is widely used in voice-over-IP, video teleconferencing, and telepresence systems. Many of these systems run over best-effort UDP/IP networks, and can suffer from packet loss and increased latency if network congestion occurs. Designing effective RTP congestion control algorithms, to adapt the transmission of RTP-based media to match the available network capacity, while also maintaining the user experience, is a difficult but important problem. Many such congestion control and media adaptation algorithms have been proposed, but to date there is no consensus on the correct approach, or even that a single standard algorithm is desirable.

This memo does not attempt to propose a new RTP congestion control algorithm. Rather, it proposes a minimal set of "circuit breakers"; conditions under which there is general agreement that an RTP flow is causing serious congestion, and ought to cease transmission. It is expected that future standards-track congestion control algorithms for RTP will operate within the envelope defined by this memo.

## 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]. This interpretation of these key words applies only when written in ALL CAPS. Mixed- or lower-case uses of these key words are not to be interpreted as carrying special significance in this memo.

## 3. Background

We consider congestion control for unicast RTP traffic flows. This is the problem of adapting the transmission of an audio/visual data flow, encapsulated within an RTP transport session, from one sender to one receiver, so that it matches the available network bandwidth. Such adaptation needs to be done in a way that limits the disruption to the user experience caused by both packet loss and excessive rate changes. Congestion control for multicast flows is outside the scope of this memo. Multicast traffic needs different solutions, since the available bandwidth estimator for a group of receivers will differ from that for a single receiver, and because multicast congestion control has to consider issues of fairness across groups of receivers that do not apply to unicast flows.

Congestion control for unicast RTP traffic can be implemented in one of two places in the protocol stack. One approach is to run the RTP traffic over a congestion controlled transport protocol, for example over TCP, and to adapt the media encoding to match the dictates of the transport-layer congestion control algorithm. This is safe for the network, but can be suboptimal for the media quality unless the transport protocol is designed to support real-time media flows. We do not consider this class of applications further in this memo, as their network safety is guaranteed by the underlying transport.

Alternatively, RTP flows can be run over a non-congestion controlled transport protocol, for example UDP, performing rate adaptation at the application layer based on RTP Control Protocol (RTCP) feedback. With a well-designed, network-aware, application, this allows highly effective media quality adaptation, but there is potential to disrupt the network's operation if the application does not adapt its sending rate in a timely and effective manner. We consider this class of applications in this memo.

Congestion control relies on monitoring the delivery of a media flow, and responding to adapt the transmission of that flow when there are signs that the network path is congested. Network congestion can be detected in one of three ways: 1) a receiver can infer the onset of congestion by observing an increase in one-way delay caused by queue build-up within the network; 2) if Explicit Congestion Notification (ECN) [RFC3168] is supported, the network can signal the presence of congestion by marking packets using ECN Congestion Experienced (CE) marks; or 3) in the extreme case, congestion will cause packet loss that can be detected by observing a gap in the received RTP sequence numbers. Once the onset of congestion is observed, the receiver has to send feedback to the sender to indicate that the transmission rate needs to be reduced. How the sender reduces the transmission rate is highly dependent on the media codec being used, and is outside the scope of this memo.

There are several ways in which a receiver can send feedback to a media sender within the RTP framework:

- o The base RTP specification [RFC3550] defines RTCP Reception Report (RR) packets to convey reception quality feedback information, and Sender Report (SR) packets to convey information about the media transmission. RTCP SR packets contain data that can be used to reconstruct media timing at a receiver, along with a count of the total number of octets and packets sent. RTCP RR packets report on the fraction of packets lost in the last reporting interval, the cumulative number of packets lost, the highest sequence number received, and the inter-arrival jitter. The RTCP RR packets also contain timing information that allows the sender to estimate the network round trip time (RTT) to the receivers. RTCP reports are sent periodically, with the reporting interval being determined by the number of SSRCs used in the session and a configured session bandwidth estimate (the number of SSRCs used is usually two in a unicast session, one for each participant, but can be greater if the participants send multiple media streams). The interval between reports sent from each receiver tends to be on the order of a few seconds on average, although it varies with the session bandwidth, and sub-second reporting intervals are possible in high bandwidth sessions, and it is randomised to avoid synchronisation

of reports from multiple receivers. RTCP RR packets allow a receiver to report ongoing network congestion to the sender. However, if a receiver detects the onset of congestion part way through a reporting interval, the base RTP specification contains no provision for sending the RTCP RR packet early, and the receiver has to wait until the next scheduled reporting interval.

- o The RTCP Extended Reports (XR) [RFC3611] allow reporting of more complex and sophisticated reception quality metrics, but do not change the RTCP timing rules. RTCP extended reports of potential interest for congestion control purposes are the extended packet loss, discard, and burst metrics [RFC3611], [RFC7002], [RFC7097], [RFC7003], [RFC6958]; and the extended delay metrics [RFC6843], [RFC6798]. Other RTCP Extended Reports that could be helpful for congestion control purposes might be developed in future.
- o Rapid feedback about the occurrence of congestion events can be achieved using the Extended RTP Profile for RTCP-Based Feedback (RTP/AVPF) [RFC4585] (or its secure variant, RTP/SAVPF [RFC5124]) in place of the RTP/AVP profile [RFC3551]. This modifies the RTCP timing rules to allow RTCP reports to be sent early, in some cases immediately, provided the RTCP transmission rate keeps within its bandwidth allocation. It also defines transport-layer feedback messages, including negative acknowledgements (NACKs), that can be used to report on specific congestion events. RTP Codec Control Messages [RFC5104] extend the RTP/AVPF profile with additional feedback messages that can be used to influence that way in which rate adaptation occurs, but do not further change the dynamics of how rapidly feedback can be sent. Use of the RTP/AVPF profile is dependent on signalling.
- o Finally, Explicit Congestion Notification (ECN) for RTP over UDP [RFC6679] can be used to provide feedback on the number of packets that received an ECN Congestion Experienced (CE) mark. This RTCP extension builds on the RTP/AVPF profile to allow rapid congestion feedback when ECN is supported.

In addition to these mechanisms for providing feedback, the sender can include an RTP header extension in each packet to record packet transmission times. There are two methods: [RFC5450] represents the transmission time in terms of a time-offset from the RTP timestamp of the packet, while [RFC6051] includes an explicit NTP-format sending timestamp (potentially more accurate, but a higher header overhead). Accurate sending timestamps can be helpful for estimating queuing delays, to get an early indication of the onset of congestion.

Taken together, these various mechanisms allow receivers to provide feedback on the senders when congestion events occur, with varying

degrees of timeliness and accuracy. The key distinction is between systems that use only the basic RTCP mechanisms, without RTP/AVPF rapid feedback, and those that use the RTP/AVPF extensions to respond to congestion more rapidly.

#### 4. RTP Circuit Breakers for Systems Using the RTP/AVP Profile

The feedback mechanisms defined in [RFC3550] and available under the RTP/AVP profile [RFC3551] are the minimum that can be assumed for a baseline circuit breaker mechanism that is suitable for all unicast applications of RTP. Accordingly, for an RTP circuit breaker to be useful, it needs to be able to detect that an RTP flow is causing excessive congestion using only basic RTCP features, without needing RTCP XR feedback or the RTP/AVPF profile for rapid RTCP reports.

RTCP is a fundamental part of the RTP protocol, and the mechanisms described here rely on the implementation of RTCP. Implementations that claim to support RTP, but that do not implement RTCP, cannot use the circuit breaker mechanisms described in this memo. Such implementations SHOULD NOT be used on networks that might be subject to congestion unless equivalent mechanisms are defined using some non-RTCP feedback channel to report congestion and signal circuit breaker conditions.

Three potential congestion signals are available from the basic RTCP SR/RR packets and are reported for each synchronisation source (SSRC) in the RTP session:

1. The sender can estimate the network round-trip time once per RTCP reporting interval, based on the contents and timing of RTCP SR and RR packets.
2. Receivers report a jitter estimate (the statistical variance of the RTP data packet inter-arrival time) calculated over the RTCP reporting interval. Due to the nature of the jitter calculation ([RFC3550], section 6.4.4), the jitter is only meaningful for RTP flows that send a single data packet for each RTP timestamp value (i.e., audio flows, or video flows where each packet comprises one video frame).
3. Receivers report the fraction of RTP data packets lost during the RTCP reporting interval, and the cumulative number of RTP packets lost over the entire RTP session.

These congestion signals limit the possible circuit breakers, since they give only limited visibility into the behaviour of the network.

RTT estimates are widely used in congestion control algorithms, as a proxy for queuing delay measures in delay-based congestion control or to determine connection timeouts. RTT estimates derived from RTCP SR and RR packets sent according to the RTP/AVP timing rules are too infrequent to be useful though, and don't give enough information to distinguish a delay change due to routing updates from queuing delay caused by congestion. Accordingly, we cannot use the RTT estimate alone as an RTP circuit breaker.

Increased jitter can be a signal of transient network congestion, but in the highly aggregated form reported in RTCP RR packets, it offers insufficient information to estimate the extent or persistence of congestion. Jitter reports are a useful early warning of potential network congestion, but provide an insufficiently strong signal to be used as a circuit breaker.

The remaining congestion signals are the packet loss fraction and the cumulative number of packets lost. If considered carefully, these can be effective indicators that congestion is occurring in networks where packet loss is primarily due to queue overflows, although loss caused by non-congestive packet corruption can distort the result in some networks. TCP congestion control [RFC5681] intentionally tries to fill the router queues, and uses the resulting packet loss as congestion feedback. An RTP flow competing with TCP traffic will therefore expect to see a non-zero packet loss fraction that has to be related to TCP dynamics to estimate available capacity. This behaviour of TCP is reflected in the congestion circuit breaker below, and will affect the design of any RTP congestion control protocol.

Two packet loss regimes can be observed: 1) RTCP RR packets show a non-zero packet loss fraction, while the extended highest sequence number received continues to increment; and 2) RR packets show a loss fraction of zero, but the extended highest sequence number received does not increment even though the sender has been transmitting RTP data packets. The former corresponds to the TCP congestion avoidance state, and indicates a congested path that is still delivering data; the latter corresponds to a TCP timeout, and is most likely due to a path failure. A third condition is that data is being sent but no RTCP feedback is received at all, corresponding to a failure of the reverse path. We derive circuit breaker conditions for these loss regimes in the following.

#### 4.1. RTP/AVP Circuit Breaker #1: Media Timeout

If RTP data packets are being sent, but the RTCP SR or RR packets reporting on that SSRC indicate a non-increasing extended highest sequence number received, this is an indication that those RTP data packets are not reaching the receiver. This could be a short-term issue affecting only a few packets, perhaps caused by a slow-to-open firewall or a transient connectivity problem, but if the issue persists, it is a sign of a more ongoing and significant problem. Accordingly, if a sender of RTP data packets receives three or more consecutive RTCP SR or RR packets from the same receiver, and those packets correspond to its transmission and have a non-increasing extended highest sequence number received field, then that sender SHOULD cease transmission (see Section 4.5). The extended highest sequence number received field is non-increasing if the sender receives at least three consecutive RTCP SR or RR packets that report the same value for this field, but it has sent RTP data packets that would have caused an increase in the reported value if they had reached the receiver.

The reason for waiting for three or more consecutive RTCP packets with a non-increasing extended highest sequence number is to give enough time for transient reception problems to resolve themselves, but to stop problem flows quickly enough to avoid causing serious ongoing network congestion. A single RTCP report showing no reception could be caused by a transient fault, and so will not cease transmission. Waiting for more than three consecutive RTCP reports before stopping a flow might avoid some false positives, but could lead to problematic flows running for a long time period (potentially tens of seconds, depending on the RTCP reporting interval) before being cut off. Equally, an application that sends few packets when the packet loss rate is high runs the risk that the media timeout circuit breaker triggers inadvertently. The chosen timeout interval is a trade-off between these extremes.

#### 4.2. RTP/AVP Circuit Breaker #2: RTCP Timeout

In addition to media timeouts, as were discussed in Section 4.1, an RTP session has the possibility of an RTCP timeout. This can occur when RTP data packets are being sent, but there are no RTCP reports returned from the receiver. This is either due to a failure of the receiver to send RTCP reports, or a failure of the return path that is preventing those RTCP reporting from being delivered. In either case, it is not safe to continue transmission, since the sender has no way of knowing if it is causing congestion. Accordingly, an RTP sender that has not received any RTCP SR or RTCP RR packets reporting on the SSRC it is using for three or more of its RTCP reporting intervals SHOULD cease transmission (see Section 4.5). When



calculating the timeout, the deterministic RTCP reporting interval,  $T_d$ , without the randomization factor, and with a fixed minimum interval  $T_{min}=5$  seconds) SHOULD be used. The rationale for this choice of timeout is as described in Section 6.2 of RFC 3550 [RFC3550].

The choice of three RTCP reporting intervals as the timeout is made following Section 6.3.5 of RFC 3550 [RFC3550]. This specifies that participants in an RTP session will timeout and remove an RTP sender from the list of active RTP senders if no RTP data packets have been received from that RTP sender within the last two RTCP reporting intervals. Using a timeout of three RTCP reporting intervals is therefore large enough that the other participants will have timed out the sender if a network problem stops the data packets it is sending from reaching the receivers, even allowing for loss of some RTCP packets.

If a sender is transmitting a large number of RTP media streams, such that the corresponding RTCP SR or RR packets are too large to fit into the network MTU, the receiver will generate RTCP SR or RR packets in a round-robin manner. In this case, the sender SHOULD treat receipt of an RTCP SR or RR packet corresponding to any SSRC it sent on the same 5-tuple of source and destination IP address, port, and protocol, as an indication that the receiver and return path are working, preventing the RTCP timeout circuit breaker from triggering.

#### 4.3. RTP/AVP Circuit Breaker #3: Congestion

If RTP data packets are being sent, and the corresponding RTCP SR or RR packets show non-zero packet loss fraction and increasing extended highest sequence number received, then those RTP data packets are arriving at the receiver, but some degree of congestion is occurring. The RTP/AVP profile [RFC3551] states that:

If best-effort service is being used, RTP receivers SHOULD monitor packet loss to ensure that the packet loss rate is within acceptable parameters. Packet loss is considered acceptable if a TCP flow across the same network path and experiencing the same network conditions would achieve an average throughput, measured on a reasonable time scale, that is not less than the RTP flow is achieving. This condition can be satisfied by implementing congestion control mechanisms to adapt the transmission rate (or the number of layers subscribed for a layered multicast session), or by arranging for a receiver to leave the session if the loss rate is unacceptably high.

The comparison to TCP cannot be specified exactly, but is intended as an "order-of-magnitude" comparison in time scale and

throughput. The time scale on which TCP throughput is measured is the round-trip time of the connection. In essence, this requirement states that it is not acceptable to deploy an application (using RTP or any other transport protocol) on the best-effort Internet which consumes bandwidth arbitrarily and does not compete fairly with TCP within an order of magnitude.

The phrase "order of magnitude" in the above means within a factor of ten, approximately. In order to implement this, it is necessary to estimate the throughput a TCP connection would achieve over the path. For a long-lived TCP Reno connection, it has been shown that the TCP throughput can be estimated using the following equation [Padhye]:

$$X = \frac{s}{R \cdot \sqrt{2 \cdot b \cdot p / 3} + (t_{\text{RTO}} \cdot (3 \cdot \sqrt{3 \cdot b \cdot p / 8} \cdot p \cdot (1 + 32 \cdot p^2)))}$$

where:

X is the transmit rate in bytes/second.

s is the packet size in bytes. If data packets vary in size, then the average size is to be used.

R is the round trip time in seconds.

p is the loss event rate, between 0 and 1.0, of the number of loss events as a fraction of the number of packets transmitted.

t\_RTO is the TCP retransmission timeout value in seconds, generally approximated by setting t\_RTO = 4 \* R.

b is the number of packets that are acknowledged by a single TCP acknowledgement; [RFC3448] recommends the use of b=1 since many TCP implementations do not use delayed acknowledgements.

This is the same approach to estimated TCP throughput that is used in [RFC3448]. Under conditions of low packet loss the second term on the denominator is small, so this formula can be approximated with reasonable accuracy as follows [Mathis]:

$$X = \frac{s}{R \cdot \sqrt{2 \cdot b \cdot p / 3}}$$

It is RECOMMENDED that this simplified throughput equation be used, since the reduction in accuracy is small, and it is much simpler to calculate than the full equation. Measurements have shown that the simplified TCP throughput equation is effective as an RTP circuit breaker for multimedia flows sent to hosts on residential networks using ADSL and cable modem links [Singh]. The data shows that the full TCP throughput equation tends to be more sensitive to packet loss and triggers the RTP circuit breaker earlier than the simplified equation. Implementations that desire this extra sensitivity MAY use the full TCP throughput equation in the RTP circuit breaker. Initial measurements in LTE networks have shown that the extra sensitivity is helpful in that environment, with the full TCP throughput equation giving a more balanced circuit breaker response than the simplified TCP equation [Sarker]; other networks might see similar behaviour.

No matter what TCP throughput equation is chosen, two parameters need to be estimated and reported to the sender in order to calculate the throughput: the round trip time,  $R$ , and the loss event rate,  $p$  (the packet size,  $s$ , is known to the sender). The round trip time can be estimated from RTCP SR and RR packets. This is done too infrequently for accurate statistics, but is the best that can be done with the standard RTCP mechanisms.

Report blocks in RTCP SR or RR packets contain the packet loss fraction, rather than the loss event rate, so  $p$  cannot be reported (TCP typically treats the loss of multiple packets within a single RTT as one loss event, but RTCP RR packets report the overall fraction of packets lost, and does not report when the packet losses occurred). Using the loss fraction in place of the loss event rate can overestimate the loss. We believe that this overestimate will not be significant, given that we are only interested in order of magnitude comparison ([Floyd] section 3.2.1 shows that the difference is small for steady-state conditions and random loss, but using the loss fraction is more conservative in the case of bursty loss).

The congestion circuit breaker is therefore: when a sender receives an RTCP SR or RR packet that contains a report block for an SSRC it is using, that sender has to check the fraction lost field in that report block to determine if there is a non-zero packet loss rate. If the fraction lost field is zero, then continue sending as normal. If the fraction lost is greater than zero, then estimate the TCP throughput using the simplified equation above, and the measured  $R$ ,  $p$  (approximated by the fraction lost), and  $s$ . Compare this with the actual sending rate. If the actual sending rate is more than ten times the estimated sending rate derived from the TCP throughput equation for three consecutive RTCP reporting intervals, the sender SHOULD cease transmission (see Section 4.5).

Systems that usually send at a high data rate, but that can reduce their data rate significantly (i.e., by at least a factor of ten), MAY first reduce their sending rate to this lower value to see if this resolves the congestion, but MUST then cease transmission if the problem does not resolve itself within a further two RTCP reporting intervals (see Section 4.5). An example of this might be a video conferencing system that backs off to sending audio only, before completely dropping the call. If such a reduction in sending rate resolves the congestion problem, the sender MAY gradually increase the rate at which it sends data after a reasonable amount of time has passed, provided it takes care not to cause the problem to recur ("reasonable" is intentionally not defined here).

The congestion circuit breaker depends on the fraction of RTP data packets lost in a reporting interval. If the number of packets sent in the reporting interval is too low, this statistic loses meaning, and it is possible that a sampling error can give the appearance of high packet loss rates. Following the guidelines in [RFC5405], an RTP sender that sends not more than one RTP packet per RTT MAY ignore a single trigger of the congestion circuit breaker, on the basis that the packet loss rate estimate is unreliable with so few samples. However, if the congestion circuit breaker triggers again after the following three RTCP reporting intervals (i.e., if there have been six or more consecutive RTCP reporting intervals where the actual sending rate is more than ten times the estimated sending rate derived from the TCP throughput equation), then the sender SHOULD cease transmission (see Section 4.5).

The RTCP reporting interval of the media sender does not affect how quickly congestion circuit breaker can trigger. The timing is based on the RTCP reporting interval of the receiver that generates the SR/RR packets from which the loss rate and RTT estimate are derived (note that RTCP requires all participants in a session to have similar reporting intervals, else the participant timeout rules in [RFC3550] will not work, so this interval is likely similar to that of the sender). If the incoming RTCP SR or RR packets are using a reduced minimum RTCP reporting interval (as specified in Section 6.2 of RFC 3550 [RFC3550] or the RTP/AVPF profile [RFC4585]), then that reduced RTCP reporting interval is used when determining if the circuit breaker is triggered.

As in Section 4.1 and Section 4.2, we use three reporting intervals to avoid triggering the circuit breaker on transient failures. This circuit breaker is a worst-case condition, and congestion control needs to be performed to keep well within this bound. It is expected that the circuit breaker will only be triggered if the usual congestion control fails for some reason.

If there are more media streams that can be reported in a single RTCP SR or RR packet, or if the size of a complete RTCP SR or RR packet exceeds the network MTU, then the receiver will report on a subset of sources in each reporting interval, with the subsets selected round-robin across multiple intervals so that all sources are eventually reported [RFC3550]. When generating such round-robin RTCP reports, priority SHOULD be given to reports on sources that have high packet loss rates, to ensure that senders are aware of network congestion they are causing (this is an update to [RFC3550]).

#### 4.4. RTP/AVP Circuit Breaker #4: Media Usability

Applications that use RTP are generally tolerant to some amount of packet loss. How much packet loss can be tolerated will depend on the application, media codec, and the amount of error correction and packet loss concealment that is applied. There is an upper bound on the amount of loss can be corrected, however, beyond which the media becomes unusable. Similarly, many applications have some upper bound on the media capture to play-out latency that can be tolerated before the application becomes unusable. The latency bound will depend on the application, but typical values can range from the order of a few hundred milliseconds for voice telephony and interactive conferencing applications, up to several seconds for some video-on-demand systems.

As a final circuit breaker, RTP senders SHOULD monitor the reported packet loss and delay to estimate whether the media is likely to be suitable for the intended purpose. If the packet loss rate and/or latency is such that the media has become unusable, and has remained unusable for a significant time period, then the application SHOULD cease transmission. Similarly, receivers SHOULD monitor the quality of the media they receive, and if the quality is unusable for a significant time period, they SHOULD terminate the session. This memo intentionally does not define a bound on the packet loss rate or latency that will result in unusable media, nor does it specify what time period is deemed significant, as these are highly application dependent.

Sending media that suffers from such high packet loss or latency that it is unusable at the receiver is both wasteful of resources, and of no benefit to the user of the application. It also is highly likely to be congesting the network, and disrupting other applications. As such, the congestion circuit breaker will almost certainly trigger to stop flows where the media would be unusable due to high packet loss or latency. However, in pathological scenarios where the congestion circuit breaker does not stop the flow, it is desirable that the RTP application cease sending useless traffic. The role of the media usability circuit breaker is to protect the network in such cases.

#### 4.5. Ceasing Transmission

What it means to cease transmission depends on the application, but the intention is that the application will stop sending RTP data packets to a particular destination 3-tuple (transport protocol, destination port, IP address), until the user makes an explicit attempt to restart the call. It is important that a human user is involved in the decision to try to restart the call, since that user will eventually give up if the calls repeatedly trigger the circuit breaker. This will help avoid problems with automatic redial systems from congesting the network. Accordingly, RTP flows halted by the circuit breaker SHOULD NOT be restarted automatically unless the sender has received information that the congestion has dissipated.

It is recognised that the RTP implementation in some systems might not be able to determine if a call set-up request was initiated by a human user, or automatically by some scripted higher-level component of the system. These implementations SHOULD rate limit attempts to restart a call to the same destination 3-tuple as used by a previous call that was recently halted by the circuit breaker. The chosen rate limit ought to not exceed the rate at which an annoyed human caller might redial a misbehaving phone.

#### 5. RTP Circuit Breakers for Systems Using the RTP/AVPF Profile

Use of the Extended RTP Profile for RTCP-based Feedback (RTP/AVPF) [RFC4585] allows receivers to send early RTCP reports in some cases, to inform the sender about particular events in the media stream. There are several use cases for such early RTCP reports, including providing rapid feedback to a sender about the onset of congestion.

Receiving rapid feedback about congestion events potentially allows congestion control algorithms to be more responsive, and to better adapt the media transmission to the limitations of the network. It is expected that many RTP congestion control algorithms will adopt the RTP/AVPF profile for this reason, defining new transport layer feedback reports that suit their requirements. Since these reports are not yet defined, and likely very specific to the details of the congestion control algorithm chosen, they cannot be used as part of the generic RTP circuit breaker.

Reduced-size RTCP reports sent under the RTP/AVPF early feedback rules that do not contain an RTCP SR or RR packet MUST be ignored by the congestion circuit breaker (they do not contain the information needed by the congestion circuit breaker algorithm), but MUST be counted as received packets for the RTCP timeout circuit breaker. Reduced-size RTCP reports sent under the RTP/AVPF early feedback rules that contain RTCP SR or RR packets MUST be processed by the

congestion circuit breaker as if they were sent as regular RTCP reports, and counted towards the circuit breaker conditions specified in Section 4 of this memo. This will potentially make the RTP circuit breaker fire earlier than it would if the RTP/AVPF profile was not used.

When using ECN with RTP (see Section 8), early RTCP feedback packets can contain ECN feedback reports. The count of ECN-CE marked packets contained in those ECN feedback reports is counted towards the number of lost packets reported if the ECN Feedback Report report is sent in a compound RTCP packet along with an RTCP SR/RR report packet. Reports of ECN-CE packets sent as reduced-size RTCP ECN feedback packets without an RTCP SR/RR packet MUST be ignored.

These rules are intended to allow the use of low-overhead RTP/AVPF feedback for generic NACK messages without triggering the RTP circuit breaker. This is expected to make such feedback suitable for RTP congestion control algorithms that need to quickly report loss events in between regular RTCP reports. The reaction to reduced-size RTCP SR/RR packets is to allow such algorithms to send feedback that can trigger the circuit breaker, when desired.

#### 6. Impact of RTCP Extended Reports (XR)

RTCP Extended Report (XR) blocks provide additional reception quality metrics, but do not change the RTCP timing rules. Some of the RTCP XR blocks provide information that might be useful for congestion control purposes, others provided non-congestion-related metrics. With the exception of RTCP XR ECN Summary Reports (see Section 8), the presence of RTCP XR blocks in a compound RTCP packet does not affect the RTP circuit breaker algorithm. For consistency and ease of implementation, only the reception report blocks contained in RTCP SR packets, RTCP RR packets, or RTCP XR ECN Summary Report packets, are used by the RTP circuit breaker algorithm.

#### 7. Impact of RTCP Reporting Groups

An optimisation for grouping RTCP reception statistics and other feedback in RTP sessions with large numbers of participants is given in [I-D.ietf-avtcore-rtp-multi-stream-optimisation]. This allows one SSRC to act as a representative that sends reports on behalf of other SSRCs that are co-located in the same endpoint and see identical reception quality. When running the circuit breaker algorithms, an endpoint MUST treat a reception report from the representative of the reporting group as if a reception report was received from all members of that group.

## 8. Impact of Explicit Congestion Notification (ECN)

The use of ECN for RTP flows does not affect the media timeout RTP circuit breaker (Section 4.1) or the RTCP timeout circuit breaker (Section 4.2), since these are both connectivity checks that simply determinate if any packets are being received.

ECN-CE marked packets SHOULD be treated as if it were lost for the purposes of congestion control, when determining the optimal media sending rate for an RTP flow. If an RTP sender has negotiated ECN support for an RTP session, and has successfully initiated ECN use on the path to the receiver [RFC6679], then ECN-CE marked packets SHOULD be treated as if they were lost when calculating if the congestion-based RTP circuit breaker (Section 4.3) has been met. The count of ECN-CE marked RTP packets is returned in RTCP XR ECN summary report packets if support for ECN has been initiated for an RTP session.

## 9. Impact of Layered Coding

Layered coding is a method of encoding a single media stream into disparate layers, such that a receiver can decode a subset of the layers to vary the quality of the media. Layered coding is often used to aid congestion control in group communication systems, where a different subset of the layers is sent to each receiver, depending on the available network capacity.

Media using layered coding can be transported within RTP in several ways: each layer can be sent as a separate RTP session; each layer can be sent using a separate SSRC within a single RTP session; or each layer can be identified by some payload-specific header field, with all layers being sent by a single SSRC within a single RTP session. The choice depends on the features provided by the RTP payload format for the layered encoding, and on the application requirements.

The RTP circuit breaker operates on a per-RTP session basis. If a layered encoding is split across multiple RTP sessions, then each session MUST be treated independently for the RTP circuit breaker.

Within an RTP session, if an application that sends a layered media encoding using a single SSRC, with the layers identified using some payload-specific mechanism, then it MUST apply the RTP circuit breaker to that layered flow as a whole, considering RTCP feedback for the SSRC sending the layered flow and applying the RTP circuit breaker as usual.

Within an RTP session, if the layered coding is sent using several SSRC values within a single RTP session, the flows for those SSRCs



MAY be treated together, so that a circuit breaker trigger for any SSRC in the layered media flow causes the entire layered flow to either cease transmission or reduce its sending rate by a factor of ten. The intent of this is to allow a layered flow to reduce its sending rate by dropping higher layers if the circuit breaker fails, rather than requiring the layer that triggered the RTP circuit breaker to cease transmission (layers are additive in many layered codecs, so forcing a lower layer to cease transmission while allowing higher layers to continue is pointless).

## 10. Security Considerations

The security considerations of [RFC3550] apply.

If the RTP/AVPF profile is used to provide rapid RTCP feedback, the security considerations of [RFC4585] apply. If ECN feedback for RTP over UDP/IP is used, the security considerations of [RFC6679] apply.

If non-authenticated RTCP reports are used, an on-path attacker can trivially generate fake RTCP packets that indicate high packet loss rates, causing the circuit breaker to trigger and disrupting an RTP session. This is somewhat more difficult for an off-path attacker, due to the need to guess the randomly chosen RTP SSRC value and the RTP sequence number. This attack can be avoided if RTCP packets are authenticated; authentication options are discussed in [RFC7201].

Timely operation of the RTP circuit breaker depends on the choice of RTCP reporting interval. If the receiver has a reporting interval that is overly long, then the responsiveness of the circuit breaker decreases. In the limit, the RTP circuit breaker can be disabled for all practical purposes by configuring an RTCP reporting interval that is many minutes duration. This issue is not specific to the circuit breaker: long RTCP reporting intervals also prevent reception quality reports, feedback messages, codec control messages, etc., from being used. Implementations SHOULD impose an upper limit on the RTCP reporting interval they are willing to negotiate (based on the session bandwidth and RTCP bandwidth fraction) when using the RTP circuit breaker. An upper limit on the reporting interval on the order of 10 seconds is a reasonable bound.

## 11. IANA Considerations

There are no actions for IANA.

## 12. Open Issues

- o Should the number of RTCP reporting intervals needed to trigger the media timeout and congestion circuit breakers scale with the

duration of the RTCP reporting interval, so the circuit breaker triggers after a fixed duration, rather than after a fixed number of reporting intervals?

### 13. Acknowledgements

The authors would like to thank Bernard Aboba, Harald Alvestrand, Gorrry Fairhurst, Kevin Gross, Cullen Jennings, Randell Jesup, Jonathan Lennox, Matt Mathis, Stephen McQuistin, Eric Rescorla, Abheek Saha, and Fabio Verdicchio, for their valuable feedback.

### 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.
- [RFC3448] Handley, M., Floyd, S., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 3448, January 2003.
- [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., Caceres, R., and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, November 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.

#### 14.2. Informative References

- [Floyd] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "Equation-Based Congestion Control for Unicast Applications", Proceedings of the ACM SIGCOMM conference, 2000, DOI 10.1145/347059.347397, August 2000.
- [I-D.ietf-avtcore-rtp-multi-stream-optimisation]

- Lennox, J., Westerlund, M., Wu, W., and C. Perkins,  
"Sending Multiple Media Streams in a Single RTP Session:  
Grouping RTCP Reception Statistics and Other Feedback",  
draft-ietf-avtcore-rtp-multi-stream-optimisation-04 (work  
in progress), August 2014.
- [Mathis] Mathis, M., Semke, J., Mahdavi, J., and T. Ott, "The  
macroscopic behavior of the TCP congestion avoidance  
algorithm", ACM SIGCOMM Computer Communication Review  
27(3), DOI 10.1145/263932.264023, July 1997.
- [Padhye] Padhye, J., Firoiu, V., Towsley, D., and J. Kurose,  
"Modeling TCP Throughput: A Simple Model and its Empirical  
Validation", Proceedings of the ACM SIGCOMM conference,  
1998, DOI 10.1145/285237.285291, August 1998.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition  
of Explicit Congestion Notification (ECN) to IP", RFC  
3168, September 2001.
- [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.
- [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.
- [RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines  
for Application Designers", BCP 145, RFC 5405, November  
2008.
- [RFC5450] Singer, D. and H. Desineni, "Transmission Time Offsets in  
RTP Streams", RFC 5450, March 2009.
- [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size  
Real-Time Transport Control Protocol (RTCP): Opportunities  
and Consequences", RFC 5506, April 2009.
- [RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion  
Control", RFC 5681, September 2009.
- [RFC6051] Perkins, C. and T. Schierl, "Rapid Synchronisation of RTP  
Flows", RFC 6051, November 2010.
- [RFC6679] Westerlund, M., Johansson, I., Perkins, C., O'Hanlon, P.,  
and K. Carlberg, "Explicit Congestion Notification (ECN)  
for RTP over UDP", RFC 6679, August 2012.

- [RFC6798] Clark, A. and Q. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Packet Delay Variation Metric Reporting", RFC 6798, November 2012.
- [RFC6843] Clark, A., Gross, K., and Q. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Delay Metric Reporting", RFC 6843, January 2013.
- [RFC6958] Clark, A., Zhang, S., Zhao, J., and Q. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Burst/Gap Loss Metric Reporting", RFC 6958, May 2013.
- [RFC7002] Clark, A., Zorn, G., and Q. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Discard Count Metric Reporting", RFC 7002, September 2013.
- [RFC7003] Clark, A., Huang, R., and Q. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Burst/Gap Discard Metric Reporting", RFC 7003, September 2013.
- [RFC7097] Ott, J., Singh, V., and I. Curcio, "RTP Control Protocol (RTCP) Extended Report (XR) for RLE of Discarded Packets", RFC 7097, January 2014.
- [RFC7201] Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", RFC 7201, April 2014.
- [Sarker] Sarker, Z., Singh, V., and C.S. Perkins, "An Evaluation of RTP Circuit Breaker Performance on LTE Networks", Proceedings of the IEEE Infocom workshop on Communication and Networking Techniques for Contemporary Video, 2014, April 2014.
- [Singh] Singh, V., McQuistin, S., Ellis, M., and C.S. Perkins, "Circuit Breakers for Multimedia Congestion Control", Proceedings of the International Packet Video Workshop, 2013, DOI 10.1109/PV.2013.6691439, December 2013.

#### Authors' Addresses

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

Email: [csp@csperkins.org](mailto:csp@csperkins.org)

Varun Singh  
Aalto University  
School of Electrical Engineering  
Otakaari 5 A  
Espoo, FIN 02150  
Finland

Email: [varun@comnet.tkk.fi](mailto:varun@comnet.tkk.fi)  
URI: <http://www.netlab.tkk.fi/~varun/>

Network Working Group  
Internet-Draft  
Obsoletes: 5117 (if approved)  
Intended status: Informational  
Expires: February 19, 2015

M. Westerlund  
Ericsson  
S. Wenger  
Vidyo  
August 18, 2014

RTP Topologies  
draft-ietf-avtcore-rtp-topologies-update-04

Abstract

This document discusses point to point and multi-endpoint topologies used in Real-time Transport Protocol (RTP)-based environments. In particular, centralized topologies commonly employed in the video conferencing industry are mapped to the RTP terminology.

This document is updated with additional topologies and is intended to replace RFC 5117.

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 February 19, 2015.

Copyright Notice

Copyright (c) 2014 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. Definitions . . . . .	3
2.1. Glossary . . . . .	3
2.2. Definitions related to RTP grouping taxonomy . . . . .	4
3. Topologies . . . . .	5
3.1. Point to Point . . . . .	5
3.2. Point to Point via Middlebox . . . . .	6
3.2.1. Translators . . . . .	6
3.2.2. Back to Back RTP sessions . . . . .	10
3.3. Point to Multipoint Using Multicast . . . . .	11
3.3.1. Any Source Multicast (ASM) . . . . .	11
3.3.2. Source Specific Multicast (SSM) . . . . .	12
3.3.3. SSM with Local Unicast Resources . . . . .	14
3.4. Point to Multipoint Using Mesh . . . . .	16
3.5. Point to Multipoint Using the RFC 3550 Translator . . . . .	19
3.5.1. Relay - Transport Translator . . . . .	19
3.5.2. Media Translator . . . . .	20
3.6. Point to Multipoint Using the RFC 3550 Mixer Model . . . . .	21
3.6.1. Media Mixing Mixer . . . . .	23
3.6.2. Media Switching . . . . .	26
3.7. Selective Forwarding Middlebox . . . . .	28
3.8. Point to Multipoint Using Video Switching MCUs . . . . .	31
3.9. Point to Multipoint Using RTCP-Terminating MCU . . . . .	33
3.10. Split Component Terminal . . . . .	34
3.11. Non-Symmetric Mixer/Translators . . . . .	37
3.12. Combining Topologies . . . . .	37
4. Comparing Topologies . . . . .	38
4.1. Topology Properties . . . . .	38
4.1.1. All to All Media Transmission . . . . .	38
4.1.2. Transport or Media Interoperability . . . . .	39
4.1.3. Per Domain Bit-Rate Adaptation . . . . .	39
4.1.4. Aggregation of Media . . . . .	40
4.1.5. View of All Session Participants . . . . .	40
4.1.6. Loop Detection . . . . .	41
4.2. Comparison of Topologies . . . . .	41
5. Security Considerations . . . . .	41
6. IANA Considerations . . . . .	43
7. Acknowledgements . . . . .	44
8. References . . . . .	44
8.1. Normative References . . . . .	44
8.2. Informative References . . . . .	44
Authors' Addresses . . . . .	45

## 1. Introduction

Real-time Transport Protocol (RTP) [RFC3550] topologies describe methods for interconnecting RTP entities and their processing behavior of RTP and RTCP. This document tries to address past and existing confusion, especially with respect to terms not defined in RTP but in common use in the conversational communication industry, such as the Multipoint Control Unit or MCU.

When the Audio-Visual Profile with Feedback (AVPF) [RFC4585] was developed the main emphasis lay in the efficient support of point to point and small multipoint scenarios without centralized multipoint control. In practice, however, most multipoint conferences operate utilizing centralized units referred to as MCUs. MCUs may implement Mixer or Translator functionality (in RTP [RFC3550] terminology), and signalling support. They may also contain additional application layer functionality. This document focuses on the media transport aspects of the MCU that can be realized using RTP, as discussed below. Further considered are the properties of Mixers and Translators, and how some types of deployed MCUs deviate from these properties.

