

AVTCORE Working Group
Internet-Draft
Updates: 3550 (if approved)
Intended status: Standards Track
Expires: August 18, 2014

C. S. Perkins
University of Glasgow
V. Singh
Aalto University
February 14, 2014

Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions
draft-ietf-avtccore-rtp-circuit-breakers-05

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 August 18, 2014.

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	7
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	12
4.5. Ceasing Transmission	13
5. RTP Circuit Breakers for Systems Using the RTP/AVPF Profile	13
6. Impact of RTCP XR	14
7. Impact of RTCP Reporting Groups	15
8. Impact of Explicit Congestion Notification (ECN)	15
9. Security Considerations	15
10. IANA Considerations	16
11. Acknowledgements	16
12. References	16
12.1. Normative References	16
12.2. Informative References	16
Authors' Addresses	18

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, 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 partway 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] in place of the more common RTP/AVP profile [RFC3551]. This modifies the RTCP timing rules to allow RTCP reports to be sent early, in some cases immediately, provided the average RTCP reporting interval remains unchanged. It also defines new transport-layer feedback messages, including negative acknowledgements (NACKs), that can be used to report on specific congestion events. The use of the RTP/AVPF profile is dependent on signalling, but is otherwise generally backwards compatible with the RTP/AVP profile, as it keeps the same average RTCP reporting interval as the base RTP specification. The 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. The dynamics of how rapidly feedback can be sent are unchanged.
- 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 which 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 far 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 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 two 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 (i.e., the sender receives at least three RTCP SR or RR packets that report the same value in the extended highest sequence number received field for an SSRC, but the sender has sent RTP data packets for that SSRC that would have caused an increase in the reported value of the extended highest sequence number received if they had reached the receiver), then that sender SHOULD cease transmission (see Section 4.5).

The reason for waiting for two 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 two 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.

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 RTCP reporting intervals SHOULD cease transmission (see Section 4.5). When calculating the timeout, the fixed minimum RTCP reporting interval SHOULD be used (based on the rationale 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, this will force the receiver to generate RTCP SR or RR packets in a round-robin manner. In this case, the sender MAY treat receipt of an RTCP SR or RR packet corresponding to an SSRC it sent using the same 5-tuple of source and destination IP address, port, and protocol, as an indication that the receiver and return path are working to prevent 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]:

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

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, this formula can be approximated as follows with reasonable accuracy [Mathis]:

$$X = \frac{s}{R * \sqrt{p^2/3}}$$

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, not caring about when the 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 two 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).

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. 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 matters (note that RTCP requires all participants in a session to have similar reporting intervals, else the participant timeout rules in [RFC3550] will not work).

As in Section 4.1, we use two 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, applications SHOULD monitor the reported packet loss and delay to estimate whether the media is suitable for the intended purpose. If the packet loss rate and/or latency is such that the media has become unusable for the application, and has remained unusable for a significant time period, then the application SHOULD cease transmission. 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.

If the extension for Reduced-Size RTCP [RFC5506] is not used, early RTCP feedback packets sent according to the RTP/AVPF profile will be compound RTCP packets that include an RTCP SR/RR packet. That RTCP SR/RR packet MUST be processed as if it were sent as a regular RTCP report 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.

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 RTP circuit breaker (they do not contain the information used by the circuit breaker algorithm). Reduced-size RTCP reports sent under the RTP/AVPF early feedback rules that contain RTCP SR or RR packets MUST be processed 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 an 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 early 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 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. 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, for example using the Secure RTP profile [RFC3711].

10. IANA Considerations

There are no actions for IANA.

11. Acknowledgements

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

12. References

12.1. 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.

12.2. 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-01 (work
in progress), January 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.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K.
Norrman, "The Secure Real-time Transport Protocol (SRTP)",
RFC 3711, March 2004.
- [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.
- [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.
- [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.
- [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@csp Perkins .org

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

Email: varun@comnet.tkk.fi
URI: <http://www.netlab.tkk.fi/~varun/>

AVTCORE
Internet-Draft
Updates: 3550, 4585 (if approved)
Intended status: Standards Track
Expires: August 18, 2014

J. Lennox
Vidyo
M. Westerlund
Ericsson
Q. Wu
Huawei
C. Perkins
University of Glasgow
February 14, 2014

Sending Multiple Media Streams in a Single RTP Session
draft-ietf-avtccore-rtp-multi-stream-03

Abstract

This document expands and clarifies the behavior of the Real-Time Transport Protocol (RTP) endpoints when they are using multiple synchronization sources (SSRCs), e.g. for sending multiple media streams, in a single RTP session. In particular, issues involving RTCP Control Protocol (RTCP) messages are described.

This document updates RFC 3550 in regards to handling of multiple SSRCs per endpoint in RTP sessions. It also updates RFC 4585 to clarify the calculation of the timeout of SSRCs and the inclusion of feedback messages.

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 August 18, 2014.

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. Terminology	3
3. Use Cases For Multi-Stream Endpoints	4
3.1. Multiple-Capturer Endpoints	4
3.2. Multi-Media Sessions	4
3.3. Multi-Stream Mixers	4
3.4. Multiple SSRCs for a Single Media Source	5
4. Multi-Stream Endpoint RTP Media Recommendations	5
5. Multi-Stream Endpoint RTCP Recommendations	5
5.1. RTCP Reporting Requirement	5
5.2. Initial Reporting Interval	6
5.3. Compound RTCP Packets	6
5.3.1. Maintaining AVG_RTCP_SIZE	7
5.3.2. Scheduling RTCP with Multiple Reporting SSRCs	8
5.4. RTP/AVPF Feedback Packets	10
5.4.1. The SSRC Used	10
5.4.2. Scheduling a Feedback Packet	11
6. RTCP Considerations for Streams with Disparate Rates	12
6.1. Timing out SSRCs	13
6.2. Tuning RTCP transmissions	14
6.2.1. RTP/AVP and RTP/SAVP	14
6.2.2. RT/AVPF and RTP/SAVPF	16
7. Security Considerations	17
8. Open Issues	17
9. IANA Considerations	18
10. References	18
10.1. Normative References	18
10.2. Informative References	18
Appendix A. Changes From Earlier Versions	19
A.1. Changes From WG Draft -02	20
A.2. Changes From WG Draft -01	20

A.3. Changes From WG Draft -00	20
A.4. Changes From Individual Draft -02	20
A.5. Changes From Individual Draft -01	20
A.6. Changes From Individual Draft -00	21
Authors' Addresses	21

1. Introduction

At the time The Real-Time Transport Protocol (RTP) [RFC3550] was originally written, and for quite some time after, endpoints in RTP sessions typically only transmitted a single media stream, and thus used a single synchronization source (SSRC) per RTP session, where separate RTP sessions were typically used for each distinct media type.

Recently, however, a number of scenarios have emerged (discussed further in Section 3) in which endpoints wish to send multiple RTP media streams, distinguished by distinct RTP synchronization source (SSRC) identifiers, in a single RTP session. Although RTP's initial design did consider such scenarios, the specification was not consistently written with such use cases in mind. The specifications are thus somewhat unclear.

The purpose of this document is to expand and clarify [RFC3550]'s language for these use cases. The authors believe this does not result in any major normative changes to the RTP specification, however this document defines how the RTP specification is to be interpreted. In these cases, this document updates RFC3550. The document also updates RFC 4585 in regards to the timeout of inactive SSRCs as specified in Section 6.1 as well as clarifying the inclusion of feedback messages.

The document starts with terminology and some use cases where multiple sources will occur. This is followed by RTP and RTCP recommendations to resolve issues. Next are security considerations and remaining open issues.

2. Terminology

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

3. Use Cases For Multi-Stream Endpoints

This section discusses several use cases that have motivated the development of endpoints that send RTP data using multiple SSRCs in a single RTP session.

3.1. Multiple-Capturer Endpoints

The most straightforward motivation for an endpoint to send multiple RTP streams in a session is the scenario where an endpoint has multiple capture devices, and thus media sources, of the same media type and characteristics. For example, telepresence endpoints, of the type described by the CLUE Telepresence Framework [I-D.ietf-clue-framework], often have multiple cameras or microphones covering various areas of a room.

3.2. Multi-Media Sessions

Recent work has been done in RTP [I-D.ietf-avtcore-multi-media-rtp-session] and SDP [I-D.ietf-mmusic-sdp-bundle-negotiation] to update RTP's historical assumption that media sources of different media types would always be sent on different RTP sessions. In this work, a single endpoint's audio and video RTP media streams (for example) are instead sent in a single RTP session.

3.3. Multi-Stream Mixers

There are several RTP topologies which can involve a central device that itself generates multiple RTP media streams in a session.

One example is a mixer providing centralized compositing for a multi-capture scenario like that described in Section 3.1. In this case, the centralized node is behaving much like a multi-capturer endpoint, generating several similar and related sources.

More complicated is the Selective Forwarding Middlebox, see Section 3.7 of [I-D.ietf-avtcore-rtp-topologies-update]. This is a middlebox that receives media streams from several endpoints, and then selectively forwards modified versions of some of the streams toward the other endpoints it is connected to. Toward one destination, a separate media source appears in the session for every other source connected to the middlebox, "projected" from the original streams, but at any given time many of them can appear to be inactive (and thus are receivers, not senders, in RTP). This sort of device is closer to being an RTP mixer than an RTP translator, in that it terminates RTCP reporting about the mixed streams, and it can re-write SSRCs, timestamps, and sequence numbers, as well as the

contents of the RTP payloads, and can turn sources on and off at will without appearing to be generating packet loss. Each projected stream will typically preserve its original RTCP source description (SDS) information.

3.4. Multiple SSRCs for a Single Media Source

There are also several cases where a single media source results in the usage of multiple SSRCs within the same RTP session. Transport robustification tools like RTP Retransmission [RFC4588] result in multiple SSRCs, one with source data, and another with the repair data. Scalable encoders and their RTP payload formats, like H.264's extension for Scalable Video Coding(SVC) [RFC6190] can be transmitted in a configuration where the scalable layers are distributed over multiple SSRCs within the same session, to enable RTP packet stream level (SSRC) selection and routing in conferencing middleboxes.

4. Multi-Stream Endpoint RTP Media Recommendations

While an endpoint MUST (of course) stay within its share of the available session bandwidth, as determined by signalling and congestion control, this need not be applied independently or uniformly to each media stream and its SSRCs. In particular, session bandwidth MAY be reallocated among an endpoint's SSRCs, for example by varying the bandwidth use of a variable-rate codec, or changing the codec used by the media stream, up to the constraints of the session's negotiated (or declared) codecs. This includes enabling or disabling media streams and their redundancy streams as more or less bandwidth becomes available.

5. Multi-Stream Endpoint RTCP Recommendations

This section contains a number of different RTCP clarifications or recommendations that enables more efficient and simpler behavior without loss of functionality.

The RTP Control Protocol (RTCP) is defined in Section 6 of [RFC3550], but it is largely documented in terms of "participants". In many cases, the specification's recommendations for "participants" are to be interpreted as applying to individual SSRCs, rather than to endpoints. This section describes several concrete cases where this applies.

5.1. RTCP Reporting Requirement

For each of an endpoint's SSRCs, whether or not they are currently sending media, SR/RR and SDS packets MUST be sent at least once per RTCP report interval. (For discussion of the content of SR or RR

packets' reception statistic reports, see
[I-D.ietf-avtcore-rtp-multi-stream-optimisation].)

5.2. Initial Reporting Interval

When a new SSRC is added to a unicast session, the sentence in [RFC3550]'s Section 6.2 applies: "For unicast sessions ... the delay before sending the initial compound RTCP packet MAY be zero." This applies to individual SSRCs as well. Thus, endpoints MAY send an initial RTCP packet for an SSRC immediately upon adding it to a unicast session.

This allowance also applies, as written, when initially joining a unicast session. However, in this case some caution needs to be exercised if the end-point or mixer has a large number of sources (SSRCs) as this can create a significant burst. How big an issue this is depends on the number of sources for which the initial SR or RR packets and Session Description CNAME items are to be sent, in relation to the RTCP bandwidth.

(tbd: Maybe some recommendation here? The aim in restricting this to unicast sessions was to avoid this burst of traffic, which the usual RTCP timing and reconsideration rules will prevent.)

5.3. Compound RTCP Packets

Section 6.1 in [RFC3550] gives the following advice to RTP translators and mixers:

"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). An example RTCP compound packet as might be produced by a mixer is shown in Fig. 1. If the overall length of a compound packet would exceed the MTU of the network path, it SHOULD be segmented into multiple shorter compound packets to be transmitted in separate packets of the underlying protocol. This does not impair the RTCP bandwidth estimation because each compound packet represents at least one distinct participant. Note that each of the compound packets MUST begin with an SR or RR packet."

Note: To avoid confusion, an RTCP packet is an individual item, such as a Sender Report (SR), Receiver Report (RR), Source Description (SDS), Goodbye (BYE), Application Defined (APP), Feedback [RFC4585] or Extended Report (XR) [RFC3611] packet. A compound packet is the combination of two or more such RTCP

packets where the first packet has to be an SR or an RR packet, and which contains a SDES packet containing an CNAME item.

The above results in compound RTCP packets that contain multiple SR or RR packets from different sources (SSRCs) as well as any of the other packet types. There are no restrictions on the order in which the packets can occur within the compound packet, except the regular compound rule, i.e., starting with an SR or RR.

This advice applies to multi-media-stream endpoints as well, with the same restrictions and considerations. (Note, however, that the last sentence does not apply to AVPF [RFC4585] or SAVPF [RFC5124] feedback packets if Reduced-Size RTCP [RFC5506] is in use.)

5.3.1. Maintaining AVG_RTCP_SIZE

When multiple local SSRCs are sending their RTCP packets in the same compound packet, this obviously results in larger RTCP compound packets. This will have an affect on the value of the average RTCP packet size metering (`avg_rtcp_size`) that is done for the purpose of RTCP transmission scheduling calculation. This section discusses the impact of this and provide recommendations with how to deal with it.

This section will use the concept of an 'RTCP Compound Packet' to represent not just proper RTCP compound packets, i.e. ones that start with an SR or RR RTCP packet and include at least one SDES CNAME item. For the purpose of the below calculation, other valid lower layer datagram units an RTCP implementation can send or receive, independently if they are an aggregate or not of RTCP packets are also considered. This especially includes Reduced-Size RTCP packets [RFC5506].

The RTCP packet scheduling algorithm that is defined in RTP [RFC3550] deals with individual SSRCs. These SSRCs transmit their set of RTCP packets at each scheduled interval. Thus, to maintain this per-SSRC property of the scheduling, the `avg_rtcp_size` needs to be updated with per-SSRC average RTCP compound packet sizes. The `avg_rtcp_size` value SHALL be updated for each received or sent RTCP compound packet with the total size (including packet overhead such as IP/UDP) divided by the number of reporting SSRCs. The number of reporting SSRCs SHALL be determined by counting the number of different SSRCs that are the source of Sender Report (SR) or Receiver Report (RR) RTCP packets within the compound. A non-compound RTCP packet, i.e. it contains no SR or RR RTCP packets at all -- as can happen with Reduced-Size RTCP packets [RFC5506] -- the SSRC count SHALL be considered to be 1.

Note: The above makes it possible to amortize the packet overhead between the number of SSRCs sharing a RTCP compound packet.

For an RTCP end-point that doesn't follow the above rule, and instead uses the full RTCP compound packet size as input, the average RTCP reporting interval will be scaled up (i.e. become longer) with a factor that is proportional to the number of SSRCs sourcing RTCP packets in an RTCP compound packet as well as the set of SSRCs being aggregated in proportion to the total number of participants. This factor can quite easily become larger than 5, e.g. with an 1500 byte MTU and an average per-SSRC sum of RTCP packets of 240 bytes, the MTU will fit 6 packets. If the receiver end-point has a single SSRC and all other endpoints fill their MTU fully, the factor will be close to 6. If the RTCP configuration is such that the transmission interval is bandwidth limited, rather than any type of minimal interval limitation (T_{min} or T_{RR_INT}), then the other end-points will likely time out this SSRC due to it using an regular RTCP interval is more than 5 times the rest of the endpoints.

5.3.2. Scheduling RTCP with Multiple Reporting SSRCs

When implementing RTCP packet scheduling for cases where multiple reporting SSRCs are aggregating their RTCP packets in the same compound packet there are a number of challenges. First of all, we have the goal of not changing the general properties of the RTCP packet transmissions, which include the general inter-packet distribution, and the behavior for dealing with flash joins as well as other dynamic events.

The below specified mechanism deals with:

- o That one can't have a-priori knowledge about which RTCP packets are to be sent, or their size, prior to generating the packets. In which case, the time from generation to transmission ought to be as short as possible to minimize the information that becomes stale.
- o That one has an MTU limit, that one ought to avoid exceeding, as that requires lower-layer fragmentation (e.g., IP fragmentation) which impacts the packets' probability of reaching the receiver(s).

Schedule all the endpoint's local SSRCs individually for transmission using the regular calculation of T_n for the profile being used. Each time a SSRC's T_n timer expires, do the regular reconsideration. If the reconsideration indicates that an RTCP packet is to be sent:

1. Consider if an additional SSRC can be added. That consideration is done by picking the SSRC which has the Tn value closest in time to now (Tc).
2. Calculate how much space for RTCP packets would be needed to add that SSRC.
3. If the considered SSRC's RTCP Packets fit within the lower layer datagram's Maximum Transmission Unit, taking the necessary protocol headers into account and the consumed space by prior SSRCs, then add that SSRC's RTCP packets to the compound packet and go again to Step 1.
4. If the considered SSRC's RTCP Packets will not fit within the compound packet, then transmit the generated compound packet.
5. Update the RTCP Parameters for each SSRC that has been included in the sent RTCP packet. The Tp value for each SSRC MUST be updated as follows:

For the first SSRC: As this SSRC was the one that was reconsidered the tp value is set to the tc as defined in RTP [RFC3550].

For any additional SSRC: The tp value SHALL be set to the transmission time this SSRC would have had it not been aggregated and given the current existing session context. This value is derived by taking this SSRC's Tn value and performing reconsideration and updating tn until $tp + T \leq tn$. Then set tp to this tn value.

6. For the sent SSRCs calculate new tn values based on the updated parameters and reschedule the timers.

Reverse reconsideration needs to be performed as specified in RTP [RFC3550]. It is important to note that under the above algorithm when performing reconsideration, the value of tp can actually be larger than tc. However, that still has the desired effect of proportionally pulling the tp value towards tc (as well as tn) as the group size shrinks in direct proportion the reduced group size.

The above algorithm has been shown in simulations to maintain the inter-RTCP-packet transmission distribution for the SSRCs and consume the same amount of bandwidth as non-aggregated packets in RTP sessions with static sets of participants. With this algorithm the actual transmission interval for any SSRC triggering an RTCP compound packet transmission is following the regular transmission rules. It also handles the cases where the number of SSRCs that can be included

in an aggregated packet varies. An SSRC that previously was aggregated and fails to fit in a packet still has its own transmission scheduled according to normal rules. Thus, it will trigger a transmission in due time, or the SSRC will be included in another aggregate.

The algorithm's behavior under SSRC group size changes is under investigation. However, it is expected to be well behaved based on the following analyses.

RTP sessions where the number of SSRC are growing: When the group size is growing, the T_d values grow in proportion to the number of new SSRCs in the group. The reconsideration when the timer for the t_n expires, that SSRC will reconsider the transmission and with a certain probability reschedule the t_n timer. This part of the reconsideration algorithm is only impacted by the above algorithm by having t_p values that are in the future instead of set to the time of the actual last transmission at the time of updating t_p . Thus the scheduling causes in worst case a plateau effect for that SSRC. That effect depends on how far into the future t_p can advance.

RTP sessions where the number of SSRC are shrinking: When the group shrinks, reverse reconsideration moves the t_p and t_n values towards t_c proportionally to the number of SSRCs that leave the session compared to the total number of participants when they left. Thus the also group size reductions need to be handled.

In general the potential issue that might exist depends on how far into the future the t_p value can drift compared to the actual packet transmissions that occur. That drift can only occur for an SSRC that never is the trigger for RTCP packet transmission and always gets aggregated and where the calculated packet transmission interval randomly occurs so that $t_n - t_p$ for this SSRC is on average larger than the ones that gets transmitted.

5.4. RTP/AVPF Feedback Packets

This section discusses the transmission of RTP/AVPF feedback packets when the transmitting endpoint has multiple SSRCs.

5.4.1. The SSRC Used

When an RTP endpoint has multiple SSRCs, it can make certain choices on which SSRC to use as the source of an RTCP Feedback Packet. This sub-section discusses some considerations of this.

- o The media type of the media the SSRC transmits is actually not a relevant factor when considering if an SSRC can transmit a particular Feedback message.
- o Feedback messages which are Notification or Indications regarding the endpoint's own RTP packet stream need to be sent using the SSRC transmitting the media it relates to. This also includes notifications that are related to a received request or command.
- o The SSRC used to send feedback messages has a role as either a media sender or a receiver. The bandwidth pools can be different for SSRCs that are senders and receivers. Thus feedback messages that expect to be more frequent can be sent from an SSRC that has the better possibility of sending frequent RTCP compound packets or reduced size packets. This also affects the consideration if the SSRC can be used in immediate mode or not.
- o Some Feedback Types requires consistency in the sender. For example TMMBR, if one sets a limitation, the same SSRC needs to be the one that increases it. Others can simply benefit from having this property.

Note that the source of the feedback RTCP packet does not need to be any of the sources (SSRC) including SR/RR packets in a compound packet. For Reduced-Size RTCP [RFC5506] the aggregation of feedback messages from multiple sources are not limited, beyond the consideration in Section 4.2.2 of [RFC5506].

5.4.2. Scheduling a Feedback Packet

When an SSRC has a need to transmit a feedback packet in early mode it follows the scheduling rules defined in Section 3.5 in RTP/AVPF [RFC4585]. When following these rules the following clarifications need to be taken into account:

- o That a session is considered to be point-to-point or multiparty not based on the number of SSRCs, but the number of endpoints directly seen in the RTP session by the endpoint. tbd: Clarify what is considered to "see" an endpoint?
- o Note that when checking if there is already a scheduled compound RTCP packet containing feedback messages (Step 2 in Section 3.5.2), that check is done considering all local SSRCs.

TBD: The above does not allow an SSRC that is unable to send either an early or regular RTCP packet with the feedback message within the `T_max_fb_delay` to trigger another SSRC to send an early packet to which it could piggyback. Nor does it allow feedback to piggyback on

even regular RTCP packet transmissions that occur within `T_max_fb_delay`. A question is if either of these behaviours ought to be allowed.

The latter appears simple and straight forward. Instead of discarding a FB message in step 4a: alternative 2, one could place such messages in a cache with a discard time equal to `T_max_fb_delay`, and in case any of the SSRCs schedule an RTCP packet for transmission within that time, it includes this message.

The former case can have more widespread impact on the application, and possibly also on the RTCP bandwidth consumption as it allows for more massive bursts of RTCP packets. Still, on a time scale of a regular reporting interval, it ought to have no effect on the RTCP bandwidth as the extra feedback messages increase the `avg_rtcp_size`.

6. RTCP Considerations for Streams with Disparate Rates

It is possible for a single RTP session to carry streams of greatly differing bandwidth. There are two scenarios where this can occur. The first is when a single RTP session carries multiple flows of the same media type, but with very different quality; for example a video switching multi-point conference unit might send a full rate high-definition video stream of the active speaker but only thumbnails for the other participants, all sent in a single RTP session. The second scenario occurs when audio and video flows are sent in a single RTP session, as discussed in [I-D.ietf-avtcore-multi-media-rtp-session].

An RTP session has a single set of parameters that configure the session bandwidth, the RTCP sender and receiver fractions (e.g., via the SDP "b=RR:" and "b=RS:" lines), and the parameters of the RTP/AVPF profile [RFC4585] (e.g., `trr-int`) if that profile (or its secure extension, RTP/SAVPF [RFC5124]) is used. As a consequence, the RTCP reporting interval will be the same for every SSRC in an RTP session. This uniform RTCP reporting interval can result in RTCP reports being sent more often than is considered desirable for a particular media type. For example, if an audio flow is multiplexed with a high quality video flow where the session bandwidth is configured to match the video bandwidth, this can result in the RTCP packets having a greater bandwidth allocation than the audio data rate. If the reduced minimum RTCP interval described in Section 6.2 of [RFC3550] is used in the session, which might be appropriate for video where rapid feedback is wanted, the audio sources could be expected to send RTCP packets more often than they send audio data packets. This is most likely undesirable, and while the mismatch can be reduced through careful tuning of the RTCP parameters, particularly `trr_int` in RTP/AVPF sessions, it is inherent in the design of the RTCP timing

rules, and affects all RTP sessions containing flows with mismatched bandwidth.

Having multiple media types in one RTP session also results in more SSRCs being present in this RTP session. This increasing the amount of cross reporting between the SSRCs. From an RTCP perspective, two RTP sessions with half the number of SSRCs in each will be slightly more efficient. If someone needs either the higher efficiency due to the lesser number of SSRCs or the fact that one can't tailor RTCP usage per media type, they need to use independent RTP sessions.