This document also codifies new multipoint architectures that have recently been introduced and which were not anticipated in RFC 5117. These architectures use scalable video coding and simulcasting, and their associated centralized units are referred to as Selective Forwarding Units (SFU). This codification provides a common information basis for future discussion and specification work.

The document's attempt to clarify and explain sections of the Real-time Transport Protocol (RTP) spec [RFC3550] is informal. It is not intended to update or change what is normatively specified within RFC 3550.

## 2. Definitions

### 2.1. Glossary

ASM: Any Source Multicast

AVPF: The Extended RTP Profile for RTCP-based Feedback

CSRC: Contributing Source

Link: The data transport to the next IP hop

Middlebox: A device that is on the Path that media travel between two Endpoints



MCU: Multipoint Control Unit

Path: The concatenation of multiple links, resulting in an end-to-end data transfer.

PtM: Point to Multipoint

PtP: Point to Point

SFU: Selective Forwarding Unit

SSM: Source-Specific Multicast

SSRC: Synchronization Source

## 2.2. Definitions related to RTP grouping taxonomy

[Note to RFC editor: The following definitions have been taken from draft-ietf-avtext-rtp-grouping-taxonomy-02 (taxonomy draft henceforth). It is avtcore working group agreement to not delay the publication of the topologies-update document through a dependency to the taxonomy draft. If, however, the taxonomy draft and this draft are in your work queue at the same time and there would be no significant additional delay (through your schedule, normative reference citations, or similar) in publishing both documents roughly in parallel, it would be preferable to replace the definition language with something like "as in [RFC YYYY]" where YYYY would be the RFC number of the published taxonomy draft.]

The following definitions have been taken from draft-ietf-avtext-rtp-grouping-taxonomy-02, and are used in capitalized form throughout the document.

**Communication Session:** A Communication Session is an association among group of participants communicating with each other via a set of Multimedia Sessions.

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

**Media Source:** A Media Source is the logical source of a reference clock synchronized, time progressing, digital media stream, called a Source Stream.

**Multimedia Session:** A multimedia session is an association among a group of participants engaged in the communication via one or more RTP Sessions.

### 3. Topologies

This subsection defines several topologies that are relevant for codec control but also RTP usage in other contexts. The section starts with point to point cases, with or without middleboxes. Then follows a number of different methods for establishing point to multipoint communication. These are structured around the most fundamental enabler, i.e., multicast, a mesh of connections, translators, mixers and finally MCUs and SFUs. The section ends by discussing de-composited terminals, asymmetric middlebox behaviors and combining topologies.

The topologies may be referenced in other documents by a shortcut name, indicated by the prefix "Topo-".

For each of the RTP-defined topologies, we discuss how RTP, RTCP, and the carried media are handled. With respect to RTCP, we also discuss the handling of RTCP feedback messages as defined in [RFC4585] and [RFC5104].

#### 3.1. Point to Point

Shortcut name: Topo-Point-to-Point

The Point to Point (PtP) topology (Figure 1) consists of two End Points, communicating using unicast. Both RTP and RTCP traffic are conveyed endpoint-to-endpoint, using unicast traffic only (even if, in exotic cases, this unicast traffic happens to be conveyed over an IP-multicast address).

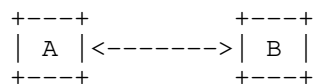


Figure 1: Point to Point

The main property of this topology is that A sends to B, and only B, while B sends to A, and only A. This avoids all complexities of handling multiple End Points and combining the requirements stemming from them. Note that an End Point can still use multiple RTP Synchronization Sources (SSRCs) in an RTP session. The number of RTP sessions in use between A and B can also be of any number, subject only to system level limitations like the number range of ports.

RTCP feedback messages for the indicated SSRCs are communicated directly between the End Points. Therefore, this topology poses minimal (if any) issues for any feedback messages. For RTP sessions which use multiple SSRC per End Point it can be relevant to implement support for cross-reporting suppression as defined in "Sending Multiple Media Streams in a Single RTP Session" [I-D.ietf-avtcore-rtp-multi-stream-optimisation].

### 3.2. Point to Point via Middlebox

This section discusses cases where two End Points communicate but have one or more middleboxes involved in the RTP session.

#### 3.2.1. Translators

Shortcut name: Topo-PtP-Translator

Two main categories of Translators can be distinguished; Transport Translators and Media translators. Both Translator types share common attributes that separate them from Mixers. For each RTP stream that the Translator receives, it generates an individual RTP stream in the other domain. A translator keeps the SSRC for an RTP stream across the translation, whereas a Mixer can select a single RTP stream from multiple received RTP streams (in cases like audio/video switching), or send out an RTP stream composed of multiple mixed media received in multiple RTP streams (in cases like audio mixing or video tiling), but always under its own SSRC, possibly using the CSRC field to indicate the source(s) of the content. Mixers are more common in point to multipoint cases than in PtP. The reason is that in PtP use cases the primary focus of a middlebox is enabling interoperability, between otherwise non-interoperable End Points, such as transcoding to a codec the receiver supports, which can be done by a media translator.

As specified in Section 7.1 of [RFC3550], the SSRC space is common for all participants in the RTP session, independent of on which side of the Translator the session resides. Therefore, it is the responsibility of the End Points (as the RTP session participants) to run SSRC collision detection, and the SSRC is thus a field the Translator cannot change. Any SDES information associated with a SSRC or CSRC also needs to be forwarded between the domains for any SSRC/CSRC used in the different domains.

A Translator commonly does not use an SSRC of its own, and is not visible as an active participant in the RTP session. One reason to have its own SSRC is when a Translator acts as a quality monitor that sends RTCP reports and therefore is required to have an SSRC. Another example is the case when a Translator is prepared to use RTCP

feedback messages. This may, for example, occur in a translator configured to detect packet loss of important video packets and wants to trigger repair by the media sending End Point, by sending feedback messages. While such feedback could use the SSRC of the target for the translator (the receiving End Point), this in turn would require translation of the targets RTCP reports to make them consistent. It may be simpler to expose an additional SSRC in the session. The only concern is End Points failing to support the full RTP specification may have issues with multiple SSRCs reporting on the RTP streams sent by that End Point, as this use case may be viewed as exotic by implementers.

In general, a Translator implementation should consider which RTCP feedback messages or codec-control messages it needs to understand in relation to the functionality of the Translator itself. This is completely in line with the requirement to also translate RTCP messages between the domains.

### 3.2.1.1. Transport Relay/Anchoring

There exist a number of different types of middleboxes that might be inserted between two End Points on the transport level, e.g., to perform changes on the IP/UDP headers, and are, therefore, basic transport translators. These middleboxes come in many variations including NAT [RFC3022] traversal by pinning the media path to a public address domain relay, network topologies where the RTP stream is required to pass a particular point for audit by employing relaying, or preserving privacy by hiding each peer's transport addresses to the other party. Other protocols or functionalities that provide this behavior are TURN [RFC5766] servers, Session Border Gateways and Media Processing Nodes with media anchoring functionalities.

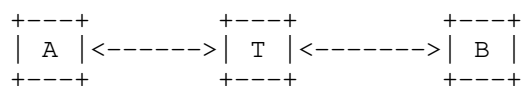


Figure 2: Point to Point with Translator

A common element in these functions is that they are normally transparent at the RTP level, i.e., they perform no changes on any RTP or RTCP packet fields and only affect the lower layers. They may affect, however, the path the RTP and RTCP packets are routed between the End Points in the RTP session, and thereby indirectly affect the RTP session. For this reason, one could believe that transport translator-type middleboxes do not need to be included in this document. This topology, however, can raise additional requirements in the RTP implementation and its interactions with the signalling

solution. Both in signalling and in certain RTCP fields, network addresses other than those of the relay can occur since B has a different network address than the relay (T). Implementations that cannot support this will also not work correctly when End Points are subject to NAT.

The transport relay implementations also have to take into account security considerations. In particular, source address filtering of incoming packets is usually important in relays, to prevent attackers to inject traffic into a session, which one peer may, in the absence of adequate security in the relay, think it comes from the other peer.

#### 3.2.1.2. Transport Translator

Transport Translators (Topo-Trn-Translator) do not modify the RTP stream itself, but are concerned with transport parameters. Transport parameters, in the sense of this section, comprise the transport addresses (to bridge different domains such unicast to multicast) and the media packetization to allow other transport protocols to be interconnected to a session (in gateways).

Translators that bridge between different protocol worlds need to be concerned about the mapping of the SSRC/CSRC (Contributing Source) concept to the non-RTP protocol. When designing a Translator to a non-RTP-based media transport, an important consideration is how to handle different sources and their identities. This problem space is not discussed henceforth.

Of the transport Translators, this memo is primarily interested in those that use RTP on both sides, and this is assumed henceforth.

The most basic transport translators that operate below the RTP level were already discussed in Section 3.2.1.1.

#### 3.2.1.3. Media Translator

Media Translators (Topo-Media-Translator) modify the media inside the RTP stream. This process is commonly known as transcoding. The modification of the media can be as small as removing parts of the stream, and it can go all the way to a full decoding and re-encoding (down to the sample level or equivalent) utilizing a different media codec. Media Translators are commonly used to connect End Points without a common interoperability point in the media encoding.

Stand-alone Media Translators are rare. Most commonly, a combination of Transport and Media Translator is used to translate both the media

and the transport aspects of the RTP stream carrying the media between two transport domains.

When media translation occurs, the Translator's task regarding handling of RTCP traffic becomes substantially more complex. In this case, the Translator needs to rewrite End Point B's RTCP Receiver Report before forwarding them to End Point A. The rewriting is needed as the RTP stream received by B is not the same RTP stream as the other participants receive. For example, the number of packets transmitted to B may be lower than what A sends, due to the different media format and data rate. Therefore, if the Receiver Reports were forwarded without changes, the extended highest sequence number would indicate that B were substantially behind in reception, while most likely it would not be. Therefore, the Translator must translate that number to a corresponding sequence number for the stream the Translator received. Similar requirements exists for most other fields in the RTCP Receiver Reports.

A media Translator may in some cases act on behalf of the "real" source (the End Point originally sending the media to the Translator) and respond to RTCP feedback messages. This may occur, for example, when a receiving End Point requests a bandwidth reduction, and the media Translator has not detected any congestion or other reasons for bandwidth reduction between the sending End Point and itself. In that case, it is sensible that the media Translator reacts to codec control messages itself, for example, by transcoding to a lower media rate.

A variant of translator behaviour worth pointing out is the one depicted in Figure 3 of an End Point A sending a RTP stream containing media (only) to B. On the path there is a device T that on A's behalf manipulates the RTP streams. One common example is that T adds a second RTP stream containing Forward Error Correction (FEC) information in order to protect A's (non FEC-protected) RTP stream. In this case, T needs to semantically bind the new FEC RTP stream to A's media-carrying RTP stream, for example by using the same CNAME as A.

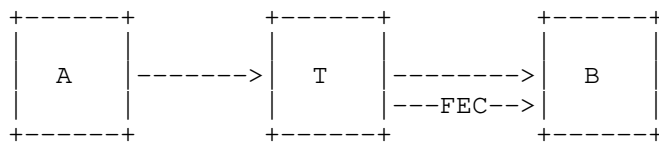


Figure 3: Media Translator adding FEC

there may also be cases where information is added into the original RTP stream, while leaving most or all of the original RTP packets

intact (with the exception of certain RTP header fields, such as the sequence number). One example is the injection of meta-data into the RTP stream, carried in their own RTP packets.

Similarly, a Media Translator can sometimes remove information from the RTP stream, while otherwise leaving the remaining RTP packets unchanged (again with the exception of certain RTP header fields).

Either type of functionality where T manipulates the RTP stream, or adds an accompanying RTP stream, on behalf of A is also covered under the media translator definition.

### 3.2.2. Back to Back RTP sessions

There exist middleboxes that interconnect two End Points A and B through themselves (MB), but not by being part of a common RTP session. They establish instead two different RTP sessions, one between A and the middlebox and another between the middlebox and B. This topology is called Topo-Back-To-Back

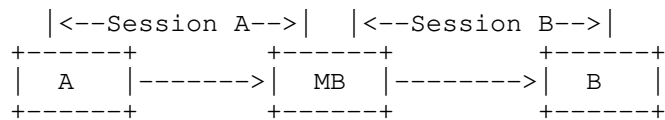


Figure 4: Back-to-back RTP sessions through Middlebox

The middlebox acts as an application-level gateway and bridges the two RTP sessions. This bridging can be as basic as forwarding the RTP payloads between the sessions, or more complex including media transcoding. The difference of this topology relative to the single RTP session context is the handling of the SSRCs and the other session-related identifiers, such as CNAMEs. With two different RTP sessions these can be freely changed and it becomes the middlebox's responsibility to maintain the correct relations.

The signalling or other above-RTP level functionalities referencing RTP streams may be what is most impacted by using two RTP sessions and changing identifiers. The structure with two RTP sessions also puts a congestion control requirement on the middlebox, because it becomes fully responsible for the media stream it sources into each of the sessions.

Adherence to congestion control can be solved locally on each of the two segments, or by bridging statistics from the receiving End Point through the middlebox to the sending End Point. From an implementation point, however, the latter requires dealing with a number of inconsistencies. First, packet loss must be detected for

an RTP stream sent from A to the middlebox, and that loss must be reported through a skipped sequence number in the RTP stream from the middlebox to B. This coupling and the resulting inconsistencies are conceptually easier to handle when considering the two RTP streams as belonging to a single RTP session.

### 3.3. Point to Multipoint Using Multicast

Multicast is an IP layer functionality that is available in some networks. Two main flavors can be distinguished: Any Source Multicast (ASM) [RFC1112] where any multicast group participant can send to the group address and expect the packet to reach all group participants; and Source Specific Multicast (SSM) [RFC3569], where only a particular IP host sends to the multicast group. Both these models are discussed below in their respective sections.

#### 3.3.1. Any Source Multicast (ASM)

Shortcut name: Topo-ASM (was Topo-Multicast)

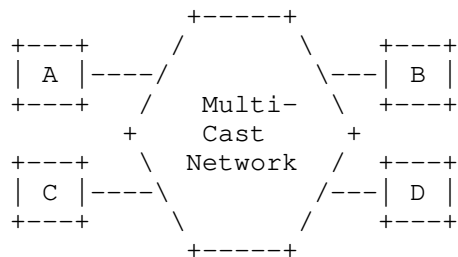


Figure 5: Point to Multipoint Using Multicast

Point to Multipoint (PtM) is defined here as using a multicast topology as a transmission model, in which traffic from any multicast group participant reaches all the other multicast group participants, except for cases such as:

- o packet loss, or
- o when a multicast group participant does not wish to receive the traffic for a specific multicast group and, therefore, has not subscribed to the IP multicast group in question. This scenario can occur, for example, where a multimedia session is distributed using two or more multicast groups and a multicast group participant is subscribed only to a subset of these sessions.

In the above context, "traffic" encompasses both RTP and RTCP traffic. The number of multicast group participants can vary between



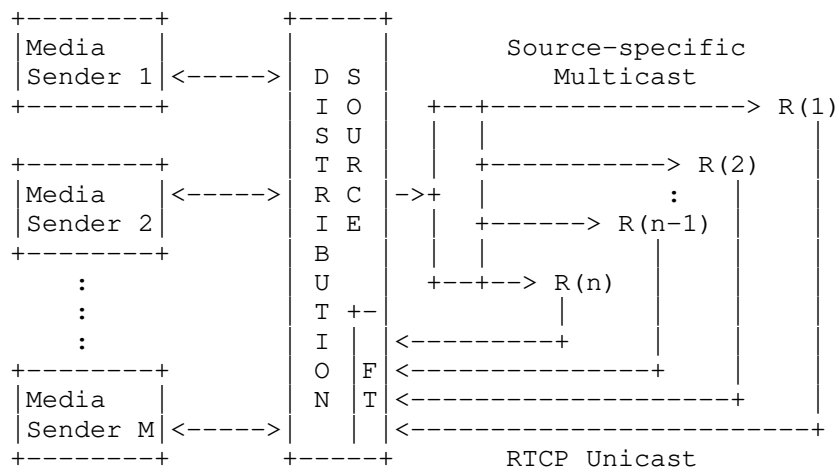
one and many, as RTP and RTCP scale to very large multicast groups (the theoretical limit of the number of participants in a single RTP session is in the range of billions). The above can be realized using Any Source Multicast (ASM).

For feedback usage, it is useful to define a "small multicast group" as a group where the number of multicast group participants is so low (and other factors such as the connectivity is so good) that it allows the participants to use early or immediate feedback, as defined in AVPF [RFC4585]. Even when the environment would allow for the use of a small multicast group, some applications may still want to use the more limited options for RTCP feedback available to large multicast groups, for example when there is a likelihood that the threshold of the small multicast group (in terms of multicast group participants) may be exceeded during the lifetime of a session.

RTCP feedback messages in multicast reach, like media data, every subscriber (subject to packet losses and multicast group subscription). Therefore, the feedback suppression mechanism discussed in [RFC4585] is typically required. Each individual End Point that is a multicast group participant needs to process every feedback message it receives, not only to determine if it is affected or if the feedback message applies only to some other End Point, but also to derive timing restrictions for the sending of its own feedback messages, if any.

### 3.3.2. Source Specific Multicast (SSM)

In Any Source Multicast, any of the multicast group participants can send to all the other multicast group participants, by sending a packet to the multicast group. In contrast, Source Specific Multicast [RFC3569][RFC4607] refers to scenarios 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.

Figure 6: Point to Multipoint using Source Specific Multicast

In the SSM topology (Figure 6) a number of RTP sending End Points (RTP sources henceforth) (1 to M) are allowed to send media to the SSM group. These sources send media to a dedicated distribution source, which forwards the RTP streams to the multicast group on behalf of the original RTP sources. The RTP streams reach the receiving End Points (Receivers henceforth) (R(1) to R(n)). The Receivers' RTCP messages cannot be sent to the multicast group, as the SSM multicast group by definition has only a single IP sender. To support RTCP, an RTP extension for SSM [RFC5760] was defined. It uses unicast transmission to send RTCP from each of the receivers to one or more Feedback Targets (FT). The feedback targets relay the RTCP unmodified, or provide a summary of the participants RTCP reports towards the whole group by forwarding the RTCP traffic to the distribution source. Figure 6 only shows a single feedback target integrated in the distribution source, but for scalability the FT can be distributed and each instance can have responsibility for sub-groups of the receivers. For summary reports, however, there typically must be a single feedback target aggregating all the summaries to a common message to the whole receiver group.

The RTP extension for SSM specifies how feedback (both reception information and specific feedback events) are handled. The more general problems associated with the use of multicast, where everyone

receives what the distribution source sends needs to be accounted for.

Aforementioned situation results in common behavior for RTP multicast:

1. Multicast applications often use a group of RTP sessions, not one. Each End Point needs to be a member of most or all of these RTP sessions in order to perform well.
2. Within each RTP session, the number of media sinks is likely to be much larger than the number of RTP sources.
3. Multicast applications need signalling functions to identify the relationships between RTP sessions.
4. Multicast applications need signalling functions to identify the relationships between SSRCs in different RTP sessions.

All multicast configurations share a signalling requirement: all of the End Points need to have the same RTP and payload type configuration. Otherwise, End Point A could, for example, be using payload type 97 to identify the video codec H.264, while End Point B would identify it as MPEG-2, with unpredictable but almost certainly not visually pleasing results.

Security solutions for this type of group communications are also challenging. First, the key-management and the security protocol must support group communication. Source authentication becomes more difficult and requires specialized solutions. For more discussion on this please review Options for Securing RTP Sessions [RFC7201].

### 3.3.3. SSM with Local Unicast Resources

[RFC6285] "Unicast-Based Rapid Acquisition of Multicast RTP Sessions" results in additional extensions to SSM Topology.

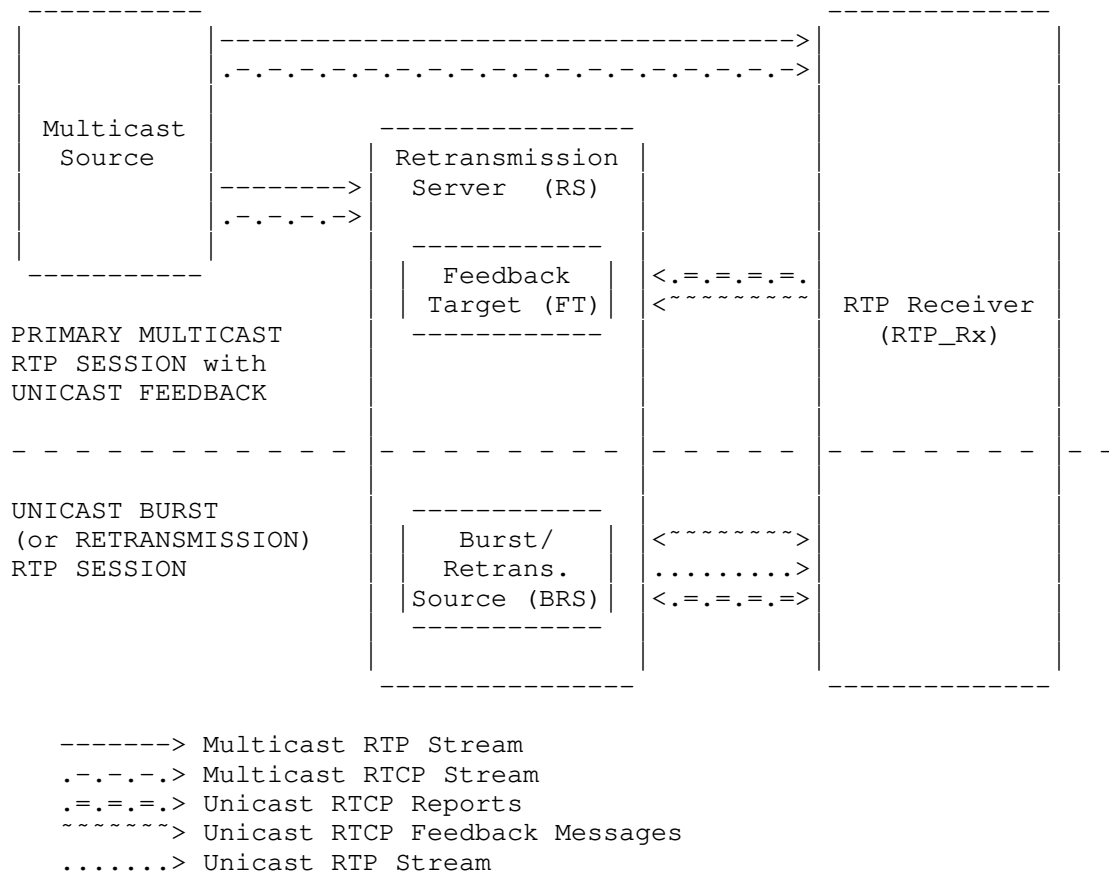


Figure 7

The Rapid acquisition extension allows an End Point joining an SSM multicast session to request media starting with the last sync-point (from where media can be decoded without requiring context established by the decoding of prior packets) to be sent at high speed until such time where, after decoding of these burst-delivered media packets, the correct media timing is established, i.e. media packets are received within adequate buffer intervals for this application. This is accomplished by first establishing a unicast PTP RTP session between the Burst/Retransmission Source (BRS, Figure 7) and the RTP Receiver. The unicast session is used to transmit cached packets from the multicast group at higher than normal speed in order to synchronize the receiver to the ongoing multicast RTP stream. Once the RTP receiver and its decoder have caught up with the multicast session's current delivery, the receiver switches over to receiving directly from the multicast group. The

(still existing) PtP RTP session is, in many deployed applications, be used as a repair channel, i.e., for RTP Retransmission traffic of those packets that were not received from the multicast group.

### 3.4. Point to Multipoint Using Mesh

Shortcut name: Topo-Mesh

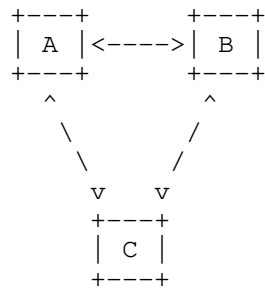


Figure 8: Point to Multi-Point using Mesh

Based on the RTP session definition, it is clearly possible to have a joint RTP session involving three or more End Points over multiple unicast transport flows, like the joint three End point session depicted above. In this case, A needs to send its RTP streams and RTCP packets to both B and C over their respective transport flows. As long as all End Points do the same, everyone will have a joint view of the RTP session.

This topology does not 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 (in which case the sending port, IP address, or SSRC can be used to demultiplex), or it might have separate local reception ports for each of the End Points.

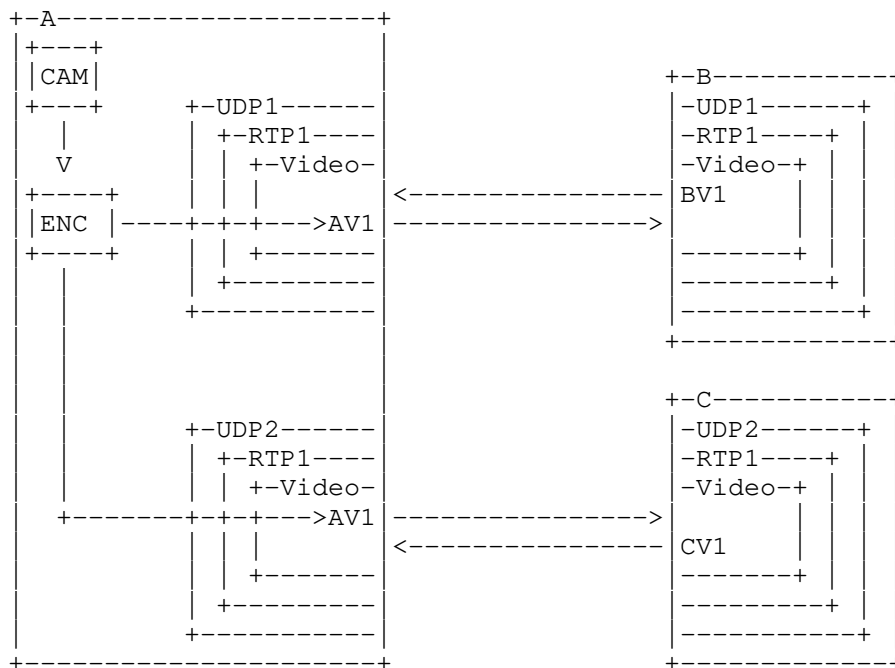


Figure 9: An Multi-unicast Mesh with a joint RTP session

A joint RTP session from End Point A's perspective for the Mesh depicted in Figure 8 with a joint RTP session have multiple transport flows, here enumerated as UDP1 and UDP2. However, there is only one RTP session (RTP1). The Media Source (CAM) is encoded and transmitted over the SSRC (AV1) across both transport layers. However, as this is a joint RTP session, the two streams must be the same. Thus, an congestion control adaptation needed for the paths A to B and A to C needs to use the most restricting path's properties.

An alternative structure for establishing the above topology is to use independent RTP sessions between each pair of peers, i.e., three different RTP sessions. In some scenarios, the same RTP stream may be sent from the transmitting End Point, however it also supports local adaptation taking place in one or more of the RTP streams, rendering them non-identical.

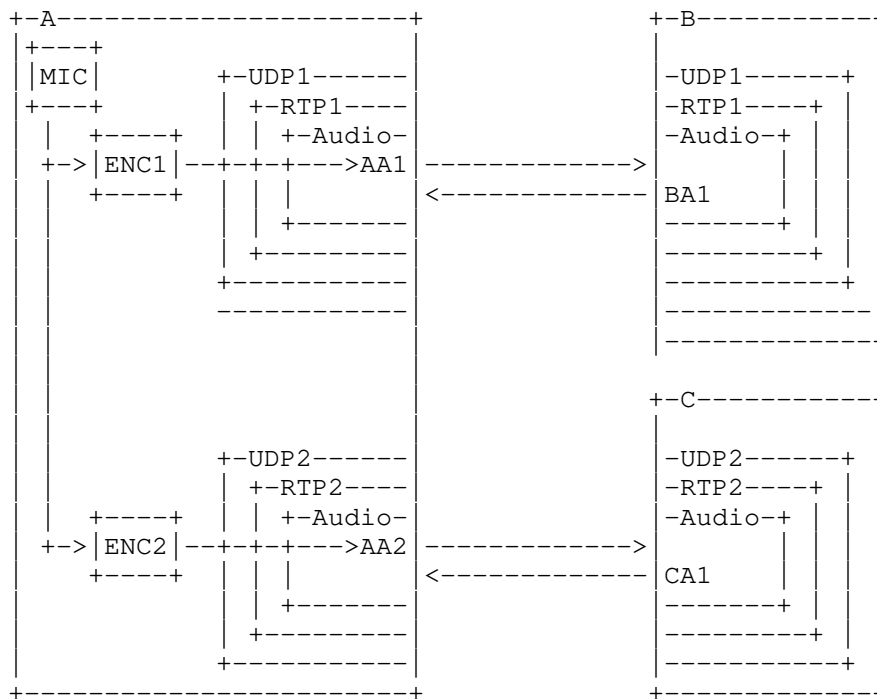


Figure 10: An Multi-unicast Mesh with independent RTP session

Lets review the topology when independent RTP sessions are used, from A's perspective in Figure 8 by considering both how the media is a handled and the RTP sessions that are set-up in Figure 10. A's microphone is captured and the digital audio can then be fed into two different encoder instances, as each being associated with two independent RTP sessions (RTP1 and RTP2). The SSRs (AA1 and AA2) in each RTP session are completely independent and the media bit-rate produced by the encoders can also be tuned differently to address any congestion control requirements differing for the paths A to B compared to A to C.

From a topologies viewpoint, an important difference exists in the behavior around RTCP. First, when a single RTP session spans all three End Points A, B, and C, and their connecting RTP streams, a common RTCP bandwidth is calculated and used for this single joint session. In contrast, when there are multiple independent RTP sessions, each RTP session has its local RTCP bandwidth allocation.

Further, when multiple sessions are used, End Points not directly involved in a session do not have any awareness of the conditions in those sessions. For example, in the case of the three End Point

configuration in Figure 8, End Point A has no awareness of the conditions occurring in the session between End Points B and C (whereas, if a single RTP session were used, it would have such awareness).

Loop detection is also affected. With independent RTP sessions, the SSRC/CSRC cannot be used to determine when an End Point receives its own media stream, or a mixed media stream including its own media stream (a condition known as a loop). The identification of loops and, in most cases, their avoidance, has to be achieved by other means, for example through signaling or the use of an RTP external name space binding SSRC/CSRC among any communicating RTP sessions in the mesh.

### 3.5. Point to Multipoint Using the RFC 3550 Translator

This section discusses some additional usages related to point to multipoint of Translators compared to the point to point only cases in Section 3.2.1.

#### 3.5.1. Relay - Transport Translator

Shortcut name: Topo-PtM-Trn-Translator

This section discusses Transport Translator only usages to enable multipoint sessions.

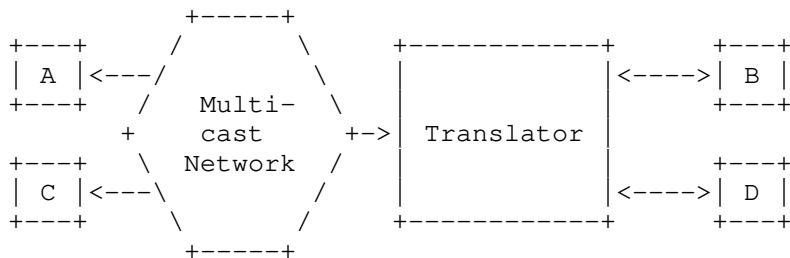


Figure 11: Point to Multipoint Using Multicast

Figure 11 depicts an example of a Transport Translator performing at least IP address translation. It allows the (non-multicast-capable) End Points B and D to take part in an any source multicast session involving End Points A and C, by having the Translator forward their unicast traffic to the multicast addresses in use, and vice versa. It must also forward B's traffic to D, and vice versa, to provide each of B and D with a complete view of the session.



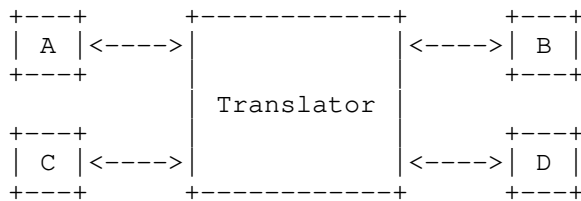


Figure 12: RTP Translator (Relay) with Only Unicast Paths

Another Translator scenario is depicted in Figure 12. The Translator in this case connects multiple End Points through unicast. This can be implemented using a very simple transport Translator which, in this document, is called a relay. The relay forwards all traffic it receives, both RTP and RTCP, to all other End Points. In doing so, a multicast network is emulated without relying on a multicast-capable network infrastructure.

For RTCP feedback this results in a similar set of considerations to those described in the ASM RTP topology. It also puts some additional signalling requirements onto the session establishment; for example, a common configuration of RTP payload types is required.

Transport translators and relays should always consider implementing source address filtering, to prevent attackers to inject traffic using the listening ports on the translator. The translator can, however, go one step further, and especially if explicit SSRC signalling is used, prevent End points to send SSRCs other than its own (that are, for example, used by other participants in the session). This can improve the security properties of the session, despite the use of group keys that on cryptographic level allows anyone to impersonate another in the same RTP session.

A Translator that doesn't change the RTP/RTCP packets content can be operated without the requiring it to have access to the security contexts used to protect the RTP/RTCP traffic between the participants.

### 3.5.2. Media Translator

In the context of multipoint communications a Media Translator is not providing new mechanisms to establish a multipoint session. It is more of an enabler, or facilitator, that ensures a given End Point or a defined sub-set of End Points can participate in the session.

If End Point B in Figure 11 were behind a limited network path, the Translator may perform media transcoding to allow the traffic received from the other End Points to reach B without overloading the

path. This transcoding can help the other End Points in the multicast part of the session, by not requiring the quality transmitted by A to be lowered to the bitrates that B is actually capable of receiving (and vice versa).

### 3.6. Point to Multipoint Using the RFC 3550 Mixer Model

Shortcut name: Topo-Mixer

A Mixer is a middlebox that aggregates multiple RTP streams that are part of a session by generating one or more new RTP streams and, in most cases, by manipulating the media data. One common application for a Mixer is to allow a participant to receive a session with a reduced amount of resources.

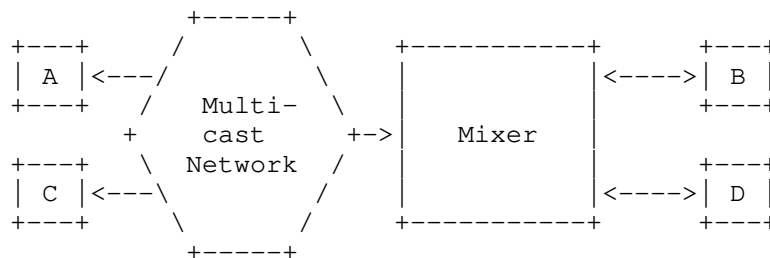


Figure 13: Point to Multipoint Using the RFC 3550 Mixer Model

A Mixer can be viewed as a device terminating the RTP streams received from other End Points in the same RTP session. Using the media data carried in the received RTP streams, a Mixer generates derived RTP streams that are sent to the receiving End Points.

The content that the Mixer provides is the mixed aggregate of what the Mixer receives over the PtP or PtM paths, which are part of the same Communication Session.

The Mixer creates the Media Source and the source RTP stream just like an End Point, as it mixes the content (often in the uncompressed domain) and then encodes and packetizes it for transmission to a receiving endpoint. The CSRC Count (CC) and CSRC fields in the RTP header can be used to indicate the contributors to the newly generated RTP stream. The SSRCs of the to-be-mixed streams on the Mixer input appear as the CSRCs at the Mixer output. That output stream uses a unique SSRC that identifies the Mixer's stream. The CSRC should be forwarded between the different End Points to allow for loop detection and identification of sources that are part of the Communication Session. Note that Section 7.1 of RFC 3550 requires the SSRC space to be shared between domains for these reasons. This

also implies that any SDES information normally needs to be forwarded across the mixer.

The Mixer is responsible for generating RTCP packets in accordance with its role. It is an RTP receiver and should therefore send RTCP receiver reports for the RTP streams it receives and terminates. In its role as an RTP sender, it should also generate RTCP sender reports for those RTP streams it sends. As specified in Section 7.3 of RFC 3550, a Mixer must not forward RTCP unaltered between the two domains.

The Mixer depicted in Figure 13 is involved in three domains that need to be separated: the any source multicast network (including End Points A and C), End Point B, and End Point D. Assuming all four End Points in the conference are interested in receiving content from each other End Point, the Mixer produces different mixed RTP streams for B and D, as the one to B may contain content received from D, and vice versa. However, the Mixer may only need one SSRC per media type in each domain where it is the receiving entity and transmitter of mixed content.

In the multicast domain, a Mixer still needs to provide a mixed view of the other domains. This makes the Mixer simpler to implement and avoids any issues with advanced RTCP handling or loop detection, which would be problematic if the Mixer were providing non-symmetric behavior. Please see Section 3.11 for more discussion on this topic. The mixing operation, however, in each domain could potentially be different.

A Mixer is responsible for receiving RTCP feedback messages and handling them appropriately. The definition of "appropriate" depends on the message itself and the context. In some cases, the reception of a codec-control message by the Mixer may result in the generation and transmission of RTCP feedback messages by the Mixer to the End Points in the other domain(s). In other cases, a message is handled by the Mixer locally and therefore not forwarded to any other domain.

When replacing the multicast network in Figure 13 (to the left of the Mixer) with individual unicast paths as depicted in Figure 14, the Mixer model is very similar to the one discussed in Section 3.9 below. Please see the discussion in Section 3.9 about the differences between these two models.



Figure 14: RTP Mixer with Only Unicast Paths

We now discuss in more detail the different mixing operations that a mixer can perform and how they can affect RTP and RTCP behavior.

### 3.6.1. Media Mixing Mixer

The media mixing mixer is likely the one that most think of when they hear the term "mixer". Its basic mode of operation is that it receives RTP streams from several End Points and selects the stream(s) to be included in a media-domain mix. The selection can be through static configuration or by dynamic, content dependent means such as voice activation. The mixer then creates a single outgoing RTP stream from this mix.

The most commonly deployed media mixer is probably the audio mixer, used in voice conferencing, where the output consists of a mixture of all the input audio signals; this needs minimal signalling to be successfully set up. From a signal processing viewpoint, audio mixing is relatively straightforward and commonly possible for a reasonable number of End Points. Assume, for example, that one wants to mix  $N$  streams from  $N$  different End Points. The mixer needs to decode those  $N$  streams, typically into the sample domain, and then produce  $N$  or  $N+1$  mixes. Different mixes are needed so that each contributing source gets a mix of all other sources except its own, as this would result in an echo. When  $N$  is lower than the number of all End points, one may produce a mix of all  $N$  streams for the group that are currently not included in the mix, thus  $N+1$  mixes. These audio streams are then encoded again, RTP packetized and sent out. In many cases, audio level normalization, noise suppression, and similar signal processing steps are also required or desirable before the actual mixing process commences.