When it comes to configuring RTCP the need for regular periodic reporting needs to be weighted against any feedback or control messages being sent. Applications using RTP/AVPF or RTP/SAVPF are RECOMMENDED to consider setting the trr-int parameter to a value suitable for the application's needs, thus potentially reducing the need for regular reporting and thus releasing more bandwidth for use for feedback or control.

Another aspect of an RTP session with multiple media types is that the RTCP packets, RTCP Feedback Messages, or RTCP XR metrics used might not be applicable to all media types. Instead, all RTP/RTCP endpoints need to correlate the media type of the SSRC being referenced in a message or packet and only use those that apply to that particular SSRC and its media type. Signalling solutions might have shortcomings when it comes to indicating that a particular set of RTCP reports or feedback messages only apply to a particular media type within an RTP session.

6.1. Timing out SSRCs

All SSRCs used in an RTP session MUST use the same timeout behaviour to avoid premature timeouts. This will depend on the RTP profile and its configuration. The RTP specification provides several options that can influence the values used when calculating the time interval. To avoid interoperability issues when using this specification, this document makes several clarifications to the calculations.

For RTP/AVP, RTP/SAVP, RTP/AVPF, and RTP/SAVPF with $T_{rr_interval} = 0$, the timeout interval SHALL be calculated using a multiplier of 5, i.e. the timeout interval becomes $5 \cdot T_d$. The T_d calculation SHALL be done using a T_{min} value of 5 seconds, not the reduced minimal interval even if used to calculate RTCP packet transmission intervals. If using either the RTP/AVPF or RTP/SAVPF profiles with $T_{rr_interval} \neq 0$ then the calculation as specified in Section 3.5.4 of RFC 4585 SHALL be used with a multiplier of 5, i.e. T_{min} in the T_d calculation is the $T_{rr_interval}$.

If endpoints implementing the RTP/AVP and RTP/AVPF profiles (or their secure variants) are combined in a single RTP session, and the RTP/AVPF endpoints use a non-zero `T_rr_interval` that is significantly lower than 5 seconds, then there is a risk that the RTP/AVPF endpoints will prematurely timeout the RTP/AVP SSRCs due to their different RTCP timeout intervals. Conversely, if the RTP/AVPF endpoints use a `T_rr_interval` that is significant larger than 5 seconds, there is a risk that the RTP/AVP endpoints will timeout the RTP/AVPF SSRCs. If such mixed RTP profiles are used, (though this is NOT RECOMMENDED), the RTP/AVPF session SHOULD use a non-zero `T_rr_interval` that is 4 seconds.

Note: It might appear strange to use a `T_rr_interval` of 4 seconds. It might be intuitive that this value ought to be 5 seconds, as then both the RTP/AVP and RTP/AVPF would use the same timeout period. However, considering regular RTCP transmission and their packet intervals for RTP/AVPF its mean value will (with non-zero `T_rr_interval`) be larger than `T_rr_interval` due to the scheduling algorithm. Thus, to enable an equal amount of regular RTCP transmissions in each directions between RTP/AVP and RTP/AVPF endpoints, taking the altered timeout intervals into account, the optimal value is around four (4), where almost four transmissions will on average occur in each direction between the different profile types given an otherwise good configuration of parameters in regards to `T_rr_interval`. If the RTCP bandwidth parameters are selected so that T_d based on bandwidth is close to 4, i.e. close to `T_rr_interval` the risk increases that RTP/AVPF SSRCs will be timed out by RTP/AVP endpoints, as the RTP/AVPF SSRC might only manage two transmissions in the timeout period.

6.2. Tuning RTCP transmissions

This sub-section discusses what tuning can be done to reduce the downsides of the shared RTCP packet intervals. First, it is considered what possibilities exist for the RTP/AVP [RFC3551] profile, then what additional tools are provided by RTP/AVPF [RFC4585].

6.2.1. RTP/AVP and RTP/SAVP

When using the RTP/AVP or RTP/SAVP profiles the tuning one can do is very limited. The controls one has are limited to the RTCP bandwidth values and whether the minimum RTCP interval is scaled according to the bandwidth. As the scheduling algorithm includes both random factors and reconsideration, one can't simply calculate the expected average transmission interval using the formula for T_d . But it does indicate the important factors affecting the transmission interval, namely the RTCP bandwidth available for the role (Active Sender or Participant), the average RTCP packet size, and the number of SSRCs

classified in the relevant role. Note that if the ratio of senders to total number of session participants is larger than the ratio of RTCP bandwidth for senders in relation to the total RTCP bandwidth, then senders and receivers are treated together.

Let's start with some basic observations:

- a. Unless the scaled minimum RTCP interval is used, then T_d prior to randomization and reconsideration can never be less than 5 seconds (assuming default T_{min} of 5 seconds).
- b. If the scaled minimum RTCP interval is used, T_d can become as low as 360 divided by RTP Session bandwidth in kilobits. In SDP the RTP session bandwidth is signalled using $b=AS$. An RTP Session bandwidth of 72 kbps results in T_{min} being 5 seconds. An RTP session bandwidth of 360 kbps of course gives a T_{min} of 1 second, and to achieve a T_{min} equal to once every frame for a 25 Hz video stream requires an RTP session bandwidth of 9 Mbps! (The use of the RTP/AVPF or RTP/SAVPF profile allows a smaller T_{min} , and hence more frequent RTCP reports, as discussed below).
- c. Let's calculate the number (n) of SSRCs in the RTP session that 5% of the session bandwidth can support to yield a T_d value equal to T_{min} with minimal scaling. For this calculation we have to make two assumptions. The first is that we will consider most or all SSRC being senders, resulting in everyone sharing the available bandwidth. Secondly we will select an average RTCP packet size. This packet will consist of an SR, containing ($n-1$) report blocks up to 31 report blocks, and an SDES item with at least a CNAME (17 bytes in size) in it. Such a basic packet will be 800 bytes for $n \geq 32$. With these parameters, and as the bandwidth goes up the time interval is proportionally decreased (due to minimal scaling), thus all the example bandwidths 72 kbps, 360 kbps and 9 Mbps all support 9 SSRCs.
- d. The actual transmission interval for a T_d value is $[0.5 \cdot T_d / 1.21828, 1.5 \cdot T_d / 1.21828]$, which means that for $T_d = 5$ seconds, the interval is actually $[2.052, 6.156]$ and the distribution is not uniform, but rather exponentially-increasing. The probability for sending at time X , given it is within the interval, is probability of picking X in the interval times the probability to randomly picking a number that is $\leq X$ within the interval with an uniform probability distribution. This results in that the majority of the probability mass is above the T_d value.

To conclude, with RTP/AVP and RTP/SAVP the key limitation for small unicast sessions is going to be the T_{min} value. Thus the RTP session bandwidth configured in RTCP has to be sufficiently high to reach the

reporting goals the application has following the rules for the scaled minimal RTCP interval.

6.2.2. RT/AVPF and RTP/SAVPF

When using RTP/AVPF or RTP/SAVPF we get a quite powerful additional tool, the setting of the `T_rr_interval` which has several effects on the RTCP reporting. First of all as `Tmin` is set to 0 after the initial transmission, the regular reporting interval is instead determined by the regular bandwidth based calculation and the `T_rr_interval`. This has the effect that we are no longer restricted by the minimal interval or even the scaling rule for the minimal rule. Instead the RTCP bandwidth and the `T_rr_interval` are the governing factors.

Now it also becomes important to separate between the application's need for regular reports and RTCP feedback packet types. In both regular RTCP mode, as in Early RTCP Mode, the usage of the `T_rr_interval` prevents regular RTCP packets, i.e. packets without any Feedback packets, to be sent more often than `T_rr_interval`. This value is applied to prevent any regular RTCP packet to be sent less than `T_rr_interval` times a uniformly distributed random value from the interval `[0.5,1.5]` after the previous regular packet packet. The random value recalculated after each regular RTCP packet transmission.

So applications that have a use for feedback packets for some media streams, for example video streams, but don't want frequent regular reporting for audio, could configure the `T_rr_interval` to a value so that the regular reporting for both audio and video is at a level that is considered acceptable for the audio. They could then use feedback packets, which will include RTCP SR/RR packets, unless reduced-size RTCP feedback packets [RFC5506] are used, and can include other report information in addition to the feedback packet that needs to be sent. That way the available RTCP bandwidth can be focused for the use which provides the most utility for the application.

Using `T_rr_interval` still requires one to determine suitable values for the RTCP bandwidth value, in fact it might make it even more important, as this is more likely to affect the RTCP behaviour and performance than when using RTP/AVP, as there are fewer limitations affecting the RTCP transmission.

When using `T_rr_interval`, i.e. having it be non zero, there are configurations that have to be avoided. If the resulting `Td` value is smaller but close to `T_rr_interval` then the interval in which the actual regular RTCP packet transmission falls into becomes very

large, from 0.5 times $T_{rr_interval}$ up to 2.73 times the $T_{rr_interval}$. Therefore for configuration where one intends to have T_d smaller than $T_{rr_interval}$, then T_d is RECOMMENDED to be targeted at values less than 1/4th of $T_{rr_interval}$ which results in that the range becomes $[0.5 * T_{rr_interval}, 1.81 * T_{rr_interval}]$.

With RTP/AVPF, using a $T_{rr_interval}$ of 0 or with another low value significantly lower than T_d still has utility, and different behaviour compared to RTP/AVP. This avoids the T_{min} limitations of RTP/AVP, thus allowing more frequent regular RTCP reporting. In fact this will result that the RTCP traffic becomes as high as the configured values.

(tbd: a future version of this memo will include examples of how to choose RTCP parameters for common scenarios)

There exists no method within the specification for using different regular RTCP reporting intervals depending on the media type or individual media stream.

7. Security Considerations

In the secure RTP protocol (SRTP) [RFC3711], the cryptographic context of a compound SRTCP packet is the SSRC of the sender of the first RTCP (sub-)packet. This could matter in some cases, especially for keying mechanisms such as Mikey [RFC3830] which allow use of per-SSRC keying.

Other than that, the standard security considerations of RTP apply; sending multiple media streams from a single endpoint does not appear to have different security consequences than sending the same number of streams.

8. Open Issues

At this stage this document contains a number of open issues. The below list tries to summarize the issues:

1. Do we need to provide a recommendation for unicast session joiners with many sources to not use 0 initial minimal interval from bit-rate burst perspective?
2. RTCP parameters for common scenarios in Section 6.2?
3. Is scheduling algorithm working well with dynamic changes?

4. Are the scheduling algorithm changes impacting previous implementations in such a way that the report aggregation has to be agreed on, and thus needs to be considered as an optimization?
5. An open question is if any improvements or clarifications ought to be allowed regarding FB message scheduling in multi-SSRC endpoints.

9. IANA Considerations

No IANA actions needed.

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.
- [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.
- [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.
- [RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, February 2008.
- [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.

10.2. Informative References

- [I-D.ietf-avtcore-multi-media-rtp-session] Westerlund, M., Perkins, C., and J. Lennox, "Sending Multiple Types of Media in a Single RTP Session", draft-ietf-avtcore-multi-media-rtp-session-04 (work in progress), January 2014.

- [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-01 (work
in progress), January 2014.
- [I-D.ietf-avtcore-rtp-topologies-update]
Westerlund, M. and S. Wenger, "RTP Topologies", draft-
ietf-avtcore-rtp-topologies-update-01 (work in progress),
October 2013.
- [I-D.ietf-clue-framework]
Duckworth, M., Pepperell, A., and S. Wenger, "Framework
for Telepresence Multi-Streams", draft-ietf-clue-
framework-14 (work in progress), February 2014.
- [I-D.ietf-mmusic-sdp-bundle-negotiation]
Holmberg, C., Alvestrand, H., and C. Jennings,
"Multiplexing Negotiation Using Session Description
Protocol (SDP) Port Numbers", draft-ietf-mmusic-sdp-
bundle-negotiation-05 (work in progress), October 2013.
- [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.
- [RFC3830] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and K.
Norrman, "MIKEY: Multimedia Internet KEYing", RFC 3830,
August 2004.
- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R.
Hakenberg, "RTP Retransmission Payload Format", RFC 4588,
July 2006.
- [RFC6190] Wenger, S., Wang, Y., Schierl, T., and A. Eleftheriadis,
"RTP Payload Format for Scalable Video Coding", RFC 6190,
May 2011.

Appendix A. Changes From Earlier Versions

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

A.1. Changes From WG Draft -02

- o Changed usage of Media Stream
- o Added Updates RFC 4585
- o Added rules for how to deal with RTCP when aggregating multiple SSRCs report in same compound packet:
 - * avg_rtcp_size calculation
 - * Scheduling rules to maintain timing
- o Started a section clarifying and discussing RTP/AVPF Feedback Packets and their scheduling.

A.2. Changes From WG Draft -01

- o None, a keep-alive version

A.3. Changes From WG Draft -00

- o Split the Reporting Group Extension from this draft into draft-ietf-avtcore-rtp-multi-stream-optimization-00.
- o Added RTCP tuning considerations from draft-ietf-avtcore-multi-media-rtp-session-02.

A.4. Changes From Individual Draft -02

- o Resubmitted as working group draft.
- o Updated references.

A.5. Changes From Individual Draft -01

- o Merged with draft-wu-avtcore-multisrc-endpoint-adver.
- o Changed how Reporting Groups are indicated in RTCP, to make it clear which source(s) is the group's reporting sources.
- o Clarified the rules for when sources can be placed in the same reporting group.
- o Clarified that mixers and translators need to pass reporting group SDES information if they are forwarding RR and SR traffic from members of a reporting group.

A.6. Changes From Individual Draft -00

- o Added the Reporting Group semantic to explicitly indicate which sources come from a single endpoint, rather than leaving it implicit.
- o Specified that Reporting Group semantics (as they now are) apply to AVPF and XR, as well as to RR/SR report blocks.
- o Added a description of the cascaded source-projecting mixer, along with a calculation of its RTCP overhead if reporting groups are not in use.
- o Gave some guidance on how the flexibility of RTCP randomization allows some freedom in RTCP multiplexing.
- o Clarified the language of several of the recommendations.
- o Added an open issue discussing how avg_rtcp_size ought to be calculated for multiplexed RTCP.
- o Added an open issue discussing how RTCP bandwidths are to be chosen for sessions where source bandwidths greatly differ.

Authors' Addresses

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

Email: jonathan@vidyo.com

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

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

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

Email: sunseawq@huawei.com

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

Email: csp@csp Perkins.org

AVTCORE WG
Internet-Draft
Intended status: Standards Track
Expires: January 02, 2015

J. Lennox
Vidyo
M. Westerlund
Ericsson
Q. Wu
Huawei
C. Perkins
University of Glasgow
July 01, 2014

Sending Multiple Media Streams in a Single RTP Session: Grouping RTCP
Reception Statistics and Other Feedback
draft-ietf-avtccore-rtp-multi-stream-optimisation-03

Abstract

RTP allows multiple media streams to be sent in a single session, but requires each Synchronisation Source (SSRC) to send RTCP reception quality reports for every other SSRC visible in the session. This causes the number of RTCP reception reports to grow with the number of SSRCs, rather than the number of endpoints. In many cases most of these RTCP reception reports are unnecessary, since all SSRCs of an endpoint are co-located and see the same reception quality. This memo defines a Reporting Group extension to RTCP to reduce the reporting overhead in such scenarios.

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 January 02, 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. RTCP Reporting Groups	3
3.1. Semantics and Behaviour of RTCP Reporting Groups	3
3.2. Identifying Members of an RTCP Reporting Group	5
3.2.1. Definition and Use of the RTCP RGRP SDES Item	5
3.2.2. Definition and Use of the RTCP RGRS Packet	6
3.3. Interactions with the RTP/AVPF Feedback Profile	8
3.4. Interactions with RTCP Extended Report (XR) Packets	9
3.5. Middlebox Considerations	9
3.6. SDP Signalling for Reporting Groups	10
4. Properties of RTCP Reporting Groups	10
4.1. Bandwidth Benefits of RTCP Reporting Groups	11
4.2. Compatibility of RTCP Reporting Groups	11
5. Security Considerations	12
6. IANA Considerations	14
7. References	15
7.1. Normative References	15
7.2. Informative References	15
Authors' Addresses	16

1. Introduction

The Real-time Transport Protocol (RTP) [RFC3550] is a protocol for group communication, supporting multiparty multimedia sessions. A single RTP session can support multiple participants sending at once, and can also support participants sending multiple simultaneous media streams. Examples of the latter might include a participant with multiple cameras who chooses to send multiple views of a scene, or a participant that sends audio and video flows multiplexed in a single RTP session. Rules for handling RTP sessions containing multiple media streams are described in [RFC3550] with some clarifications in [I-D.ietf-avtcore-rtp-multi-stream].

An RTP endpoint will have one or more synchronisation sources (SSRCs) that send media streams. It will have at least one SSRC for each

media stream it sends, and might use multiple SSRCs when using media scalability features [RFC6190], forward error correction, RTP retransmission [RFC4588], or similar mechanisms. An endpoint that is not sending any media streams, will have at least one SSRC to use for reporting and any feedback messages. Each SSRC has to send RTCP sender reports corresponding to the RTP packets it sends, and receiver reports for traffic it receives. That is, every SSRC will send RTCP packets to report on every other SSRC. This rule is simple, but can be quite inefficient for endpoints that send large numbers of media streams in a single RTP session. Consider a session comprising ten participants, each sending three media streams with their own SSRC. There will be 30 SSRCs in such an RTP session, and 30 RTCP reception reports will be sent per reporting interval as each SSRC reports on all the others. However, the three SSRCs comprising each participant will almost certainly see identical reception quality, since they are co-located. If there was a way to indicate that several SSRCs are co-located, and see the same reception quality, then two-thirds of those RTCP reports could be suppressed. This would allow the remaining RTCP reports to be sent more often, while keeping within the same RTCP bandwidth fraction.

This memo defines such an RTCP extension, RTCP Reporting Groups. This extension is used to indicate the SSRCs that originate from the same endpoint, and therefore have identical reception quality, hence allowing the endpoints to suppress unnecessary RTCP reception quality reports.

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 [RFC2119].

3. RTCP Reporting Groups

An RTCP Reporting Group is a set of synchronization sources (SSRCs) that are co-located at a single endpoint (which could be an end host or a middlebox) in an RTP session. Since they are co-located, every SSRC in the RTCP reporting group will have an identical view of the network conditions, and see the same lost packets, jitter, etc. This allows a single representative to send RTCP reception quality reports on behalf of the rest of the reporting group, reducing the number of RTCP packets that need to be sent without loss of information.

3.1. Semantics and Behaviour of RTCP Reporting Groups

A group of co-located SSRCs that see identical network conditions can form an RTCP reporting group. If reporting groups are in use, an RTP

endpoint with multiple SSRCs MAY put those SSRCs into a reporting group if their view of the network is identical; i.e., if they report on traffic received at the same interface of an RTP endpoint. SSRCs with different views of the network MUST NOT be put into the same reporting group.

An endpoint that has combined its SSRCs into an RTCP reporting group will choose one (or a subset) of those SSRCs as a "reporting source" for that RTCP reporting group. A reporting source will send RTCP SR/RR reception quality reports on behalf of the other members of the RTCP reporting group. A reporting source MUST suppress the RTCP SR/RR reports that relate to other members of the reporting group, and only report on remote SSRCs. The other members (non reporting sources) of the RTCP reporting group will suppress their RTCP reception quality reports, and instead send an RTCP RGRS packet (see Section 3.2.2) to indicate that they are part of an RTCP reporting group and give the SSRCs of the reporting sources.

If there are large numbers of remote SSRCs in the RTP session, then the reception quality reports generated by the reporting source might grow too large to fit into a single compound RTCP packet, forcing the reporting source to use a round-robin policy to determine what remote SSRCs it includes in each compound RTCP packet, and so reducing the frequency of reports on each SSRC. To avoid this, in sessions with large numbers of remote SSRCs, an RTCP reporting group MAY use more than one reporting source. If several SSRCs are acting as reporting sources for an RTCP reporting group, then each reporting source MUST have non-overlapping sets of remote SSRCs it reports on.

An endpoint SHOULD NOT create an RTCP reporting group that comprises only a single local SSRC (i.e., an RTCP reporting group where the reporting source is the only member of the group), unless it is anticipated that the group might have additional SSRCs added to it in the future.

If a reporting source leaves the RTP session (i.e., if it sends a RTCP BYE packet, or leaves the session without sending BYE under the rules of [RFC3550] section 6.3.7), the remaining members of the RTCP reporting group MUST either (a) have another reporting source, if existing, report on the remote SSRCs the leaving SSRC reported on, (b) choose a new reporting source, or (c) disband the RTCP reporting group and begin sending reception quality reports following [RFC3550] and [I-D.ietf-avtcore-rtp-multi-stream].

The RTCP timing rules assign different bandwidth fractions to senders and receivers. This lets senders transmit RTCP reception quality reports more often than receivers. If a reporting source in an RTCP reporting group is a receiver, but one or more non-reporting SSRCs in

the RTCP reporting group are senders, then the endpoint MAY treat the reporting source as a sender for the purpose of RTCP bandwidth allocation, increasing its RTCP bandwidth allocation, provided it also treats one of the senders as if it were a receiver and makes the corresponding reduction in RTCP bandwidth for that SSRC.

3.2. Identifying Members of an RTCP Reporting Group

When RTCP Reporting Groups are in use, the other SSRCs in the RTP session need to be able to identify which SSRCs are members of an RTCP reporting group. Two RTCP extensions are defined to support this: the RTCP RGRP SDES item is used by the reporting source(s) to identify an RTCP reporting group, and the RTCP RGRS packet is used by other members of an RTCP reporting group to identify the reporting source(s).

3.2.1. Definition and Use of the RTCP RGRP SDES Item

A new RTCP SDES item is defined to identify an RTCP reporting group. The motivation for giving a reporting group an identify is to ensure that the RTCP reporting group and its member SSRCs can be correctly associated when there are multiple reporting sources, and to ensure that a reporting SSRC can be associated with the correct reporting group if an SSRC collision occurs.

The RTCP Source Description (SDES) RGRP item is defined. The RTCP SDES RGRP item MUST be sent by the reporting sources in a reporting group, and MUST NOT be sent by other members of the reporting group or by SSRCs that are not members of any RTCP reporting group. Specifically, every reporting source in an RTCP reporting group MUST include an RTCP SDES packet containing an RGRP item in every compound RTCP packet in which it sends an RR or SR packet (i.e., in every RTCP packet it sends, unless Reduced-Size RTCP [RFC5506] is in use).

Syntactically, the format of the RTCP SDES RGRP item is identical to that of the RTCP SDES CNAME item [RFC7022], except that the SDES item type field MUST have value RGRP=(TBA) instead of CNAME=1. The value of the RTCP SDES RGRP item MUST be chosen with the same concerns about global uniqueness and the same privacy considerations as the RTCP SDES CNAME item. The value of the RTCP SDES RGRP item MUST be stable throughout the lifetime of the reporting group, even if the some or all of the reporting sources change their SSRC due to collisions, or if the set of reporting sources changes.

Note to RFC Editor: please replace (TBA) in the above paragraph with the RTCP SDES item type number assigned to the RGRP item, then delete this note.

An RTP mixer or translator that forwards RTCP SR or RR packets from members of a reporting group MUST forward the corresponding RTCP SDES RGRP items as well, even if it otherwise strips SDES items other than the CNAME item.

3.2.2. Definition and Use of the RTCP RGRS Packet

A new RTCP packet type is defined to allow the members of an RTCP reporting group to identify the reporting sources for that group. This allows participants in an RTP session to distinguish an SSRC that is sending empty RTCP reception reports because it is a member of an RTCP reporting group, from an SSRC that is sending empty RTCP reception reports because it is not receiving any traffic. It also explicitly identifies the reporting sources, allowing other members of the RTP session to know which SSRCs are acting as the reporting sources for an RTCP reporting group, and allowing them to detect if RTCP packets from any of the reporting sources are being lost.