In video, the term "mixing" has a different interpretation than audio. It is commonly used to refer to the process of spatially combining contributed video streams, which is also known as "tiling". The reconstructed, appropriately scaled down videos can be spatially arranged in a set of tiles, each tile containing the video from an End Point (typically showing a human participant). Tiles can be of different sizes, so that, for example, a particularly important

participant, or the loudest speaker, is being shown on in larger tile than other participants. A self-view picture can be included in the tiling, which can either be locally produced or be a feedback from a mixer-received and reconstructed video image. Such remote loopback allows for confidence monitoring, i.e., it enables the participant to see himself/herself in the same quality as other participants see him/her. The tiling normally operates on reconstructed video in the sample domain. The tiled image is encoded, packetized, and sent by the mixer to the receiving End Points. It is possible that a middlebox with media mixing duties contains only a single mixer of the aforementioned type, in which case all participants necessarily see the same tiled video, even if it is being sent over different RTP streams. More common, however, are mixing arrangements where an individual mixer is available for each outgoing port of the middlebox, allowing individual compositions for each receiving End Point (a feature commonly referred to as personalized layout).

One problem with media mixing is that it consumes both large amounts of media processing resources (for the decoding and mixing process in the uncompressed domain) and encoding resources (for the encoding of the mixed signal). Another problem is the quality degradation created by decoding and re-encoding the media, which is the result of the lossy nature of most commonly used media codecs. A third problem is the latency introduced by the media mixing, which can be substantial and annoyingly noticeable in case of video, or in case of audio if that mixed audio is lip-synchronized with high latency video. The advantage of media mixing is that it is straightforward for the End Points to handle the single media stream (which includes the mixed aggregate of many sources), as they don't need to handle multiple decodings, local mixing and composition. In fact, mixers were introduced in pre-RTP times so that legacy, single stream receiving endpoints (that, in some protocol environments, actually didn't need to be aware of the multipoint nature of the conference) could successfully participate in what a user would recognize as a multiparty video conference.

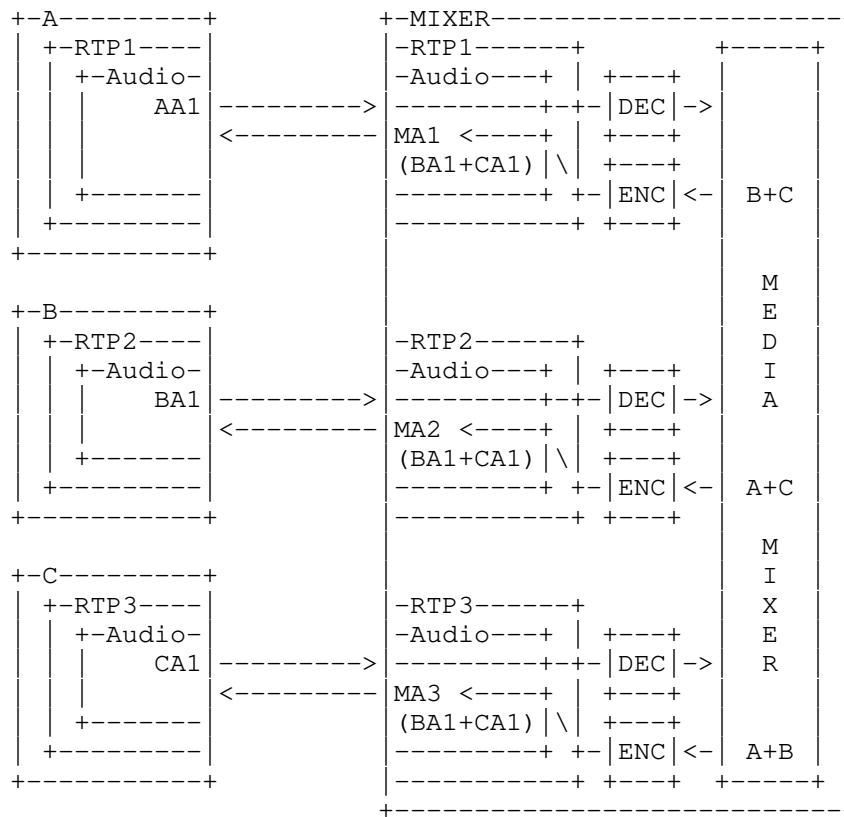


Figure 15: Session and SSRC details for Media Mixer

From an RTP perspective media mixing can be a very simple process, as can be seen in Figure 15. The mixer presents one SSRC towards the receiving End Point, e.g., MA1 to Peer A, where the associated stream is the media mix of the other End Points. As each peer, in this example, receives a different version of a mix from the mixer, there is no actual relation between the different RTP sessions in terms of actual media or transport level information. There are, however, common relationships between RTP1-RTP3, namely SSRC space and identity information. When A receives the MA1 stream which is a combination of BA1 and CA1 streams, the mixer may include CSRC information in the MA1 stream to identify the contributing source BA1 and CA1, allowing the receiver to identify the contributing sources even if this were not possible through the media itself or through other signaling means.

The CSRC has, in turn, utility in RTP extensions, like the Mixer to Client audio levels RTP header extension [RFC6465]. If the SSRCs

from the End Point to mixer paths are used as CSRCs in another RTP session, then RTP1, RTP2 and RTP3 become one joint session as they have a common SSRC space. At this stage, the mixer also needs to consider which RTCP information it needs to expose in the different paths. In the above scenario, a mixer would normally expose nothing more than the Source Description (SDES) information and RTCP BYE for a CSRC leaving the session. The main goal would be to enable the correct binding against the application logic and other information sources. This also enables loop detection in the RTP session.

### 3.6.2. Media Switching

Media switching mixers are used in limited functionality scenarios where no, or only very limited, concurrent presentation of multiple sources is required by the application, to more complex multi-stream usages with receiver mixing or tiling, including combined with simulcast and/or scalability between source and mixer. An RTP Mixer based on media switching avoids the media decoding and encoding operations in the mixer, as it conceptually forwards the encoded media stream as it was being sent to the mixer. It does not avoid, however, the decryption and re-encryption cycle as it rewrites RTP headers. Forwarding media (in contrast to reconstructing-mixing-encoding media) reduces the amount of computational resources needed in the mixer and increases the media quality (both in terms of fidelity and reduced latency).

A media switching mixer maintains a pool of SSRCs representing conceptual or functional RTP streams that the mixer can produce. These RTP streams are created by selecting media from one of the RTP streams received by the mixer and forwarded to the peer using the mixer's own SSRCs. The mixer can switch between available sources if that is required by the concept for the source, like the currently active speaker. Note that the mixer, in most cases, still needs to perform a certain amount of media processing, as many media formats do not allow to "tune into" the stream at arbitrary points in their bitstream.

To achieve a coherent RTP stream from the mixer's SSRC, the mixer needs to rewrite the incoming RTP packet's header. First the SSRC field must be set to the value of the Mixer's SSRC. Second, the sequence number must be the next in the sequence of outgoing packets it sent. Third, the RTP timestamp value needs to be adjusted using an offset that changes each time one switches media source. Finally, depending on the negotiation of the RTP payload type, the value representing this particular RTP payload configuration may have to be changed if the different End Point-to-mixer paths have not arrived on the same numbering for a given configuration. This also requires that the different End Points support a common set of codecs,

otherwise media transcoding for codec compatibility would still be required.

We now consider the operation of a media switching mixer that supports a video conference with six participating End Points (A-F) where the two most recent speakers in the conference are shown to each receiving End Point. The mixer has thus two SSRCs sending video to each peer, and each peer is capable of locally handling two video streams simultaneously.

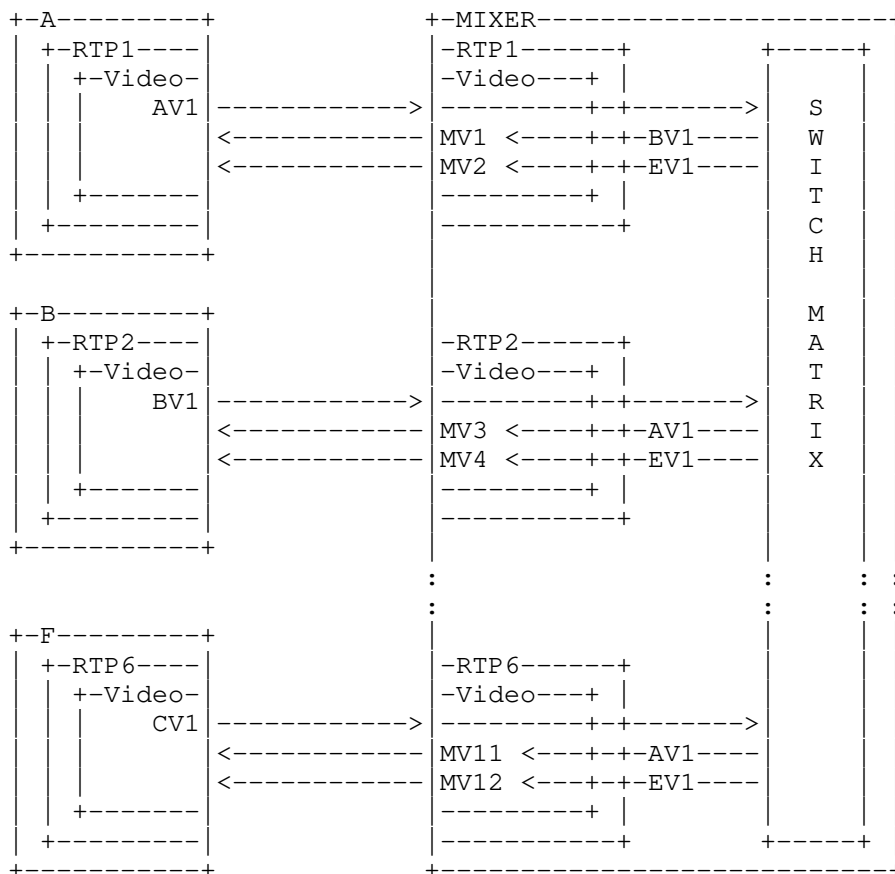


Figure 16: Media Switching RTP Mixer

The Media Switching RTP mixer can, similarly to the Media Mixing Mixer, reduce the bit-rate required for media transmission towards the different peers by selecting and forwarding only a sub-set of RTP streams it receives from the sending End Points. In cases the mixer



receives simulcast transmissions or a scalable encoding of the media source, the mixer has more degrees of freedom to select streams or sub-sets of stream to forward to a receiving End Point, both based on transport or End Point restrictions as well as application logic.

To ensure that a media receiver in an End Point can correctly decode the media in the RTP stream after a switch, a codec that uses temporal prediction needs to start its decoding from independent refresh points, or points in the bitstream offering similar functionality (like "dirty refresh points"). For some codecs, for example frame based speech and audio codecs, this is easily achieved by starting the decoding at RTP packet boundaries, as each packet boundary provides a refresh point (assuming proper packetization on the encoder side). For other codecs, particularly in video, refresh points are less common in the bitstream or may not be present at all without an explicit request to the respective encoder. The Full Intra Request [RFC5104] RTCP codec control message has been defined for this purpose.

In this type of mixer one could consider to fully terminate the RTP sessions between the different End Point and mixer paths. The same arguments and considerations as discussed in Section 3.9 need to be taken into consideration and apply here.

### 3.7. Selective Forwarding Middlebox

Another method for handling media in the RTP mixer is to "project", or make available, all potential RTP sources (SSRCs) into a per-End Point, independent RTP session. The middlebox can select which of the potential sources that are currently actively transmitting media will be sent to each of the End Points. This is similar to the media switching Mixer but has some important differences in RTP details.

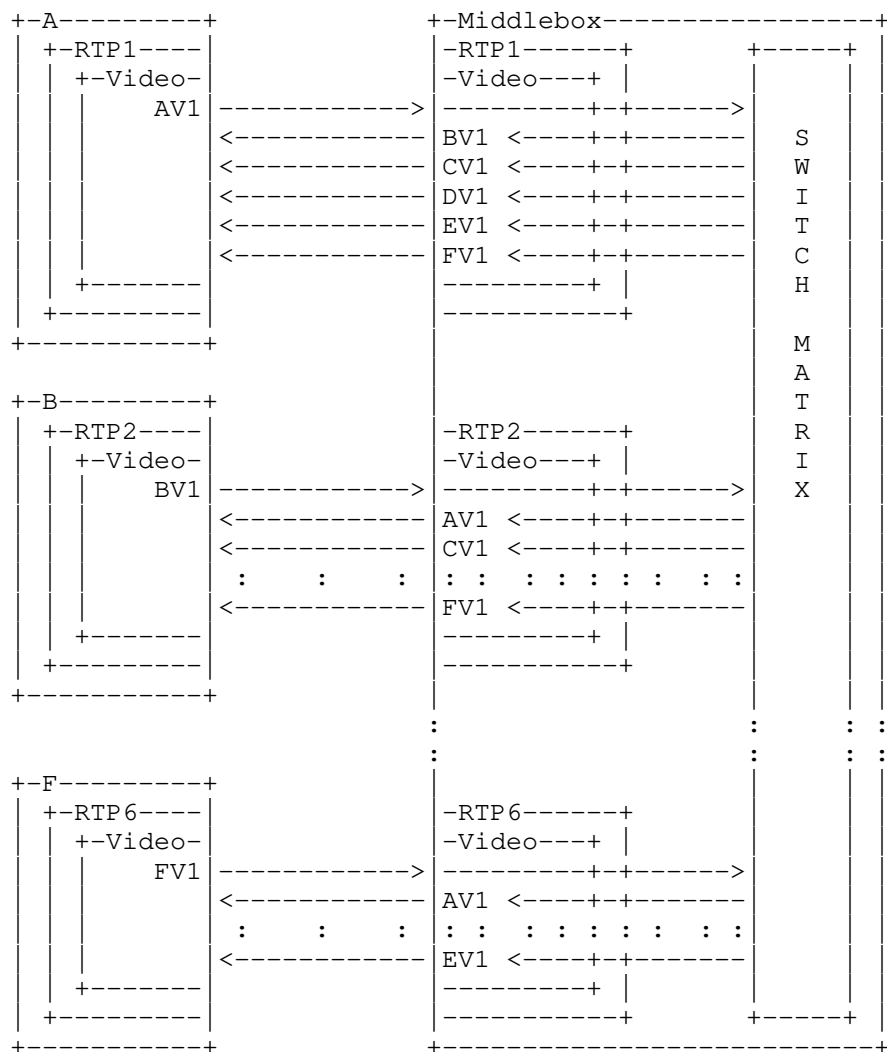


Figure 17: Selective Forwarding Middlebox

In the six End Point conference depicted above in (Figure 17) one can see that End Point A is aware of five incoming SSRCs, BV1-FV1. If this middlebox intends to have a similar behavior as in Section 3.6.2 where the mixer provides the End Points with the two latest speaking End Points, then only two out of these five SSRCs need concurrently transmit media to A. As the middlebox selects the source in the different RTP sessions that transmit media to the End points, each RTP stream requires rewriting of certain RTP header fields when being projected from one session into another. In particular, the sequence

number needs to be consecutively incremented based on the packet actually being transmitted in each RTP session. Therefore, the RTP sequence number offset will change each time a source is turned on in a RTP session. The timestamp (possibly offset) stays the same.

As the RTP sessions are independent, the SSRC numbers used can also be handled independently, thereby bypassing the requirement for SSRC collision detection and avoidance. On the other hand, tools such as remapping tables between the RTP sessions are required. For example, the RTP stream that is being sent by End Point B to the middlebox (BV1) may use an SSRC value of 12345678. When that RTP stream is sent to End Point F by the middlebox, it can use any SSRC value, e.g. 87654321. As a result, each End Point may have a different view of the application usage of a particular SSRC. Any RTP level identity information, such as SDP items also needs to update the SSRC referenced, if the included SDP items are intended to be global. Thus the application must not use SSRC as references to RTP streams when communicating with other peers directly. This also affects loop detection which will fail to work, as there is no common namespace and identities across the different legs in the communication session on RTP level. Instead this responsibility falls onto higher layers.

The middlebox is also responsible to receive any RTCP codec control requests coming from an End Point, and decide if it can act on the request locally or needs to translate the request into the RTP session that contains the media source. Both End Points and the middlebox need to implement conference related codec control functionalities to provide a good experience. Commonly used are Full Intra Request to request from the media source to provide switching points between the sources, and Temporary Maximum Media Bit-rate Request (TMMBR) to enable the middlebox to aggregate congestion control responses towards the media source so to enable it to adjust its bit-rate (obviously only in case the limitation is not in the source to middlebox link).

The selective forwarding middlebox has been introduced in recently developed videoconferencing systems in conjunction with, and to capitalize on, scalable video coding as well as simulcasting. An example of scalable video coding is Annex G of H.264, but other codecs, including H.264 AVC and VP8 also exhibit scalability, albeit only in the temporal dimension. In both scalable coding and simulcast cases the video signal is represented by a set of two or more bitstreams, providing a corresponding number of distinct fidelity points. The middlebox selects which parts of a scalable bitstream (or which bitstream, in the case of simulcasting) to forward to each of the receiving End Points. The decision may be driven by a number of factors, such as available bit rate, desired layout, etc. Contrary to transcoding MCUs, these "Selective

Forwarding Units" (SFUs) have extremely low delay, and provide features that are typically associated with high-end systems (personalized layout, error localization) without any signal processing at the middlebox. They are also capable of scaling to a large number of concurrent users, and--due to their very low delay--can also be cascaded.

This version of the middlebox also puts different requirements on the End Point when it comes to decoder instances and handling of the RTP streams providing media. As each projected SSRC can, at any time, provide media, the End Point either needs to be able to handle as many decoder instances as the middlebox received, or have efficient switching of decoder contexts in a more limited set of actual decoder instances to cope with the switches. The application also gets more responsibility to update how the media provided is to be presented to the user.

Note that this topology could potentially be seen as a media translator which include an on/off logic as part of its media translation. The main difference would be a common global SSRC space in the case of the Media Translator and the mapped one used in the above. It also has mixer aspects, as the streams it provides are not basically translated version, but instead they have conceptual property assigned to them. Thus this topology appears to be some hybrid between the translator and mixer model.

The differences between selective forwarding middlebox and a switching mixer (Section 3.6.2) are minor, and they share most properties. The above requirement on having a large number of decoding instances or requiring efficient switching of decoder contexts, are one point of difference. The other is how the identification is performed, where the Mixer uses CSRC to provide information on what is included in a particular RTP stream that represent a particular concept. Selective forwarding gets the source information through the SSRC, and instead have to use other mechanism to make clear the streams current purpose.

### 3.8. Point to Multipoint Using Video Switching MCUs

Shortcut name: Topo-Video-switch-MCU

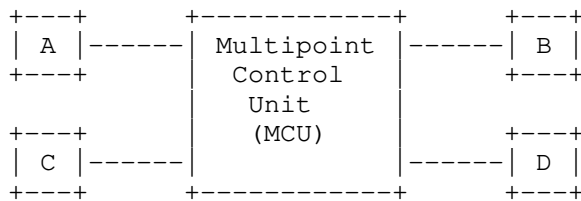


Figure 18: Point to Multipoint Using a Video Switching MCU

This PtM topology was popular in early implementations of multipoint videoconferencing systems due to its simplicity, and the corresponding middlebox design has been known as a "video switching MCU". The more complex RTCP-terminating MCUs, discussed in the next section, became the norm, however, when technology allowed implementations at acceptable costs.

A video switching MCU forwards to a participant a single media stream, selected from the available streams. The criteria for selection are often based on voice activity in the audio-visual conference, but other conference management mechanisms (like presentation mode or explicit floor control) are known to exist as well.

The video switching MCU may also perform media translation to modify the content in bit-rate, encoding, or resolution. However, it still may indicate the original sender of the content through the SSRC. In this case, the values of the CC and CSRC fields are retained.

If not terminating RTP, the RTCP Sender Reports are forwarded for the currently selected sender. All RTCP Receiver Reports are freely forwarded between the End points. In addition, the MCU may also originate RTCP control traffic in order to control the session and/or report on status from its viewpoint.

The video switching MCU has most of the attributes of a Translator. However, its stream selection is a mixing behavior. This behavior has some RTP and RTCP issues associated with it. The suppression of all but one RTP stream results in most participants seeing only a subset of the sent RTP streams at any given time, often a single RTP stream per conference. Therefore, RTCP Receiver Reports only report on these RTP streams. Consequently, the End Points emitting RTP streams that are not currently forwarded receive a view of the session that indicates their RTP streams disappear somewhere en route. This makes the use of RTCP for congestion control, or any type of quality reporting, very problematic.

To avoid the aforementioned issues, the MCU needs to implement two features. First, it needs to act as a Mixer (see Section 3.6) and forward the selected RTP stream under its own SSRC and with the appropriate CSRC values. Second, the MCU needs to modify the RTCP RRs it forwards between the domains. As a result, it is recommended that one implement a centralized video switching conference using a Mixer according to RFC 3550, instead of the shortcut implementation described here.

### 3.9. Point to Multipoint Using RTCP-Terminating MCU

Shortcut name: Topo-RTCP-terminating-MCU

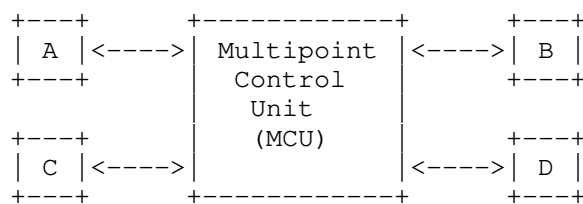


Figure 19: Point to Multipoint Using Content Modifying MCUs

In this PtM scenario, each End Point runs an RTP point-to-point session between itself and the MCU. This is a very commonly deployed topology in multipoint video conferencing. The content that the MCU provides to each participant is either:

- a selection of the content received from the other End Points, or
- the mixed aggregate of what the MCU receives from the other PtP paths, which are part of the same Communication Session.

In case (a), the MCU may modify the content in terms of bit-rate, encoding format, or resolution. No explicit RTP mechanism is used to establish the relationship between the original RTP stream of the media being sent RTP stream the MCU sends. In other words, the outgoing RTP streams typically use a different SSRC, and may well use a different payload type (PT), even if this different PT happens to be mapped to the same media type. This is a result of the individually negotiated RTP session for each End Point.

In case (b), the MCU is the Media Source and generates the Source RTP Stream as it mixes the received content and then encodes and packetizes it for transmission to an End Point. According to RTP [RFC3550], the SSRC of the contributors are to be signalled using the CSRC/CC mechanism. In practice, today, most deployed MCUs do not implement this feature. Instead, the identification of the End

Points whose content is included in the Mixer's output is not indicated through any explicit RTP mechanism. That is, most deployed MCUs set the CSRC Count (CC) field in the RTP header to zero, thereby indicating no available CSRC information, even if they could identify the original sending End Points as suggested in RTP.

The main feature that sets this topology apart from what RFC 3550 describes is the breaking of the common RTP session across the centralized device, such as the MCU. This results in the loss of explicit RTP-level indication of all participants. If one were using the mechanisms available in RTP and RTCP to signal this explicitly, the topology would follow the approach of an RTP Mixer. The lack of explicit indication has at least the following potential problems:

1. Loop detection cannot be performed on the RTP level. When carelessly connecting two misconfigured MCUs, a loop could be generated.
2. There is no information about active media senders available in the RTP packet. As this information is missing, receivers cannot use it. It also deprives the client of information related to currently active senders in a machine-usable way, thus preventing clients from indicating currently active speakers in user interfaces, etc.

Note that many/most deployed MCUs (and video conferencing endpoints) rely on signalling layer mechanisms for the identification of the contributing sources, for example, a SIP conferencing package [RFC4575]. This alleviates, to some extent, the aforementioned issues resulting from ignoring RTP's CSRC mechanism.

### 3.10. Split Component Terminal

Shortcut name: Topo-Split-Terminal

In some applications, for example in some telepresence systems, terminals may be not integrated into a single functional unit, but composed of more than one subunits. For example, a telepresence room terminal employing multiple cameras and monitors may consist of multiple video conferencing subunits, each capable of handling a single camera and monitor. Another example would be a video conferencing terminal in which audio is handled by one subunit, and video by another. Each of these subunits uses its own physical network interface (for example: Ethernet jack) and network address.

The various (media processing) subunits need (logically and physically) to be interconnected by control functionality, but their media plane functionality may be split. This type of terminals is

referred to as split component terminals. Historically, the earliest split component terminals were perhaps the (independent) audio and video conference software tools used over the MBONE in the late 1990s.

An example for such a split component terminal is depicted in Figure 20. Within split component terminal A, at least audio and video subunits are addressed by their own network addresses. In some of these systems, the control stack subunit may also have its own network address.

From an RTP viewpoint, each of the subunits terminates RTP, and acts as an End Point in the sense that each subunit includes its own, independent RTP stack. However, as the subunits are semantically part of the same terminal, it is appropriate that this semantic relationship is expressed in RTCP protocol elements, namely in the CNAME.

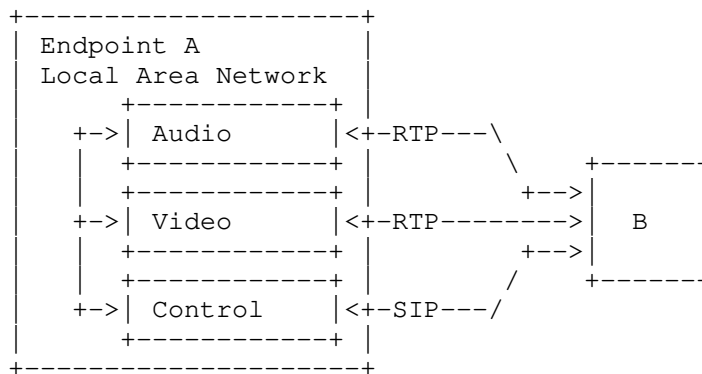


Figure 20: Split Component Terminal

It is further sensible that the subunits share a common clock from which RTP and RTCP clocks are derived, to facilitate synchronization and avoid clock drift.

To indicate that audio and video Source Streams generated by different sub-units share a common clock, and can be synchronized, the RTP streams generated from those Source Streams need to include the same CNAME in their RTCP SDES packets. The use of a common CNAME for RTP flows carried in different transport-layer flows is entirely normal for RTP and RTCP senders, and fully compliant RTP End Points, middle-boxes, and other tools should have no problem with this.

However, outside of the split component terminal scenario (and perhaps a multi-homed End Point scenario, which is not further



discussed herein), the use of a common CNAME in RTP streams sent from separate endpoints (as opposed to a common CNAME for RTP streams sent on different transport layer flows between two endpoints) is rare. It has been reported that at least some third party tools like some network monitors do not handle endpoints that use of a common CNAME across multiple transport layer flows gracefully: they report an error condition that two separate End Points are using the same CNAME. Depending on the sophistication of the support staff, such erroneous reports can lead to support issues.

Aforementioned support issue can sometimes be avoided if each of the subunits of a split component terminal is configured to use a different CNAME, with the synchronization between the RTP streams being indicated by some non-RTP signaling channel rather than using a common CNAME sent in RTCP. This complicates the signaling, especially in cases where there are multiple SSRCs in use with complex synchronization requirements, as is the same in many current telepresence systems. Unless one uses RTCP terminating topologies such as Topo-RTCP-terminating-MCU, sessions involving more than one video subunit with a common CNAME are close to unavoidable.

The different RTP streams comprising a split terminal system can form a single RTP session or they can form multiple RTP sessions, depending on the visibility of their SSRC values in RTCP reports. If the receiver of the RTP streams sent by the split terminal sends reports relating to all of the RTP flows (i.e., to each SSRC) in each RTCP report then a single RTP session is formed. Alternatively, if the receiver of the RTP streams sent by the split terminal does not send cross-reports in RTCP, then the audio and video form separate RTP sessions.

For example, in the Figure 20, B will send RTCP reports to each of the sub-units of A. If the RTCP packets that B sends to the audio sub-unit of A include reports on the reception quality of the video as well as the audio, and similarly if the RTCP packets that B sends to the video sub-unit of A include reports on the reception quality of the audio as well as video, then a single RTP session is formed. However, if the RTCP packets B sends to the audio sub-unit of A only report on the received audio, and the RTCP packet B sends to the video sub-unit of A only report on the received video, then there are two separate RTP sessions.

Forming a single RTP session across the RTP streams sent by the different sub-units of a split terminal gives each sub-unit visibility into reception quality of RTP streams sent by the other sub-units. This information can help diagnose reception quality problems, but at the cost of increased RTCP bandwidth use.

RTP streams sent by the sub-units of a split terminal need to use the same CNAME in their RTCP packets if they are to be synchronized, irrespective of whether a single RTP session is formed or not.

### 3.11. Non-Symmetric Mixer/Translators

Shortcut name: Topo-Asymmetric

It is theoretically possible to construct an MCU that is a Mixer in one direction and a Translator in another. The main reason to consider this would be to allow topologies similar to Figure 13, where the Mixer does not need to mix in the direction from B or D towards the multicast domains with A and C. Instead, the RTP streams from B and D are forwarded without changes. Avoiding this mixing would save media processing resources that perform the mixing in cases where it isn't needed. However, there would still be a need to mix B's media towards D. Only in the direction B -> multicast domain or D -> multicast domain would it be possible to work as a Translator. In all other directions, it would function as a Mixer.

The Mixer/Translator would still need to process and change the RTCP before forwarding it in the directions of B or D to the multicast domain. One issue is that A and C do not know about the mixed-media stream the Mixer sends to either B or D. Therefore, any reports related to these streams must be removed. Also, receiver reports related to A and C's RTP streams would be missing. To avoid A and C thinking that B and D aren't receiving A and C at all, the Mixer needs to insert locally generated reports reflecting the situation for the streams from A and C into B and D's Sender Reports. In the opposite direction, the Receiver Reports from A and C about B's and D's stream also need to be aggregated into the Mixer's Receiver Reports sent to B and D. Since B and D only have the Mixer as source for the stream, all RTCP from A and C must be suppressed by the Mixer.

This topology is so problematic and it is so easy to get the RTCP processing wrong, that it is not recommended for implementation.

### 3.12. Combining Topologies

Topologies can be combined and linked to each other using Mixers or Translators. However, care must be taken in handling the SSRC/CSRC space. A Mixer does not forward RTCP from sources in other domains, but instead generates its own RTCP packets for each domain it mixes into, including the necessary Source Description (SDS) information for both the CSRCs and the SSRCs. Thus, in a mixed domain, the only SSRCs seen will be the ones present in the domain, while there can be CSRCs from all the domains connected together with a combination of

Mixers and Translators. The combined SSRC and CSRC space is common over any Translator or Mixer. It is important to facilitate loop detection, something that is likely to be even more important in combined topologies due to the mixed behavior between the domains. Any hybrid, like the Topo-Video-switch-MCU or Topo-Asymmetric, requires considerable thought on how RTCP is dealt with.

#### 4. Comparing Topologies

The topologies discussed in Section 3 have different properties. This section first describes these properties and then analyzes how these properties are supported by the different topologies. Note that, even if a certain property is supported within a particular topology concept, the necessary functionality may be optional to implement.

##### 4.1. Topology Properties

###### 4.1.1. All to All Media Transmission

To recapitulate, multicast, and in particular Any Source Multicast (ASM), provides the functionality that everyone may send to, or receive from, everyone else within the session. Source-specific Multicast (SSM) can provide a similar functionality by having anyone intending to participate as sender to send its media to the SSM distribution source. The SSM distribution source forwards the media to all receivers subscribed to the multicast group. Mesh, MCUs, Mixers, SFMs and Translators may all provide that functionality at least on some basic level. However, there are some differences in which type of reachability they provide.

Closest to true IP-multicast-based, all-to-all transmission comes perhaps the transport Translator function called "relay" in in Section 3.5, as well as the Mesh with joint RTP sessions. Media Translators, Mesh with independent RTP Sessions, Mixers, SFUs and the MCU variants do not provide a fully meshed forwarding on the transport level; instead, they only allow limited forwarding of content from the other session participants.

The "all to all media transmission" requires that any media transmitting End Point considers the path to the least capable receiving End Point. Otherwise, the media transmissions may overload that path. Therefore, a sending End Point needs to monitor the path from itself to any of the receiving End Points, to detect the currently least capable receiver, and adapt its sending rate accordingly. As multiple End Points may send simultaneously, the available resources may vary. RTCP's Receiver Reports help performing this monitoring, at least on a medium time scale.

The resource consumption for performing all to all transmission varies depending with the topology. Both ASM and SSM have the benefit that only one copy of each packet traverses a particular link. Using a relay causes the transmission of one copy of a packet per End Point-to-relay path and packet transmitted. However, in most cases the links carrying the multiple copies will be the ones close to the relay (which can be assumed to be part of the network infrastructure with good connectivity to the backbone), rather than the End Points (which may be behind slower access links). The Mesh causes  $N-1$  streams of transmitted packets to traverse the first hop link from the End Point, in an  $N$  End Point mesh. How long the different paths are common, is highly situation dependent.

The transmission of RTCP by design adapts to any changes in the number of participants due to the transmission algorithm, defined in the RTP specification [RFC3550], and the extensions in AVPF [RFC4585] (when applicable). That way, the resources utilized for RTCP stay within the bounds configured for the session.

#### 4.1.2. Transport or Media Interoperability

All Translators, Mixers, and RTCP-terminating MCU, and Mesh with individual RTP sessions, allow changing the media encoding or the transport to other properties of the other domain, thereby providing extended interoperability in cases where the End Points lack a common set of media codecs and/or transport protocols. Selective Forwarding Middleboxes can adopt the transport, and (at least) selectively forward the encoded streams that match a receiving End Point's capability. It requires an additional translator to change the media encoding if the encoded streams do not match the receiving End Point's capabilities.

#### 4.1.3. Per Domain Bit-Rate Adaptation

End Points are often connected to each other with a heterogeneous set of paths. This makes congestion control in a Point to Multipoint set problematic. For the ASM, SSM, Mesh with common RTP session, and Transport Relay scenario, each individual sending End Point has to adapt to the receiving End Point behind the least capable path, yielding suboptimal quality for the End Points behind the more capable paths. This is no longer an issue when Media Translators, Mixers, SFM or MCUs are involved, as each End Point only needs to adapt to the slowest path within its own domain. The Translator, Mixer, SFM, or MCU topologies all require their respective outgoing RTP streams to adjust the bit-rate, packet-rate, etc., to adapt to the least capable path in each of the other domains. That way one can avoid lowering the quality to the least-capable End Point in all the domains at the cost (complexity, delay, equipment) of the Mixer,

SFM or Translator, and potentially media sender (multicast/layered encoding and sending the different representations).

#### 4.1.4. Aggregation of Media

In the all-to-all media property mentioned above and provided by ASM, SSM, Mesh with common RTP session, and relay, all simultaneous media transmissions share the available bit-rate. For End Points with limited reception capabilities, this may result in a situation where even a minimal acceptable media quality cannot be accomplished, because multiple RTP streams need to share the same resources. One solution to this problem is to provide for a Mixer, or MCU to aggregate the multiple RTP streams into a single one, where the single RTP stream takes up less resources in terms of bit-rate. This aggregation can be performed according to different methods. Mixing or selection are two common methods. Selection is almost always possible and easy to implement. Mixing requires resources in the mixer, and may be relatively easy and not impairing the quality too badly (audio) or quite difficult (video tiling, which is not only computationally complex but also reduces the pixel count per stream, with corresponding loss in perceptual quality).

#### 4.1.5. View of All Session Participants

The RTP protocol includes functionality to identify the session participants through the use of the SSRC and CSRC fields. In addition, it is capable of carrying some further identity information about these participants using the RTCP Source Descriptors (SDS). In topologies that provide a full all-to-all functionality, i.e. ASM, Mesh with common RTP session, Relay a compliant RTP implementation offers the functionality directly as specified in RTP. In topologies that do not offer all-to-all communication, it is necessary that RTCP is handled correctly in domain bridging function. RTP includes explicit specification text for Translators and Mixers, and for SFMs the required functionality can be derived from that text. However, the MCU described in Section 3.8 cannot offer the full functionality for session participant identification through RTP means. The topologies that create independent RTP sessions per End Point or pair of End Points, like Back-to-Back RTP session, MESH with independent RTP sessions, and the RTCP terminating MCU RTCP terminating MCU (Section 3.9) do not support RTP based identification of session participants. In all those cases, other non-RTP based mechanisms need to be implemented if such knowledge is required or desirable.

#### 4.1.6. Loop Detection

In complex topologies with multiple interconnected domains, it is possible to unintentionally form media loops. RTP and RTCP support detecting such loops, as long as the SSRC and CSRC identities are maintained and correctly set in forwarded packets. Loop detection will work in ASM, SSM, Mesh with joint RTP session, and Relay. It is likely that loop detection works for the video switching MCU Section 3.8, at least as long as it forwards the RTCP between the End Points. However, the Back-to-Back RTP sessions, Mesh with independent RTP sessions, SFM, will definitely break the loop detection mechanism.

#### 4.2. Comparison of Topologies

The table below attempts to summarize the properties of the different topologies. The legend to the topology abbreviations are: Topo-Point-to-Point (PtP), Topo-ASM (ASM), Topo-SSM (SSM), Topo-Trns-Translator (TT), Topo-Media-Translator (including Transport Translator) (MT), Topo-Mesh with joint session (MJS), Topo-Mesh with individual sessions (MIS), Topo-Mixer (Mix), Topo-Asymmetric (ASY), Topo-Video-switch-MCU (VSM), and Topo-RTCP-terminating-MCU (RTM), Selective Forwarding Middlebox (SFM). In the table below, Y indicates Yes or full support, N indicates No support, (Y) indicates partial support, and N/A indicates not applicable.

Property	PtP	ASM	SSM	TT	MT	MJS	MIS	Mix	ASY	VSM	RTM	SFM
All to All media	N	Y	(Y)	Y	Y	Y	(Y)	(Y)	(Y)	(Y)	(Y)	(Y)
Interoperability	N/A	N	N	Y	Y	Y	Y	Y	Y	N	Y	Y
Per Domain Adaptation	N/A	N	N	N	Y	N	Y	Y	Y	N	Y	Y
Aggregation of media	N	N	N	N	N	N	N	Y	(Y)	Y	Y	N
Full Session View	Y	Y	Y	Y	Y	Y	N	Y	Y	(Y)	N	Y
Loop Detection	Y	Y	Y	Y	Y	Y	N	Y	Y	(Y)	N	N

Please note that the Media Translator also includes the transport Translator functionality.

#### 5. Security Considerations

The use of Mixers, SFMs and Translators has impact on security and the security functions used. The primary issue is that both Mixers, SFMs and Translators modify packets, thus preventing the use of integrity and source authentication, unless they are trusted devices that take part in the security context, e.g., the device can send Secure Realtime Transport Protocol (SRTCP) and Secure Realtime Transport Control Protocol (SRTCP) [RFC3711] packets to End Points in the Communication Session. If encryption is employed, the media

Translator, SFM and Mixer need to be able to decrypt the media to perform its function. A transport Translator may be used without access to the encrypted payload in cases where it translates parts that are not included in the encryption and integrity protection, for example, IP address and UDP port numbers in a media stream using SRTP [RFC3711]. However, in general, the Translator, SFM or Mixer needs to be part of the signalling context and get the necessary security associations (e.g., SRTP crypto contexts) established with its RTP session participants.