The format of the RTCP RGRS packet is defined below. It comprises the fixed RTCP header that indicates the packet type and length, the SSRC of the packet sender, and a list of reporting sources for the RTCP reporting group of which the packet sender is a member.

```

      0                   1                   2                   3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|V=2|P|          SC          |PT=RGRS(TBA)|          length          |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|          SSRC of packet sender          |
+===+===+===+===+===+===+===+===+===+===+===+===+===+===+===+===+
:          List of SSRCs for the Reporting Source(s)          :
:          ...          :
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

The fields in the RTCP RGRS packet have the following definition:

version (V): This field identifies the RTP version. The current RTP version is 2.

padding (P): If set, the padding bit indicates that the RTCP packet contains additional padding octets at the end that are not part of the control information but are included in the length field. See [RFC3550].

Source Count (SC): Indicates the number of reporting source SSRCs that are included in this RTCP packet. As the RTCP RGRS packet MUST NOT be sent by reporting sources, all the SSRCs in the

list of reporting sources will be different from the SSRC of the packet sender. Every RTCP RGRS packet MUST contain at least one reporting source SSRC.

Payload type (PT): The RTCP packet type number that identifies the packet as being an RTCP RGRS packet. The RGRS RTCP packet has the value [TBA].

Note to RFC Editor: please replace [TBA] here, and in the packet format diagram above, with the RTCP packet type that IANA assigns to the RTCP RGRS packet.

Length: The length of this packet in 32-bit words minus one, including the header and any padding. This is in line with the definition of the length field used in RTCP sender and receiver reports [RFC3550]. Since all RTCP RGRS packets include at least one reporting source SSRC, the length will always be 2 or greater.

SSRC of packet sender: The SSRC of the sender of this packet.

List of SSRCs for the Reporting Source(s): A variable length size (as indicated by SC header field) of the 32 bit SSRC values of the reporting sources for the RTCP Reporting Group of which the packet sender is a member.

Every source that belongs to an RTCP reporting group but is not a reporting source MUST include an RTCP RGRS packet in every compound RTCP packet in which it sends an RR or SR packet (i.e., in every RTCP packet it sends, unless Reduced-Size RTCP [RFC5506] is in use). Each RTCP RGRS packet MUST contain the SSRC identifier of at least one reporting source. If there are more reporting sources in an RTCP reporting group than can fit into an RTCP RGRS packet, the members of that reporting group MUST send the SSRCs of the reporting sources in a round-robin fashion in consecutive RTCP RGRS packets, such that all the SSRCs of the reporting sources are included over the course of several RTCP reporting intervals.

An RTP mixer or translator that forwards RTCP SR or RR packets from members of a reporting group MUST also forward the corresponding RGRS RTCP packets. If the RTP mixer or translator rewrites SSRC values of the packets it forwards, it MUST make the corresponding changes to the RTCP RGRS packets.

3.3. Interactions with the RTP/AVPF Feedback Profile

Use of the RTP/AVPF Feedback Profile [RFC4585] allows SSRCs to send rapid RTCP feedback requests and codec control messages. If use of the RTP/AVPF profile has been negotiated in an RTP session, members of an RTCP reporting group can send rapid RTCP feedback and codec control messages following [RFC4585] and [RFC5104], as updated by Section 5.4 of [I-D.ietf-avtcore-rtp-multi-stream], and by the following considerations.

The members of an RTCP reporting group will all see identical network conditions. Accordingly, one might therefore think that it doesn't matter which SSRC in the reporting group sends the RTP/AVPF feedback or codec control messages. There might be, however, cases where the sender of the feedback/codec control message has semantic importance, or when only a subset of the members of an RTCP reporting group might want to send RTP/AVPF feedback or a codec control message in response to a particular event. For example, an RTP video sender might choose to treat packet loss feedback received from SSRCs known to be audio receivers with less urgency than feedback that it receives from video receivers when deciding what packets to retransmit, and a multimedia receiver using reporting groups might want to choose the outgoing SSRC for feedback packets to reflect this.

Each member of an RTCP reporting group SHOULD therefore send RTP/AVPF feedback/codec control messages independently of the other members of the reporting group, to respect the semantic meaning of the message sender. The suppression rules of [RFC4585] will ensure that only a single copy of each feedback packet is (typically) generated, even if several members of a reporting group send the same feedback. When an endpoint knows that several members of its RTCP reporting group will be sending identical feedback, and that the sender of the feedback is not semantically important, then that endpoint MAY choose to send all its feedback from the reporting source and deterministically suppress feedback packets generated by the other sources in the reporting group.

It is important to note that the RTP/AVPF timing rules operate on a per-SSRC basis. Using a single reporting source to send all feedback for a reporting group will hence limit the amount of feedback that can be sent to that which can be sent by one SSRC. If this limit is a problem, then the reporting group can allow each of its members to send its own feedback, using its own SSRC.

If the RTP/AVPF feedback messages or codec control requests are sent as compound RTCP packets, then those compound RTCP packets MUST include either an RTCP RGRS packet or an RTCP SDES RGRP item, depending on whether they are sent by the reporting source or a non-

reporting source in the RTCP reporting group respectively. The contents of non-compound RTCP feedback or codec control messages are not affected by the use of RTCP reporting groups.

3.4. Interactions with RTCP Extended Report (XR) Packets

When using RTCP Extended Reports (XR) [RFC3611] with RTCP reporting groups, it is RECOMMENDED that the reporting source is used to send the RTCP XR packets. If multiple reporting sources are in use, the reporting source that sends the SR/RR packets that relate to a particular remote SSRC SHOULD send the RTCP XR reports about that SSRC. This is motivated as one commonly combine the RTCP XR metrics with the regular report block to more fully understand the situation. Receiving these blocks in different compound packets reduces their value as the measuring intervals are not synchronized in those cases.

Some RTCP XR report blocks are specific to particular types of media, and might be relevant to only some members of a reporting group. For example, it would make no sense for an SSRC that is receiving video to send a VoIP metric RTCP XR report block. Such media specific RTCP XR report blocks MUST be sent by the SSRC to which they are relevant, and MUST NOT be included in the common report sent by the reporting source. This might mean that some SSRCs send RTCP XR packets in compound RTCP packets that contain an empty RTCP SR/RR packet, and that the time period covered by the RTCP XR packet is different to that covered by the RTCP SR/RR packet. If it is important that the RTCP XR packet and RTCP SR/RR packet cover the same time period, then that source SHOULD be removed from the RTCP reporting group, and send standard RTCP packets instead.

3.5. Middlebox Considerations

Many different types of middlebox are used with RTP. RTCP reporting groups are potentially relevant to those types of RTP middlebox that have their own SSRCs and generate RTCP reports for the traffic they receive. RTP middleboxes that do not have their own SSRC, and that don't send RTCP reports on the traffic they receive, cannot use the RTCP reporting groups extension, since they generate no RTCP reports to group.

An RTP middlebox that has several SSRCs of its own can use the RTCP reporting groups extension to group the RTCP reports it generates. This can occur, for example, if a middlebox is acting as an RTP mixer for both audio and video flows that are multiplexed onto a single RTP session, where the middlebox has one SSRC for the audio mixer and one for the video mixer part, and when the middlebox wants to avoid cross reporting between audio and video.

A middlebox cannot use the RTCP reporting groups extension to group RTCP packets from the SSRCs that it is forwarding. It can, however, group the RTCP packets from the SSRCs it is forwarding into compound RTCP packets following the rules in Section 6.1 of [RFC3550] and Section 5.3 of [I-D.ietf-avtcore-rtp-multi-stream]. If the middlebox is using RTCP reporting groups for its own SSRCs, it MAY include RTCP packets from the SSRCs that it is forwarding as part of the compound RTCP packets its reporting source generates.

A middlebox that forwards RTCP SR or RR packets sent by members of a reporting group MUST forward the corresponding RTCP SDES RGRP items, as described in Section 3.2.1. A middlebox that forwards RTCP SR or RR packets sent by member of a reporting group MUST also forward the corresponding RTCP RGRS packets, as described in Section 3.2.2. Failure to forward these packets can cause compatibility problems, as described in Section 4.2.

If a middlebox rewrites SSRC values in the RTP and RTCP packets that it is forwarding, then it MUST make the corresponding changes in RTCP SDES packets containing RGRP items and in RTCP RGRS packets, to allow them to be associated with the rewritten SSRCs.

3.6. SDP Signalling for Reporting Groups

This document defines the "a=rtcp-rgrp" Session Description Protocol (SDP) [RFC4566] attribute to indicate if the session participant is capable of supporting RTCP Reporting Groups for applications that use SDP for configuration of RTP sessions. A participant that proposes the use of RTCP Reporting Groups SHALL itself support the reception of RTCP Reporting Groups.

An offering client that wishes to use RTCP Reporting Groups MUST include the attribute "a=rtcp-rgrp" in the SDP offer. If "a=rtcp-rgrp" is present in the offer SDP, the answerer that supports RTCP Reporting Groups and wishes to use it SHALL include the "a=rtcp-rgrp" attribute in the answer.

In declarative usage of SDP, such as the Real Time Streaming Protocol (RTSP) [RFC2326] and the Session Announcement Protocol (SAP) [RFC2974], the presence of the attribute indicates that the session participant MAY use RTCP Reporting Groups in its RTCP transmissions.

4. Properties of RTCP Reporting Groups

This section provides additional information on what the resulting properties are with the design specified in Section 3. The content of this section is non-normative.

4.1. Bandwidth Benefits of RTCP Reporting Groups

To understand the benefits of RTCP reporting groups, consider a scenario in which the two endpoints in a session each have a hundred sources, of which eight each are sending within any given reporting interval.

For ease of analysis, we can make the simplifying approximation that the duration of the RTCP reporting interval is equal to the total size of the RTCP packets sent during an RTCP interval, divided by the RTCP bandwidth. (This will be approximately true in scenarios where the bandwidth is not so high that the minimum RTCP interval is reached.) For further simplification, we can assume RTCP senders are following the recommendations regarding Compound RTCP Packets in [I-D.ietf-avtcore-rtp-multi-stream]; thus, the per-packet transport-layer overhead will be small relative to the RTCP data. Thus, only the actual RTCP data itself need be considered.

In a report interval in this scenario, there will, as a baseline, be 200 SDES packets, 184 RR packets, and 16 SR packets. This amounts to approximately 6.5 kB of RTCP per report interval, assuming 16-byte CNAMEs and no other SDES information.

Using the original [RFC3550] everyone-reports-on-every-sender feedback rules, each of the 184 receivers will send 16 report blocks, and each of the 16 senders will send 15. This amounts to approximately 76 kB of report block traffic per interval; 92% of RTCP traffic consists of report blocks.

If reporting groups are used, however, there is only 0.4 kB of reports per interval, with no loss of useful information. Additionally, there will be (assuming 16-byte RGRPs, and a single reporting source per reporting group) an additional 2.4 kB per cycle of RGRP SDES items and RGRS packets. Put another way, the unmodified [RFC3550] reporting interval is approximately 8.9 times longer than if reporting groups are in use.

4.2. Compatibility of RTCP Reporting Groups

The RTCP traffic generated by receivers using RTCP Reporting Groups might appear, to observers unaware of these semantics, to be generated by receivers who are experiencing a network disconnection, as the non-reporting sources appear not to be receiving a given sender at all.

This could be a potentially critical problem for such a sender using RTCP for congestion control, as such a sender might think that it is sending so much traffic that it is causing complete congestion collapse.

However, such an interpretation of the session statistics would require a fairly sophisticated RTCP analysis. Any receiver of RTCP statistics which is just interested in information about itself needs to be prepared that any given reception report might not contain information about a specific media source, because reception reports in large conferences can be round-robin.

Thus, it is unclear to what extent such backward compatibility issues would actually cause trouble in practice.

5. Security Considerations

The security considerations of [RFC3550] and [I-D.ietf-avtcore-rtp-multi-stream] apply. If the RTP/AVPF profile is in use, then the security considerations of [RFC4585] (and [RFC5104], if used) also apply. If RTCP XR is used, the security consideration of [RFC3611] and any XR report blocks used also apply.

The RTCP SDES RGRP item is vulnerable to malicious modifications unless integrity protection is used. A modification of this item's length field could cause the parsing of the RTCP packet in which it is contained to fail. Depending on the implementation, parsing of the full compound RTCP packet can also fail causing the whole packet to be discarded. A modification to the value of this SDES item would make the receiver of the report think that the sender of the report was a member of a different RTCP reporting group. This will potentially create an inconsistency, when the RGRS reports the source as being in the same reporting group as another source with another reporting group identifier. What impact on a receiver implementation such inconsistencies would have are difficult to fully predict. One case is when congestion control or other adaptation mechanisms are used, an inconsistent report can result in a media sender to reduce its bit-rate. However, a direct modification of the receiver report or a feedback message itself would be a more efficient attack, and equally costly to perform.

The new RGRS RTCP Packet type is very simple. The common RTCP packet type header shares the security risks with previous RTCP packet types. Errors or modification of the length field can cause the full compound packet to fail header validation (see Appendix A.2 in [RFC3550]) resulting in the whole compound RTCP packet being discarded. Modification of the SC or P fields would cause inconsistency when processing the RTCP packet, likely resulting in

being classified as invalid. A modification of the PT field would cause the packet being interpreted under some other packet type's rules. In such case the result might be more or less predictable but packet type specific. Modification of the SSRC of packet sender would attribute this packet to another sender. Resulting in a receiver believing the reporting group applies also for this SSRC, if it exists. If it doesn't exist, unless also corresponding modifications are done on a SR/RR packet and a SDES packet the RTCP packet SHOULD be discarded. If consistent changes are done, that could be part of a resource exhaustion attack on a receiver implementation. Modification of the "List of SSRCs for the Reporting Source(s)" would change the SSRC the receiver expect to report on behalf of this SSRC. If that SSRC exist, that could potentially change the report group used for this SSRC. A change to another reporting group belonging to another endpoint is likely detectable as there would be a mismatch between the SSRC of the packet sender's endpoint information, transport addresses, SDES CNAME etc and the corresponding information from the reporting group indicated.

In general the reporting group is providing limited impacts attacks. The most significant result from an deliberate attack would be to cause the information to be discarded or be inconsistent, including discard of all RTCP packets that are modified. This causes a lack of information at any receiver entity, possibly disregarding the endpoints participation in the session.

To protect against this type of attacks from external non trusted entities, integrity and source authentication SHOULD be applied. This can be done, for example, by using SRTP [RFC3711] with appropriate key-management, other options exist as discussed in RTP Security Options [RFC7201].

The Report Group Identifier has a potential privacy impacting properties. If this would be generated by an implementation in such a way that is long term stable or predictable, it could be used for tracking a particular end-point. Therefore it is RECOMMENDED that it be generated as a short-term persistent RGRP, following the rules for short-term persistent CNAMEs in [RFC7022]. The rest of the information revealed, i.e. the SSRCs, the size of reporting group and the number of reporting sources in a reporting group is of less sensitive nature, considering that the SSRCs and the communication would anyway be revealed without this extension. By encrypting the report group extensions the SSRC values would preserved confidential, but can still be revealed if SRTP [RFC3711] is used. The size of the reporting groups and number of reporting sources are likely determinable from analysis of the packet pattern and sizes. However, this information appears to have limited value.

6. IANA Considerations

(Note to the RFC-Editor: in the following, please replace "TBA" with the IANA-assigned value, and "XXXX" with the number of this document, then delete this note)

The IANA is requested to register one new RTCP SDES items in the "RTCP SDES Item Types" registry, as follows:

Value	Abbrev	Name	Reference
TBA	RGRP	Reporting Group Identifier	[RFCXXXX]

The definition of the RTCP SDES RGRP item is given in Section 3.2.1 of this memo.

The IANA is also requested to register one new RTCP packet type in the RTCP Control Packet Types (PT) Registry as follows:

Value	Abbrev	Name	Reference
TBA	RGRS	Reporting Group Reporting Sources	[RFCXXXX]

The definition of the RTCP RGRS packet type is given in Section 3.2.2 of this memo.

The IANA is also requested to register one new SDP attribute:

SDP Attribute ("att-field"):

Attribute name:	rtcp-rgrp
Long form:	RTCP Reporting Groups
Type of name:	att-field
Type of attribute:	Media or session level
Subject to charset:	No
Purpose:	Negotiate or configure the use of the RTCP Reporting Group Extension.
Reference:	[RFCXXXX]
Values:	See [RFCXXXX]

The definition of the "a=rtcp-rgrp" SDES attribute is given in Section 3.6 of this memo.

7. References

7.1. Normative References

- [I-D.ietf-avtcore-rtp-multi-stream]
Lennox, J., Westerlund, M., Wu, W., and C. Perkins,
"Sending Multiple Media Streams in a Single RTP Session",
draft-ietf-avtcore-rtp-multi-stream-04 (work in progress),
May 2014.
- [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.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session
Description Protocol", RFC 4566, July 2006.
- [RFC7022] Begen, A., Perkins, C., Wing, D., and E. Rescorla,
"Guidelines for Choosing RTP Control Protocol (RTCP)
Canonical Names (CNAMEs)", RFC 7022, September 2013.

7.2. Informative References

- [RFC2326] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time
Streaming Protocol (RTSP)", RFC 2326, April 1998.
- [RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session
Announcement Protocol", RFC 2974, October 2000.
- [RFC3611] Friedman, T., Caceres, R., and A. Clark, "RTP Control
Protocol Extended Reports (RTCP XR)", RFC 3611, November
2003.
- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K.
Norrman, "The Secure Real-time Transport Protocol (SRTP)",
RFC 3711, March 2004.
- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey,
"Extended RTP Profile for Real-time Transport Control
Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July
2006.
- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R.
Hakenberg, "RTP Retransmission Payload Format", RFC 4588,
July 2006.

- [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.
- [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.
- [RFC6190] Wenger, S., Wang, Y.-K., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, May 2011.
- [RFC7201] Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", RFC 7201, April 2014.

Authors' Addresses

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

Email: jonathan@vidyo.com

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

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

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

Email: sunseawq@huawei.com

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

Email: csp@csp Perkins.org

AVTCORE Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 18, 2014

D. McGrew
D. Wing
F. Andreasen
Cisco
February 14, 2014

Encrypted Key Transport for Secure RTP
draft-ietf-avtcore-srtp-ekt-02

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 August 18, 2014.

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	5
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	12
2.4. Synchronizing Operation	13
2.5. Transport	13
2.6. Timing and Reliability Consideration	14
3. Use of EKT with SDP Security Descriptions	15
3.1. SDP Security Descriptions Recap	15
3.2. Relationship between EKT and SDP Security Descriptions	16
3.3. Overview of Combined EKT and SDP Security Description Operation	18
3.4. EKT Extensions to SDP Security Descriptions	18
3.4.1. EKT_Cipher	18
3.4.2. EKT_Key	19
3.4.3. EKT_SPI	19
3.5. Offer/Answer Procedures	19
3.5.1. Generating the Initial Offer - Unicast Streams	19
3.5.2. Generating the Initial Answer - Unicast Streams	20
3.5.3. Processing of the Initial Answer - Unicast Streams	21
3.6. SRTP-Specific Use Outside Offer/Answer	22
3.7. Modifying the Session	23
3.8. Backwards Compatibility Considerations	23
3.9. Grammar	24
4. Use of EKT with DTLS-SRTP Key Transport	25
4.1. DTLS-SRTP Recap	25
4.2. EKT Extensions to DTLS-SRTP	25
4.3. Offer/Answer Considerations	27
4.3.1. Generating the Initial Offer	27
4.3.2. Generating the Initial Answer	28
4.3.3. Processing the Initial Answer	28
4.3.4. Sending DTLS EKT Key Reliably	29

4.3.5. Modifying the Session	29
5. Use of EKT with MIKEY	29
5.1. EKT extensions to MIKEY	31
5.2. Offer/Answer considerations	32
5.2.1. Generating the Initial Offer	32
5.2.2. Generating the Initial Answer	33
5.2.3. Processing the Initial Answer	33
5.2.4. Modifying the Session	34
6. Using EKT for interoperability between key management systems	34
7. Design Rationale	35
7.1. Alternatives	36
8. Security Considerations	36
9. IANA Considerations	37
10. Acknowledgements	38
11. References	39
11.1. Normative References	39
11.2. Informative References	40
Appendix A. Using EKT to Optimize Interworking DTLS-SRTP with	
Security Descriptions	41
Authors' Addresses	43

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 (this document), SRTP master keys have to be always generated randomly and never shared, 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").

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.

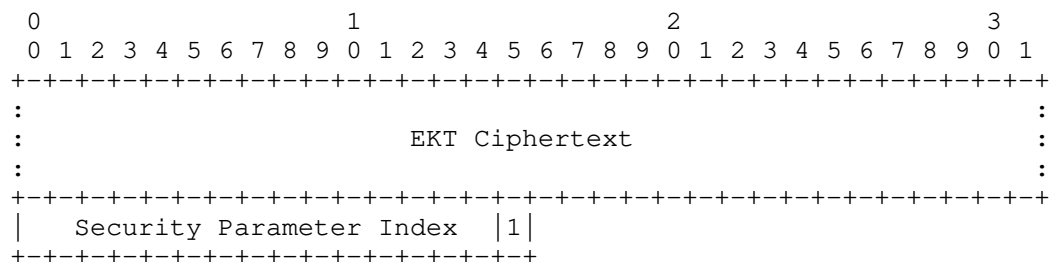


Figure 2: Full EKT Field format

```

      0 1 2 3 4 5 6 7 8
+--+--+--+--+--+--+--+
|   Reserved   |0|
+--+--+--+--+--+--+--+

```

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], 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^{32} 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 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 and because SRTP rekeying can occur without concern for RTCP transmission limits.

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 keys distributed with EKT, rather than with call signaling. This means a receiver can immediately decrypt an SRTP (or SRTCP packet) using that new key, provided the SRTP packet (or SRTCP packet) also contains the EKT key. However, a receiver cannot immediately authenticate or decrypt an SRTCP packets without the EKT key. Fortunately, the inability to authenticate or decrypt SRTCP has little immediate impact on the end user.

This section describes how to reliably and expediently deliver EKT keys to receivers.

There are three cases to consider. The first case is a new sender joining a session which needs to communicate its key to all the receivers. The second case is a sender changing its 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 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 EKT 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 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 key, which increases the chance a new receiver won't have seen the new key.

New receiver: When a new receiver joins a session it does not need to communicate its sending 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, SDES 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_S RTP
a=crypto:2 F8_128_HMAC_SHA1_80
    inline:MTIzNDU2Nzg5QUJDREUwMTIzNDU2Nzg5QUJjiiKt|2^20
    FEC_ORDER=FEC_S RTP
```

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, two SRTP master keys and associated salts. Finally, the FEC_ORDER session parameter indicates the order of Forward Error Correction used.

3.2. Relationship between EKT and SDP Security Descriptions

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 SSRCS 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 the shortcomings of SDESC.

3.3. Overview of Combined EKT and SDP Security Description Operation

We define three session extension parameters 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 following new SDESC session parameters are defined. These MUST NOT appear more than once in a given crypto attribute:

EKT_Cipher: The EKT cipher used to encrypt the SRTP Master Key

EKT_Key: The EKT key used to encrypt the SRTP Master Key

EKT_SPI: The EKT Security Parameter Index

Below are details on each of these attributes.

3.4.1. EKT_Cipher

The (optional) EKT_Cipher parameter 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. In the Offer/Answer model, the EKT_Cipher parameter is a negotiated parameter.

3.4.2. EKT_Key

The (mandatory) EKT_Key parameter is the key K 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]). In the Offer/Answer model, the EKT_Key parameter is a negotiated parameter.

3.4.3. EKT_SPI

The (mandatory) EKT_SPI parameter 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. In the Offer/Answer model, the EKT_SPI parameter is a negotiated parameter.

3.5. Offer/Answer Procedures

In this section, we provide the offer/answer procedures associated with use of the three new SDESC parameters 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 one EKT_Key parameter and one EKT_SPI parameter in at least one "crypto" attribute (see [RFC4568]). The EKT_SPI parameter serves to identify the EKT parameter set used for a particular 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.

If a given crypto attribute includes more than one set of SRTP key parameters (SRTP master key, salt, lifetime), they MUST all use the same salt. (EKT requires a single shared salt between all the participants in the direct SRTP session).

As EKT is not defined for use with MKI, the offer and the answer 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 EKT session parameters can either be included as optional or mandatory parameters, however within a given crypto attribute, they MUST all be either optional or mandatory.

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 EKT parameters. If mandatory EKT parameters are included with a "crypto" attribute, the answerer MUST support those parameters in order to accept that offered crypto attribute. If optional EKT parameters are 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 neither optional nor mandatory EKT parameters 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 mixture of optional and mandatory EKT parameters, or an incomplete set of mandatory EKT parameters, 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 offer (however, if the default EKT cipher is being used, it may be omitted). Furthermore, the EKT parameters included MUST be mandatory (i.e., no "-" prefix).