Including the Mixer, SFM and Translator in the security context allows the entity, if subverted or misbehaving, to perform a number of very serious attacks as it has full access. It can perform all the attacks possible (see RFC 3550 and any applicable profiles) as if the media session were not protected at all, while giving the impression to the human session participants that they are protected.

Transport Translators have no interactions with cryptography that works above the transport layer, such as SRTP, since that sort of Translator leaves the RTP header and payload unaltered. Media Translators, on the other hand, have strong interactions with cryptography, since they alter the RTP payload. A media Translator in a session that uses cryptographic protection needs to perform cryptographic processing to both inbound and outbound packets.

A media Translator may need to use different cryptographic keys for the inbound and outbound processing. For SRTP, different keys are required, because an RFC 3550 media Translator leaves the SSRC unchanged during its packet processing, and SRTP key sharing is only allowed when distinct SSRCs can be used to protect distinct packet streams.

When the media Translator uses different keys to process inbound and outbound packets, each session participant needs to be provided with the appropriate key, depending on whether they are listening to the Translator or the original source. (Note that there is an architectural difference between RTP media translation, in which participants can rely on the RTP Payload Type field of a packet to determine appropriate processing, and cryptographically protected media translation, in which participants must use information that is not carried in the packet.)

When using security mechanisms with Translators, SFMs and Mixers, it is possible that the Translator, SFM or Mixer could create different security associations for the different domains they are working in. Doing so has some implications:

First, it might weaken security if the Mixer/Translator accepts a weaker algorithm or key in one domain than in another. Therefore, care should be taken that appropriately strong security parameters are negotiated in all domains. In many cases, "appropriate" translates to "similar" strength. If a key management system does allow the negotiation of security parameters resulting in a different strength of the security, then this system should notify the participants in the other domains about this.

Second, the number of crypto contexts (keys and security related state) needed (for example, in SRTP [RFC3711]) may vary between Mixers, SFMs and Translators. A Mixer normally needs to represent only a single SSRCs per domain and therefore needs to create only one security association (SRTP crypto context) per domain. In contrast, a Translator needs one security association per participant it translates towards, in the opposite domain. Considering Figure 11, the Translator needs two security associations towards the multicast domain, one for B and one for D. It may be forced to maintain a set of totally independent security associations between itself and B and D respectively, so as to avoid two-time pad occurrences. These contexts must also be capable of handling all the sources present in the other domains. Hence, using completely independent security associations (for certain keying mechanisms) may force a Translator to handle  $N \cdot DM$  keys and related state; where N is the total number of SSRCs used over all domains and DM is the total number of domains.

The multicast based (ASM and SSM), Relay and Mesh with common RTP session are all topologies with multiple End Points that require shared knowledge about the different crypto contexts for the End Points. These multi-party topologies have special requirements on the key-management as well as the security functions. Specifically source-authentication in these environments has special requirements.

There exist a number of different mechanisms to provide keys to the different participants. One example is the choice between group keys and unique keys per SSRC. The appropriate keying model is impacted by the topologies one intends to use. The final security properties are dependent on both the topologies in use and the keying mechanisms' properties, and need to be considered by the application. Exactly which mechanisms are used is outside of the scope of this document. Please review RTP Security Options [RFC7201] to get a better understanding of most of the available options.

## 6. IANA Considerations

This document makes no request of IANA.



Note to RFC Editor: this section may be removed on publication as an RFC.

## 7. Acknowledgements

The authors would like to thank Mark Baugher, Bo Burman, Umesh Chandra, Alex Eleftheriadis, Roni Even, Ladan Gharai, Geoff Hunt, Keith Lantz, and Colin Perkins for their help in reviewing and improving this document.

## 8. References

### 8.1. 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.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.
- [RFC4575] Rosenberg, J., Schulzrinne, H., and O. Levin, "A Session Initiation Protocol (SIP) Event Package for Conference State", RFC 4575, August 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.

### 8.2. Informative References

- [I-D.ietf-avtcore-rtp-multi-stream-optimisation] Lennox, J., Westerlund, M., Wu, W., and C. Perkins, "Sending Multiple Media Streams in a Single RTP Session: Grouping RTCP Reception Statistics and Other Feedback", draft-ietf-avtcore-rtp-multi-stream-optimisation-03 (work in progress), July 2014.
- [RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, August 1989.
- [RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

- [RFC3569] Bhattacharyya, S., "An Overview of Source-Specific Multicast (SSM)", RFC 3569, July 2003.
- [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.
- [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.
- [RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.
- [RFC6465] Ivov, E., Marocco, E., and J. Lennox, "A Real-time Transport Protocol (RTP) Header Extension for Mixer-to-Client Audio Level Indication", RFC 6465, December 2011.
- [RFC7201] Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", RFC 7201, April 2014.

#### Authors' Addresses

Magnus Westerlund  
Ericsson  
Farogatan 6  
SE-164 80 Kista  
Sweden

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

Stephan Wenger  
Vidyo  
433 Hackensack Ave  
Hackensack, NJ 07601  
USA

Email: stewe@stewe.org

AVTCORE Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: April 23, 2015

J. Mattsson, Ed.  
Ericsson  
D. McGrew  
D. Wing  
F. Andreassen  
Cisco  
October 20, 2014

Encrypted Key Transport for Secure RTP  
draft-ietf-avtcore-srtp-ekt-03

Abstract

Encrypted Key Transport (EKT) is an extension to Secure Real-time Transport Protocol (SRTP) that provides for the secure transport of SRTP master keys, Rollover Counters, and other information. This facility enables SRTP to work for decentralized conferences with minimal control.

This note defines EKT, and also describes how to use it with SDP Security Descriptions, DTLS-SRTP, and MIKEY. With EKT, these other key management protocols provide an EKT key to everyone in a session, and EKT coordinates the SRTP keys within the session.

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 23, 2015.

Copyright Notice

Copyright (c) 2014 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
1.1. History . . . . .	4
1.2. Conventions Used In This Document . . . . .	5
2. Encrypted Key Transport . . . . .	5
2.1. EKT Field Formats . . . . .	6
2.2. Packet Processing and State Machine . . . . .	8
2.2.1. Outbound Processing . . . . .	8
2.2.2. Inbound Processing . . . . .	9
2.3. Ciphers . . . . .	11
2.3.1. The Default Cipher . . . . .	12
2.3.2. Other EKT Ciphers . . . . .	13
2.4. Synchronizing Operation . . . . .	13
2.5. Transport . . . . .	13
2.6. Timing and Reliability Consideration . . . . .	15
3. Use of EKT with SDP Security Descriptions . . . . .	16
3.1. SDP Security Descriptions Recap . . . . .	16
3.2. Relationship between EKT and SDESC . . . . .	17
3.3. Overview of Combined EKT and SDESC Operation . . . . .	19
3.4. EKT Extensions to SDP Security Descriptions . . . . .	19
3.5. Offer/Answer Considerations . . . . .	20
3.5.1. Generating the Initial Offer - Unicast Streams . . . . .	20
3.5.2. Generating the Initial Answer - Unicast Streams . . . . .	21
3.5.3. Processing of the Initial Answer - Unicast Streams . . . . .	22
3.6. SRTP-Specific Use Outside Offer/Answer . . . . .	23
3.7. Modifying the Session . . . . .	23
3.8. Backwards Compatibility Considerations . . . . .	24
3.9. Grammar . . . . .	25
4. Use of EKT with DTLS-SRTP . . . . .	25
4.1. DTLS-SRTP Recap . . . . .	26
4.2. EKT Extensions to DTLS-SRTP . . . . .	26
4.3. Offer/Answer Considerations . . . . .	28
4.3.1. Generating the Initial Offer . . . . .	28
4.3.2. Generating the Initial Answer . . . . .	29
4.3.3. Processing the Initial Answer . . . . .	29
4.3.4. Sending DTLS EKT Key Reliably . . . . .	30
4.3.5. Modifying the Session . . . . .	30

5. Use of EKT with MIKEY . . . . .	30
5.1. EKT Extensions to MIKEY . . . . .	32
5.2. Offer/Answer Considerations . . . . .	33
5.2.1. Generating the Initial Offer . . . . .	33
5.2.2. Generating the Initial Answer . . . . .	34
5.2.3. Processing the Initial Answer . . . . .	34
5.2.4. Modifying the Session . . . . .	35
6. Using EKT for Interoperability between Key Management Systems	35
7. Design Rationale . . . . .	36
7.1. Alternatives . . . . .	37
8. Security Considerations . . . . .	37
9. IANA Considerations . . . . .	39
10. Acknowledgements . . . . .	39
11. References . . . . .	40
11.1. Normative References . . . . .	40
11.2. Informative References . . . . .	41
Appendix A. Using EKT to Optimize Interworking DTLS-SRTP with Security Descriptions . . . . .	42
Authors' Addresses . . . . .	44

## 1. Introduction

RTP is designed to allow decentralized groups with minimal control to establish sessions, such as for multimedia conferences. Unfortunately, Secure RTP (SRTP [RFC3711]) cannot be used in many minimal-control scenarios, because it requires that SSRC values and other data be coordinated among all of the participants in a session. For example, if a participant joins a session that is already in progress, that participant needs to be told the SRTP keys (and SSRC, ROC and other details) of the other SRTP sources.

The inability of SRTP to work in the absence of central control was well understood during the design of the protocol; the omission was considered less important than optimizations such as bandwidth conservation. Additionally, in many situations SRTP is used in conjunction with a signaling system that can provide most of the central control needed by SRTP. However, there are several cases in which conventional signaling systems cannot easily provide all of the coordination required. It is also desirable to eliminate the layer violations that occur when signaling systems coordinate certain SRTP parameters, such as SSRC values and ROCs.

This document defines Encrypted Key Transport (EKT) for SRTP, an extension to SRTP that fits within the SRTP framework and reduces the amount of external signaling control that is needed in an SRTP session. EKT securely distributes the SRTP master key and other information for each SRTP source (SSRC), using SRTCP or SRTP to transport that information. With this method, SRTP entities are free

to choose SSRC values as they see fit, and to start up new SRTP sources (SSRC) with new SRTP master keys (see Section 2.2) within a session without coordinating with other entities via external signaling or other external means. This fact allows to reinstate the RTP collision detection and repair mechanism, which is nullified by the current SRTP specification because of the need to control SSRC values closely. An SRTP endpoint using EKT can generate new keys whenever an existing SRTP master key has been overused, or start up a new SRTP source (SSRC) to replace an old SRTP source that has reached the packet-count limit. However, EKT does not allow SRTP's ROC to rollover; that requires re-keying outside of EKT (e.g., using MIKEY or DTLS-SRTP). EKT also solves the problem in which the burst loss of the N initial SRTP packets can confuse an SRTP receiver, when the initial RTP sequence number is greater than or equal to  $2^{16} - N$ . These features can simplify many architectures that implement SRTP.

EKT provides a way for an SRTP session participant, either a sender or receiver, to securely transport its SRTP master key and current SRTP rollover counter to the other participants in the session. This data, possibly in conjunction with additional data provided by an external signaling protocol, furnishes the information needed by the receiver to instantiate an SRTP/SRTCP receiver context.

EKT does not control the manner in which the SSRC is generated; it is only concerned with their secure transport. Those values may be generated on demand by the SRTP endpoint, or may be dictated by an external mechanism such as a signaling agent or a secure group controller.

EKT is not intended to replace external key establishment mechanisms such as SDP Security Descriptions [RFC4568], DTLS-SRTP [RFC5764], or MIKEY [RFC3830][RFC4563]. Instead, it is used in conjunction with those methods, and it relieves them of the burden of tightly coordinating every SRTP source (SSRC) among every SRTP participant.

### 1.1. History

[[RFC Editor Note: please remove this section prior to publication as an RFC.]]

A substantial change occurred between the EKT documents draft-ietf-avt-srtp-ekt-03 and draft-ietf-avtcore-srtp-ekt-00. The change makes it possible for the EKT data to be removed from a packet without affecting the ability of the receiver to correctly process the data that is present in that packet. This capability facilitates interoperability between SRTP implementations with different SRTP key management methods. The changes also greatly simplify the EKT

processing rules, and makes the EKT data that must be carried in SRTP and/or SRTCP packets somewhat larger.

In draft-ietf-avtcore-srtp-ekt-02, SRTP master keys have to be always generated randomly and not re-used, MKI is no longer allowed with EKT (as MKI duplicates some of EKT's functions), and text clarifies that EKT must be negotiated during call setup. Some text was consolidated and re-written, notably Section 2.6 ("Timing and Reliability"). Support for re-directing the DTLS-SRTP handshake to another host was removed, as it needed NAT traversal support.

In draft-ietf-avtcore-srtp-ekt-03, the SRTCP compound packet problem is discussed. Updates and clarifications were made to the SDESC and MIKEY sections.

## 1.2. Conventions Used In This Document

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 [RFC2119].

## 2. Encrypted Key Transport

In EKT, an SRTP master key is encrypted with a key encrypting key and the resulting ciphertext is transported in selected SRTCP packets or in selected SRTP packets. The key encrypting key is called an EKT key. A single such key suffices for a single SRTP session, regardless of the number of participants in that session. However, there can be multiple EKT keys used within a particular session.

EKT defines a new method of providing SRTP master keys to an endpoint. In order to convey the ciphertext of the SRTP master key, and other additional information, an additional EKT field is added to SRTP or SRTCP packets. When added to SRTCP, the EKT field appears at the end of the packet, after the authentication tag, if that tag is present, or after the SRTCP index otherwise. When added to SRTP, The EKT field appears at the end of the SRTP packet, after the authentication tag (if that tag is present), or after the ciphertext of the encrypted portion of the packet otherwise.

EKT MUST NOT be used in conjunction with SRTP's MKI (Master Key Identifier) or with SRTP's <From, To> [RFC3711], as those SRTP features duplicate some of the functions of EKT.

## 2.1. EKT Field Formats

The EKT Field uses one of the two formats defined below. These two formats can always be unambiguously distinguished on receipt by examining the final bit of the EKT Field, which is also the final bit of the SRTP or SRTCP packet. The first format is the Full EKT Field (or Full\_EKT\_Field), and the second is the Short EKT Field (or Short\_EKT\_Field). The formats are defined as

```
EKT_Plaintext = SRTP_Master_Key || SSRC || ROC || ISN  
  
EKT_Ciphertext = EKT_Encrypt(EKT_Key, EKT_Plaintext)  
  
Full_EKT_Field = EKT_Ciphertext || SPI || '1'  
  
Short_EKT_Field = Reserved || '0'
```

Figure 1: EKT data formats

Here || denotes concatenation, and '1' and '0' denote single one and zero bits, respectively. These fields and data elements are defined as follows:

**EKT\_Plaintext:** The data that is input to the EKT encryption operation. This data never appears on the wire, and is used only in computations internal to EKT.

**EKT\_Ciphertext:** The data that is output from the EKT encryption operation, described in Section 2.3. This field is included in SRTP and SRTCP packets when EKT is in use. The length of this field is variable, and is equal to the ciphertext size N defined in Section 2.3. Note that the length of the field is inferable from the SPI field, since the particular EKT cipher used by the sender of a packet can be inferred from that field.

**SRTP\_Master\_Key:** On the sender side, the SRTP Master Key associated with the indicated SSRC. The length of this field depends on the cipher suite negotiated during call setup for SRTP or SRTCP.

**SSRC:** On the sender side, this field is the SSRC for this SRTP source. The length of this field is fixed at 32 bits.

**Rollover Counter (ROC):** On the sender side, this field is set to the current value of the SRTP rollover counter in the SRTP context associated with the SSRC in the SRTP or SRTCP packet. The length of this field is fixed at 32 bits.



**Initial Sequence Number (ISN):** If this field is nonzero, it indicates the RTP sequence number of the initial RTP packet that is protected using the SRTP master key conveyed (in encrypted form) by the EKT Ciphertext field of this packet. When this field is present in an RTCP packet it indicates the RTP sequence number of the first RTP packet encrypted by this master key. If the ISN field is zero, it indicates that the initial RTP/RTCP packet protected using the SRTP master key conveyed in this packet preceded, or was concurrent with, the last roll-over of the RTP sequence number, and thus should be used as the current master key for processing this packet. The length of this field is fixed at 16 bits.

**Security Parameter Index (SPI):** This field is included in SRTP and SRTCP packets when EKT is in use. It indicates the appropriate EKT key and other parameters for the receiver to use when processing the packet. It is an "index" into a table of possibilities (which are established via signaling or some other out-of-band means), much like the IPsec Security Parameter Index [RFC4301]. The length of this field is fixed at 15 bits. The parameters identified by this field are:

- \* The EKT key used to process the packet.
- \* The EKT cipher used to process the packet.
- \* The Secure RTP parameters associated with the SRTP Master Key carried by the packet and the SSRC value in the packet. Section 8.2. of [RFC3711] summarizes the parameters defined by that specification.
- \* The Master Salt associated with the Master Key. (This value is part of the parameters mentioned above, but we call it out for emphasis.) The Master Salt is communicated separately, via signaling, typically along with the EKT key.

Together, these data elements are called an EKT parameter set. Within each SRTP session, each distinct EKT parameter set that may be used MUST be associated with a distinct SPI value, to avoid ambiguity.

**Reserved:** The length of this field is 7 bits. MUST be all zeros on transmission, and MUST be ignored on reception.

The Full\_EKT\_Field and Short\_EKT\_Field formats are shown in Figure 2 and Figure 3, respectively. These figures show the on-the-wire data. The Ciphertext field holds encrypted data, and thus has no apparent inner structure.

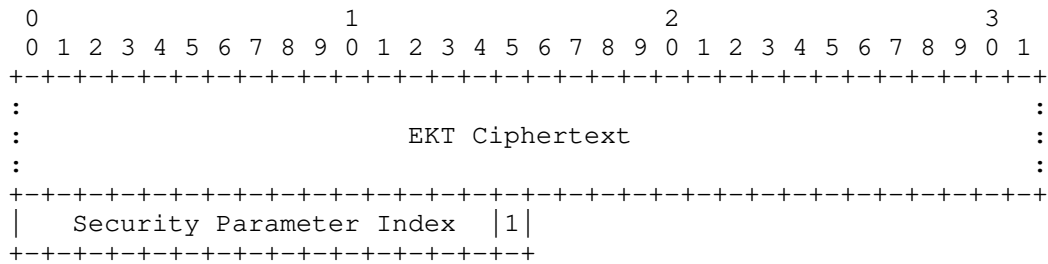


Figure 2: Full EKT Field format

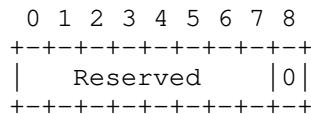


Figure 3: Short EKT Field format

## 2.2. Packet Processing and State Machine

At any given time, each SRTP/SRTCP source (SSRC) has associated with it a single EKT parameter set. This parameter set is used to process all outbound packets, and is called the outbound parameter set. There may be other EKT parameter sets that are used by other SRTP/SRTCP sources in the same session, including other SRTP/SRTCP sources on the same endpoint (e.g., one endpoint with voice and video might have two EKT parameter sets, or there might be multiple video sources on an endpoint each with their own EKT parameter set). All of these EKT parameter sets SHOULD be stored by all of the participants in an SRTP session, for use in processing inbound SRTP and SRTCP traffic.

All SRTP master keys **MUST NOT** be re-used, **MUST** be randomly generated according to [RFC4086], and **MUST NOT** be equal to or derived from other SRTP master keys.

### 2.2.1. Outbound Processing

See Section 2.6 which describes when to send an EKT packet and describes if a Full EKT Field or Short EKT Field is sent.

When an SRTP or SRTCP packet is to be sent, the EKT field for that packet is created as follows, or uses an equivalent set of steps. The creation of the EKT field MUST precede the normal SRTP or SRTCP packet processing. The ROC used in EKT processing MUST be the same as the one used in the SRTP processing.

If the Short format is used, an all-zero octet is appended to the packet. Otherwise, processing continues as follows.

The Rollover Counter field in the packet is set to the current value of the SRTP rollover counter (represented as an unsigned integer in network byte order).

The Initial Sequence Number field is set to zero, if the initial RTP packet protected using the current SRTP master key for this source preceded, or was concurrent with, the last roll-over of the RTP sequence number. Otherwise, that field is set to the value of the RTP sequence number of the initial RTP packet that was or will be protected by that key. See "rekey" in Section 2.6. The rekeying event MUST NOT change the value of ROC (otherwise, the current value of the ROC would not be known to late joiners of existing sessions). This means rekeying near the end of sequence number space (e.g., 100 packets before sequence number 65535) is not possible because ROC needs to roll over.

The Security Parameter Index field is set to the value of the Security Parameter Index that is associated with the outbound parameter set.

The EKT\_Plaintext field is computed from the SRTP Master Key, SSRC, ROC, and ISN fields, as shown in Figure 1.

The EKT\_Ciphertext field is set to the ciphertext created by encrypting the EKT\_Plaintext with the EKT cipher, using the EKT Key as the encryption key. The encryption process is detailed in Section 2.3. Implementations MAY cache the value of this field to avoid recomputing it for each packet that is sent.

Implementation note: Because of the format of the Full EKT Field, a packet containing the Full EKT Field MUST be sent when the ROC changes (i.e., every  $2^{16}$  packets).

#### 2.2.2. Inbound Processing

When an SRTP or SRTCP packet containing a Full EKT Field or Short EKT Field is received, it is processed as follows or using an equivalent set of steps. Inbound EKT processing MUST take place prior to the usual SRTP or SRTCP processing. Implementation note: the receiver may want to have a sliding window to retain old master keys for some brief period of time, so that out of order packets can be processed. The following steps show processing as packets are received in order.

1. The final bit is checked to determine which EKT format is in use. If the packet contains a Short EKT Field then the Short EKT Field

is removed and normal SRTP or SRTCP processing is applied. If the packet contains a Full EKT Field, then processing continues as described below.

2. The Security Parameter Index (SPI) field is checked to determine which EKT parameter set should be used when processing the packet. If multiple parameter sets have been defined for the SRTP session, then the one that is associated with the value of the SPI field in the packet is used. This parameter set is called the matching parameter set below. If there is no matching SPI, then the verification function MUST return an indication of authentication failure, and the steps described below are not performed.
3. The EKT\_Ciphertext is decrypted using the EKT\_Key and EKT\_Cipher in the matching parameter set, as described in Section 2.3. If the EKT decryption operation returns an authentication failure, then the packet processing halts with an indication of failure. Otherwise, the resulting EKT\_Plaintext is parsed as described in Figure 1, to recover the SRTP Master Key, SSRC, ROC, and ISN fields.
4. The SSRC field output from the decryption operation is compared to the SSRC field from the SRTP header if EKT was received over SRTP, or from the SRTCP header if EKT was received over SRTCP. If the values of the two fields do not match, then packet processing halts with an indication of failure. Otherwise, it continues as follows.
5. If an SRTP context associated with the SSRC in the previous step already exists and the ROC from the EKT\_Plaintext is less than the ROC in the SRTP context, then EKT processing halts and the packet is processed as an out-of-order packet (if within the implementation's sliding window) or dropped (as it is a replay). Otherwise, the ROC in the SRTP context is set to the value of the ROC from the EKT\_Plaintext, and the SRTP Master Key from the EKT\_Plaintext is accepted as the SRTP master key corresponding to the SSRC indicated in the EKT\_Plaintext, beginning at the sequence number indicated by the ISN (see next step).
6. If the ISN from the EKT\_Plaintext is less than the RTP sequence number of an authenticated received SRTP packet, then EKT processing halts (as this is a replay). If the Initial Sequence Number field is nonzero, then the initial sequence number for the SRTP master key is set to the packet index created by appending that field to the current rollover counter and treating the result as a 48-bit unsigned integer. The initial sequence number for the master key is equivalent to the "From" value of the

<From, To> pair of indices (Section 8.1.1 of [RFC3711]) that can be associated with a master key.

7. The newly accepted SRTP master key, the SRTP parameters from the matching parameter set, and the SSRC from the packet are stored in the crypto context associated with the SRTP source (SSRC). The SRTP Key Derivation algorithm is run in order to compute the SRTP encryption and authentication keys, and those keys are stored for use in SRTP processing of inbound packets. The Key Derivation algorithm takes as input the newly accepted SRTP master key, along with the Master Salt from the matching parameter set.
8. At this point, EKT processing has successfully completed, and the normal SRTP or SRTCP processing takes place.

Implementation note: the value of the EKT Ciphertext field is identical in successive packets protected by the same EKT parameter set and the same SRTP master key, ROC, and ISN. This ciphertext value MAY be cached by an SRTP receiver to minimize computational effort by noting when the SRTP master key is unchanged and avoiding repeating Steps 2 through 6.

### 2.3. Ciphers

EKT uses an authenticated cipher to encrypt the EKT Plaintext, which is comprised of the SRTP master keys, SSRC, ROC, and ISN. We first specify the interface to the cipher, in order to abstract the interface away from the details of that function. We then define the cipher that is used in EKT by default. The default cipher described in Section 2.3.1 MUST be implemented, but another cipher that conforms to this interface MAY be used, in which case its use MUST be coordinated by external means (e.g., key management).

The master salt length for the offered cipher suites MUST be the same. In practice the easiest way to achieve this is by offering the same crypto suite.

An EKT cipher consists of an encryption function and a decryption function. The encryption function  $E(K, P)$  takes the following inputs:

- o a secret key  $K$  with a length of  $L$  bytes, and
- o a plaintext value  $P$  with a length of  $M$  bytes.

The encryption function returns a ciphertext value  $C$  whose length is  $N$  bytes, where  $N$  is at least  $M$ . The decryption function  $D(K, C)$  takes the following inputs:

- o a secret key  $K$  with a length of  $L$  bytes, and
- o a ciphertext value  $C$  with a length of  $N$  bytes.

The decryption function returns a plaintext value  $P$  that is  $M$  bytes long, or returns an indication that the decryption operation failed because the ciphertext was invalid (i.e. it was not generated by the encryption of plaintext with the key  $K$ ).

These functions have the property that  $D(K, E(K, P)) = P$  for all values of  $K$  and  $P$ . Each cipher also has a limit  $T$  on the number of times that it can be used with any fixed key value. For each key, the encryption function **MUST NOT** be invoked on more than  $T$  distinct values of  $P$ , and the decryption function **MUST NOT** be invoked on more than  $T$  distinct values of  $C$ .

The length of the EKT Plaintext is ten bytes, plus the length of the SRTP Master Key.

Security requirements for EKT ciphers are discussed in Section 8.

### 2.3.1. The Default Cipher

The default EKT Cipher is the Advanced Encryption Standard (AES) [FIPS197] Key Wrap with Padding [RFC5649] algorithm. It requires a plaintext length  $M$  that is at least one octet, and it returns a ciphertext with a length of  $N = M + 8$  octets. It can be used with key sizes of  $L = 16, 24$ , and  $32$ , and its use with those key sizes is indicated as AESKW\_128, AESKW\_192, and AESKW\_256, respectively. The key size determines the length of the AES key used by the Key Wrap algorithm. With this cipher,  $T=2^{48}$ .

SRTP transform	EKT transform	length of EKT plaintext	length of EKT ciphertext	length of Full EKT Field
AES-128	AESKW_128 (m)	26	40	42
AES-192	AESKW_192	34	48	50
AES-256	AESKW_256	42	56	58
F8-128	AESKW_128	26	40	42
SEED-128	AESKW_128	26	40	42

Figure 4: AESKW Table

The mandatory to implement transform is AESKW\_128, indicated by (m).

As AES-128 is the mandatory to implement transform in SRTP [RFC3711], AESKW\_128 MUST be implemented for EKT.

For all the SRTP transforms listed in the table, the corresponding EKT transform MUST be used, unless a stronger EKT transform is negotiated by key management.

#### 2.3.2. Other EKT Ciphers

Other specifications may extend this one by defining other EKT ciphers per Section 9. This section defines how those ciphers interact with this specification.

An EKT cipher determines how the EKT Ciphertext field is written, and how it is processed when it is read. This field is opaque to the other aspects of EKT processing. EKT ciphers are free to use this field in any way, but they SHOULD NOT use other EKT or SRTP fields as an input. The values of the parameters L, M, N, and T MUST be defined by each EKT cipher, and those values MUST be inferable from the EKT parameter set.

#### 2.4. Synchronizing Operation

A participant in a session MAY opt to use a particular EKT parameter set to protect outbound packets after it accepts that EKT parameter set for protecting inbound traffic. In this case, the fact that one participant has changed to using a new EKT key for outbound traffic can trigger other participants to switch to using the same key.

If a source has its EKT key changed by key management, it MUST also change its SRTP master key, which will cause it to send out a new Full EKT Field. This ensures that if key management thought the EKT key needs changing (due to a participant leaving or joining) and communicated that in key management to a source, the source will also change its SRTP master key, so that traffic can be decrypted only by those who know the current EKT key.

The use of EKT MUST be negotiated during key management or call setup (e.g., using DTLS-SRTP, Security Descriptions, MIKEY, or similar).

#### 2.5. Transport

EKT SHOULD be used over SRTP, and MAY be used over SRTCP. SRTP is preferred because it shares fate with transmitted media, because SRTP rekeying can occur without concern for RTCP transmission limits, and to avoid SRTCP compound packets with RTP translators and mixers.

This specification requires the EKT SSRC match the SSRC in the RTCP header, but Section 6.1 of [RFC3550] encourages creating SRTCP compound packets:

It is RECOMMENDED that translators and mixers combine individual RTCP packets from the multiple sources they are forwarding into one compound packet whenever feasible in order to amortize the packet overhead (see Section 7).

These compound SRTCP packets might have an SSRC that does not match the EKT SSRC. To reduce the occasion of this occurring, EKT-aware RTP mixers and translators which are generating SRTCP compound packets SHOULD attempt to place an SRTCP payload containing an EKT tag at the front of the compound packet (so that the EKT receiver will process it), and MAY be even more robust and implement more sophisticated algorithms to ensure all EKT tags from different senders are sent at the front of the compound packet. However, no robust algorithm exists which ensures robust EKT delivery in conjunction with SRTCP compound packets. This impact to RTP translators and mixers, and the inability to reliably determine an RTP translator or mixer might be involved in an RTP session, provides further incentive to send EKT over RTP.

The packet processing, state machine, and Authentication Tag format for EKT over SRTP are nearly identical to that for EKT over SRTCP. Differences are highlighted in Section 2.2.1 and Section 2.2.2.

The Full EKT Field is appended to the SRTP or SRTCP payload and is 42, 50, or 58 octets long for AES-128, AES-192, or AES-256, respectively. This length impacts the maximum payload size of the SRTP (or SRTCP) packet itself. To remain below the network path MTU, senders SHOULD constrain the SRTP (or SRTCP) payload size by this length of the Full EKT Field.

EKT can be transported over SRTCP, but some of the information that it conveys is used for SRTP processing; some elements of the EKT parameter set apply to both SRTP and SRTCP. Furthermore, SRTCP packets can be lost and both SRTP and SRTCP packets may be delivered out of order. This can lead to various race conditions if EKT is transported over SRTCP but not SRTP, which we review below.

The ROC signaled via EKT over SRTCP may be off by one when it is received by the other party(ies) in the session. In order to deal with this, receivers should simply follow the SRTP packet index estimation procedures defined in Section 3.3.1 [RFC3711].



## 2.6. Timing and Reliability Consideration

A system using EKT has the SRTP master keys distributed with EKT, rather than with call signaling. A receiver can immediately decrypt an SRTP (or SRTCP packet) using that new key, provided the SRTP packet (or SRTCP packet) also contains a Full EKT Field.

This section describes how to reliably and expediently deliver new SRTP master keys to receivers.

There are three cases to consider. The first case is a new sender joining a session which needs to communicate its SRTP master key to all the receivers. The second case is a sender changing its SRTP master key which needs to be communicated to all the receivers. The third case is a new receiver joining a session already in progress which needs to know the sender's SRTP master key.

**New sender:** A new sender SHOULD send a packet containing the Full EKT Field as soon as possible, always before or coincident with sending its initial SRTP packet. To accommodate packet loss, it is RECOMMENDED that three consecutive packets contain the Full EKT Field be transmitted. Inclusion of that Full EKT Field can be stopped early if the sender determines all receivers have received the new SRTP master key by receipt of an SRTCP receiver report or explicit ACK for a sequence number with the new key.

**Rekey:** By sending EKT over SRTP, the rekeying event shares fate with the SRTP packets protected with that new SRTP master key. To avoid sending large SRTP packets (such as video key frames) with the Full EKT Field, it can be advantageous to send smaller SRTP packets with the Full EKT Field with the Initial Sequence Number prior to the actual rekey event, but this does eliminate the benefits of fate-sharing EKT with the SRTP packets with the new SRTP master key, which increases the chance a new receiver won't have seen the new SRTP master key.

**New receiver:** When a new receiver joins a session it does not need to communicate its sending SRTP master key (because it is a receiver). When a new receiver joins a session the sender is generally unaware of the receiver joining the session. Thus, senders SHOULD periodically transmit the Full EKT Field. That interval depends on how frequently new receivers join the session, the acceptable delay before those receivers can start processing SRTP packets, and the acceptable overhead of sending the Full EKT Field. The RECOMMENDED frequency is the same as the key frame frequency if sending video or every 5 seconds. When joining a session it is likely that SRTP or SRTCP packets might be received before a packet containing the Full EKT Field is received. Thus, to avoid doubling the authentication

effort, an implementation joining an EKT session SHOULD buffer received SRTP and SRTCP packets until it receives the Full EKT Field packet and use the information in that packet to authenticate and decrypt the received SRTP/SRTCP packets.

### 3. Use of EKT with SDP Security Descriptions

The SDP Security Descriptions (SDESC) [RFC4568] specification defines a generic framework for negotiating security parameters for media streams negotiated via the Session Description Protocol with the "crypto" attribute and the Offer/Answer procedures defined in [RFC3264]. In addition to the general framework, SDESC also defines how to use that framework specifically to negotiate security parameters for Secure RTP. Below, we first provide a brief recap of the crypto attribute when used for SRTP and we then explain how it is complementary to EKT. In the rest of this Section, we provide extensions to the crypto attribute and associated offer/answer procedures to define its use with EKT.

#### 3.1. SDP Security Descriptions Recap

The SRTP crypto attribute defined for SDESC contains a tag followed by three types of parameters (refer to [RFC4568] for details):

- o Crypto-suite. Identifies the encryption and authentication transform.
- o Key parameters. SRTP keying material and parameters.
- o Session parameters. Additional (optional) SRTP parameters such as Key Derivation Rate, Forward Error Correction Order, use of unencrypted SRTP, and other parameters defined by SDESC.

The crypto attributes in the example SDP in Figure 5 illustrate these parameters.

```
v=0
o=sam 2890844526 2890842807 IN IP4 192.0.2.5
s=SRTP Discussion
i=A discussion of Secure RTP
u=http://www.example.com/seminars/srtp.pdf
e=marge@example.com (Marge Simpson)
c=IN IP4 192.0.2.12
t=2873397496 2873404696
m=audio 49170 RTP/SAVP 0
a=crypto:1 AES_CM_128_HMAC_SHA1_80
    inline:WVNfX19zZW1jdGwgKCKgewkyMjA7fQp9CnVubGVz|2^20
    FEC_ORDER=FEC_SRTP
a=crypto:2 F8_128_HMAC_SHA1_80
    inline:MTIzNDU2Nzg5QUJDREUwMTIzNDU2Nzg5QUJjiiKt|2^20
    FEC_ORDER=FEC_SRTP
```

Figure 5: SDP Security Descriptions example

For legibility the SDP shows line breaks that are not present on the wire.

The first crypto attribute has the tag "1" and uses the crypto-suite AES\_CM\_128\_HMAC\_SHA1\_80. The "inline" parameter provides the SRTP master key and salt and the master key lifetime (number of packets). Finally, the FEC\_ORDER session parameter indicates the order of Forward Error Correction used (FEC is applied before SRTP processing by the sender of the SRTP media).

The second crypto attribute has the tag "2", the crypto-suite F8\_128\_HMAC\_SHA1\_80, a SRTP master key, and its associated salt. Finally, the FEC\_ORDER session parameter indicates the order of Forward Error Correction used.

### 3.2. Relationship between EKT and SDESC

SDP Security Descriptions [RFC4568] define a generic framework for negotiating security parameters for media streams negotiated via the Session Description Protocol by use of the Offer/Answer procedures defined in [RFC3264]. In addition to the general framework, SDESC also defines how to use it specifically to negotiate security parameters for Secure RTP.

EKT and SDP Security Descriptions are complementary. SDP Security Descriptions can negotiate several of the SRTP security parameters (e.g., cipher and use of Master Key Identifier) as well as SRTP master keys. SDESC, however, does not negotiate SSRs and their associated Rollover Counter (ROC). Instead, SDESC relies on a so-called "late binding", where a newly observed SSRC will have its

crypto context initialized to a ROC value of zero. Clearly, this does not work for participants joining an SRTP session that has been established for a while and hence has a non-zero ROC. It is impossible to use SDESC to join an SRTP session that is already in progress. In this case, EKT on the endpoint running SDESC can provide the additional signaling necessary to communicate the ROC (Section 6.4.1 of [RFC4568]). The use of EKT solves this problem by communicating the ROC associated with the SSRC in the media plane.

SDP Security Descriptions negotiates different SRTP master keys in the send and receive direction. The offer contains the master key used by the offerer to send media, and the answer contains the master key used by the answerer to send media. Consequently, if media is received by the offerer prior to the answer being received, the offerer does not know the master key being used. Use of SDP security preconditions can solve this problem, however it requires an additional round-trip as well as a more complicated state machine. EKT solves this problem by simply sending the master key used in the media plane thereby avoiding the need for security preconditions.

If multiple crypto-suites were offered, the offerer also will not know which of the crypto-suites offered was selected until the answer is received. EKT solves this problem by using a correlator, the Security Parameter Index (SPI), which uniquely identifies each crypto attribute in the offer.

One of the primary call signaling protocols using offer/answer is the Session Initiation Protocol (SIP) [RFC3261]. SIP uses the INVITE message to initiate a media session and typically includes an offer SDP in the INVITE. An INVITE may be "forked" to multiple recipients which potentially can lead to multiple answers being received. SDESC, however, does not properly support this scenario, mainly because SDP and RTP/RTCP does not contain sufficient information to allow for correlation of an incoming RTP/RTCP packet with a particular answer SDP. Note that extensions providing this correlation do exist (e.g., Interactive Connectivity Establishment (ICE)). SDESC addresses this point-to-multipoint problem by moving each answer to a separate RTP transport address thereby turning a point-to-multipoint scenario into multiple point-to-point scenarios. There are however significant disadvantages to doing so. As long as the crypto attribute in the answer does not contain any declarative parameters that differ from those in the offer, EKT solves this problem by use of the SPI correlator and communication of the answerer's SRTP master key in EKT.

As can be seen from the above, the combination of EKT and SDESC provides a better solution to SRTP negotiation for offer/answer than either of them alone. SDESC negotiates the various SRTP crypto

parameters (which EKT does not), whereas EKT addresses some of the shortcomings of SDESC.

### 3.3. Overview of Combined EKT and SDESC Operation

We define a new session parameter to SDESC to communicate the EKT cipher, EKT key, and Security Parameter Index to the peer. The original SDESC parameters are used as defined in [RFC4568], however the procedures associated with the SRTP master key differ slightly, since both SDESC and EKT communicate an SRTP master key. In particular, the SRTP master key communicated via SDESC is used only if there is currently no crypto context established for the SSRC in question. This will be the case when an entity has received only the offer or answer, but has yet to receive a valid EKT packet from the peer. Once a valid EKT packet is received for the SSRC, the crypto context is initialized accordingly, and the SRTP master key will then be derived from the EKT packet. Subsequent offer/answer exchanges do not change this: The most recent SRTP master key negotiated via EKT will be used, or, if none is available for the SSRC in question, the most recent SRTP master key negotiated via offer/answer will be used. This is done to avoid race conditions between the offer/answer exchange and EKT, even though this breaks some offer/answer rules. Note that with the rules described in this paragraph, once a valid EKT packet has been received for a given SSRC, rekeying for that SSRC can only be done via EKT. The associated SRTP crypto parameters however can be changed via SDESC.

### 3.4. EKT Extensions to SDP Security Descriptions

In order to use EKT and SDESC in conjunction with each other, the new SDESC session parameter "EKT" is defined. It MUST NOT appear more than once in a given crypto attribute. In the Offer/Answer model, the EKT parameter is a negotiated parameter. The "EKT" session parameter consists of three parts (the formal grammar is provided in Section 3.9):

"EKT=" <EKT\_Cipher> "|" <EKT\_Key> "|" <EKT\_SPI>

Below are details on each of these attributes.

**EKT\_Cipher:** The (optional) EKT\_Cipher field defines the EKT cipher used to encrypt the EKT key within SRTP and SRTCP packets. The default value is "AESKW\_128" in accordance with Section 2.3.1. For the AES Key Wrap cipher, the values "AESKW\_128", "AESKW\_192", and "AESKW\_256" are defined for values of L=16, 24, and 32 respectively.

**EKT\_Key:** The (mandatory) EKT\_Key field is the EKT key used to encrypt the SRTP Master Key within SRTP and SRTCP packets. The value is base64 encoded with "=" padding if padding is necessary (see Section 3.2 and 4 of [RFC4648]).

**EKT\_SPI:** The (mandatory) EKT\_SPI field is the Security Parameter Index. It is encoded as an ASCII string representing the hexadecimal value of the Security Parameter Index. The SPI identifies the \*offer\* crypto attribute (including the EKT Key and Cipher) being used for the associated SRTP session. A crypto attribute corresponds to an EKT Parameter Set and hence the SPI effectively identifies a particular EKT parameter set. Note that the scope of the SPI is the SRTP session, which may or may not be limited to the scope of the associated SIP dialog. In particular, if one of the participants in an SRTP session is an SRTP translator, the scope of the SRTP session is not limited to the scope of a single SIP dialog. However, if all of the participants in the session are endpoints or mixers, the scope of the SRTP session will correspond to a single SIP dialog.

### 3.5. Offer/Answer Considerations

In this section, we provide the offer/answer procedures associated with use of the new SDESC session parameter defined in Section 3.4. Since SDESC is defined only for unicast streams, we provide only offer/answer procedures for unicast streams here as well.

#### 3.5.1. Generating the Initial Offer - Unicast Streams

When the initial offer is generated, the offerer MUST follow the steps defined in [RFC4568] Section 7.1.1 as well as the following steps.

[[Editor's Note: following paragraph would benefit from rewording.]]

For each unicast media line using Security Descriptions and where use of EKT is desired, the offerer MUST include the EKT parameter in at least one "crypto" attribute (see [RFC4568]). The EKT parameter MUST contain the EKT\_Key and EKT\_SPI fields. The EKT\_SPI field serves to identify the EKT parameter set used for a particular SRTP or SRTCP packet. Consequently, within a single media line, a given EKT\_SPI value MUST NOT be used with multiple crypto attributes. Note that the EKT parameter set to use for the session is not yet established at this point; each offered crypto attribute contains a candidate EKT parameter set. Furthermore, if the media line refers to an existing SRTP session, then any SPI values used for EKT parameter sets in that session MUST NOT be remapped to any different EKT parameter sets. When an offer describes an SRTP session that is already in progress,

the offer SHOULD use an EKT parameter set (including EKT\_SPI and EKT\_KEY) that is already in use.

As EKT is not defined for use with MKI, a "crypto" attribute containing the EKT parameter MUST NOT contain MKI.

**Important Note:** The scope of the offer/answer exchange is the SIP dialog(s) established as a result of the INVITE, however the scope of EKT is the direct SRTP session, i.e., all the participants that are able to receive SRTP and SRTCP packets directly. If an SRTP session spans multiple SIP dialogs, the EKT parameter sets MUST be synchronized between all the SIP dialogs where SRTP and SRTCP packets can be exchanged. In the case where the SIP entity operates as an RTP mixer (and hence re-originates SRTP and SRTCP packets with its own SSRC), this is not an issue, unless the mixer receives traffic from the various participants on the same destination IP address and port, in which case further coordination of SPI values and crypto parameters may be needed between the SIP dialogs (note that SIP forking with multiple early media senders is an example of this). However, if it operates as a transport translator (relay) then synchronized negotiation of the EKT parameter sets on *\*all\** the involved SIP dialogs will be needed. This is non-trivial in a variety of use cases, and hence use of the combined SDES/EKT mechanism with RTP translators should be considered very carefully. It should be noted, that use of SRTP with RTP translators in general should be considered very carefully as well.

The session parameter "EKT" can either be included as an optional or mandatory parameter.

### 3.5.2. Generating the Initial Answer - Unicast Streams

When the initial answer is generated, the answerer MUST follow the steps defined in [RFC4568] Section 7.1.2 as well as the following steps.

For each unicast media line using SDESC, the answerer examines the associated crypto attribute(s) for the presence of the session parameter "EKT". If a mandatory EKT parameter is included with a "crypto" attribute, the answerer MUST support those parameters in order to accept that offered crypto attribute. If an optional EKT parameter is included instead, the answerer MAY accept the offered crypto attribute without using EKT. However, doing so will prevent the offerer from processing any packets received before the answer. If no EKT parameter are included with a crypto attribute, and that crypto attribute is accepted in the answer, EKT MUST NOT be used. If

a given a crypto attribute includes a malformed EKT parameter, that crypto attribute MUST be considered invalid.

When EKT is used with SDESC, the offerer and answerer MUST use the same SRTP master salt. Thus, the SRTP key parameter(s) in the answer crypto attribute MUST use the same master salt as the one accepted from the offer.

When the answerer accepts the offered media line and EKT is being used, the crypto attribute included in the answer MUST include the same EKT parameter values as found in the accepted crypto attribute from the offerer (however, if the default EKT cipher is being used, it may be omitted). Furthermore, the EKT parameter included MUST be mandatory (i.e., no "-" prefix).

Acceptance of a crypto attribute with an EKT parameter leads to establishment of the EKT parameter set for the corresponding SRTP session. Consequently, the answerer MUST send packets in accordance with that particular EKT parameter set only. If the answerer wants to enable the offerer to process SRTP packets received by the offerer before it receives the answer, the answerer MUST NOT include any declarative session parameters that either were not present in the offered crypto attribute, or were present but with a different value. Otherwise, the offerer's view of the EKT parameter set would differ from the answerer's until the answer is received. Similarly, unless the offerer and answerer has other means for correlating an answer with a particular SRTP session, the answer SHOULD NOT include any declarative session parameters that either were not present in the offered crypto attribute, or were present but with a different value. If this recommendation is not followed and the offerer receives multiple answers (e.g., due to SIP forking), the offerer may not be able to process incoming media stream packets correctly.

### 3.5.3. Processing of the Initial Answer - Unicast Streams

When the offerer receives the answer, it MUST perform the steps in [RFC4568] Section 7.1.3 as well as the following steps for each SRTP media stream it offered with one or more crypto lines containing EKT parameters in it.

[[Editor's Note: following paragraph would benefit from rewording.]]

If the answer crypto line contains an EKT parameter, and the corresponding crypto line from the offer contained the same EKT values, use of EKT has been negotiated successfully and MUST be used for the media stream. When determining whether the values match, an optional and mandatory parameter MUST be considered equal.



Furthermore, if the default EKT cipher is being used, it MAY be either present or absent in the offer and/or answer.

If the answer crypto line does not contain an EKT parameter, then EKT MUST NOT be used for the corresponding SRTP session. Note that if the accepted crypto attribute contained a mandatory EKT parameter in the offer, and the crypto attribute in the answer does not contain an EKT parameter, then negotiation has failed (Section 5.1.3 of [RFC4568]).

If the answer crypto line contains an EKT parameter but the corresponding offered crypto line did not, or if the values don't match or are invalid, then the offerer MUST consider the crypto line invalid (see Section 7.1.3 of [RFC4568] for further operation).

The EKT parameter set is established when the answer is received, however there are a couple of special cases to consider here. First of all, if an SRTP packet containing a Full EKT Field is received prior to the answer, then the EKT parameter set is established provisionally based on the SPI included. Once the answer (which may include declarative session parameters) is received, the EKT parameter set is fully established. The second case involves receipt of multiple answers due to SIP forking. In this case, there will be multiple EKT parameter sets; one for each SRTP session. As mentioned earlier, reliable correlation of SIP dialogs to SRTP sessions requires extensions, and hence if one or more of the answers include declarative session parameters, it may be difficult to fully establish the EKT parameter set for each SRTP session. In the absence of a specific correlation mechanism, it is RECOMMENDED, that such correlation be done based on the signaled receive IP-address in the SDP and the observed source IP-address in incoming SRTP/SRTCP packets, and, if necessary, the signaled receive UDP port and the observed source UDP port.

### 3.6. SRTP-Specific Use Outside Offer/Answer

Security Descriptions use for SRTP is not defined outside offer/answer and hence neither does Security Descriptions with EKT.

### 3.7. Modifying the Session

When a media stream using the SRTP security descriptions has been established, and a new offer/answer exchange is performed, the offerer and answerer MUST follow the steps in Section 7.1.4 of [RFC4568] as well as the following steps. SDESC allows for all parameters of the session to be modified, and the EKT session parameter are no exception to that, however, there are a few additional rules to be adhered to when using EKT.

It is permissible to start a session without the use of EKT, and then subsequently start using EKT, however the converse is not. Thus, once use of EKT has been negotiated on a particular media stream, EKT MUST continue to be used on that media stream in all subsequent offer/answer exchanges.

The reason for this is that both SDESC and EKT communicate the SRTP master key with EKT communicated master keys taking precedence. Reverting back to an SDESC-controlled master key in a synchronized manner is difficult.

Once EKT is being used, the salt for the direct SRTP session MUST NOT be changed. Thus, a new offer/answer which does not create a new SRTP session (e.g., because it reuses the same IP address and port) MUST use the same salt for all crypto attributes as is currently used for the direct SRTP session.

[[Editor's Note: following paragraph would benefit from re-arranging into earlier-described steps.]]

Finally, subsequent offer/answer exchanges MUST NOT remap a given SPI value to a different EKT parameter set until  $2^{15}$  other mappings have been used within the SRTP session. In practice, this requirements is most easily met by using a monotonically increasing SPI value (modulo  $2^{15}$  and starting with zero) per direct SRTP session. Note that a direct SRTP session may span multiple SIP dialogs, and in such cases coordination of SPI values across those SIP dialogs will be required. In the simple point-to-point unicast case without translators, the requirement simply applies within each media line in the SDP. In the point-to-multipoint case, the requirement applies across all the associated SIP dialogs.

### 3.8. Backwards Compatibility Considerations

Backwards compatibility can be achieved in a couple of ways. First of all, Security Descriptions allows for session parameters to be prefixed with "-" to indicate that they are optional. If the answerer does not support the EKT session parameter, such optional parameters will simply be ignored. When the answer is received, absence of the parameter will indicate that EKT is not being used. Receipt of SRTP or SRTCP packets prior to receipt of such an answer will obviously be problematic (as is normally the case for Security Descriptions without EKT).

Alternatively, Security Descriptions allows for multiple crypto lines to be included for a particular media stream. Thus, two crypto lines that differ in their use of EKT parameters (presence in one, absence in the other) can be used as a way to negotiate use of EKT. When the

answer is received, the accepted crypto attribute will indicate whether EKT is being used or not.

### 3.9. Grammar

The ABNF [RFC5234] syntax for the one new SDP Security Descriptions session parameter, EKT, comprising three parts is shown in Figure 6.

```

ekt      = "EKT=" cipher "|" key "|" spi
cipher   = cipher-ext / "AESKW_128" / "AESKW_192" / "AESKW_256"
cipher-ext = 1*64(ALPHA / DIGIT / "_")
key      = 1*(base64) ; See Section 4 of [RFC4648]
base64   = ALPHA / DIGIT / "+" / "/" / "="
spi      = 4HEXDIG ; See [RFC5234]
```

Figure 6: ABNF for the EKT session parameters

Using the example from Figure 6 with the EKT extensions to SDP Security Descriptions results in the following example SDP:

```

v=0
o=sam 2890844526 2890842807 IN IP4 192.0.2.5
s=SRTP Discussion
i=A discussion of Secure RTP
u=http://www.example.com/seminars/srtp.pdf
e=marge@example.com (Marge Simpson)
c=IN IP4 192.0.2.12
t=2873397496 2873404696
m=audio 49170 RTP/SAVP 0
a=crypto:1 AES_CM_128_HMAC_SHA1_80
    inline:WVNfX19zZW1jdGwgKCkgewkyMjA7fQp9CnVubGVz|2^20
    FEC_ORDER=FEC_SRTP EKT=AESKW_128|WWVzQUxvdmVseUVLVGtleQ|AAE0
a=crypto:2 F8_128_HMAC_SHA1_80
    inline:MTIzNDU2Nzg5QUJDREUwMTIzNDU2Nzg5QUJjZGVm|2^20
    FEC_ORDER=FEC_SRTP EKT=AESKW_128|VHdvTG92ZWx5RUtUa2V5cw|AAE1
```

For legibility the SDP shows line breaks that are not present on the wire.

Figure 7: SDP Security Descriptions example with EKT

## 4. Use of EKT with DTLS-SRTP

This document defines an extension to DTLS-SRTP called Key Transport. The EKT with the DTLS-SRTP Key Transport enables secure transport of EKT keying material from one DTLS-SRTP peer to another. This enables those peers to process EKT keying material in SRTP (or SRTCP) and retrieve the embedded SRTP keying material. This combination of

protocols is valuable because it combines the advantages of DTLS (strong authentication of the endpoint and flexibility) with the advantages of EKT (allowing secure multiparty RTP with loose coordination and efficient communication of per-source keys).

#### 4.1. DTLS-SRTP Recap

DTLS-SRTP [RFC5764] uses an extended DTLS exchange between two peers to exchange keying material, algorithms, and parameters for SRTP. The SRTP flow operates over the same transport as the DTLS-SRTP exchange (i.e., the same 5-tuple). DTLS-SRTP combines the performance and encryption flexibility benefits of SRTP with the flexibility and convenience of DTLS-integrated key and association management. DTLS-SRTP can be viewed in two equivalent ways: as a new key management method for SRTP, and a new RTP-specific data format for DTLS.

#### 4.2. EKT Extensions to DTLS-SRTP

This document adds a new TLS negotiated extension called "ekt". This adds a new TLS content type, EKT, and a new negotiated extension EKT. The negotiated extension MUST only be requested in conjunction with the "use\_srtp" extension (Section 3.2 of [RFC5764]). The DTLS server MUST include "dtls-srtp-ekt" in its SDP (as a session or media level attribute) and "ekt" in its TLS ServerHello message. If a DTLS client includes "ekt" in its ClientHello, but does not receive "ekt" in the ServerHello, the DTLS client MUST NOT send DTLS packets with the "ekt" content-type.

The formal description of the dtls-srtp-ekt attribute is defined by the following ABNF [RFC5234] syntax:

```
attribute = "a=dtls-srtp-ekt"
```

Using the syntax described in DTLS [RFC6347], the following structures are used:

```
enum {
    ekt_key(0),
    ekt_key_ack(1),
    ekt_key_error(254),
    (255)
} SRTPKeyTransportType;

struct {
    SRTPKeyTransportType keytrans_type;
    uint24 length;
    uint16 message_seq;
    uint24 fragment_offset;
    uint24 fragment_length;
    select (SRTPKeyTransportType) {
        case ekt_key:
            EKTkey;
    };
} KeyTransport;

enum {
    RESERVED(0),
    AESKW_128(1),
    AESKW_192(2),
    AESKW_256(3),
} ektcipher;

struct {
    ektcipher EKT_Cipher;
    uint EKT_Key_Value<1..256>;
    uint EKT_Master_Salt<1..256>;
    uint16 EKT_SPI;
} EKTkey;
```

Figure 8: Additional TLS Data Structures

The diagram below shows a message flow of DTLS client and DTLS server using the DTLS-SRTP Key Transport extension. SRTP packets exchanged prior to the `ekt_message` are encrypted using the SRTP master key derived from the normal DTLS-SRTP key derivation function. After the `ekt_key` message, they can be encrypted using the SRTP key carried by EKT.

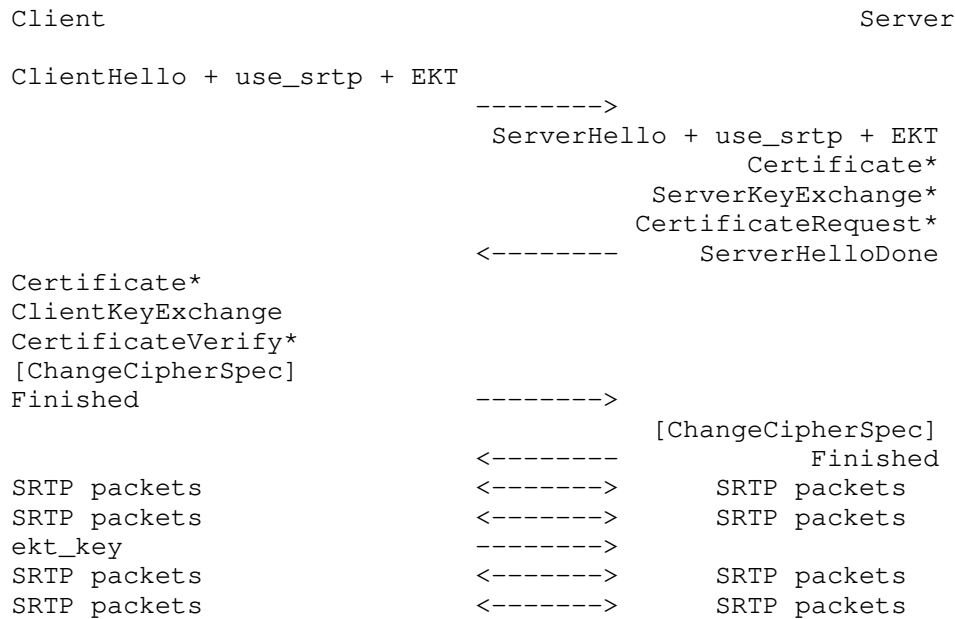


Figure 9: Handshake Message Flow

#### 4.3. Offer/Answer Considerations

This section describes Offer/Answer considerations for the use of EKT together with DTLS-SRTP for unicast and multicast streams. The offerer and answerer MUST follow the procedures specified in [RFC5764] as well as the following ones.

As most DTLS-SRTP processing is performed on the media channel, rather than in SDP, there is little processing performed in SDP other than informational and to redirect DTLS-SRTP to an alternate host. Advertising support for the extension is necessary in SDP because in some cases it is required to establish an SRTP call. For example, a mixer may be able to only support SRTP listeners if those listeners implement DTLS Key Transport (because it lacks the CPU cycles necessary to encrypt SRTP uniquely for each listener).

##### 4.3.1. Generating the Initial Offer

The initial offer contains a new SDP attribute, "dtls-srtp-ekt", which contains no value. This attribute MUST only appear at the media level. This attribute indicates the offerer is capable of supporting DTLS-SRTP with EKT extensions, and indicates the desire to use the "ekt" extension during the DTLS-SRTP handshake.

An example of SDP containing the dtls-srtp-ekt attribute::

```
v=0
o=sam 2890844526 2890842807 IN IP4 192.0.2.5
s=SRTP Discussion
i=A discussion of Secure RTP
u=http://www.example.com/seminars/srtp.pdf
e=marge@example.com (Marge Simpson)
c=IN IP4 192.0.2.12
t=2873397496 2873404696
m=audio 49170 UDP/TLS/RTP/SAVP 0
a=fingerprint:SHA-1
  4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB
a=dtls-srtp-ekt
```

For legibility the SDP shows line breaks that are not present on the wire.

#### 4.3.2. Generating the Initial Answer

Upon receiving the initial offer, the presence of the dtls-srtp-ekt attribute indicates a desire to receive the EKT extension in the DTLS-SRTP handshake. DTLS messages should be constructed according to those two attributes.

If the answerer does not wish to perform EKT, it MUST NOT include a=dtls-srtp-ekt in the SDP answer, and it MUST NOT negotiate EKT during its DTLS-SRTP exchange.

Otherwise, the dtls-srtp-ekt attribute SHOULD be included in the answer, and EKT SHOULD be negotiated in the DTLS-SRTP handshake.

#### 4.3.3. Processing the Initial Answer

The presence of the dtls-srtp-ekt attribute indicates a desire by the answerer to perform DTLS-SRTP with EKT extensions. There are two indications the remote peer does not want to do EKT: the dtls-srtp-ekt attribute is not present in the answer, or the DTLS-SRTP exchange fails to negotiate the EKT extension. If the dtls-srtp-ekt attribute is not present in the answer, the DTLS-SRTP exchange MUST NOT attempt to negotiate the EKT extension. If the dtls-srtp-ekt attribute is present in the answer but the DTLS-SRTP exchange fails to negotiate the EKT extension, EKT MUST NOT be used with that media stream.

After successful DTLS negotiation of the EKT extension, the DTLS client and server MAY exchange SRTP packets, encrypted using the KDF described in [RFC5764]. This is normal and expected, even if Key Transport was negotiated by both sides, as neither side may (yet)

have a need to alter the SRTP key. However, it is also possible that one (or both) peers will immediately send an EKT packet before sending any SRTP, and also possible that SRTP, encrypted with an unknown key, may be received before the EKT packet is received.

#### 4.3.4. Sending DTLS EKT Key Reliably

In the absence of a round trip time estimate, the DTLS `ekt_key` message is sent using an exponential backoff initialized to 250ms, so that if the first message is sent at time 0, the next transmissions are at 250ms, 500ms, 1000ms, and so on. If a recent round trip time estimate is available, exponential backoff is used with the first transmission at 1.5 times the round trip time estimate. In either case, re-transmission stops when `ekt_key_ack` or `ekt_key_error` message is received for the matching `message_seq`.

#### 4.3.5. Modifying the Session

As DTLS-SRTP-EKT processing is done on the DTLS-SRTP channel (media channel) rather than signaling, no special processing for modifying the session is necessary.

If the initial offer and initial answer both contained EKT attributes (indicating the answerer desired to perform EKT), a subsequent offer/answer exchange MUST also contain those same EKT attributes. If not, operation is undefined and the session MAY be terminated. If the initial offer and answer failed to negotiate EKT (that is, the answer did not contain EKT attributes), EKT negotiation failed and a subsequent offer SHOULD NOT include EKT attributes.

### 5. Use of EKT with MIKEY

The advantages outlined in Section 1 are useful in some scenarios in which MIKEY is used to establish SRTP sessions. In this section, we briefly review MIKEY and related work, and discuss these scenarios.

An SRTP sender or a group controller can use MIKEY to establish a SRTP cryptographic context. This capability includes the distribution of a TEK generation key (TGK) or the TEK itself, security policy payload, crypto session bundle ID (CSB\_ID) and a crypto session ID (CS\_ID). The TEK directly maps to an SRTP master key, whereas the TGK is used along with the CSB\_ID and a CS\_ID to generate a TEK. The CS\_ID is used to generate multiple TEKs (SRTP master keys) from a single TGK. For a media stream in SDP, MIKEY allocates two consecutive numbers for the crypto session IDs, so that each direction uses a different SRTP master key (see [RFC4567]).



The MIKEY specification [RFC3830] defines three modes to exchange keys, associated parameters and to protect the MIKEY message: pre-shared key, public-key encryption and Diffie-Hellman key exchange. In the first two modes the MIKEY initiator only chooses and distributes the TKG or TEK, whereas in the third mode both MIKEY entities (the initiator and responder) contribute to the keys. All three MIKEY modes have in common that for establishing a SRTP session the exchanged key is valid for the send and receive direction. Especially for group communications it is desirable to update the SRTP master key individually per direction. EKT provides this property by distributing the SRTP master key within the SRTP/SRTCP packet.

MIKEY already supports synchronization of ROC values between the MIKEY initiator and responder. The SSRC / ROC value pair is part of the MIKEY Common Header payload. This allows providing the current ROC value to late joiners of a session. However, in some scenarios a key management based ROC synchronization is not sufficient. For example, in mobile and wireless environments, members may go in and out of coverage and may miss a sequence number overrun. In point-to-multipoint translator scenarios it is desirable to not require the group controller to track the ROC values of each member, but to provide the ROC value by the originator of the SRTP packet. A better alternative to synchronize the ROC values is to send them directly via SRTP/SRTCP as EKT does. A separate SRTP extension [RFC4771] includes the ROC in a modified authentication tag but that extension does not support updating the SRTP master key.

Besides the ROC, MIKEY synchronizes also the SSRC values of the SRTP streams. Each sender of a stream sends the associated SSRC within the MIKEY message to the other party. If an SRTP session participant starts a new SRTP source (SSRC) or a new participant is added to a group, subsequent SDP offer/answer and MIKEY exchanges are necessary to update the SSRC values. EKT improves these scenarios by updating the keys and SSRC values without coordination on the signaling channel. With EKT, SRTP can handle early media, since the EKT SPI allows the receiver to identify the cryptographic keys and parameters used by the source.

The MIKEY specification [RFC3830] suggests the use of unicast for rekeying. This method does not scale well to large groups or interactive groups. The EKT extension of SRTP/SRTCP provides a solution for rekeying the SRTP master key and for ROC/SSRC synchronization. EKT is not a substitution for MIKEY, but rather a complementary addition to address the above described limitations of MIKEY.

In the next section we provide an extension to MIKEY for support of EKT. EKT can be used only with the pre-shared key or public-key encryption MIKEY mode of [RFC3830]. The Diffie-Hellman exchange mode is not suitable in conjunction with EKT, because it is not possible to establish one common EKT key over multiple EKT entities. Additional MIKEY modes specified in separate documents are not considered for EKT.

### 5.1. EKT Extensions to MIKEY

In order to use EKT with MIKEY, the EKT cipher, EKT key and EKT SPI is negotiated in the MIKEY message exchange.

The following parameters are added to the MIKEY Security Protocol Parameters namespace ([RFC3830], Section 6.10.1). (TBD will be requested from IANA [NOTE TO RFC EDITOR])

Type	Meaning	Possible values
TBD	EKT cipher	see below
TBD	EKT SPI	a 15-bit value

Figure 10: MIKEY Security Protocol Parameters

EKT cipher	Value
(reserved)	0
AESKW_128	1
AESKW_192	2
AESKW_256	3

Figure 11: EKT Cipher Parameters

EKT\_Key is transported in the MIKEY KEMAC payload within one separate Key Data sub-payload. As specified in Section 6.2 of [RFC3830], the KEMAC payload carries the TEK Generation Key (TGK) or the Traffic Encryption Key (TEK). One or more TGKs or TEKs are carried in individual Key Data sub-payloads within the KEMAC payload. The KEMAC payload is encrypted as part of MIKEY. The Key Data sub-payload, specified in Section 6.13 of [RFC3830], has the following format:

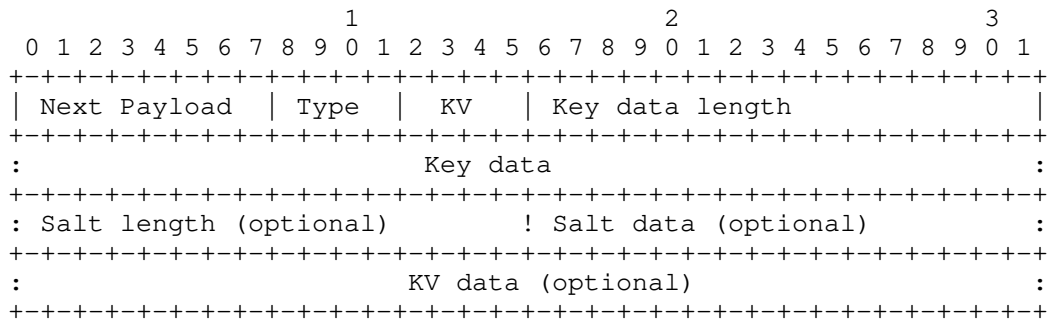


Figure 12: Key Data Sub-Payload of MIKEY

These fields are described below:

**Type:** 4 bits in length, indicates the type of key included in the payload. We define Type = TBD (will be requested from IANA [NOTE TO RFC EDITOR]) to indicate transport of the EKT key.

**KV:** (4 bits): indicates the type of key validity period specified. KV=1 is currently specified as an SPI. We use that value to indicate the KV data contains the EKT\_SPI for the key type EKT\_Key. KV data would be 16 bits in length, but it is also possible to interpret the length from the 'Key data len' field. KV data MUST be present for the key type EKT\_Key when KV=1.

**Salt length, Salt Data:** These optional fields SHOULD be omitted for the key type EKT\_Key, if the SRTP master salt is already present in the TKG or TEK Key Data sub-payload. The EKT\_Key sub-payload MUST contain a SRTP master salt, if the SRTP master salt is not already present in the TKG or TEK Key Data sub-payload.

**KV Data:** length determined by Key Data Length field.

## 5.2. Offer/Answer Considerations

This section describes Offer/Answer considerations for the use of EKT together with MIKEY for unicast streams. The offerer and answerer MUST follow the procedures specified in [RFC3830] and [RFC4567] as well as the following ones.

### 5.2.1. Generating the Initial Offer

If it is intended to use MIKEY together with EKT, the offerer MUST include at least one MIKEY key-mgmt attribute with one EKT\_Key Key Data sub-payload and the SRTP Security Policy payload (SP) with the policy parameter EKT SPI. The policy parameter EKT Cipher is

OPTIONAL, The default value is "AESKW\_128" in accordance with Section 2.3.1. MIKEY can be used on session or media level. On session level, MIKEY provides the keys for multiple SRTP sessions in the SDP offer. The EKT SPI references a EKT parameter set including the Secure RTP parameters as specified in Section 8.2 in [RFC3711]. If MIKEY is used on session level, it is only possible to use one EKT SPI value. Therefore, the session-level MIKEY message MUST contain one SRTP Security Policy payload only, which is valid for all related SRTP media lines. If MIKEY is used on media level, different SRTP Security Policy parameters (and consequently different EKT SPI values) can be used for each media line. If MIKEY is used on session and media level, the media level content overrides the session level content.

EKT requires a single shared SRTP master salt between all participants in the direct SRTP session. If a MIKEY key-mgmt attribute contains more than one TGK or TEK Key Data sub-payload, all the sub-payloads MUST contain the same master salt value. Consequently, the EKT\_Key Key Data sub-payload MAY also contain the same salt or MAY omit the salt value. If the SRTP master salt is not present in the TGK and TEK Key Data sub-payloads, the EKT\_Key sub-payload MUST contain a master salt.

#### 5.2.2. Generating the Initial Answer

For each media line in the offer using MIKEY, provided on session and/or on media level, the answerer examines the related MIKEY key-mgmt attributes for the presence of EKT parameters. In order to accept the offered key-mgmt attribute, the MIKEY message MUST contain one EKT\_Key Key Data sub-payload and the SRTP Security Policy payload with policy parameter EKT SPI. The answerer examines also the existence of a SRTP master salt in the TGK/TEK and/or the EKT\_Key sub-payloads. If multiple salts are available, all values MUST be equal. If the salt values differ or no salt is present, the key-mgmt attribute MUST be considered as invalid.

The MIKEY responder message in the SDP answer does not contain a MIKEY KEMAC or Security Policy payload and consequently does not contain any EKT parameters. If a key-mgmt attribute for a media line was accepted by the answerer, the EKT parameter set of the offerer is valid for both directions of the SRTP session.

#### 5.2.3. Processing the Initial Answer

On reception of the answer, the offerer examines if EKT has been accepted for the offered media lines. If a MIKEY key-mgmt attribute is received containing a valid MIKEY responder message, EKT has been successfully negotiated. On receipt of a MIKEY error message, EKT

negotiation has failed. For example, this may happen if an EKT extended MIKEY initiator message is sent to a MIKEY entity not supporting EKT. A MIKEY error code 'Invalid SPpar' or 'Invalid DT' is returned to indicate that the EKT parameters (EKT Cipher and EKT SPI) in the SRTP Security Policy payload or the EKT\_Key sub-payload is not supported. In this case, the offerer may send a second SDP offer with a MIKEY key-mgmt attribute without the additional EKT extensions.

This behavior can be improved by offering two key-mgmt SDP attributes. One attribute offers MIKEY with SRTP and EKT and the other attribute offers MIKEY with SRTP without EKT.

#### 5.2.4. Modifying the Session

Once an SRTP stream has been established, a new offer/answer exchange can modify the session including the EKT parameters. If the EKT key or EKT cipher is modified (i.e., a new EKT parameter set is created) the offerer MUST also provide a new EKT SPI value. The offerer MUST NOT remap an existing EKT SPI value to a new EKT parameter set. Similar, a modification of the SRTP Security Policy leads to a new EKT parameter set and requires a fresh EKT SPI, even if the EKT key or cipher did not change.

Once EKT is being used, the SRTP master salt for the SRTP session MUST NOT be changed. The salt in the Key Data sub-payloads within the subsequent offers MUST be the same as the one already used.

After EKT has been successfully negotiated for a session and an SRTP master key has been transported by EKT, it is difficult to switch back to a pure MIKEY based key exchange in a synchronized way. Therefore, once EKT is being used for a session, EKT MUST be used also in all subsequent offer/answer exchanges for that session.

### 6. Using EKT for Interoperability between Key Management Systems

A media gateway (MGW) can provide interoperability between an SRTP-EKT endpoint and a non-EKT SRTP endpoint. When doing this function, the MGW can perform non-cryptographic transformations on SRTP packets outlined above. However, there are some uses of cryptography that will be required for that gateway. If a new SRTP master key is communicated to the MGW (via EKT from the EKT leg, or via Security Descriptions without EKT from the Security Descriptions leg), the MGW needs to convert that information for the other leg, and that process will incur some cryptographic operations. Specifically, if the new key arrived via EKT, the key must be decrypted and then sent in Security Descriptions (e.g., as a SIP re-INVITE); likewise, if a new

key arrives via Security Descriptions that must be encrypted via EKT and sent in SRTP/SRTCP.

Additional non-normative information can be found in Appendix A.

## 7. Design Rationale

From [RFC3550], a primary function of RTCP is to carry the CNAME, a "persistent transport-level identifier for an RTP source" since "receivers require the CNAME to keep track of each participant." EKT works in much the same way but uses SRTP to carry information needed for the proper processing of the SRTP traffic.

With EKT, SRTP gains the ability to synchronize the creation of cryptographic contexts across all of the participants in a single session. This feature provides some, but not all, of the functionality that is present in IKE phase two (but not phase one). Importantly, EKT does not provide a way to indicate SRTP options.

With EKT, external signaling mechanisms provide the SRTP options and the EKT Key, but need not provide the key(s) for each individual SRTP source. EKT provides a separation between the signaling mechanisms and the details of SRTP. The signaling system need not coordinate all SRTP streams, nor predict in advance how many sources will be present, nor communicate SRTP-level information (e.g., rollover counters) of current sessions.

EKT is especially useful for multi-party sessions, and for the case where multiple RTP sessions are sent to the same destination transport address (see the example in the definition of "RTP session" in [RFC3550]). A SIP offer that is forked in parallel (sent to multiple endpoints at the same time) can cause multiple RTP sessions to be sent to the same transport address, making EKT useful for use with SIP.

EKT can also be used in conjunction with a scalable group-key management system like GDOI [RFC6407]. In such a combination GDOI would provide a secure entity authentication method for group members, and a scalable way to revoke group membership; by itself, EKT does not attempt to provide either capability.

EKT carries the encrypted key in a new SRTP field (at the end of the SRTP packet). This maintains compatibility with the existing SRTP specification by defining a new crypto function that incorporates the encrypted key, and a new authentication transform to provide implicit authentication of the encrypted key.

The main motivation for the use of the variable-length EKT format is bandwidth conservation. When EKT is sent over SRTP, there will be a loss of (usable) bandwidth due to the additional EKT bytes in each RTP packet. For some applications, this bandwidth loss is significant.

### 7.1. Alternatives

In its current design, EKT requires that the Master Salt be established out of band. That requirement is undesirable. In an offer/answer environment, it forces the answerer to re-use the same Master Salt value used by the offerer. The Master Salt value could be carried in EKT packets though that would consume yet more bandwidth.

In some scenarios, two SRTP sessions may be combined into a single session. When using EKT in such sessions, it is desirable to have an SPI value that is larger than 15 bits, so that collisions between SPI values in use in the two different sessions are unlikely (since each collision would confuse the members of one of the sessions).

An alternative that addresses both of these needs is as follows: the SPI value can be lengthened from 15 bits to 63 bits, and the Master Salt can be identical to, or constructed from, the SPI value. SRTP conventionally uses a 14-byte Master Salt, but shorter values are acceptable. This alternative would add six bytes to each EKT packet; that overhead may be a reasonable tradeoff for addressing the problems outlined above. This is considered too high a bandwidth penalty.

## 8. Security Considerations

EKT inherits the security properties of the SRTP keying it uses: Security Descriptions, DTLS-SRTP, or MIKEY.

With EKT, each SRTP sender and receiver MUST generate distinct SRTP master keys. This property avoids any security concern over the re-use of keys, by empowering the SRTP layer to create keys on demand. Note that the inputs of EKT are the same as for SRTP with key-sharing: a single key is provided to protect an entire SRTP session. However, EKT remains secure even in the absence of out-of-band coordination of SSRCs, and even when SSRC values collide.

The EKT Cipher includes its own authentication/integrity check. For an attacker to successfully forge a full EKT packet, it would need to defeat the authentication mechanisms of both the EKT Cipher and the SRTP authentication mechanism.