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

If the answer crypto line contains EKT parameters but the corresponding offered crypto line did not, or if the parameters 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 parameters 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 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 requirement 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 parameters, such optional parameters will simply be ignored. When the answer is received, absence of the parameters 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|1:4
  FEC_ORDER=FEC_SRTP EKT=AESKW_128|WWVzQUxvdmVseUVLVGtleQ|AAE0
a=crypto:2 F8_128_HMAC_SHA1_80
  inline:MTIzNDU2Nzg5QUJDREUwMTIzNDU2Nzg5QUJjZGVm|2^20|1:4;
  inline:QUJjZGVmMTIzNDU2Nzg5QUJDREUwMTIzNDU2Nzg5|2^20|2:4
  FEC_ORDER=FEC_SRTP EKT=AESKW_128|VHdvTG92ZWx5RUtUa2V5cw|AAE0
```

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

4. Use of EKT with DTLS-SRTP Key Transport

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 7: 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.

Client		Server
ClientHello + use_srtp + EKT	----->	
		ServerHello + use_srtp + EKT Certificate*

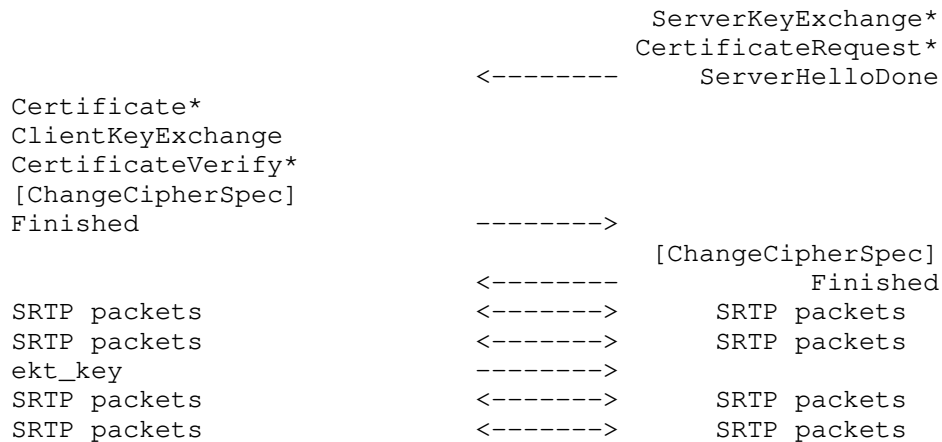


Figure 8: Handshake Message Flow

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 must be negotiated in the MIKEY message exchange.

For EKT we specify a new SRTP Policy Type in the Security Policy (SP) payload of MIKEY (see Section 6.10 of [RFC3830]). The SP payload contains a set of policies. Each policy consists of a number of Policy Param TLVs.

Prot type	Value
EKT	TBD (will be assigned by IANA) NOTE TO RFC EDITOR

Figure 9: EKT Security Policy

The EKT Security Policy has one parameter representing the EKT cipher.

Type	Meaning	Possible values
0	EKT cipher	see below

Figure 10: EKT Security Policy Parameters

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

Figure 11: EKT Cipher Parameters

The two mandatory EKT parameters (EKT_Key and EKT_SPI) are 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:

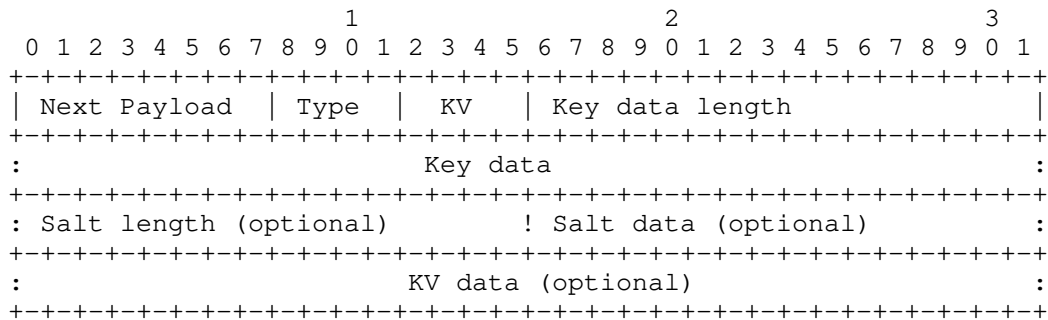


Figure 12: Key Data Sub-Payload of MIKEY

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 EKT_Cipher Security Policy payload. 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 EKT_Cipher Security Policy payload. 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 SP' or 'Invalid DT' is returned to indicate that the EKT_Cipher 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 defining an additional key-mgmt prtcl-id value 'mikeyekt' and 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'.
- o From the Security Policy table name space, a new value to be assigned for 'EKT' (see Figure 9).

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, and Rob Raymond 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 John Mattsson for his detailed security review highlighting the duplicative interaction between SRTP MKI with EKT ISN and encouraged the EKT specification to prohibit sharing SRTP master keys. Thanks to Magnus Westerlund for his review.

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.

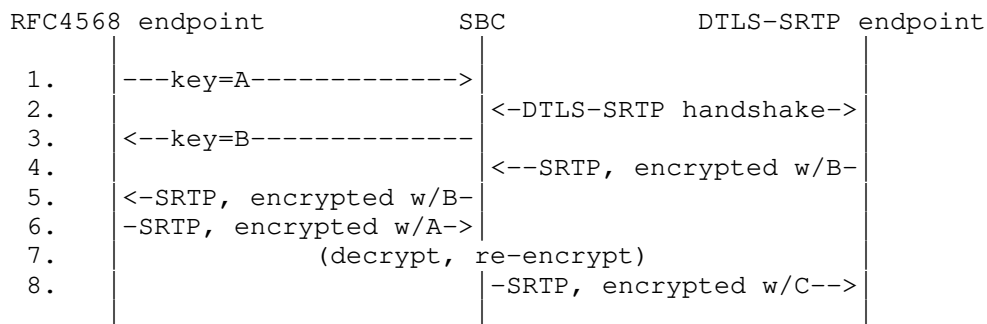


Figure 13: Interworking Security Descriptions and DTLS-SRTP

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.

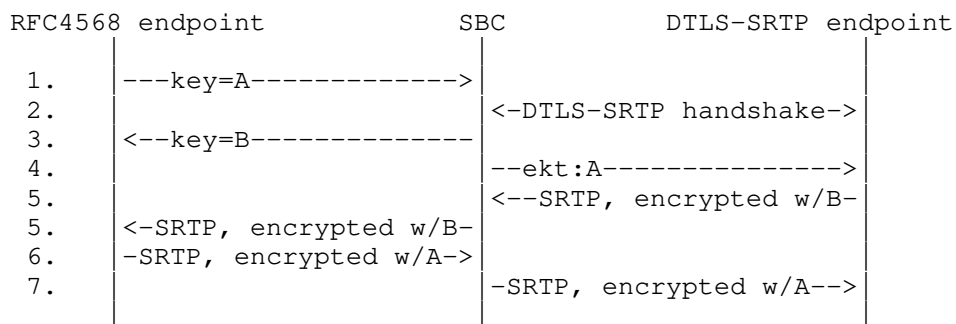


Figure 14: Interworking Security Descriptions and EKT

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

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

Phone: (408) 525 8651
Email: 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

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

Email: fandreas@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2014

M. Westerlund
B. Burman
Ericsson
S. Nandakumar
Cisco
October 22, 2013

Using Simulcast in RTP Sessions
draft-westerlund-avtcore-rtp-simulcast-03

Abstract

In some application scenarios it may be desirable to send multiple differently encoded versions of the same Media Source in independent Source Packet Streams. This is called Simulcast. This document discusses the best way of accomplishing Simulcast in RTP and how to signal it in SDP. A solution is defined by making three extensions to SDP, and using RTP/RTCP identification methods to relate RTP Source Packet Streams. The first SDP extension consists of two new session level SDP attributes that express capability to send or receive Simulcast Source Packet Streams, respectively. The second SDP extension introduces an SDP media level attribute that groups and identifies a selected set of media level parameters for a specific direction, called a media configuration. The third SDP extension describes how to group such media configurations on SDP session or media level for Simulcast purposes.

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 25, 2014.

Copyright Notice

Copyright (c) 2013 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. Terminology	3
2.2. Requirements Language	4
3. Use Cases	4
3.1. Reaching a Diverse Set of Receivers	5
3.2. Application Specific Media Source Handling	6
3.3. Receiver Adaptation in Multicast/Broadcast	7
3.4. Receiver Media Source Preferences	7
4. Requirements	8
5. Proposed Solution Overview	9
6. Proposed Signaling	10
6.1. Simulcast Capability	11
6.1.1. Declarative Use	12
6.1.2. Offer/Answer Use	12
6.2. Media Configuration	13
6.2.1. Simulcast Limitations	16
6.2.2. Declarative Use	17
6.2.3. Offer/Answer Use	17
6.3. Grouping Simulcast Configurations	18
6.3.1. Declarative Use	19
6.3.2. Offer/Answer Use	19
6.4. Relating Simulcast Versions	20
6.5. Two-Phase Negotiation	20
6.6. Signaling Examples	21
6.6.1. Unified Plan Client	21
6.6.2. Multi-Transport Client	24
6.6.3. Multi-Source Client	26
7. Network Aspects	28
8. IANA Considerations	29
9. Security Considerations	29
10. Contributors	29
11. Acknowledgements	30

12. References	30
12.1. Normative References	30
12.2. Informative References	31
Appendix A. Discussion on Receiver Diversity	32
Authors' Addresses	34

1. Introduction

Most of today's multiparty video conference solutions make use of centralized servers to reduce the bandwidth and CPU consumption in the endpoints. Those servers receive Source Packet Streams from each participant and send some suitable set of possibly modified streams to the rest of the participants, which usually have heterogeneous capabilities (screen size, CPU, bandwidth, codec, etc). One of the biggest issues is how to perform stream adaptation to different participants' constraints with the minimum possible impact on video quality and server performance.

Simulcast is the act of simultaneously sending multiple different versions of the same media content, e.g. the same video source encoded with different video encoder types or image resolutions. This can be done in several ways and for different purposes. This document focuses on the case where it is desirable to provide a Media Source as multiple Source Packet Streams over RTP [RFC3550] towards an intermediary so that the intermediary can provide the wanted functionality by selecting which Source Packet Stream to forward to other participants in the session, and more specifically how the identification and grouping of the involved Source Packet Streams are done. From an RTP perspective, Simulcast is a specific application of the aspects discussed in RTP Multiplexing Guidelines [I-D.ietf-avtcore-multiplex-guidelines].

The purpose of this document is to describe a few scenarios where it is motivated to use Simulcast, and propose a suitable solution for signaling and performing RTP Simulcast.

2. Definitions

2.1. Terminology

This document makes use of the terminology defined in RTP Taxonomy [I-D.lennox-raiarea-rtp-grouping-taxonomy]. In addition, the following terms are used:

Media Configuration: A specific set of parameter values applied on the encoding and packetization process that creates a specific Source Packet Stream. In SDP, the applicable parameter values are described by the joint set of "rtpmap" parameters, "fmtp"

parameters, and the "config-id" (Section 6.2) parameters, including extensions.

2.2. Requirements Language

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

3. Use Cases

Many use cases of Simulcast as described in this document relate to a multi-party Communication Session where one or more central nodes are used to adapt the view of the Communication Session towards individual Participants, and facilitate the Media Transport between Participants. Thus, these cases targets the RTP Mixer topology defined in [RFC5117] (Section 3.4: Topo-Mixer), further elaborated and extended with other topologies in [I-D.ietf-avtcore-rtp-topologies-update] (Section 3.6 to 3.9).

There are two principle approaches for an RTP Mixer to provide this adapted view of the Communication Session to each receiving Participant:

- o Transcoding (decoding and re-encoding) received Source Packet Streams with characteristics adapted to each receiving Participant. This often include mixing or composition of Media Sources from multiple Participants into a mixed Media Source originated by the RTP Mixer. The main advantage of this approach is that it achieves close to optimal adaptation to individual receiving Participants. The main disadvantages are that it can be very computationally expensive to the RTP Mixer and typically also degrades media Quality of Experience (QoE) such as end-to-end delay for the receiving Participants.
- o Switching a subset of all received Source Packet Streams or sub-streams to each receiving Participant, where the used subset is typically specific to each receiving Participant. The main advantages of this approach are that it is computationally cheap to the RTP Mixer and it has very limited impact on media QoE. The main disadvantage is that it can be difficult to combine a subset of received Source Packet Streams into a perfect fit to the resource situation of a receiving Participant.

The use of Simulcast is relates to the latter approach, where it is more important to reduce the load on the RTP Mixer and/or minimize QoE impact than to achieve an optimal adaptation of resource usage.

A multicast/broadcast case where the receivers themselves selects the most appropriate simulcast version and tune in to the right transport to receive that version is also considered (Section 3.3) . This enables large receiver populations with heterogeneity where it comes to capabilities and the use network paths bandwidth.

In this section, an "RTP switch" is used as a common short term for the terms "switching RTP mixer", "source projecting middlebox", and "video switching MCU" as discussed in [I-D.ietf-avtcore-rtp-topologies-update].

3.1. Reaching a Diverse Set of Receivers

The Media Sources provided by a sending Participant potentially need to reach several receiving Participants that differ in terms of available resources. A discussion on that topic is included in Appendix A. The receiver resources that typically differ include, but are not limited to:

Codec: This includes codec type (such as SDP MIME type) and can include codec configuration options (e.g. SDP fmp parameters). A couple of codec resources that differ only in codec configuration will be "different" if they are somehow not "compatible", like if they differ in video codec profile, or the transport packetization configuration.

Sampling: This relates to how the Media Source is sampled, in spatial as well as in temporal domain. For video streams, spatial sampling affects image resolution and temporal sampling affects video frame rate. For audio, spatial sampling relates to the number of audio channels and temporal sampling affects audio bandwidth. This may be used to suit different rendering capabilities or needs at the receiving endpoints, as well as a method to achieve different transport capabilities, bitrates and eventually QoE by controlling the amount of source data.

Bitrate: This relates to the amount of bits spent per second to transmit the Media Source as an Source Packet Stream, which typically also affects the Quality of Experience (QoE) for the receiving user.

Letting the sending Participant create a Simulcast of a few differently configured Source Packet Streams per Media Source can be a good trade-off when using an RTP switch as middlebox, instead of sending a single Source Packet Stream and using an RTP Mixer to create individual transcodings to each receiving Participant.

This requires that the receiving Participants can be categorized in terms of available resources and that the sending Participant can choose a matching configuration for a single Source Packet Stream per category and Media Source.

For example, assume for simplicity a set of receiving Participants that differ only in that some have support to receive Codec A, and the others have support to receive Codec B. Further assume that the sending participant can send both Codec A and B. It can then reach all receivers by creating two Simulcasted Source Packet Streams from each Media Source; one for Codec A and one for Codec B.

In another simple example, a set of receiving Participants differ only in screen resolution; some are able to display video with at most 360p resolution and some support 720p resolution. A sending Participant can then reach all receivers by creating a Simulcast of Source Packet Streams with 360p and 720p resolution for each sent video Media Source.

In more elaborate cases, the receiving Participants differ both in available Sampling and Bitrate, and maybe also Codec, and it is up to the RTP switch to find a good trade-off in which Simulcasted stream to choose for each intended receiver. It is also the responsibility of the RTP switch to negotiate a good fit of Simulcast streams with the sending Participant.

The maximum number of Simulcasted Source Packet Streams that can be sent is mainly limited by the amount of processing and uplink network resources available to the sending Participant.

3.2. Application Specific Media Source Handling

The application logic that controls the Communication Session may include special handling of some Media Sources. It is for example commonly the case that the media from a sending Participant is not sent back to itself.

It is also common that a currently active speaker Participant is shown in larger size or higher quality than other Participants (the Sampling or Bitrate aspects of Section 3.1). Not sending the active speaker media back to itself means there is some other Participant's media instead that receive special handling towards the active speaker; typically the previous active speaker. This way, the previously active speaker is needed both in larger size (to current active speaker) and in small size (to the rest of the Participants), which can be solved with a Simulcast from the previously active speaker to the RTP switch.

3.3. Receiver Adaptation in Multicast/Broadcast

When using Broadcast or Multicast technology to distribute real-time media streams to large populations of receivers there can still be significant heterogeneity among the receiver population. This can depend on several factors:

Network Bandwidth: The network paths to individual receivers will have variations in the bandwidth. Thus putting different limits on the supported bit-rates that can be received.

Endpoint Capabilities: The endpoint's hardware and software can have varying capabilities in relation to screen resolution, decoding capabilities, and supported media codecs.

To handle these variations, a transmitter of real-time media may want to apply Simulcast to its Source Packet Streams and provide a set of media configurations, enabling the receivers to select the best fit from these sets themselves. The endpoint capabilities will usually result in a single initial choice. However, the network bandwidth can vary over time, which requires a client to continuously monitor its reception to determine if the received media streams still fit within the available bandwidth. If not, another Simulcast media configuration containing a thinner set of Source Packet Streams will have to be chosen.

When one uses IP multicast, the level of Simulcast granularity that the receiver can select from is by choosing different multicast addresses. Thus, different Simulcast versions need to be put on different Media Transports using different multicast addresses. If these Simulcast versions are described using SDP, they need to be part of different SDP media descriptions, as SDP binds to transport on media description level. To enable more than the initial choice to function well, there is a need to enable correct mapping of Source Packet Streams in one Simulcast media configuration to a corresponding Source Packet Stream in another Simulcast media configuration on another multicast group.

3.4. Receiver Media Source Preferences

The application logic that controls the Communication Session may allow receiving Participants to apply preferences to the characteristics of the Source Packet Stream they receive, for example in terms of the aspects listed in Section 3.1. Sending a Simulcast of Source Packet Streams is one way of accommodating receivers with conflicting or otherwise incompatible preferences.

4. Requirements

The following requirements need to be met to support the use cases in previous sections:

REQ-1: Identification. It must be possible to identify a set of simulcasted Source Packet Streams as originating from the same Media Source:

REQ-1.1: In SDP signaling.

REQ-1.2: On RTP/RTCP level.

REQ-2: Transport usage. The solution must work when distributing different Simulcast versions on:

REQ-2.1: Same Media Transport and RTP session.

REQ-2.2: Different Media Transports and RTP sessions.

REQ-3: Capability negotiation. It must be possible that:

REQ-3.1: Sender can express capability of sending simulcast.

REQ-3.2: Receiver can express capability of receiving simulcast.

REQ-3.3: Sender can express maximum number of Simulcast versions that can be provided.

REQ-3.4: Receiver can express maximum number of Simulcast versions that can be received.

REQ-3.5: Sender can detail the characteristics of the Simulcast versions that can be provided.

REQ-3.6: Receiver can detail the characteristics of the Simulcast versions that it prefers to receive.

REQ-4: Distinguishing features. It must be possible to have different Simulcast versions use different values for any combination of:

REQ-4.1: Codec. This includes both codec type and configuration options for both codec and RTP packetization. It also includes different layers from a scalable codec, but only as long as those layers are possible to identify on RTP level.

REQ-4.2: Bitrate of Source Packet Stream.

REQ-4.3: Sampling in spatial as well as in temporal domain.

REQ-5: Compatibility. It must be possible to use Simulcast in combination with other RTP mechanisms that generate additional Source Packet Streams:

REQ-5.1: RTP Retransmission [RFC4588].

REQ-5.2: RTP Forward Error Correction [RFC5109].

REQ-6: Interoperability. The solution must also be able to use in:

REQ-6.1: Interworking with non-simulcast legacy clients using a single Media Source per media type.

REQ-6.2: WebRTC "Unified Plan" environment.

5. Proposed Solution Overview

Signaling Simulcast is about negotiating between media sender and receiver what the different Simulcast versions should be, how to identify them in terms of Source Packet Streams, and how to inter-relate those Source Packet Streams.

The proposed solution consists of:

- o Signaling Simulcast capability in an optional, pre-stage Offer/Answer:
 - * Separate send and receive Simulcast capabilities as SDP session level attributes.
 - * Media properties that are supported as base for different Simulcast versions are listed as parameters that are also possible to rank.
 - * Early indication of maximum number of available encoding/decoding resources on SDP media level.
- o Including detailed information for the Simulcast in a main Offer/Answer:
 - * Including Simulcast capability indications, as described above, being kept from the pre-stage Offer/Answer, if any.
 - * Defining and labeling of the media configuration for each Simulcast version to be sent or received.

- * The media configuration for a Simulcast version can include acceptable parameter ranges for parameters that are most likely used to distinguish Simulcast versions.
 - * Indicating the use of Simulcast, separately per direction, by grouping the defined media configurations, not individual streams, that will constitute the Simulcast.
 - * Allowing that any one of the media configurations in a specific Simulcast is signaled inactive from the start of the session. This is defined as equivalent to the affected Source Packet Stream being in PAUSED state [I-D.westerlund-avtext-rtp-stream-pause].
 - * Adding and/or modifying SDP media descriptions as needed to accommodate the negotiated Simulcast streams.
 - * Parameter limits to the aggregate of media configurations are signaled by existing SDP attributes on session and media description level.
 - * Including media level indication of maximum number of available encoding/decoding resources on SDP media level. They MAY be modified compared to the pre-stage Offer/Answer, if any.
 - * Identifying which Source Packet Stream corresponds to which media configuration by including the configuration label as part of the SDES item SRCNAME [I-D.westerlund-avtext-rtcp-sdes-srcname] information include in the RTP and RTCP packets. The optional mechanism for source specific signalling defined in SRCNAME could be used to let Simulcast sender pre-announce such a relationship before sending the Source Packet Stream.
- o Adding Simulcast information to the Source Packet Stream:
- * Identifying Source Packet Streams from same Media Source using the new RTCP SDES Item SRCNAME [I-D.westerlund-avtext-rtcp-sdes-srcname], and as described there including the possibility to send the same information as an RTP Header Extension [RFC5285].
 - * Using PAUSE/RESUME [I-D.westerlund-avtext-rtp-stream-pause] functionality to temporarily turn individual Simulcast versions on or off.

6. Proposed Signaling

This section further details the signaling solution outlined above (Section 5).

6.1. Simulcast Capability

There are numerous media properties that can be varied to construct a set of Simulcast versions. A Simulcast enabled endpoint could also support Simulcast based on several of those properties. As long as those properties are relatively independent and if each Simulcast version need explicit definition in the SDP, this would lead to an exponential number of Simulcast version candidates and a very long SDP that is likely also hard to interpret. There is thus a need to limit the Simulcast version candidates included in the SDP to cover as small set of properties as possible.

If a legacy endpoint not supporting Simulcast were to be presented with an SDP including media descriptions for a set of Simulcast versions, it may not know how to correctly handle or interpret these "surplus" media descriptions.

Based on the functionality that Simulcast is intended to achieve, it should be clear that the reasons to send Simulcast versions are not the same as to receive Simulcast versions, seen from a single endpoint.

For these reasons, it is proposed to define two new SDP session level attributes, "a=sim-send-cap" and "a=sim-recv-cap", which explicitly signal support for Simulcast media transmission and Simulcast media reception, respectively, for that media description. "a=sim-send-cap" and "a=sim-recv-cap" MAY be used independently and simultaneously. These attributes are also proposed to have parameters indicating the media properties used to create the Simulcast versions, and their preferred ranking. The meaning of the attributes on SDP media level is undefined and MUST NOT be used.