The presence of the SSRC in the EKT\_Plaintext ensures that an attacker cannot substitute an EKT\_Ciphertext from one SRTP stream into another SRTP stream.

An attacker who strips a Full\_EKT\_Field from an SRTP packet may prevent the intended receiver of that packet from being able to decrypt it. This is a minor denial of service vulnerability. Similarly, an attacker who adds a Full\_EKT\_Field can disrupt service.

An attacker could send packets containing either Short EKT Field or Full EKT Field, in an attempt to consume additional CPU resources of the receiving system. In the case of the Short EKT Field, this field is stripped and normal SRTP or SRTCP processing is performed. In the case of the Full EKT Field, the attacker would have to have guessed or otherwise determined the SPI being used by the receiving system. If an invalid SPI is provided by the attacker, processing stops. If a valid SPI is provided by the attacker, the receiving system will decrypt the EKT ciphertext and return an authentication failure (Step 3 of Section 2.2.2).

EKT can rekey an SRTP stream until the SRTP rollover counter (ROC) needs to roll over. EKT does not extend SRTP's rollover counter (ROC), and like SRTP itself EKT cannot properly handle a ROC rollover. Thus even if using EKT, new (master or session) keys need to be established after  $2^{48}$  packets are transmitted in a single SRTP stream as described in Section 3.3.1 of [RFC3711]. Due to the relatively low packet rates of typical RTP sessions, this is not expected to be a burden.

The confidentiality, integrity, and authentication of the EKT cipher MUST be at least as strong as the SRTP cipher.

Part of the EKT\_Plaintext is known, or easily guessable to an attacker. Thus, the EKT Cipher MUST resist known plaintext attacks. In practice, this requirement does not impose any restrictions on our choices, since the ciphers in use provide high security even when much plaintext is known.

An EKT cipher MUST resist attacks in which both ciphertexts and plaintexts can be adaptively chosen. An EKT cipher MUST resist attacks in which both ciphertexts and plaintexts can be adaptively chosen and adversaries that can query both the encryption and decryption functions adaptively.



## 9. IANA Considerations

IANA is requested to register EKT from Section 3.9 into the Session Description Protocol (SDP) Security Descriptions [iana-sdp-sdesc] registry for "SRTP Session Parameters".

IANA is requested to register the following new attributes into the SDP Attributes registry [iana-sdp-attr].

Attribute name: dtls-srtp-ekt

Long form name: DTLS-SRTP with EKT

Type of attribute: Media-level

Subject to charset: No

Purpose: Indicates support for DTLS-SRTP with EKT

Appropriate values: No values

Contact name: Dan Wing, dwing@cisco.com

We request the following IANA assignments from the existing [iana-mikey] name spaces in the IETF consensus range (0-240) [RFC3830]:

- o From the Key Data payload name spaces, a value to indicate the type as the 'EKT\_Key'.

Furthermore, we need the following two new IANA registries created, populated with the initial values in this document. New values for both of these registries can be defined via Specification Required [RFC5226].

- o EKT parameter type, initially populated with the list from Figure 10
- o EKT cipher, initially populated with the list from Figure 11

## 10. Acknowledgements

Thanks to Lakshminath Dondeti for assistance with earlier versions of this document. Thanks to Kai Fischer for writing the MIKEY section.

Thanks to Nermeen Ismail, Eddy Lem, Rob Raymond, and Yi Cheng for fruitful discussions and comments. Thanks to Felix Wyss for his review and comments regarding ciphers. Thanks to Michael Peck for

his review. Thanks to Magnus Westerlund for his review. Thanks to Michael Peck and Jonathan Lennox for their review comments.

## 11. References

### 11.1. Normative References

- [FIPS197] National Institute of Standards and Technology (NIST), "The Advanced Encryption Standard (AES)", FIPS-197 Federal Information Processing Standard, November 2001.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.
- [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.
- [RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.
- [RFC4563] Carrara, E., Lehtovirta, V., and K. Norrman, "The Key ID Information Type for the General Extension Payload in Multimedia Internet KEYing (MIKEY)", RFC 4563, June 2006.
- [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] Andreasen, F., Baugher, M., and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, July 2006.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [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.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

## 11.2. Informative References

- [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.
- [RFC3830] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and K. Norrman, "MIKEY: Multimedia Internet KEYing", RFC 3830, August 2004.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.
- [RFC4771] Lehtovirta, V., Naslund, M., and K. Norrman, "Integrity Transform Carrying Roll-Over Counter for the Secure Real-time Transport Protocol (SRTP)", RFC 4771, January 2007.
- [RFC5649] Housley, R. and M. Dworkin, "Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm", RFC 5649, September 2009.
- [RFC6407] Weis, B., Rowles, S., and T. Hardjono, "The Group Domain of Interpretation", RFC 6407, October 2011.
- [iana-mikey] IANA, , "Multimedia Internet KEYing (Mikey) Payload Name Spaces", 2011, <<http://www.iana.org/assignments/mikey-payloads/mikey-payloads.xhtml>>.
- [iana-sdp-attr] IANA, , "SDP Parameters", 2011, <<http://www.iana.org/assignments/sdp-parameters/sdp-parameters.xml>>.

```
[iana-sdp-sdesc]
    IANA, , "Session Description Protocol (SDP) Security
    Descriptions: SRTP Session Parameters", 2011,
    <http://www.iana.org/assignments/sdp-security-
    descriptions/sdp-security-descriptions.xml#sdp-security-
    descriptions-4>.
```

#### Appendix A. Using EKT to Optimize Interworking DTLS-SRTP with Security Descriptions

Today, SDP Security Descriptions [RFC4568] is used for distributing SRTP keys in several different IP PBX systems. The IP PBX systems are typically used within a single enterprise. A Session Border Controller is a reasonable solution to interwork between Security Descriptions in one network and DTLS-SRTP in another network. For example, a mobile operator (or an Enterprise) could operate Security Descriptions within their network and DTLS-SRTP towards the Internet.

However, due to the way Security Descriptions and DTLS-SRTP manage their SRTP keys, such an SBC has to authenticate, decrypt, re-encrypt, and re-authenticate the SRTP (and SRTCP) packets in one direction, as shown in Figure 13, below. This is computationally expensive.

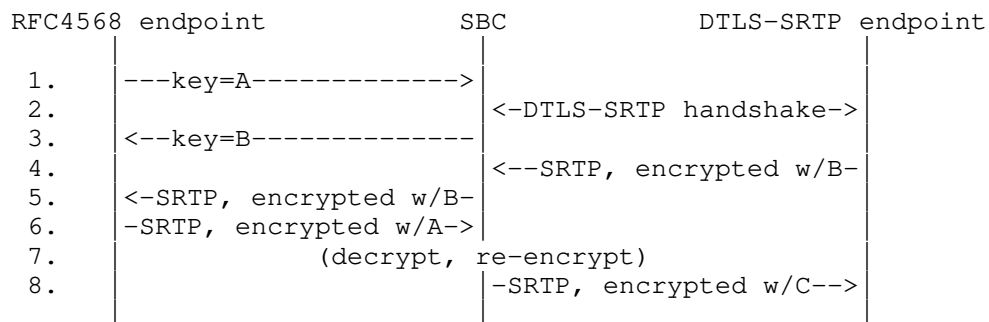


Figure 13: Interworking Security Descriptions and DTLS-SRTP

The message flow is as follows (similar steps occur with SRTCP):

1. The Security Descriptions [RFC4568] endpoint discloses its SRTP key to the SBC, using a=crypto in its SDP.
2. SBC completes DTLS-SRTP handshake. From this handshake, the SBC derives the SRTP key for traffic from the DTLS-SRTP endpoint (key B) and to the DTLS-SRTP endpoint (key C).

3. The SBC communicates the SRTP encryption key (key B) to the Security Descriptions endpoint (using a=crypto). (There is no way, with DTLS-SRTP, to communicate the Security Descriptions key to the DTLS-SRTP key endpoint.)
4. The DTLS-SRTP endpoint sends an SRTP key, encrypted with its key B. This is received by the SBC.
5. The received SRTP packet is simply forwarded; the SBC does not need to do anything with this packet as its key (key B) was already communicated in step 3.
6. The Security Descriptions endpoint sends an SRTP packet, encrypted with its key A.
7. The SBC has to authenticate and decrypt the SRTP packet (using key A), and re-encrypt it and generate an HMAC (using key C).
8. The SBC sends the new SRTP packet.

If EKT is deployed on the DTLS-SRTP endpoints, EKT helps to avoid the computationally expensive operation so the SBC does not need to perform any per-packet operations on the SRTP (or SRTCP) packets in either direction. With EKT the SBC can simply forward the SRTP (and SRTCP) packets in both directions without per-packet HMAC or cryptographic operations.

To accomplish this interworking, DTLS-SRTP EKT must be supported on the DTLS-SRTP endpoint, which allows the SBC to transport the Security Description key to the EKT endpoint and send the DTLS-SRTP key to the Security Descriptions endpoint. This works equally well for both incoming and outgoing calls. An abbreviated message flow is shown in Figure 14, below.

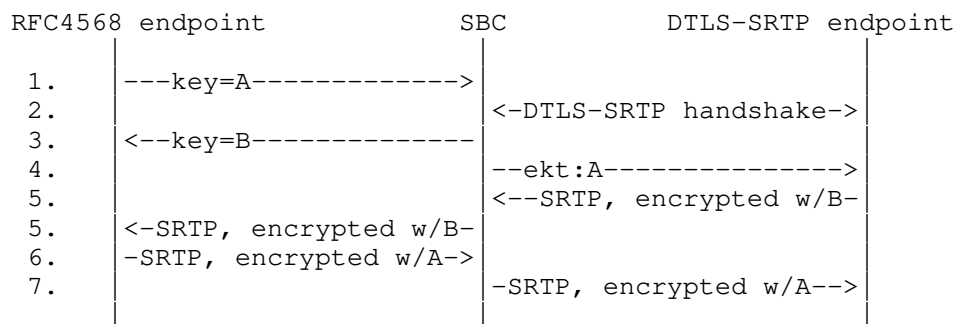


Figure 14: Interworking Security Descriptions and EKT

The message flow is as follows (similar steps occur with SRTCP):

1. Security Descriptions endpoint discloses its SRTP key to the SBC (a=crypto).
2. SBC completes DTLS-SRTP handshake. From this handshake, the SBC derives the SRTP key for traffic from the DTLS-SRTP endpoint (key B) and to the DTLS-SRTP endpoint (key C).
3. The SBC communicates the SRTP encryption key (key B) to the Security Descriptions endpoint.
4. The SBC sends an EKT packet indicating that SRTP will be encrypted with 'key A' towards the DTLS-SRTP endpoint.
5. The DTLS-SRTP endpoint sends an SRTP key, encrypted with its key B. This is received by the SBC.
6. The received SRTP packet is simply forwarded; the SBC does not need to do anything with this packet as its key (key B) was communicated in step 3.
7. The Security Descriptions endpoint sends an SRTP packet, encrypted with its key A.
8. The received SRTP packet is simply forwarded; the SBC does not need to do anything with this packet as its key (key A) was communicated in step 4.

#### Authors' Addresses

John Mattsson (editor)  
Ericsson AB  
SE-164 80 Stockholm  
Sweden

Phone: +46 10 71 43 501  
Email: john.mattsson@ericsson.com

David A. McGrew  
Cisco Systems, Inc.  
510 McCarthy Blvd.  
Milpitas, CA 95035  
US

Phone: (408) 525 8651  
Email: [mcgrew@cisco.com](mailto:mcgrew@cisco.com)  
URI: <http://www.mindspring.com/~dmcgrew/dam.htm>

Dan Wing  
Cisco Systems, Inc.  
510 McCarthy Blvd.  
Milpitas, CA 95035  
US

Phone: (408) 853 4197  
Email: [dwing@cisco.com](mailto:dwing@cisco.com)

Flemming Andreason  
Cisco Systems, Inc.  
499 Thornall Street  
Edison, NJ 08837  
US

Email: [fandreas@cisco.com](mailto:fandreas@cisco.com)

Network Working Group  
Internet Draft  
Intended status: Informational  
Expires: April 27, 2015

P. Jones (Ed.)  
N. Ismail  
D. Benham  
N. Buckles  
Cisco Systems  
J. Mattsson  
Y. Cheng  
Ericsson  
R. Barnes  
Mozilla  
October 27, 2014

Requirements for Private Media in a Switched Conferencing Environment  
draft-jones-avtcore-private-media-reqts-00

Abstract

This document specifies the requirements for ensuring the privacy and integrity of real-time media flows between two or more endpoints communicating in a switched conferencing environment. This document also provides a high-level overview of switched conferencing in order to establish a common understanding of the goals and objectives of this work.

Status of this Memo

This Internet-Draft is submitted to IETF 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, 2015.

Copyright Notice

Copyright (c) 2014 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.....	2
2. Requirements Language.....	3
3. Terminology.....	3
4. Background.....	4
5. Motivation for Private Media in Switched Conferencing.....	5
5.1. Switched Conferencing in Cloud Services.....	5
5.2. Private Media Security through Switching.....	7
6. Goals and Non-Goals.....	8
6.1. Goals.....	8
6.1.1. Ensure End-To-End Confidentiality.....	8
6.1.2. Ensure End-To-End Source Authentication of Media.....	9
6.1.3. Provide a More Efficient Service than "Full-Mesh".....	9
6.1.4. Support Cloud-Based Conferencing.....	9
6.1.5. Limiting a User's Access to Content.....	9
6.1.6. Compatibility with the WebRTC Security Architecture.....	10
6.2. Non-Goals.....	10
6.2.1. Securing the Endpoints.....	10
6.2.2. Concealing that Communication Occurs.....	10
6.2.3. Individual Media Source Authentication.....	11
6.2.4. Support for Multicast in Switched Conferencing.....	11
7. Requirements.....	11
8. IANA Considerations.....	12
9. Security Considerations.....	12
10. References.....	12
10.1. Normative References.....	12
10.2. Informative References.....	13
11. Acknowledgments.....	13
12. Contributors.....	13
Authors' Addresses.....	14

## 1. Introduction

Users of multimedia communication products and services have privacy expectations that are largely satisfied with the use of SRTP [RFC3711] and related technologies when communicating point-to-point over the Internet. When communicating in a conferencing environment with two or more participants, though, it is necessary for an endpoint to share the SRTP master key and salt with the conference server so that it can authenticate and decrypt received RTP and RTCP packets. The conference server also needs the master key and salt in order to transmit media packets it receives to other participants in

the conference. The need for conferencing servers to have the master key is a security risk for users.

Within a corporate or other isolated environment where conferencing servers are tightly controlled, this security risk can be effectively managed. However, managing this risk is becoming increasingly difficult as conferencing resources are being deployed in networks that are less than fully trusted, including virtualized conferencing servers deployed in cloud environments.

There are also public voice and video conferencing service providers in which users must place full trust in order to use those services, as it is necessary for an endpoint to share the SRTP master key with those conferencing servers. This exposes corporations, for example, to a higher risk of being subjected to corporate espionage. While it is not the intent of this draft to suggest that any existing service provider would permit or condone any illicit use of its service, the fact is that security threats can come from external sources and remain undiscovered for long periods of time.

It is possible to ensure communication privacy within the context of a switched conferencing environment with limited changes in the security mechanisms used today. This document discusses this possibility in more detail and presents a set of requirements for meeting this objective.

## 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] when they appear in ALL CAPS. These words may also appear in this document in lower case as plain English words, absent their normative meanings.

## 3. Terminology

[Editor's Note: we may want to refine these or add/remove terms]

**Adversary** - An unauthorized entity that may attempt to compromise the performance of a conference server through various means, including, but not limited to, the transmission of bogus media packets or attempt to gain access to the plaintext of the media.

**Switching conference server** - A conference server that does not decrypt RTP media flows or perform processing on the media payload, but instead simply forwards the received media from a sender to the other participants in a multimedia conference. A switching conference server may modify some RTP headers.

#### 4. Background

Traditional multimedia conferencing servers would mix, transcode, transrate, and/or recompose media flows from one or more conference participants, sending out a different audio and video flow to each participant. For audio, this might entail mixing some number of input flows that appear to contain audio intended to be heard by the other participants, with each participant receiving a flow that does not contain that participant's own audio. For video, the conference server may elect to send only video showing the current active speaker, a tiled composition of all participants or the most recent active speakers, a video flow with the active speaker presented prominently with other participants presented as thumbnail images, or some other composite arrangement. It is also common for audio or video to be transcoded. A typical traditional conferencing server is depicted in Figure 1.

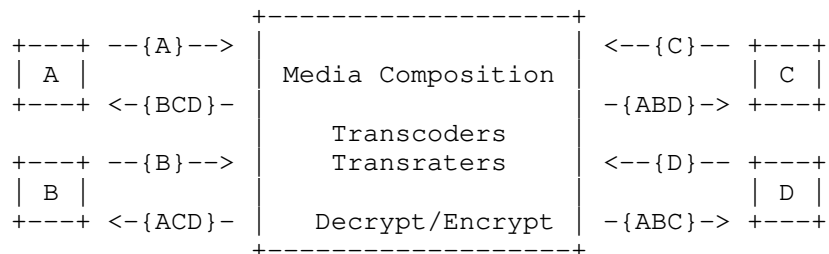


Figure 1 – Traditional Conferencing Server

Traditional conference servers require a significant amount of processing power, which in turn translates into a high cost for conferencing hardware manufacturers. Significantly, too, it is very difficult to deploy these servers in a cloud environment due to the high processing demands, as the specialized hardware found in the traditional voice and video conferencing server does not exist in a cloud environment.

To enable the traditional conferencing server to perform its job, the server establishes an SRTP session with each of the conference participants so that it can get the keys required to decrypt and encrypt media flows from and to each participant. This means that the conference server is necessarily a fully trusted entity in the communication path. Anytime these servers are deployed in a network that is not tightly controlled, it increases the risk that an attacker might gain access to cryptographic key material, thus allowing the attacker to be able to see and listen to ongoing conferences. In some instances, depending on how the hardware is designed and how keys and certificates are managed, it might be possible for an attacker to see and listen to previously recorded conferences or future conferences.

The Secure Real-time Transport Protocol (SRTP) [RFC3711] is a profile of RTP, which can provide confidentiality, message authentication, and replay protection to the RTP traffic and to the RTP Control Protocol (RTCP). Encryption of header extension in SRTP [RFC6904] provides a mechanism extending the mechanisms of [RFC3711], to selectively encrypt RTP header extensions in SRTP. [RFC3711] and [RFC6904] solves end-to-end use cases between two endpoints, and does not consider use cases where a sender delivers media to a receiver via a cloud-based conferencing service.

## 5. Motivation for Private Media in Switched Conferencing

### 5.1. Switched Conferencing in Cloud Services

There is a trend in the industry for enterprises to use cloud services to host multi-party conferences and meet-me services, either exclusively or to meet peak loads on-demand. At the same time, there is huge shift toward using light-weight, cost-effective switching conference servers in cloud services that do not necessarily need to mix audio or composite/transcode video. Also fueling the use of such light-weight conference servers is the desire to fully exploit virtualized computing resources and dynamic scalability potential available in cloud computing environments.

The increased use of cloud services has exposed a problem. There are two different trust domains from a media perspective: endpoints and other devices in a trusted domain, and conference servers controlled by the cloud service in an untrusted domain. Other examples of conference devices spread across trusted and untrusted domains are likely, but the cloud service trend is triggering the urgency to address the need to allow for lightweight media conference while enabling media privacy at the same time.

With a switching conference server, each participant transmits media to the server as it would with a traditional conferencing server. However, the switching conference server merely forwards media to the other participants in the conference (where the other participant may be associated with a cascaded conference server or an endpoint on the same server), leaving composition to the receiving endpoint. Since some endpoints may have a limited amount of bandwidth, each endpoint might negotiate with the switching conference server to receive only a subset of the available media flows. Each transmitting endpoint might also send multiple media flows of varying frame sizes and/or frame rates (e.g., simulcast or scalability layers), so that the server can select the streams most appropriate for each receiver's bandwidth and capabilities. This allows, for example, an endpoint to receive and display higher quality video for the active speaker and thumbnails for other participants. It is also worth noting that, for switched media to work successfully, each endpoint in the conference must support the media formats transmitted by all other entities in

the conference. More modern endpoints support multiple codecs and formats, making this commercially practical.

Figure 2 depicts an example of a switching conference server wherein each participant is receiving the media flows transmitted by each of the other participants in the conference.

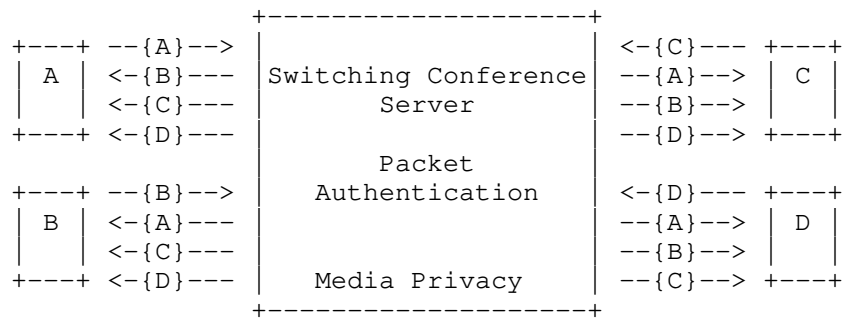


Figure 2 - Switching Conference Server

Note - The use of multiple arrows directed toward each endpoint is not intended to suggest the use of separate RTP sessions.

By using methods such as those described in [RFC6464], it is possible for the switching conference server to transmit the appropriate audio and video flows to conference participants without having knowledge of the contents of the encrypted media. The examples that follow help to illustrate this point.

In the Figure 3 below, endpoints A, B and D receive the video streams from endpoint C, the currently active speaker, which is receiving video from endpoint A, the previous active speaker. Later when endpoint B becomes the active speaker (Figure 4), endpoints A, C and D will start to receive video from B, while endpoint B continues to receive video from endpoint C. Finally in Figure 5, endpoint A becomes the active speaker.

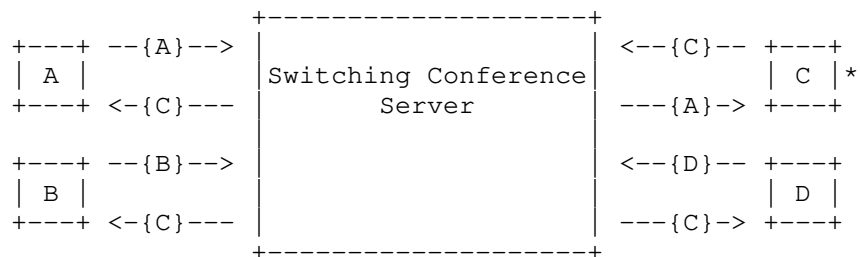


Figure 3 - Endpoint "C" is the Active Speaker

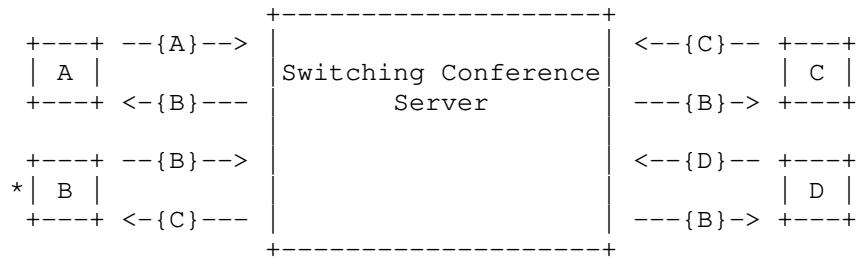


Figure 4 - Endpoint "B" is the Active Speaker

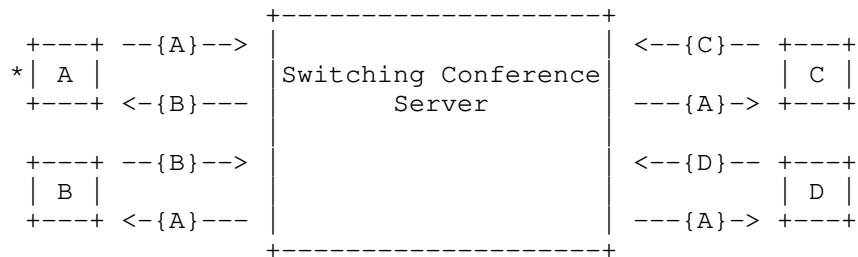


Figure 5 - Endpoint "A" is the Active Speaker

Switched conferencing can also enable conferences to scale to include many more simultaneous participants than would be possible with a traditional conferencing server. Like traditional conferencing servers, switching conference servers can also be cascaded or interconnected in a meshed topology to increase the size of the conference without putting undue burden on any particular server.

## 5.2. Private Media Security through Switching

A traditional conferencing server, or MCU, establishes an SRTP session with each participating endpoint separately, and needs to decrypt packets containing media presented to other endpoints. By using a switching conference server, it is possible to keep the media encryption keys private to the endpoints such that the conference server does not have access to the keys used for media encryption. The switching conference server just forwards media received to each of the other participants in the conference.

This provides for a significantly improved security model, as one can, for example, utilize conferencing resources in the cloud that do not necessarily have to be trusted. That said, there may be situations where the switching conference server needs to modify the RTP packet received from an endpoint, such as by adding or removing an RTP header extension, modifying the payload type value, etc. It would be the responsibility of the switching conference server to ensure that media of the expected type and containing the correct information is received by a recipient.

Thus, there is a need to utilize an end-to-end encryption and authentication key (or pair of keys) and a hop-by-hop encryption and authentication key (or pair of keys). The purpose for the hop-by-hop encryption key is to optionally encrypt RTP header extensions. The current SRTP specification and related specifications do not define use of a dual-key approach presently. However, such an approach is possible and would result in ensuring the privacy of media while also enabling the more scalable switched conferencing model.

The assumptions with this model are that the endpoints are trusted entities, as they clearly have access to the media keys for encryption. Some call processing functions for the administrative domain, such as SIP [RFC3261] proxy servers or B2BUAs, are trusted in exactly the same way they are with the traditional conferencing model, meaning they must be trusted to keep signaling secure as certificate information (e.g., fingerprints) might be conveyed via signaling. The switching conference server is not fully trusted and is not given visibility into the actual contents of the SRTP payload. However, the switching conference server in the untrusted domain is at least trusted to perform its core duties of forwarding media and processing signaling; it simply isn't trusted with the media encryption keys.

The assumption is that no changes are made to SRTCP, i.e. SRTCP is protected hop-by-hop with a single security context.

This dual-key model does necessitate a change in the way that keys are managed. However, the topic of key management is outside the scope of this requirements document. However, high-level assumptions like if the end-to-end contexts use a group key as SRTP master key or if individual SRTP master keys (that may be derived/negotiated from another group key) is likely to influence the solution derived from this document.

## 6. Goals and Non-Goals

### 6.1. Goals

#### 6.1.1. Ensure End-To-End Confidentiality

The content of the communication and all media needs to be confidential within the group of entities explicitly invited into the conference. An external monitoring adversary should not be able to deduce the human-to-human communication that actually occurred from capturing the media packets.

At the same time, it is necessary to allow switching media servers to manipulate certain RTP header fields like the payload type value.

#### 6.1.2. Ensure End-To-End Source Authentication of Media

In a conference system with multiple participants it is vital that the multimedia content presented to any of the human participants is from the stated participant, and not an adversary that attempts to inject misleading content. Nor should an adversary be able to fool the system into becoming a trusted party in the conference. Only explicitly invited parties shall be able to contribute content.

#### 6.1.3. Provide a More Efficient Service than "Full-Mesh"

A multi-party conference that has the goals of confidentiality and source authentication can be established as a "full mesh" (i.e., each participating endpoint directly addresses each of the other participants). However, this has a significant issue with the amount of consumed resources in both the uplink and the downlink from each participant.

A switched conferencing model would yield the efficiencies desired.

#### 6.1.4. Support Cloud-Based Conferencing

To achieve cost-effective and scalable conferencing, it must be possible to run the conference node instances in a cloud-based virtualized environment.

From a security standpoint, this is a significant issue since the virtualized server instance and the underlying hardware and software upon which it runs might not be secure from an adversary.

#### 6.1.5. Limiting a User's Access to Content

Since an invited user will be provided with the content protection keys, the user can decrypt content from time periods before and after the user joined the conference. However, this is not always desirable. It should be possible to re-key the content protection keys every time a user joins or leaves the conference so each particular set of conference participants uses a unique key.

This also changes the trust level required on the conference roster handling at any point and how to keep that accurate and secured.

It should be noted that timely completion of the re-keying operations become an obstacle in system design and operation. Thus, it is a goal to allow for this possibility when it is deemed essential, but it should not be a requirement on a system to re-key each time the participant list changes.



#### 6.1.6. Compatibility with the WebRTC Security Architecture

It is a goal of this work to ensure compatibility with the WebRTC security architecture as described in [I.D-rtcweb-security-arch]. As an example, local resources that are considered a part of the trusted computing base (TCB), such as keying material derived using DTLS-SRTP, will remain within the TCB and not exposed to untrusted entities.

The browser is reliant on an external calling service to convey signaling information that may open the door for a man-in-the-middle attack, such as the conveyance of certificate fingerprints over the interface between the browser and the calling service. However, as described in [I.D-rtcweb-security-arch], the browser may utilize additional services, such as a trusted identify provider, to mitigate such risks.

#### 6.2. Non-Goals

##### 6.2.1. Securing the Endpoints

The security of a communication session requires that the endpoints are not compromised and that the users are trustworthy. If not, credentials and decrypted content may be shared with third parties. However, this is hard to prevent through system design. Thus, it should be assumed that the endpoint is secure and the user is trustworthy; how to achieve this is out of scope this document.

##### 6.2.2. Concealing that Communication Occurs

A non-goal is to attempt to prevent a pervasive monitoring adversary from knowing that the communication session has occurred. The reason for excluding this as a goal is that it is extremely difficult to achieve, as a pervasive monitoring adversary can be expected to be able to have knowledge of all IP flows that enter or exit local ISPs, across links that straddle nation borders or internet exchange points. To hide the fact communication occurred, the flows required to achieve the communication session need to be highly difficult to correlate between different legs of the communication.

At this stage this is deemed too difficult to attempt and will need to be a subject for further study. Existing attempts include The Onion Router (TOR), against which it has been claimed to be possible to monitor, at least partially, by an adversary with sufficient reach.

Also of consideration is that trying to conceal the fact that communication occurred actually makes it more difficult for network administrators to effectively manage and troubleshoot issues with conference calls.

### 6.2.3. Individual Media Source Authentication

Although the participants in the conference are authenticated, it is not a goal to provide source authentication of the media at the individual user level, instead being satisfied with being able to authenticate media as coming from an invited conference participant or not.

There exist solutions that can provide individual media source authentication (e.g., TESLA). However, they impact the performance or security properties they provide. Thus, further study is required to determine impact and resulting security properties if desired to have individual source authentication.

### 6.2.4. Support for Multicast in Switched Conferencing

Multicast traffic is, by design, transmitted to every participant in a conference. The focus of this document is only on centralized unicast conferencing that utilizes a switched conferencing architecture.

## 7. Requirements

The following are the security solution requirements for switched conferencing that enable end-to-end media privacy between all conference participants.

Note that while some switching media servers might be fully trusted entities, the intent of this solution and purpose for these private media (PM) requirements is to address those servers that are not fully trusted.

PM-01: Switching conference server MUST be able to switch the media between participants in a conference without having access to the media encryption keys.

PM-02: Solution MUST maintain all current SRTP security goals, namely the ability to provide for confidentiality, provide replay protection, and ensure message integrity.

PM-03: Solution MUST extend replay attacks protection to cover each hop in the media path. It MUST be possible to detect if a packet received by either an endpoint or a switching conference server was previously received by that entity or if the packet is not intended for that entity.

PM-04: Keys used for end-to-end encryption and authentication of RTP payloads and other information deemed unsuitable for access by the switching conference server MUST NOT be generated by or accessible to any component that is not in the fully trusted domain.

- PM-05: The switching conference server **MUST** be capable of making changes to the RTP header and, optionally, the RTP header extensions.
- PM-06: The switching conference server, or any entity that is not fully trusted, **MUST NOT** be involved in the authentication of identities for the purpose of media key distribution.
- PM-07: The switching conference server **MUST** be able to switch an already active SRTP stream to a new receiver, while guaranteeing the timely synchronization between the SRTP context of the transmitter and its current and new receivers.
- PM-08: It **MUST** be possible for the switching conference server to determine if a received media packet was transmitted by a valid conference participant.
- PM-09: It **MUST** be possible for a conference to be optionally re-keyed as desired, such as each time a participant joins or leaves the conference.
- PM-10: To decrypt packets, the receiving endpoint needs to be able to know the SSRC and RTP sequence number used by the sending endpoint. These values need to be integrity protected end-to-end, either explicitly by inclusion in an end-to-end MAC or implicitly like the MKI field in [RFC3711].

## 8. IANA Considerations

There are no IANA considerations for this document.

## 9. Security Considerations

[TBD]

## 10. References

### 10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.
- [RFC6464] Lennox, J., Ivov, E., and E. Marocco, "A Real-time Transport Protocol (RTP) Header Extension for Client-to-Mixer Audio Level Indication", RFC 6464, December 2011.

[I.D-rtcweb-security-arch]

E. Rescorla, "WebRTC Security Architecture", Work in Progress, July 2014.

[RFC6904] J. Lennox, "Encryption of Header Extensions in the Secure Real-time Transport Protocol (SRTP)", RFC 6904, December 2013.

## 10.2. Informative References

[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.

## 11. Acknowledgments

The authors would like to thank Marcello Caramma, Matthew Miller, Christian Oien, Magnus Westerlund, Cullen Jennings, Christer Holmberg, and Bo Burman for their invaluable input.

## 12. Contributors

[TBD]

## Authors' Addresses

Paul E. Jones  
Cisco Systems, Inc.  
7025 Kit Creek Rd.  
Research Triangle Park, NC 27709  
USA

Phone: +1 919 476 2048  
Email: paulej@packetizer.com

Nermeen Ismail  
Cisco Systems, Inc.  
170 W Tasman Dr.  
San Jose  
USA

Email: nermeen@cisco.com

David Benham  
Cisco Systems, Inc.  
170 W Tasman Dr.  
San Jose  
USA

Email: dbenham@cisco.com

Nathan Buckles  
Cisco Systems, Inc.  
170 W Tasman Dr.  
San Jose  
USA

Email: nbuckles@cisco.com

John Mattsson  
Ericsson AB  
SE-164 80 Stockholm  
Sweden

Phone: +46 10 71 43 501  
Email: john.mattsson@ericsson.com

Yi Cheng  
Ericsson  
SE-164 80 Stockholm

Sweden

Phone: +46 10 71 17 589

Email: yi.cheng@ericsson.com

Richard Barnes

Mozilla

331 E Evelyn Ave.

Mountain View

USA

Email: rlb@ipv.sx



PAYLOAD  
Internet-Draft  
Intended status: Standards Track  
Expires: April 4, 2015

V. Singh  
Aalto University  
A. Begen  
Cisco Systems  
M. Zanaty  
Cisco  
October 1, 2014

RTP Payload Format for Non-Interleaved and Interleaved Parity Forward  
Error Correction (FEC)  
draft-singh-payload-rtp-ld2d-parity-scheme-00

Abstract

This document defines new RTP payload formats for the Forward Error Correction (FEC) packets that are generated by the non-interleaved and interleaved parity codes from a source media encapsulated in RTP. These parity codes are systematic codes, where a number of repair symbols are generated from a set of source symbols. These repair symbols are sent in a repair flow separate from the source flow that carries the source symbols. The non-interleaved and interleaved parity codes offer a good protection against random and bursty packet losses, respectively, at a cost of decent complexity. The RTP payload formats that are defined in this document address the scalability issues experienced with the earlier specifications including RFC 2733, RFC 5109 and SMPTE 2022-1, and offer several improvements. Due to these changes, the new payload formats are not backward compatible with the earlier specifications, but endpoints that do not implement the scheme can still work by simply ignoring the FEC packets.

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 4, 2015.



## Copyright Notice

Copyright (c) 2014 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
1.1. Use Cases for 1-D FEC Protection . . . . .	6
1.2. Use Cases for 2-D Parity FEC Protection . . . . .	7
1.3. Overhead Computation . . . . .	9
2. Requirements Notation . . . . .	9
3. Definitions and Notations . . . . .	10
3.1. Definitions . . . . .	10
3.2. Notations . . . . .	10
4. Packet Formats . . . . .	10
4.1. Source Packets . . . . .	10
4.2. Repair Packets . . . . .	10
5. Payload Format Parameters . . . . .	14
5.1. Media Type Registration . . . . .	14
5.1.1. Registration of audio/non-interleaved-parityfec . . . . .	14
5.1.2. Registration of video/non-interleaved-parityfec . . . . .	15
5.1.3. Registration of text/non-interleaved-parityfec . . . . .	17
5.1.4. Registration of application/non-interleaved-parityfec . . . . .	18
5.1.5. Registration of audio/interleaved-parityfec . . . . .	19
5.1.6. Registration of video/interleaved-parityfec . . . . .	21
5.1.7. Registration of text/interleaved-parityfec . . . . .	22
5.1.8. Registration of application/interleaved-parityfec . . . . .	23
5.2. Mapping to SDP Parameters . . . . .	25
5.2.1. Offer-Answer Model Considerations . . . . .	25
5.2.2. Declarative Considerations . . . . .	26
6. Protection and Recovery Procedures . . . . .	26
6.1. Overview . . . . .	26
6.2. Repair Packet Construction . . . . .	26
6.3. Source Packet Reconstruction . . . . .	28
6.3.1. Associating the Source and Repair Packets . . . . .	28
6.3.2. Recovering the RTP Header . . . . .	30
6.3.3. Recovering the RTP Payload . . . . .	31

6.3.4. Iterative Decoding Algorithm for the 2-D Parity FEC Protection . . . . .	32
7. SDP Examples . . . . .	34
7.1. Example SDP for 1-D Parity FEC Protection . . . . .	34
7.2. Example SDP for 2-D Parity FEC Protection . . . . .	35
8. Congestion Control Considerations . . . . .	35
9. Security Considerations . . . . .	36
10. IANA Considerations . . . . .	37
11. Acknowledgments . . . . .	37
12. Change Log . . . . .	37
12.1. draft-singh-payload-1d2d-parity-scheme-00 . . . . .	37
12.2. draft-ietf-fecframe-1d2d-parity-scheme-00 . . . . .	37
13. References . . . . .	37
13.1. Normative References . . . . .	37
13.2. Informative References . . . . .	38
Authors' Addresses . . . . .	39

## 1. Introduction

This document defines new RTP payload formats for the Forward Error Correction (FEC) that is generated by the non-interleaved and interleaved parity codes from a source media encapsulated in RTP [RFC3550]. The type of the source media protected by these parity codes can be audio, video, text or application. The FEC data are generated according to the media type parameters, which are communicated out-of-band (e.g., in SDP). Furthermore, the associations or relationships between the source and repair flows may be communicated in-band or out-of-band. Situations where adaptivity of FEC parameters is desired, the endpoint can use the in-band mechanism, whereas when the FEC parameters are fixed, the endpoint may prefer to negotiate them out-of-band.

Both the non-interleaved and interleaved parity codes use the exclusive OR (XOR) operation to generate the repair symbols. In a nutshell, the following steps take place:

1. The sender determines a set of source packets to be protected by FEC based on the media type parameters.
2. The sender applies the XOR operation on the source symbols to generate the required number of repair symbols.
3. The sender packetizes the repair symbols and sends the repair packet(s) along with the source packets to the receiver(s) (in different flows). The repair packets may be sent proactively or on-demand.

Note that the source and repair packets belong to different source and repair flows, and the sender must provide a way for the receivers to demultiplex them, even in the case they are sent in the same 5-tuple (i.e., same source/destination address/port with UDP). This is required to offer backward compatibility for endpoints that do not understand the FEC packets (See Section 4). At the receiver side, if all of the source packets are successfully received, there is no need for FEC recovery and the repair packets are discarded. However, if there are missing source packets, the repair packets can be used to recover the missing information. Figure 1 and Figure 2 describe example block diagrams for the systematic parity FEC encoder and decoder, respectively.

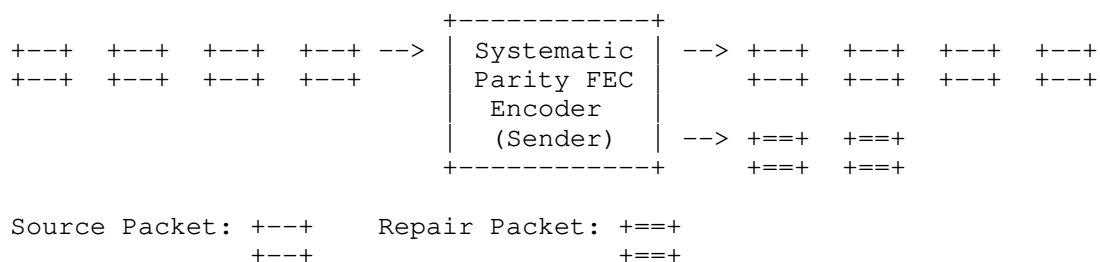


Figure 1: Block diagram for systematic parity FEC encoder

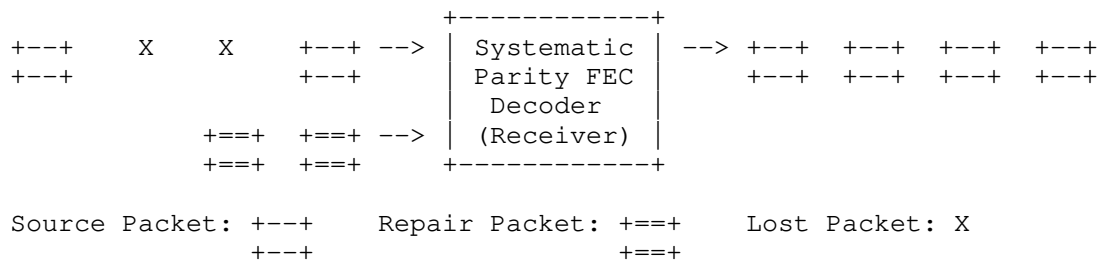


Figure 2: Block diagram for systematic parity FEC decoder

In Figure 2, it is clear that the FEC packets have to be received by the endpoint within a certain amount of time for the FEC recovery process to be useful. In this document, we refer to the time that spans a FEC block, which consists of the source packets and the corresponding repair packets, as the repair window. At the receiver side, the FEC decoder should wait at least for the duration of the repair window after getting the first packet in a FEC block, to allow all the repair packets to arrive. (The waiting time can be adjusted if there are missing packets at the beginning of the FEC block.) The FEC decoder can start decoding the already received packets sooner;

however, it should not register a FEC decoding failure until it waits at least for the duration of the repair window.

Suppose that we have a group of  $D \times L$  source packets that have sequence numbers starting from 1 running to  $D \times L$ , and a repair packet is generated by applying the XOR operation to every  $L$  consecutive packets as sketched in Figure 3. This process is referred to as 1-D non-interleaved FEC protection. As a result of this process,  $D$  repair packets are generated, which we refer to as non-interleaved (or row) FEC packets.

$$\begin{array}{rcl}
 \begin{array}{|c|c|c|c|c|} \hline S_1 & S_2 & S_3 & \dots & S_L \\ \hline \end{array} & + \begin{array}{|c|} \hline \text{XOR} \\ \hline \end{array} & = \begin{array}{|c|} \hline R_1 \\ \hline \end{array} \\
 \begin{array}{|c|c|c|c|c|} \hline S_{L+1} & S_{L+2} & S_{L+3} & \dots & S_{2 \times L} \\ \hline \end{array} & + \begin{array}{|c|} \hline \text{XOR} \\ \hline \end{array} & = \begin{array}{|c|} \hline R_2 \\ \hline \end{array} \\
 \vdots & & \vdots \\
 \begin{array}{|c|c|c|c|c|} \hline S_{(D-1) \times L+1} & S_{(D-1) \times L+2} & S_{(D-1) \times L+3} & \dots & S_{D \times L} \\ \hline \end{array} & + \begin{array}{|c|} \hline \text{XOR} \\ \hline \end{array} & = \begin{array}{|c|} \hline R_D \\ \hline \end{array}
 \end{array}$$

Figure 3: Generating non-interleaved (row) FEC packets

If we apply the XOR operation to the group of the source packets whose sequence numbers are  $L$  apart from each other, as sketched in Figure 4. In this case the endpoint generates  $L$  repair packets. This process is referred to as 1-D interleaved FEC protection, and the resulting  $L$  repair packets are referred to as interleaved (or column) FEC packets.

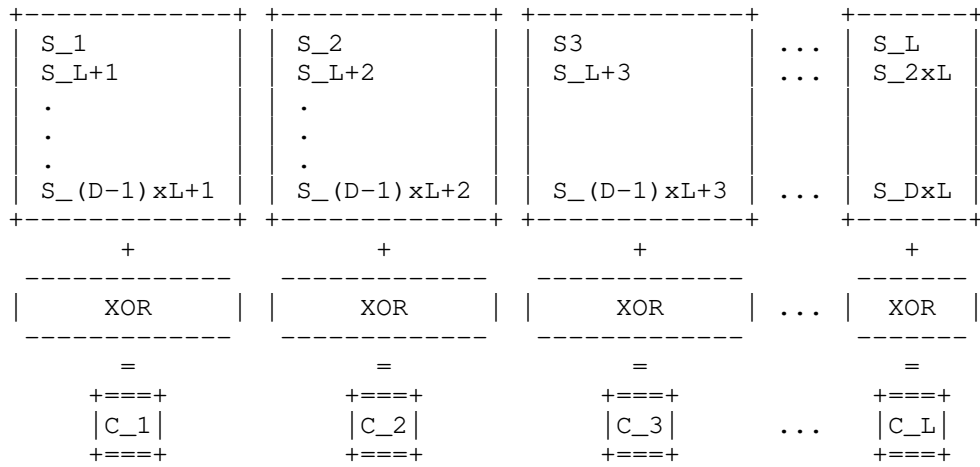


Figure 4: Generating interleaved (column) FEC packets

#### 1.1. Use Cases for 1-D FEC Protection

We generate one non-interleaved repair packet out of  $L$  consecutive source packets or one interleaved repair packet out of  $D$  non-consecutive source packets. Regardless of whether the repair packet is a non-interleaved or an interleaved one, it can provide a full recovery of the missing information if there is only one packet missing among the corresponding source packets. This implies that 1-D non-interleaved FEC protection performs better when the source packets are randomly lost. However, if the packet losses occur in bursts, 1-D interleaved FEC protection performs better provided that  $L$  is chosen large enough, i.e.,  $L$ -packet duration is not shorter than the observed burst duration. If the sender generates non-interleaved FEC packets and a burst loss hits the source packets, the repair operation fails. This is illustrated in Figure 5.

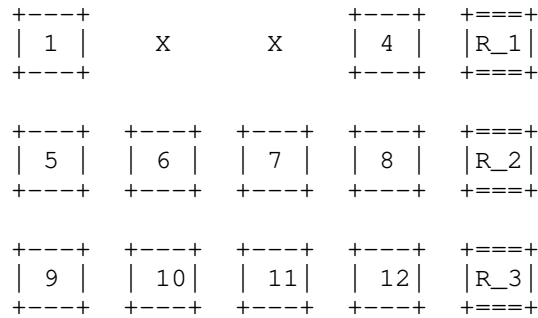


Figure 5: Example scenario where 1-D non-interleaved FEC protection fails error recovery (Burst Loss)

The sender may generate interleaved FEC packets to combat with the bursty packet losses. However, two or more random packet losses may hit the source and repair packets in the same column. In that case, the repair operation fails as well. This is illustrated in Figure 6. Note that it is possible that two burst losses may occur back-to-back, in which case interleaved FEC packets may still fail to recover the lost data.

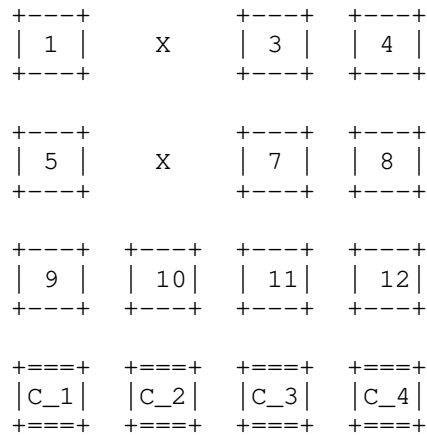


Figure 6: Example scenario where 1-D interleaved FEC protection fails error recovery (Periodic Loss)

## 1.2. Use Cases for 2-D Parity FEC Protection

In networks where the source packets are lost both randomly and in bursts, the sender ought to generate both non-interleaved and interleaved FEC packets. This type of FEC protection is known as 2-D parity FEC protection. At the expense of generating more FEC

packets, thus increasing the FEC overhead, 2-D FEC provides superior protection against mixed loss patterns. However, it is still possible for 2-D parity FEC protection to fail to recover all of the lost source packets if a particular loss pattern occurs. An example scenario is illustrated in Figure 7.

+----+			+----+	+====+
1	X	X	4	R_1
+----+			+----+	+====+
+----+	+----+	+----+	+----+	+====+
5	6	7	8	R_2
+----+	+----+	+----+	+----+	+====+
+----+			+----+	+====+
9	X	X	12	R_3
+----+			+----+	+====+
+====+	+====+	+====+	+====+	
C_1	C_2	C_3	C_4	
+====+	+====+	+====+	+====+	

Figure 7: Example scenario #1 where 2-D parity FEC protection fails error recovery

2-D parity FEC protection also fails when at least two rows are missing a source and the FEC packet and the missing source packets (in at least two rows) are aligned in the same column. An example loss pattern is sketched in Figure 8. Similarly, 2-D parity FEC protection cannot repair all missing source packets when at least two columns are missing a source and the FEC packet and the missing source packets (in at least two columns) are aligned in the same row.

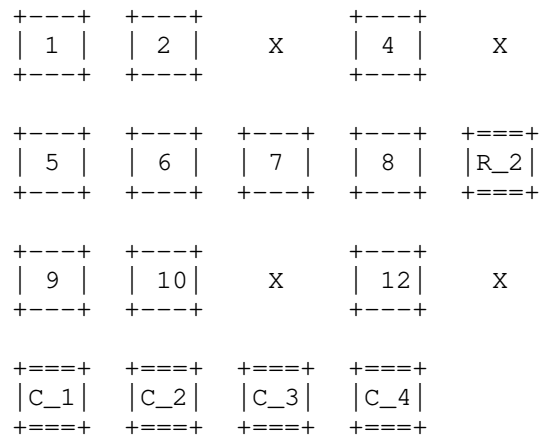


Figure 8: Example scenario #2 where 2-D parity FEC protection fails error recovery

### 1.3. Overhead Computation

The overhead is defined as the ratio of the number of bytes belonging to the repair packets to the number of bytes belonging to the protected source packets.

Generally, repair packets are larger in size compared to the source packets. Also, not all the source packets are necessarily equal in size. However, if we assume that each repair packet carries an equal number of bytes carried by a source packet, we can compute the overhead for different FEC protection methods as follows:

- o 1-D Non-interleaved FEC Protection: Overhead =  $1/L$
- o 1-D Interleaved FEC Protection: Overhead =  $1/D$
- o 2-D Parity FEC Protection: Overhead =  $1/L + 1/D$

where L and D are the number of columns and rows in the source block, respectively.

### 2. Requirements Notation

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 [RFC2119].



### 3. Definitions and Notations

#### 3.1. Definitions

This document uses a number of definitions from [RFC6363].

#### 3.2. Notations

- o L: Number of columns of the source block.
- o D: Number of rows of the source block.
- o bitmask: Run-length encoding of packets protected by a FEC packet. If the bit  $i$  in the mask is set to 1, the source packet number  $N + i$  is protected by this FEC packet. Here,  $N$  is the sequence number base, which is indicated in the FEC packet as well.

### 4. Packet Formats

This section defines the formats of the source and repair packets.

#### 4.1. Source Packets

The source packets MUST contain the information that identifies the source block and the position within the source block occupied by the packet. Since the source packets that are carried within an RTP stream already contain unique sequence numbers in their RTP headers [RFC3550], we can identify the source packets in a straightforward manner and there is no need to append additional field(s). The primary advantage of not modifying the source packets in any way is that it provides backward compatibility for the receivers that do not support FEC at all. In multicast scenarios, this backward compatibility becomes quite useful as it allows the non-FEC-capable and FEC-capable receivers to receive and interpret the same source packets sent in the same multicast session.

#### 4.2. Repair Packets

The repair packets MUST contain information that identifies the source block they pertain to and the relationship between the contained repair symbols and the original source block. For this purpose, we use the RTP header of the repair packets as well as another header within the RTP payload, which we refer to as the FEC header, as shown in Figure 9.

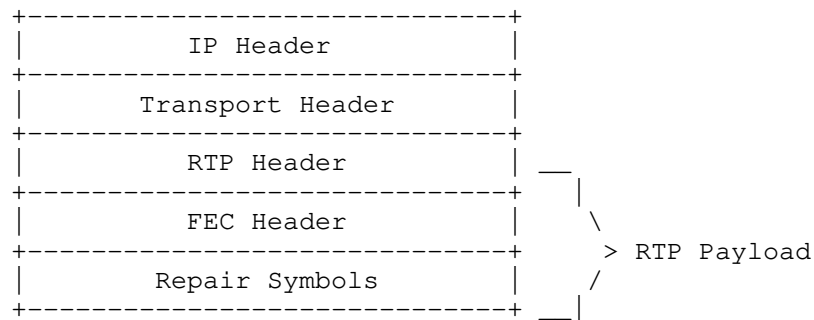


Figure 9: Format of repair packets

The RTP header is formatted according to [RFC3550] with some further clarifications listed below:

- o Marker (M) Bit: This bit is not used for this payload type, and SHALL be set to 0.
- o Payload Type: The (dynamic) payload type for the repair packets is determined through out-of-band means. Note that this document registers new payload formats for the repair packets (Refer to Section 5 for details). According to [RFC3550], an RTP receiver that cannot recognize a payload type must discard it. This provides backward compatibility. If a non-FEC-capable receiver receives a repair packet, it will not recognize the payload type, and hence, will discard the repair packet.
- o Sequence Number (SN): The sequence number has the standard definition. It MUST be one higher than the sequence number in the previously transmitted repair packet. The initial value of the sequence number SHOULD be random (unpredictable, based on [RFC3550]).
- o Timestamp (TS): The timestamp SHALL be set to a time corresponding to the repair packet's transmission time. Note that the timestamp value has no use in the actual FEC protection process and is usually useful for jitter calculations.
- o Synchronization Source (SSRC): The SSRC value SHALL be randomly assigned as suggested by [RFC3550]. This allows the sender to multiplex the source and repair flows on the same port, or multiplex multiple repair flows on a single port. The repair flows SHOULD use the RTCP CNAME field to associate themselves with the source flow.

In some networks, the RTP Source, which produces the source packets and the FEC Source, which generates the repair packets from the source packets may not be the same host. In such scenarios, using the same CNAME for the source and repair flows means that the RTP Source and the FEC Source MUST share the same CNAME (for this specific source-repair flow association). A common CNAME may be produced based on an algorithm that is known both to the RTP and FEC Source [RFC7022]. This usage is compliant with [RFC3550].

Note that due to the randomness of the SSRC assignments, there is a possibility of SSRC collision. In such cases, the collisions MUST be resolved as described in [RFC3550].

The format of the FEC header is shown in Figure 10.

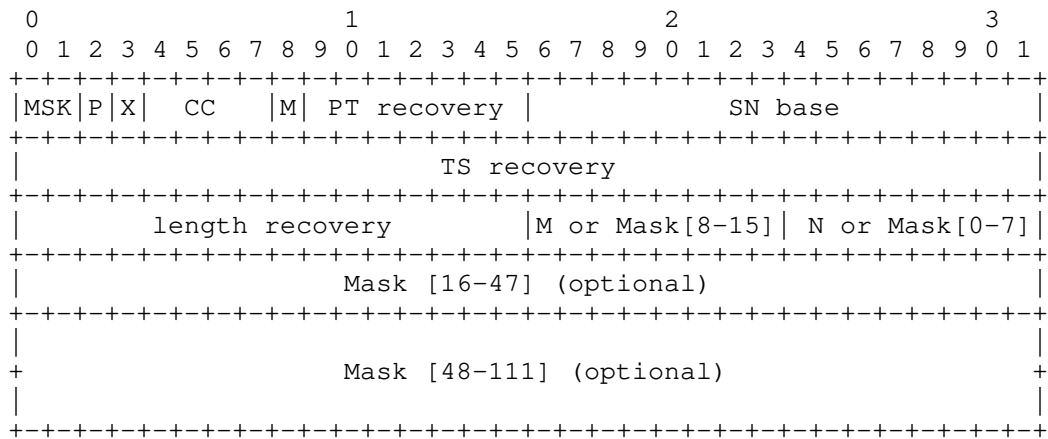


Figure 10: Format of the FEC header

The FEC header consists of the following fields:

- o The MSK field (2 bits) indicates the type of the mask. Namely:

MSK bits	Use
00	16-bit mask
01	48-bit mask
10	112-bit mask
11	packets indicated by offset M and N

Figure 11: MSK bit values

- o The P, X, CC, M and PT recovery fields are used to determine the corresponding fields of the recovered packets.
- o The SN base field is used to indicate the lowest sequence number, taking wrap around into account, of those source packets protected by this repair packet.
- o The TS recovery field is used to determine the timestamp of the recovered packets.
- o The Length recovery field is used to determine the length of the recovered packets.
- o Mask is a run-length encoding of packets protected by the FEC packet. Where a bit  $i$  set to 1 indicates that the source packet with sequence number (SN base +  $i$ ) is protected by this FEC packet.
- o If the the MSK field is set to 11, it indicates the offset of packets protected by this FEC packet. Consequently, the following conditions may occur:

If  $M=0$ ,  $N=0$ , regular protection pattern code with the values of L and D are indicated in the SDP description.

If  $M>0$ ,  $N=0$ , indicates a non-interleaved (row) FEC of M packets starting at SN base.  
Hence,  $FEC = SN, SN+1, SN+2, \dots, SN+(M-1), SN+M$ .

If  $M>0$ ,  $N>0$ , indicates interleaved (column) FEC of every M packet in a group of N packets starting at SN base.  
Hence,  $FEC = SN+(M \times 0), SN+(M \times 1), \dots, SN+(M \times N)$ .

Figure 12: Interpreting the M and N field values

The details on setting the fields in the FEC header are provided in Section 6.2.

It should be noted that a mask-based approach (similar to the ones specified in [RFC2733] and [RFC5109]) may not be very efficient to indicate which source packets in the current source block are associated with a given repair packet. In particular, for the applications that would like to use large source block sizes, the size of the mask that is required to describe the source-repair packet associations may be prohibitively large. The 8-bit fields proposed in [SMPTE2022-1] indicate a systematized approach. Instead the approach in this document uses the 8-bit fields to indicate packet offsets protected by the FEC packet. The approach in [SMPTE2022-1] is inherently more efficient for regular patterns, it

does not provide flexibility to represent other protection patterns (e.g., staircase).

## 5. Payload Format Parameters

This section provides the media subtype registration for the non-interleaved and interleaved parity FEC. The parameters that are required to configure the FEC encoding and decoding operations are also defined in this section.

### 5.1. Media Type Registration

This registration is done using the template defined in [RFC6838] and following the guidance provided in [RFC3555].

Note to the RFC Editor: In the following sections, please replace "XXXX" with the number of this document prior to publication as an RFC.

#### 5.1.1. Registration of audio/non-interleaved-parityfec

Type name: audio

Subtype name: non-interleaved-parityfec

Required parameters:

- o rate: The RTP timestamp (clock) rate. The rate SHALL be larger than 1000 Hz to provide sufficient resolution to RTCP operations. However, it is RECOMMENDED to select the rate that matches the rate of the protected source RTP stream.
- o L: Number of columns of the source block. L is a positive integer.
- o D: Number of rows of the source block. D is a positive integer.
- o ToP: Type of the protection applied by the sender: 0 for 1-D interleaved FEC protection, 1 for 1-D non-interleaved FEC protection, and 2 for 2-D parity FEC protection. The ToP value of 3 is reserved for future uses.
- o repair-window: The time that spans the source packets and the corresponding repair packets. The size of the repair window is specified in microseconds.

Optional parameters: None.

Encoding considerations: This media type is framed (See Section 4.8 in the template document [RFC6838]) and contains binary data.

Security considerations: See Section 9 of [RFCXXXX].

Interoperability considerations: None.

Published specification: [RFCXXXX].

Applications that use this media type: Multimedia applications that want to improve resiliency against packet loss by sending redundant data in addition to the source media.

Fragment identifier considerations: None.

Additional information: None.

Person & email address to contact for further information: Varun Singh <varun.singh@iki.fi> and IETF Audio/Video Transport Payloads Working Group.

Intended usage: COMMON.

Restriction on usage: This media type depends on RTP framing, and hence, is only defined for transport via RTP [RFC3550].

Author: Varun Singh <varun.singh@iki.fi>.

Change controller: IETF Audio/Video Transport Working Group delegated from the IESG.

Provisional registration? (standards tree only): Yes.

#### 5.1.2. Registration of video/non-interleaved-parityfec

Type name: video

Subtype name: non-interleaved-parityfec

Required parameters:

- o rate: The RTP timestamp (clock) rate. The rate SHALL be larger than 1000 Hz to provide sufficient resolution to RTCP operations. However, it is RECOMMENDED to select the rate that matches the rate of the protected source RTP stream.
- o L: Number of columns of the source block. L is a positive integer.

- o D: Number of rows of the source block. D is a positive integer.
- o ToP: Type of the protection applied by the sender: 0 for 1-D interleaved FEC protection, 1 for 1-D non-interleaved FEC protection, and 2 for 2-D parity FEC protection. The ToP value of 3 is reserved for future uses.
- o repair-window: The time that spans the source packets and the corresponding repair packets. The size of the repair window is specified in microseconds.

Optional parameters: None.

Encoding considerations: This media type is framed (See Section 4.8 in the template document [RFC6838]) and contains binary data.

Security considerations: See Section 9 of [RFCXXXX].

Interoperability considerations: None.

Published specification: [RFCXXXX].

Applications that use this media type: Multimedia applications that want to improve resiliency against packet loss by sending redundant data in addition to the source media.

Fragment identifier considerations: None.

Additional information: None.

Person & email address to contact for further information: Varun Singh <varun.singh@iki.fi> and IETF Audio/Video Transport Payloads Working Group.

Intended usage: COMMON.

Restriction on usage: This media type depends on RTP framing, and hence, is only defined for transport via RTP [RFC3550].

Author: Varun Singh <varun.singh@iki.fi>.

Change controller: IETF Audio/Video Transport Working Group delegated from the IESG.

Provisional registration? (standards tree only): Yes.

## 5.1.3. Registration of text/non-interleaved-parityfec

Type name: text

Subtype name: non-interleaved-parityfec

Required parameters:

- o rate: The RTP timestamp (clock) rate. The rate SHALL be larger than 1000 Hz to provide sufficient resolution to RTCP operations. However, it is RECOMMENDED to select the rate that matches the rate of the protected source RTP stream.
- o L: Number of columns of the source block. L is a positive integer.
- o D: Number of rows of the source block. D is a positive integer.
- o ToP: Type of the protection applied by the sender: 0 for 1-D interleaved FEC protection, 1 for 1-D non-interleaved FEC protection, and 2 for 2-D parity FEC protection. The ToP value of 3 is reserved for future uses.
- o repair-window: The time that spans the source packets and the corresponding repair packets. The size of the repair window is specified in microseconds.

Optional parameters: None.

Encoding considerations: This media type is framed (See Section 4.8 in the template document [RFC6838]) and contains binary data.

Security considerations: See Section 9 of [RFCXXXX].

Interoperability considerations: None.

Published specification: [RFCXXXX].

Applications that use this media type: Multimedia applications that want to improve resiliency against packet loss by sending redundant data in addition to the source media.

Fragment identifier considerations: None.

Additional information: None.



Person & email address to contact for further information: Varun Singh <varun.singh@iki.fi> and IETF Audio/Video Transport Payloads Working Group.

Intended usage: COMMON.

Restriction on usage: This media type depends on RTP framing, and hence, is only defined for transport via RTP [RFC3550].

Author: Varun Singh <varun.singh@iki.fi>.

Change controller: IETF Audio/Video Transport Working Group delegated from the IESG.

Provisional registration? (standards tree only): Yes.

#### 5.1.4. Registration of application/non-interleaved-parityfec

Type name: application

Subtype name: non-interleaved-parityfec

Required parameters:

- o rate: The RTP timestamp (clock) rate. The rate SHALL be larger than 1000 Hz to provide sufficient resolution to RTCP operations. However, it is RECOMMENDED to select the rate that matches the rate of the protected source RTP stream.
- o L: Number of columns of the source block. L is a positive integer.
- o D: Number of rows of the source block. D is a positive integer.
- o ToP: Type of the protection applied by the sender: 0 for 1-D interleaved FEC protection, 1 for 1-D non-interleaved FEC protection, and 2 for 2-D parity FEC protection. The ToP value of 3 is reserved for future uses.
- o repair-window: The time that spans the source packets and the corresponding repair packets. The size of the repair window is specified in microseconds.

Optional parameters: None.

Encoding considerations: This media type is framed (See Section 4.8 in the template document [RFC6838]) and contains binary data.

Security considerations: See Section 9 of [RFCXXXX].

Interoperability considerations: None.

Published specification: [RFCXXXX].

Applications that use this media type: Multimedia applications that want to improve resiliency against packet loss by sending redundant data in addition to the source media.

Fragment identifier considerations: None.

Additional information: None.

Person & email address to contact for further information: Varun Singh <varun.singh@iki.fi> and IETF Audio/Video Transport Payloads Working Group.

Intended usage: COMMON.

Restriction on usage: This media type depends on RTP framing, and hence, is only defined for transport via RTP [RFC3550].

Author: Varun Singh <varun.singh@iki.fi>.

Change controller: IETF Audio/Video Transport Working Group delegated from the IESG.

Provisional registration? (standards tree only): Yes.

#### 5.1.5. Registration of audio/interleaved-parityfec

Type name: audio

Subtype name: interleaved-parityfec

Required parameters:

- o rate: The RTP timestamp (clock) rate. The rate SHALL be larger than 1000 Hz to provide sufficient resolution to RTCP operations. However, it is RECOMMENDED to select the rate that matches the rate of the protected source RTP stream.
- o L: Number of columns of the source block. L is a positive integer.
- o D: Number of rows of the source block. D is a positive integer.

- o ToP: Type of the protection applied by the sender: 0 for 1-D interleaved FEC protection, 1 for 1-D non-interleaved FEC protection, and 2 for 2-D parity FEC protection. The ToP value of 3 is reserved for future uses.
- o repair-window: The time that spans the source packets and the corresponding repair packets. The size of the repair window is specified in microseconds.

Optional parameters: None.

Encoding considerations: This media type is framed (See Section 4.8 in the template document [RFC6838]) and contains binary data.

Security considerations: See Section 9 of [RFCXXXX].

Interoperability considerations: None.

Published specification: [RFCXXXX].

Applications that use this media type: Multimedia applications that want to improve resiliency against packet loss by sending redundant data in addition to the source media.

Fragment identifier considerations: None.

Additional information: None.

Person & email address to contact for further information: Varun Singh <varun.singh@iki.fi> and IETF Audio/Video Transport Payloads Working Group.

Intended usage: COMMON.

Restriction on usage: This media type depends on RTP framing, and hence, is only defined for transport via RTP [RFC3550].

Author: Varun Singh <varun.singh@iki.fi>.

Change controller: IETF Audio/Video Transport Working Group delegated from the IESG.

Provisional registration? (standards tree only): Yes.

## 5.1.6. Registration of video/interleaved-parityfec

Type name: video

Subtype name: interleaved-parityfec

Required parameters:

- o rate: The RTP timestamp (clock) rate. The rate SHALL be larger than 1000 Hz to provide sufficient resolution to RTCP operations. However, it is RECOMMENDED to select the rate that matches the rate of the protected source RTP stream.
- o L: Number of columns of the source block. L is a positive integer.
- o D: Number of rows of the source block. D is a positive integer.
- o ToP: Type of the protection applied by the sender: 0 for 1-D interleaved FEC protection, 1 for 1-D non-interleaved FEC protection, and 2 for 2-D parity FEC protection. The ToP value of 3 is reserved for future uses.
- o repair-window: The time that spans the source packets and the corresponding repair packets. The size of the repair window is specified in microseconds.

Optional parameters: None.

Encoding considerations: This media type is framed (See Section 4.8 in the template document [RFC6838]) and contains binary data.

Security considerations: See Section 9 of [RFCXXXX].

Interoperability considerations: None.

Published specification: [RFCXXXX].

Applications that use this media type: Multimedia applications that want to improve resiliency against packet loss by sending redundant data in addition to the source media.

Fragment identifier considerations: None.

Additional information: None.

Person & email address to contact for further information: Varun Singh <varun.singh@iki.fi> and IETF Audio/Video Transport Payloads Working Group.

Intended usage: COMMON.

Restriction on usage: This media type depends on RTP framing, and hence, is only defined for transport via RTP [RFC3550].

Author: Varun Singh <varun.singh@iki.fi>.

Change controller: IETF Audio/Video Transport Working Group delegated from the IESG.

Provisional registration? (standards tree only): Yes.

#### 5.1.7. Registration of text/interleaved-parityfec

Type name: text

Subtype name: interleaved-parityfec

Required parameters:

- o rate: The RTP timestamp (clock) rate. The rate SHALL be larger than 1000 Hz to provide sufficient resolution to RTCP operations. However, it is RECOMMENDED to select the rate that matches the rate of the protected source RTP stream.
- o L: Number of columns of the source block. L is a positive integer.
- o D: Number of rows of the source block. D is a positive integer.
- o ToP: Type of the protection applied by the sender: 0 for 1-D interleaved FEC protection, 1 for 1-D non-interleaved FEC protection, and 2 for 2-D parity FEC protection. The ToP value of 3 is reserved for future uses.
- o repair-window: The time that spans the source packets and the corresponding repair packets. The size of the repair window is specified in microseconds.

Optional parameters: None.

Encoding considerations: This media type is framed (See Section 4.8 in the template document [RFC6838]) and contains binary data.

Security considerations: See Section 9 of [RFCXXXX].

Interoperability considerations: None.

Published specification: [RFCXXXX].

Applications that use this media type: Multimedia applications that want to improve resiliency against packet loss by sending redundant data in addition to the source media.

Fragment identifier considerations: None.

Additional information: None.

Person & email address to contact for further information: Varun Singh <varun.singh@iki.fi> and IETF Audio/Video Transport Payloads Working Group.

Intended usage: COMMON.

Restriction on usage: This media type depends on RTP framing, and hence, is only defined for transport via RTP [RFC3550].

Author: Varun Singh <varun.singh@iki.fi>.

Change controller: IETF Audio/Video Transport Working Group delegated from the IESG.

Provisional registration? (standards tree only): Yes.

#### 5.1.8. Registration of application/interleaved-parityfec

Type name: application

Subtype name: interleaved-parityfec

Required parameters:

- o rate: The RTP timestamp (clock) rate. The rate SHALL be larger than 1000 Hz to provide sufficient resolution to RTCP operations. However, it is RECOMMENDED to select the rate that matches the rate of the protected source RTP stream.
- o L: Number of columns of the source block. L is a positive integer.
- o D: Number of rows of the source block. D is a positive integer.

- o ToP: Type of the protection applied by the sender: 0 for 1-D interleaved FEC protection, 1 for 1-D non-interleaved FEC protection, and 2 for 2-D parity FEC protection. The ToP value of 3 is reserved for future uses.
- o repair-window: The time that spans the source packets and the corresponding repair packets. The size of the repair window is specified in microseconds.

Optional parameters: None.

Encoding considerations: This media type is framed (See Section 4.8 in the template document [RFC6838]) and contains binary data.

Security considerations: See Section 9 of [RFCXXXX].

Interoperability considerations: None.

Published specification: [RFCXXXX].

Applications that use this media type: Multimedia applications that want to improve resiliency against packet loss by sending redundant data in addition to the source media.

Fragment identifier considerations: None.

Additional information: None.

Person & email address to contact for further information: Varun Singh <varun.singh@iki.fi> and IETF Audio/Video Transport Payloads Working Group.

Intended usage: COMMON.

Restriction on usage: This media type depends on RTP framing, and hence, is only defined for transport via RTP [RFC3550].

Author: Varun Singh <varun.singh@iki.fi>.

Change controller: IETF Audio/Video Transport Working Group delegated from the IESG.

Provisional registration? (standards tree only): Yes.

## 5.2. Mapping to SDP Parameters

Applications that are using RTP transport commonly use Session Description Protocol (SDP) [RFC4566] to describe their RTP sessions. The information that is used to specify the media types in an RTP session has specific mappings to the fields in an SDP description. In this section, we provide these mappings for the media subtypes registered by this document. Note that if an application does not use SDP to describe the RTP sessions, an appropriate mapping must be defined and used to specify the media types and their parameters for the control/description protocol employed by the application.

The mapping of the media type specification for "non-interleaved-parityfec" and "interleaved-parityfec" and their parameters in SDP is as follows:

- o The media type (e.g., "application") goes into the "m=" line as the media name.
- o The media subtype goes into the "a=rtpmap" line as the encoding name. The RTP clock rate parameter ("rate") also goes into the "a=rtpmap" line as the clock rate.
- o The remaining required payload-format-specific parameters go into the "a=fmtp" line by copying them directly from the media type string as a semicolon-separated list of parameter=value pairs.

SDP examples are provided in Section 7.

### 5.2.1. Offer-Answer Model Considerations

When offering 1-D interleaved parity FEC over RTP using SDP in an Offer/Answer model [RFC3264], the following considerations apply:

- o Each combination of the L and D parameters produces a different FEC data and is not compatible with any other combination. A sender application may desire to offer multiple offers with different sets of L and D values as long as the parameter values are valid. The receiver SHOULD normally choose the offer that has a sufficient amount of interleaving. If multiple such offers exist, the receiver may choose the offer that has the lowest overhead or the one that requires the smallest amount of buffering. The selection depends on the application requirements.
- o The value for the repair-window parameter depends on the L and D values and cannot be chosen arbitrarily. More specifically, L and D values determine the lower limit for the repair-window size.



The upper limit of the repair-window size does not depend on the L and D values.

- o Although combinations with the same L and D values but with different repair-window sizes produce the same FEC data, such combinations are still considered different offers. The size of the repair-window is related to the maximum delay between the transmission of a source packet and the associated repair packet. This directly impacts the buffering requirement on the receiver side and the receiver must consider this when choosing an offer.
- o There are no optional format parameters defined for this payload. Any unknown option in the offer MUST be ignored and deleted from the answer. If FEC is not desired by the receiver, it can be deleted from the answer.

#### 5.2.2. Declarative Considerations

In declarative usage, like SDP in the Real-time Streaming Protocol (RTSP) [RFC2326] or the Session Announcement Protocol (SAP) [RFC2974], the following considerations apply:

- o The payload format configuration parameters are all declarative and a participant MUST use the configuration that is provided for the session.
- o More than one configuration may be provided (if desired) by declaring multiple RTP payload types. In that case, the receivers should choose the repair flow that is best for them.

### 6. Protection and Recovery Procedures

This section provides a complete specification of the 1-D and 2-D parity codes and their RTP payload formats.

#### 6.1. Overview

The following sections specify the steps involved in generating the repair packets and reconstructing the missing source packets from the repair packets.

#### 6.2. Repair Packet Construction

The RTP header of a repair packet is formed based on the guidelines given in Section 4.2.

The FEC header includes 12 octets (or upto 28 octets when the longer optional masks are used). It is constructed by applying the XOR

operation on the bit strings that are generated from the individual source packets protected by this particular repair packet. The set of the source packets that are associated with a given repair packet can be computed by the formula given in Section 6.3.1.

The bit string is formed for each source packet by concatenating the following fields together in the order specified:

- o The first 64 bits of the RTP header (64 bits).
- o Unsigned network-ordered 16-bit representation of the source packet length in bytes minus 12 (for the fixed RTP header), i.e., the sum of the lengths of all the following if present: the CSRC list, extension header, RTP payload and RTP padding (16 bits).

By applying the parity operation on the bit strings produced from the source packets, we generate the FEC bit string. The FEC header is generated from the FEC bit string as follows:

- o The first (most significant) 2 bits in the FEC bit string are skipped. The MSK bits in the FEC header are set to the appropriate value, i.e., it depends on the chosen bitmask length.
- o The next bit in the FEC bit string is written into the P recovery bit in the FEC header.
- o The next bit in the FEC bit string is written into the X recovery bit in the FEC header.
- o The next 4 bits of the FEC bit string are written into the CC recovery field in the FEC header.
- o The next bit is written into the M recovery bit in the FEC header.
- o The next 7 bits of the FEC bit string are written into the PT recovery field in the FEC header.
- o The next 16 bits are skipped.
- o The next 32 bits of the FEC bit string are written into the TS recovery field in the FEC header.
- o The next 16 bits are written into the length recovery field in the FEC header.
- o Depending on the chosen MSK value, the bit mask of appropriate length will be set to the appropriate values.

As described in Section 4.2, the SN base field of the FEC header MUST be set to the lowest sequence number of the source packets protected by this repair packet. When MSK represents a bitmask (MSK=00,01,10), the SN base field corresponds to the lowest sequence number indicated in the bitmask. When MSK=11, the following considerations apply: 1) for the interleaved FEC packets, this corresponds to the lowest sequence number of the source packets that forms the column, 2) for the non-interleaved FEC packets, the SN base field MUST be set to the lowest sequence number of the source packets that forms the row.

The repair packet payload consists of the bits that are generated by applying the XOR operation on the payloads of the source RTP packets. If the payload lengths of the source packets are not equal, each shorter packet MUST be padded to the length of the longest packet by adding octet 0's at the end.

Due to this possible padding and mandatory FEC header, a repair packet has a larger size than the source packets it protects. This may cause problems if the resulting repair packet size exceeds the Maximum Transmission Unit (MTU) size of the path over which the repair flow is sent.

### 6.3. Source Packet Reconstruction

This section describes the recovery procedures that are required to reconstruct the missing source packets. The recovery process has two steps. In the first step, the FEC decoder determines which source and repair packets should be used in order to recover a missing packet. In the second step, the decoder recovers the missing packet, which consists of an RTP header and RTP payload.

In the following, we describe the RECOMMENDED algorithms for the first and second steps. Based on the implementation, different algorithms MAY be adopted. However, the end result MUST be identical to the one produced by the algorithms described below.

Note that the same algorithms are used by the 1-D parity codes, regardless of whether the FEC protection is applied over a column or a row. The 2-D parity codes, on the other hand, usually require multiple iterations of the procedures described here. This iterative decoding algorithm is further explained in Section 6.3.4.

#### 6.3.1. Associating the Source and Repair Packets

We denote the set of the source packets associated with repair packet  $p^*$  by set  $T(p^*)$ . Note that in a source block whose size is  $L$  columns by  $D$  rows, set  $T$  includes  $D$  source packets plus one repair packet for the FEC protection applied over a column, and  $L$  source packets plus

one repair packet for the FEC protection applied over a row. Recall that 1-D interleaved and non-interleaved FEC protection can fully recover the missing information if there is only one source packet missing in set T. If there are more than one source packets missing in set T, 1-D FEC protection will not work.

#### 6.3.1.1. Signaled in SDP

The first step is associating the source and repair packets. If the endpoint relies entirely on out-of-band signaling (MSK=11, and M=N=0), then this information may be inferred from the media type parameters specified in the SDP description. Furthermore, the payload type field in the RTP header, assists the receiver distinguish an interleaved or non-interleaved FEC packet.

Mathematically, for any received repair packet,  $p^*$ , we can determine the sequence numbers of the source packets that are protected by this repair packet as follows:

$$p^*_{\text{snb}} + i * X\_1 \text{ (modulo 65536)}$$

where  $p^*_{\text{snb}}$  denotes the value in the SN base field of  $p^*$ 's FEC header,  $X\_1$  is set to L and 1 for the interleaved and non-interleaved FEC packets, respectively, and

$$0 \leq i < X\_2$$

where  $X\_2$  is set to D and L for the interleaved and non-interleaved FEC packets, respectively.

#### 6.3.1.2. Using bitmasks

When using fixed size bitmasks (16-, 48-, 112-bits), the SN base field in the FEC header indicates the lowest sequence number of the source packets that forms the FEC packet. Finally, the bits marked by "1" in the bitmask are offsets from the SN base and make up the rest of the packets protected by the FEC packet. The bitmasks are able to represent arbitrary protection patterns, for example, 1-D interleaved, 1-D non-interleaved, 2-D, staircase.

#### 6.3.1.3. Using M and N Offsets

When value of M is non-zero, the 8-bit fields indicate the offset of packets protected by an interleaved ( $N>0$ ) or non-interleaved ( $N=0$ ) FEC packet. Using a combination of interleaved and non-interleaved FEC packets can form 2-D protection patterns.

Mathematically, for any received repair packet,  $p^*$ , we can determine the sequence numbers of the source packets that are protected by this repair packet are as follows:

When  $N = 0$ :  
 $p^*_{\text{snb}}, p^*_{\text{snb}+1}, \dots, p^*_{\text{snb}+(M-1)}, p^*_{\text{snb}+M}$   
When  $N > 0$ :  
 $p^*_{\text{snb}}, p^*_{\text{snb}+(M \times 1)}, p^*_{\text{snb}+(M \times 2)}, \dots, p^*_{\text{snb}+(M \times (N-1))}, p^*_{\text{snb}+(M \times N)}$

#### 6.3.2. Recovering the RTP Header

For a given set  $T$ , the procedure for the recovery of the RTP header of the missing packet, whose sequence number is denoted by  $\text{SEQNUM}$ , is as follows:

1. For each of the source packets that are successfully received in  $T$ , compute the 80-bit string by concatenating the first 64 bits of their RTP header and the unsigned network-ordered 16-bit representation of their length in bytes minus 12.
2. For the repair packet in  $T$ , compute the FEC bit string from the first 80 bits of the FEC header.
3. Calculate the recovered bit string as the XOR of the bit strings generated from all source packets in  $T$  and the FEC bit string generated from the repair packet in  $T$ .
4. Create a new packet with the standard 12-byte RTP header and no payload.
5. Set the version of the new packet to 2. Skip the first 2 bits in the recovered bit string.
6. Set the Padding bit in the new packet to the next bit in the recovered bit string.
7. Set the Extension bit in the new packet to the next bit in the recovered bit string.
8. Set the CC field to the next 4 bits in the recovered bit string.
9. Set the Marker bit in the new packet to the next bit in the recovered bit string.
10. Set the Payload type in the new packet to the next 7 bits in the recovered bit string.

11. Set the SN field in the new packet to SEQNUM. Skip the next 16 bits in the recovered bit string.
12. Set the TS field in the new packet to the next 32 bits in the recovered bit string.
13. Take the next 16 bits of the recovered bit string and set the new variable Y to whatever unsigned integer this represents (assuming network order). Convert Y to host order. Y represents the length of the new packet in bytes minus 12 (for the fixed RTP header), i.e., the sum of the lengths of all the following if present: the CSRC list, header extension, RTP payload and RTP padding.
14. Set the SSRC of the new packet to the SSRC of the source RTP stream.

This procedure recovers the header of an RTP packet up to (and including) the SSRC field.

#### 6.3.3. Recovering the RTP Payload

Following the recovery of the RTP header, the procedure for the recovery of the RTP payload is as follows:

1. Append Y bytes to the new packet.
2. For each of the source packets that are successfully received in T, compute the bit string from the Y octets of data starting with the 13th octet of the packet. If any of the bit strings generated from the source packets has a length shorter than Y, pad them to that length. The padding of octet 0 MUST be added at the end of the bit string. Note that the information of the first 8 octets are protected by the FEC header.
3. For the repair packet in T, compute the FEC bit string from the repair packet payload, i.e., the Y octets of data following the FEC header. Note that the FEC header may be 12, 16, 32 octets depending on the length of the bitmask.
4. Calculate the recovered bit string as the XOR of the bit strings generated from all source packets in T and the FEC bit string generated from the repair packet in T.
5. Append the recovered bit string (Y octets) to the new packet generated in Section 6.3.2.

#### 6.3.4. Iterative Decoding Algorithm for the 2-D Parity FEC Protection

In 2-D parity FEC protection, the sender generates both non-interleaved and interleaved FEC packets to combat with the mixed loss patterns (random and bursty). At the receiver side, these FEC packets are used iteratively to overcome the shortcomings of the 1-D non-interleaved/interleaved FEC protection and improve the chances of full error recovery.

The iterative decoding algorithm runs as follows:

1. Set num\_recovered\_until\_this\_iteration to zero
2. Set num\_recovered\_so\_far to zero
3. Recover as many source packets as possible by using the non-interleaved FEC packets as outlined in Section 6.3.2 and Section 6.3.3, and increase the value of num\_recovered\_so\_far by the number of recovered source packets.
4. Recover as many source packets as possible by using the interleaved FEC packets as outlined in Section 6.3.2 and Section 6.3.3, and increase the value of num\_recovered\_so\_far by the number of recovered source packets.
5. If num\_recovered\_so\_far > num\_recovered\_until\_this\_iteration  
---num\_recovered\_until\_this\_iteration = num\_recovered\_so\_far  
---Go to step 3  
Else  
---Terminate

The algorithm terminates either when all missing source packets are fully recovered or when there are still remaining missing source packets but the FEC packets are not able to recover any more source packets. For the example scenarios when the 2-D parity FEC protection fails full recovery, refer to Section 1.2. Upon termination, variable num\_recovered\_so\_far has a value equal to the total number of recovered source packets.

Example:

Suppose that the receiver experienced the loss pattern sketched in Figure 13.

X	X	+---+   3   +---+	+---+   4   +---+	+===+   R_1   +===+
+---+   5   +---+	+---+   6   +---+	+---+   7   +---+	+---+   8   +---+	+===+   R_2   +===+
+---+   9   +---+	X	X	+---+   12   +---+	+===+   R_3   +===+
+===+   C_1   +===+	+===+   C_2   +===+	+===+   C_3   +===+	+===+   C_4   +===+	

Figure 13: Example loss pattern for the iterative decoding algorithm

The receiver executes the iterative decoding algorithm and recovers source packets #1 and #11 in the first iteration. The resulting pattern is sketched in Figure 14.

+---+   1   +---+	X	+---+   3   +---+	+---+   4   +---+	+===+   R_1   +===+
+---+   5   +---+	+---+   6   +---+	+---+   7   +---+	+---+   8   +---+	+===+   R_2   +===+
+---+   9   +---+	X	+---+   11   +---+	+---+   12   +---+	+===+   R_3   +===+
+===+   C_1   +===+	+===+   C_2   +===+	+===+   C_3   +===+	+===+   C_4   +===+	

Figure 14: The resulting pattern after the first iteration

Since the if condition holds true, the receiver runs a new iteration. In the second iteration, source packets #2 and #10 are recovered, resulting in a full recovery as sketched in Figure 15.



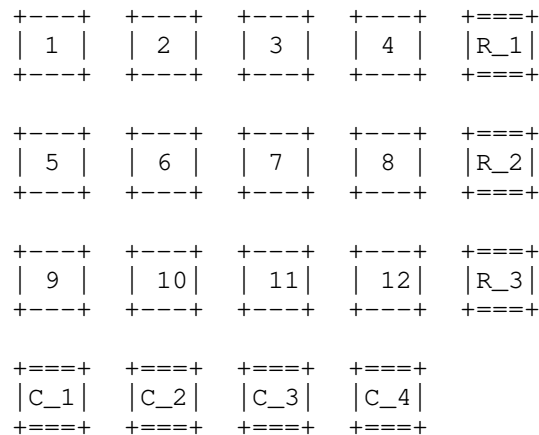


Figure 15: The resulting pattern after the second iteration

## 7. SDP Examples

This section provides two SDP [RFC4566] examples. The examples use the FEC grouping semantics defined in [RFC4756].

### 7.1. Example SDP for 1-D Parity FEC Protection

In this example, we have one source video stream (mid:S1) and one FEC repair stream (mid:R1). We form one FEC group with the "a=group:FEC S1 R1" line. The source and repair streams are sent to the same port on different multicast groups. The repair window is set to 200 ms.

```

v=0
o=ali 1122334455 1122334466 IN IP4 fec.example.com
s=1-D Interleaved Parity FEC Example
t=0 0
a=group:FEC S1 R1
m=video 30000 RTP/AVP 100
c=IN IP4 233.252.0.1/127
a=rtpmap:100 MP2T/90000
a=mid:S1
m=application 30000 RTP/AVP 110
c=IN IP4 233.252.0.2/127
a=rtpmap:110 interleaved-parityfec/90000
a=fmtp:110 L:5; D:10; ToP:0; repair-window:200000
a=mid:R1

```

## 7.2. Example SDP for 2-D Parity FEC Protection

In this example, we have one source video stream (mid:S1) and two FEC repair streams (mid:R1 and mid:R2). We form one FEC group with the "a=group:FEC S1 R1 R2" line. The source and repair streams are sent to the same port on different multicast groups. The repair window is set to 200 ms.

```
v=0
o=ali 1122334455 1122334466 IN IP4 fec.example.com
s=2-D Parity FEC Example
t=0 0
a=group:FEC S1 R1 R2
m=video 30000 RTP/AVP 100
c=IN IP4 233.252.0.1/127
a=rtpmap:100 MP2T/90000
a=mid:S1
m=application 30000 RTP/AVP 110
c=IN IP4 233.252.0.2/127
a=rtpmap:110 interleaved-parityfec/90000
a=fmtp:110 L:5; D:10; ToP:2; repair-window:200000
a=mid:R1
m=application 30000 RTP/AVP 111
c=IN IP4 233.252.0.3/127
a=rtpmap:111 non-interleaved-parityfec/90000
a=fmtp:111 L:5; D:10; ToP:2; repair-window:200000
a=mid:R2
```

Note that the sender might be generating two repair flows carrying non-interleaved and interleaved FEC packets, however the receiver might be interested only in the interleaved FEC packets. The receiver can identify the repair flow carrying the desired repair data by checking the payload types associated with each repair flow described in the SDP description.

## 8. Congestion Control Considerations

FEC is an effective approach to provide applications resiliency against packet losses. However, in networks where the congestion is a major contributor to the packet loss, the potential impacts of using FEC SHOULD be considered carefully before injecting the repair flows into the network. In particular, in bandwidth-limited networks, FEC repair flows may consume most or all of the available bandwidth and consequently may congest the network. In such cases, the applications MUST NOT arbitrarily increase the amount of FEC protection since doing so may lead to a congestion collapse. If desired, stronger FEC protection MAY be applied only after the source rate has been reduced [I-D.singh-rmcat-adaptive-fec].

In a network-friendly implementation, an application SHOULD NOT send/receive FEC repair flows if it knows that sending/receiving those FEC repair flows would not help at all in recovering the missing packets. However, it MAY still continue to use FEC if considered for bandwidth estimation instead of speculatively probe for additional capacity [Holmer13][Nagy14]. It is RECOMMENDED that the amount of FEC protection is adjusted dynamically based on the packet loss rate observed by the applications.

In multicast scenarios, it may be difficult to optimize the FEC protection per receiver. If there is a large variation among the levels of FEC protection needed by different receivers, it is RECOMMENDED that the sender offers multiple repair flows with different levels of FEC protection and the receivers join the corresponding multicast sessions to receive the repair flow(s) that is best for them.

Editor's note: Additional congestion control considerations regarding the use of 2-D parity codes should be added here.

## 9. Security Considerations

RTP packets using the payload format defined in this specification are subject to the security considerations discussed in the RTP specification [RFC3550] and in any applicable RTP profile. The main security considerations for the RTP packet carrying the RTP payload format defined within this memo are confidentiality, integrity and source authenticity. Confidentiality is achieved by encrypting the RTP payload. Integrity of the RTP packets is achieved through a suitable cryptographic integrity protection mechanism. Such a cryptographic system may also allow the authentication of the source of the payload. A suitable security mechanism for this RTP payload format should provide confidentiality, integrity protection, and at least source authentication capable of determining if an RTP packet is from a member of the RTP session.

Note that the appropriate mechanism to provide security to RTP and payloads following this memo may vary. It is dependent on the application, transport and signaling protocol employed. Therefore, a single mechanism is not sufficient, although if suitable, using the Secure Real-time Transport Protocol (SRTP) [RFC3711] is recommended. Other mechanisms that may be used are IPsec [RFC4301] and Transport Layer Security (TLS) [RFC5246] (RTP over TCP); other alternatives may exist.

## 10. IANA Considerations

New media subtypes are subject to IANA registration. For the registration of the payload formats and their parameters introduced in this document, refer to Section 5.

## 11. Acknowledgments

Some parts of this document are borrowed from [RFC5109]. Thus, the author would like to thank the editor of [RFC5109] and those who contributed to [RFC5109].

## 12. Change Log

### 12.1. draft-singh-payload-ld2d-parity-scheme-00

This is the initial version, which is based on draft-ietf-fecframe-ld2d-parity-scheme-00. The following are the major changes compared to that document:

- o Updated packet format with 16-, 48-, 112- bitmask.
- o Updated the sections on: repair packet construction, source packet construction.
- o Updated the media type registration and aligned to RFC6838.

### 12.2. draft-ietf-fecframe-ld2d-parity-scheme-00

- o Some details were added regarding the use of CNAME field.
- o Offer-Answer and Declarative Considerations sections have been completed.
- o Security Considerations section has been completed.
- o The timestamp field definition has changed.

## 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.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC3555] Casner, S. and P. Hoschka, "MIME Type Registration of RTP Payload Formats", RFC 3555, July 2003.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC4756] Li, A., "Forward Error Correction Grouping Semantics in Session Description Protocol", RFC 4756, November 2006.
- [RFC6363] Watson, M., Begen, A., and V. Roca, "Forward Error Correction (FEC) Framework", RFC 6363, October 2011.
- [RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, January 2013.
- [RFC7022] Begen, A., Perkins, C., Wing, D., and E. Rescorla, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMES)", RFC 7022, September 2013.

### 13.2. Informative References

- [Holmer13] Holmer, S., Shemer, M., and M. Paniconi, "Handling Packet Loss in WebRTC", Proc. of IEEE International Conference on Image Processing (ICIP 2013) , 9 2013.
- [I-D.singh-rmcat-adaptive-fec] Singh, V., Nagy, M., Ott, J., and L. Eggert, "Congestion Control Using FEC for Conversational Media", draft-singh-rmcat-adaptive-fec-00 (work in progress), July 2014.
- [Nagy14] Nagy, M., Singh, V., Ott, J., and L. Eggert, "Congestion Control using FEC for Conversational Multimedia Communication", Proc. of 5th ACM International Conference on Multimedia Systems (MMSys 2014) , 3 2014.
- [RFC2326] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.
- [RFC2733] Rosenberg, J. and H. Schulzrinne, "An RTP Payload Format for Generic Forward Error Correction", RFC 2733, December 1999.

- [RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, October 2000.
- [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.
- [RFC5109] Li, A., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, December 2007.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.
- [SMPTE2022-1] SMPTE 2022-1-2007, , "Forward Error Correction for Real-Time Video/Audio Transport over IP Networks", 2007.

## Authors' Addresses

Varun Singh  
Aalto University  
Espoo, FIN  
Finland

Email: varun@comnet.tkk.fi

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

Email: abegen@cisco.com

Mo Zanaty  
Cisco  
Raleigh, NC  
USA

Email: mzanaty@cisco.com

Network Working Group  
Internet-Draft

Intended status: Informational  
Expires: April 30, 2015

H. Stokking  
O. van Deventer  
R. van Brandenburg  
TNO  
F. Boronat  
M. Montagud  
Universitat Politecnica de  
Valencia  
October 27, 2014

Inter-Destination Media Synchronizatoin for IPTV Environments  
draft-stokking-avtcore-idms-for-iptv-00.txt

Abstract

[RFC7272] describes the use of RTCP for the purpose of Inter-Destination Media Synchronization (IDMS) between Synchronization Clients (SCs) and an Media Synchronization Application Server (MSAS). This document extends that work for application in the area of IPTV. First, RTCP can be used according to the single source multicast (SSM) principles from [RFC5760] in the IPTV application area. This document specifies the use of a feedback target for collecting and possibly summarizing IDMS reports. For this, the document defines 2 new sub-report blocks for the use of IDMS according to the SSM principles. Alternatively, the MSAS can be co-located with the Feedback Target, for synchronizing small groups of receivers. Secondly, in an IPTV environment, different viewers may receive the same content, but in non-identical streams. The IDMS solution presented in [RFC7272] will no longer work in such a case. This document provides a solution for this.

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].

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 30, 2015.

#### Copyright Notice

Copyright (c) 2014 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.

#### Table of Contents

1. Introduction . . . . .	3
1.1. IDMS in an IPTV environment . . . . .	3
1.1.1. IDMS for Single Source Multicast . . . . .	3
1.1.2. IDMS for different streams providing the same content . . . . .	4
2. IDMS report aggregation in SSM session . . . . .	5
2.1. IDMS Packet Received Sub-Report Block . . . . .	5
2.2. IDMS Packet Presented Sub-Report Block . . . . .	7
3. IDMS with separate synchronization server with unicast synchronization settings . . . . .	7
4. IDMS in case of multiple RTP streams . . . . .	8
5. IANA Considerations . . . . .	9
6. Security Considerations . . . . .	9
7. Acknowledgements . . . . .	9
8. Normative References . . . . .	9
Authors' Addresses . . . . .	10



## 1. Introduction

The Real-time Transport Protocol (RTP) provides a real-time transport mechanism suitable for unicast and multicast communication between multimedia applications. An important component of the RTP protocol is the control channel, defined as the RTP Control Protocol (RTCP). RTP and RTCP have been extended in numerous RFCs. Two such extensions are the extensions for Single-Source Multicast (SSM) Sessions with Unicast Feedback in [RFC5760] and the use of RTCP for Inter-Destination Media Synchronization (IDMS) in [RFC7272].

This Internet draft provides a number of extensions on the use of RTCP for IDMS in an IPTV environment. The IPTV environment has a number of characteristics that are currently not dealt with [RFC7272]. The introduction discusses the IPTV environment and identifies various gaps in the current IDMS solution. The next sections discuss solution directions for dealing with these gaps. The purpose of this Internet Draft is to build upon [RFC7272] so that the IDMS solution can be applied in an IPTV specific environment.

### 1.1. IDMS in an IPTV environment

An IPTV environment has specific characteristics, which [RFC7272] does not deal with properly. These characteristics are:

- o Single Source Multicast (SSM, [RFC5760]) setting with a large number of viewers.
- o Different receivers may receive different versions of the same content, i.e. they receive non-identical streams, e.g. different unicast streams, different encoded streams, streams from different media senders.

#### 1.1.1. IDMS for Single Source Multicast

The first characteristic of IPTV is the large-scale Single Source Multicast (SSM) setting. Regular linear television is offered using SSM. Such SSM sessions have a large number of viewers, often in the millions, which requires a highly scalable approach. Applying IDMS to such an SSM session can be done in two ways:

1. Synchronize all receivers. In this case, [RFC7272] does not offer the scalability to synchronize all viewers in such large-scale sessions. In such a case each receiver contains a synchronization client (SC) which communicates with the Media Synchronization Application Server (MSAS). [RFC5760] offers a unicast feedback system using feedback targets (FTs) to collect and possibly aggregate RTCP reports of groups of receivers. IDMS can be performed using this

same feedback system, providing more scalability. Section 2 of this document specifies how to accomplish this.

2. Synchronize independent groups of receivers, depending on the application. Use cases for synchronization, such as social TV or such as multiple receivers in a single physical location, require only a limited number of receivers to be synchronized together. For example, when millions of viewers watch the same television show, only specific groups of users viewing the show together would have to be synchronized, and only within their own group. [RFC5760] describes a system that provides all receivers with the same information. If only a limited subset of the receivers are synchronized together, not all receivers need to receive the same synchronization instructions. Section 3 of this document provides a unicast way of sending synchronization instructions to receivers, which requires the MSAS to be co-located with the Feedback Target.

The choice between options 1 and 2 depends on a number of factors. If only a limited number of receivers use a service that requires IDMS, it is inefficient to synchronize all viewers. Also, playout timing differences between various receivers can be relatively large due to e.g. variable propagation delays. If that is the case, and every receiver is synchronizing to the slowest receiver, a lot of buffering needs to be done. This is not efficient, and also significantly increases channel changing delays. In these cases, it makes sense to use option 2. If on the other hand many viewers use synchronization sensitive services, and playout timing differences are relatively small, it may make sense to synchronize all receivers by using option 1.

#### 1.1.2. IDMS for different streams providing the same content

Different viewers may watch the same content, but use a different media stream in an IPTV environment. An example of this is when one viewer receives an High Definition (HD) stream and another viewer receives an Standard Definition (SD) stream. Another example is when multiple receivers view the same video-on-demand, receiving this using different unicast streams. Services such as social TV, where different viewers remotely view media content together, require synchronization of these different streams. The IDMS solution is based on RTP timestamps. For different streams, these timestamps are not aligned, i.e. there is no relation between the timestamps in one stream and the timestamps in another stream, because of the different random offset of the RTP timestamps, as well as potential clock skew. Because of this, the MSAS cannot determine proper synchronization instructions.

There are two possible solution directions for this problem.

The first is to have the media source output the different streams with the same timing. The NTP timestamp in the RTP packets will then be synchronous, i.e. IDMS can be based on this NTP timestamp analogue to inter-stream synchronization (lip-sync). This does require the MSAS to be informed on the RTP/NTP relationships of the various streams. This information is available in the RTCP SRs of the various streams. If the MSAS is part of the media source, this is implicitly available. If this is not the case, the MSAS should receive these SRs somehow. This document presents various options to achieve this.

The other solution for this problem, is to have the media source determine and signal the relationship between the various RTP timestamps of the various streams. Again, if the MSAS is part of with the media source, then this information is locally available. If the MSAS is a separate entity, the media source can sent this information to the MSAS. Section 4 of this document shows how that is done.

## 2. IDMS report aggregation in SSM session

[RFC5760] describes how to use Feedback Targets (FTs) or the Distribution Source (DS) for summarizing RTCP packets from receivers, using the Receiver Summary Information (RSI) Packet. This section describes two new sub-report blocks, to be used in those RSI packets.

One sub-report block is for summarizing reported RTP packet received timestamps, the other is for summarizing reported RTP packet presented timestamps.

A feedback target or distribution source MUST only summarize IDMS information from SCs, if they belong to the same synchronization group, i.e. when the reports from the receivers contain the same Media Stream Correlation Identifier [RFC7272]. If at least one of the receivers in a certain synchronization group reports on both packet received timestamps and packet presented timestamps, a feedback target or distribution source SHOULD also include packet presented timestamps. If all receivers report on packet presented timestamps, a feedback target or distribution source MUST include packet presented timestamps. If a feedback target or distribution source summarizes the packet received timestamps, it SHOULD also summarize the packet presented timestamps.

### 2.1. IDMS Packet Received Sub-Report Block

To summarize the packet received timestamps in the IDMS information from SCs, a feedback target or distribution source can use the following sub-report block. The name of this sub-report block is "IDMS Received", the long name is "IDMS Packet Received NTP

Timestamp".

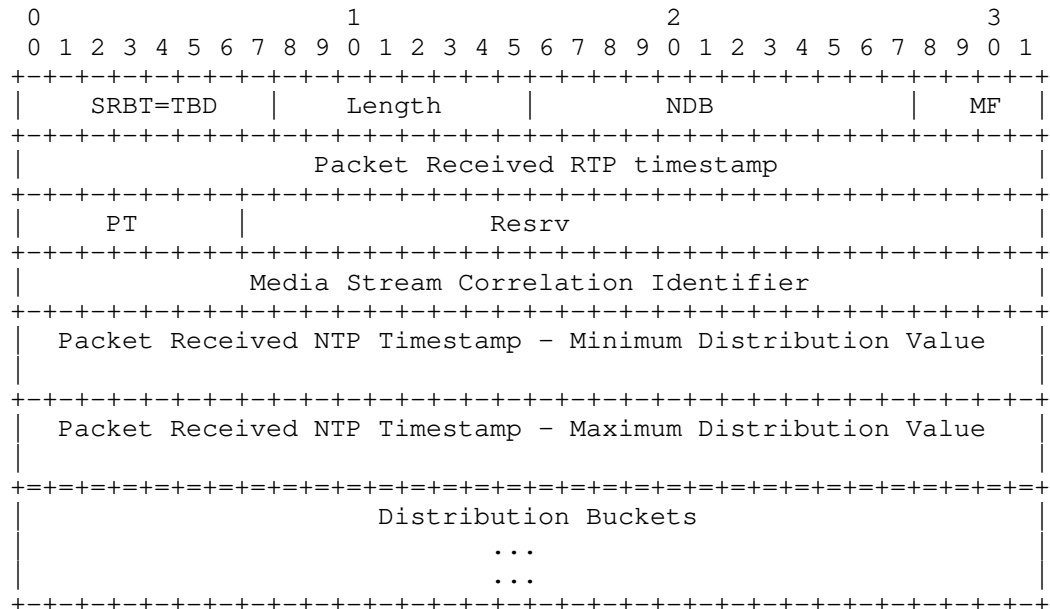


Figure 1: IDMS Packet Received Sub-Report Block

Sub-Report Block Type (SRBT): 8 bits, TBD upon IANA registration.

Length: 8 bits, the length of the sub-report in 32-bit words, as defined in RFC 5760.

Number of Distribution Buckets (NDB): 12 bits, as defined in RFC 5760, except for the calculation of the size of each bucket. Since the header is longer than the sub-report blocks defined in RFC 5760, the size of each bucket can be calculated using the formula  $((\text{length} * 4) - 32) * 8 / \text{NDB number of bits}$ .

Multiplicative Factor (MF): 4 bits, as defined in [RFC5760].

Packet Received RTP Timestamp: 32 bits, as defined in [RFC7272].

Payload Type (PT): 7 bits, as defined in [RFC7272].

Reserved Bits (Resrv): 25 bits, as defined in [RFC7272].

Media Stream Correlation Identifier: 32 bits, as defined in [RFC7272].

Packet Received NTP timestamp - Minimum Distribution Value: 64 bits, as defined in [RFC7272].

Packet Received NTP Timestamp - Maximum Distribution Value: 64 bits, as defined in [RFC7272].

Distribution Buckets: each bucket has  $((\text{Length} * 4) - 28) * 8 / \text{NDB}$  bits.

The whole sub-report block contains only a single packet received RTP timestamp value. Since various receivers will normally report on different packet received RTP timestamps, a feedback target MUST recalculate all packet received NTP timestamps to match the single packet received RTP timestamp. This will give an overview of the packet received times of all receivers for that specific RTP timestamp. This recalculation is necessary for all reported timestamps: minimum, maximum and those in the distribution buckets.

## 2.2. IDMS Packet Presented Sub-Report Block

To summarize the packet presented timestamps in the IDMS information from SCs, a feedback target or distribution source can use the following sub-report block. The name of this sub-report block is "IDMS Presented", the long name is "IDMS Packet Presented NTP Timestamp".

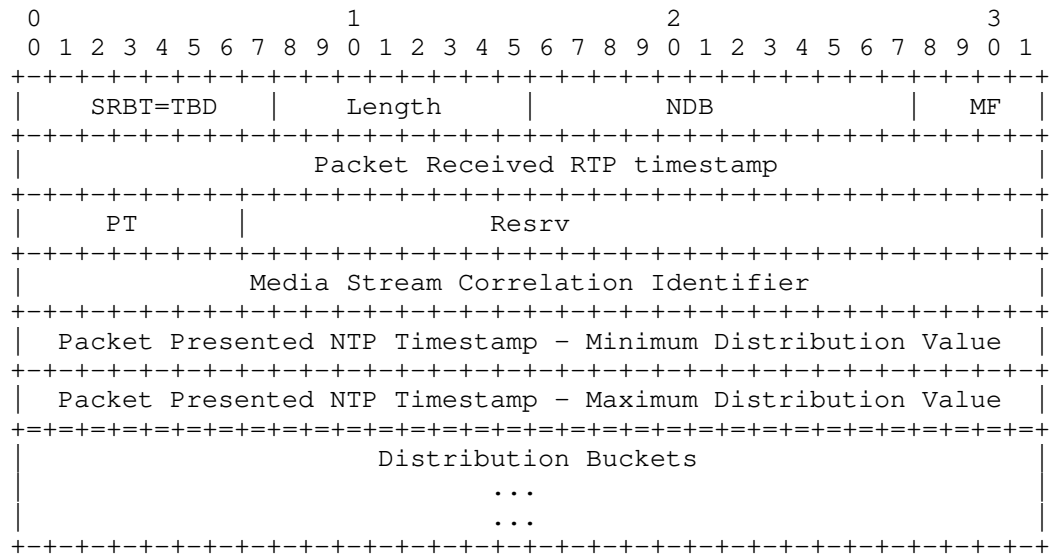


Figure 2: IDMS Packet Presented Sub-Report Block

Sub-Report Block Type (SRBT): 8 bits, TBD upon IANA registration.

Length: 8 bits, the length of the sub-report in 32-bit words, as defined in RFC 5760.

Number of Distribution Buckets (NDB): 12 bits, as defined in RFC 5760, except for the calculation of the size of each bucket. Since the header is longer than of the sub-report blocks defined in RFC 5760, the size of each bucket can be calculated using the formula  $((\text{length} * 4) - 28) * 8 / \text{NDB number of bits}$ .

Multiplicative Factor (MF): 4 bits, as defined in [RFC5760].

Packet Received RTP Timestamp: 32 bits, as defined in [RFC7272].

Payload Type (PT): 7 bits, as defined in [RFC7272].

Reserved Bits (Resrv): 25 bits, as defined in [RFC7272].

Media Stream Correlation Identifier: 32 bits, as defined in [RFC7272].

Packet Presented NTP timestamp - Minimum Distribution Value: 32 bits, as defined in [RFC7272].

Packet Presented NTP Timestamp - Maximum Distribution Value: 32 bits,

as defined in [RFC7272].

Distribution Buckets: each bucket has  $((\text{Length} * 4) - 28) * 8 / \text{NDB}$  bits.

The whole sub-report block contains only a single packet received RTP timestamp value. Since various receivers will report on different packet presented NTP timestamps, a feedback target MUST recalculate all packet presented NTP timestamps to match the single packet received RTP timestamp. This is true for all reported timestamps: minimum, maximum and those in the distribution buckets.

### 3. IDMS with separate MSAS with unicast synchronization settings

The second alternative for IDMS in a large-scale SSM context is to synchronize small groups of receivers that need to be synchronized with each other because of the service requirements. Different groups may still receive the same RTP streams, but can be synchronized independent of each other. In that case, each group MUST receive its own synchronization settings instructions in the form of IDMS Settings Packets as defined in [RFC7272]. Normally the Receiver Reports (RRs) or the Received Summary Information (RSI) is sent to all receivers. But, since different groups of receivers may need different synchronization settings instructions, these instructions cannot be multicast. As it happens, multicasting all instructions would lead to a situation where all receivers would receive a multitude of different settings instructions. They would have to find their own instructions based on the MSCSI of their group, which is possible. But, with a large number of groups, this would be highly inefficient. This is why a unicast method is taken here.

To unicast the instructions to the various SCs, the MSAS needs to directly receive the IDMS reports from the various SCs. This means the MSAS MUST be co-located with the feedback target. When supplying SCs with the unicast address to which they should send their reports, different SCs in the same synchronization group MUST be allocated the same feedback target. Also, because the synchronization information is no longer relevant upstream of the MSAS, the feedback target SHOULD terminate these RTCP blocks and not forward them or summarize them.

How the receivers receive the unicast address of the feedback target is out of scope of this draft. [RFC5760] only defines pre-configuration for this. Alternatively, the RTCP-attribute as specified in [RFC3605] can be used on the session level to provide receivers of a shared session with the unicast address of the MSAS, similar to how this is done in [TS183063].

Synchronization settings instructions MUST be sent by the MSAS to the source IP addresses of the received synchronization information, using the same destination port as the received synchronization information.

#### 4. IDMS in case of multiple RTP streams

IDMS is based on various receivers reporting on the packet received times or packet presentation times. This document describes situations in which the MSAS is not part of the media source. If all receivers receive the exact same RTP streams, e.g. in case of multiple receivers of a single multicast streams, this will work fine. The MSAS can relate the various received IDMS information. Even if different receivers report on different RTP timestamps, the MSAS can calculate the timing differences between clients by extrapolation using the RTP clock frequency derived from the reported payload type.

However, when the MSAS is not co-located with the media source and the receivers receive the same content in different RTP streams, an MSAS cannot perform the necessary calculations for achieving synchronization. To perform these calculations, there has to exist some common timeline in the reports by the various receivers. To determine a common timeline, the MSAS needs some kind of information correlating the RTP timestamps in the various streams. This section provides two alternatives for this.

The first alternative is to use the RTCP Sender Reports that belong to the various RTP streams. In these SRs, the RTP timestamps are linked to the employed wallclock time. This is normally used for intra- and inter-media synchronization. The media sources need to ensure that the same part of the content in different streams corresponds to the same wallclock time (NTP timestamp in the SR). This way the SRs of the various RTP streams can be used to establish a common timeline between those RTP streams. The easiest way to send the SR to the MSAS is by having the receiver append it to its report blocks. Another option is to have the MSAS act as a third party monitor, as described in [RFC3550].

The second alternative is that the media source sends information on the correlation of the various timestamps to the MSAS. This can be done by using the IDMS report block from [RFC7272], using the Synchronization Packet Sender Types (SPST) 3 and 4, as specified in [TS183063] Annex W, and registered with IANA in the IDMS XR Block SPST Registry. These SPSTs are normally used for synchronization in case a transcoder is changing the media stream such that the RTP timestamps also change. In this case, synchronization would be impossible between users receiving the original stream and users



receiving the transcoded version. A transcoder can link an incoming RTP timestamps (SPST=3) to an outgoing RTP timestamp (SPST=4), and thus enable correlating the timelines. These SPSTs 3 and 4 can also be used if one source sends out two version of the same content, linking the timestamps of one stream to those of the other stream.

## 5. IANA Considerations

This document defines two new RSI Sub-Report Blocks, the "IDMS Received" and the "IDMS Presented". Based on the specification in section 2, these two sub-report blocks are added to the IANA registry for Sub-Report Block Type (SRBT) Values for the RSI Packet, as part of the RTP parameters registration.

## 6. Security Considerations

The content of this ID does not pose any security risks above or beyond those mentioned in [RFC5760] and [RFC7272].

## 7. Acknowledgements

## 8. 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.
- [RFC3605] Huitema, C., "Real Time Control Protocol (RTCP) attribute in Session Description Protocol (SDP)", RFC 3605, October 2003.
- [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.
- [RFC7272] Brandenburg, R. van, Stokking, H., Deventer, O. van, Boronat, F., Montagud, M., Gross, K., "Inter-Destination Media Synchronization (IDMS) Using the RTP Control Protocol (RTCP)", RFC 7272, June 2014.
- [TS183063] ETSI, "Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN); IMS-based IPTV stage 3 specification", TS 183 063 v3.6.1, August

2014.

Authors' Addresses

Hans Stokking  
TNO  
Brassersplein 2  
Delft, 2292CH  
The Netherlands

Phone: 0031-(0)88-86 67 278  
Fax:  
Email: hans.stokking@tno.nl  
URI:

Oskar van Deventer  
TNO  
Brassersplein 2  
Delft, 2292CH  
The Netherlands

Phone: 0031-(0)88-86 67 078  
Fax:  
Email: oskar.vandeventer@tno.nl  
URI:

Ray van Brandenburg  
TNO  
Brassersplein 2  
Delft, 2292CH  
The Netherlands

Phone: 0031-(0)88-86 63 609  
Fax:  
Email: ray.vanbrandenburg@tno.nl  
URI:

Fernando Boronat  
Universitat Politecnica de Valencia  
IGIC Institute, Universitat Politecnica de Valencia-Campus de Gandia  
(UPV), C/ Paraninfo, 1, Grao de Gandia Valencia, 46730  
Spain

Phone: +34 962 849 341  
Fax:

Email: fboronat@dcom.upv.es  
URI:

Mario Montagud  
Universitat Politecnica de Valencia  
IGIC Institute, Universitat Politecnica de Valencia-Campus de Gandia  
(UPV), C/ Paraninfo, 1, Grao de Gandia Valencia, 46730  
Spain

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