```

simulcast-cap    = "a="( "sim-send-cap:" / "sim-recv-cap:" )
                  cap-prop-list
cap-prop-list    = cap-prop-entry *(WSP cap-prop-entry)
cap-prop-entry   = cap-prop ["=" q-value]
cap-prop         = "rtptime"
                  / "fmt"
                  / "imageattr"
                  / "framerate"
                  / token ; for future extensions
q-value          = ( "0" "." 1*2DIGIT )
                  / ( "1" "." 1*2("0") )
                  ; Values between 0.00 and 1.00
; WSP and DIGIT defined in [RFC5234]
```

; token defined in [RFC4566]

Figure 1: ABNF for Simulcast Capability

The media property values are taken from existing (and could be extended to cover other or future) SDP attributes that express media properties that can be varied to create different Simulcast versions:

rtpmap: Differences in codec type, sampling rate (see Section 4), and number of channels.

fmp: Differences in codec-specific encoding parameters.

imageattr: Differences in video resolution and aspect ratio [RFC6236].

framerate: Differences in framerate.

The optional q-value expresses the relative preference to base a Simulcast version on that media property, with 1.00 meaning maximum (100%) preference and 0.00 meaning no (0%) preference. Several media properties can share the same q-value, in which case they are equally preferred. Not including any q-value for a media property value SHALL default to a q-value of 1.00.

The list of media properties is made extensible, to allow introducing additional dimensions for Simulcast versions.

6.1.1. Declarative Use

When used as a declarative media description, sim-recv-cap indicates the configured end-point's required capability to recognize and receive a specified set of Source Packet Streams as Simulcast streams. In the same fashion, sim-send-cap requests the end-point to send a specified set of Source Packet Streams as Simulcast streams. sim-recv-cap and sim-send-cap MAY be used independently and at the same time and they need not specify the same capability properties.

6.1.2. Offer/Answer Use

An offerer wanting to use Simulcast SHALL include either one or both of those attributes, depending on in which direction(s) Simulcast is both supported and desirable. An offerer that receives an answer without "a=sim-send-cap" or "a=sim-recv-cap" MUST NOT define or use any Simulcast alternatives in that direction to the answerer.

An answerer that does not understand the concept of Simulcast will also not know those attributes and will remove them in the SDP answer, as defined in existing SDP Offer/Answer procedures. An answerer that does understand the attributes and that wants to support Simulcast in the indicated direction SHALL reverse directionality of the attribute; "sim-send-cap" becomes "sim-recv-cap" and vice versa, and include it in the answer.

An offerer that intends to send Simulcast alternatives and thus includes "a=sim-send-cap", MUST also include at least one media property parameter that it intends to use to construct the Simulcast alternatives, but it MAY include more media property parameters. Including multiple media property parameters in "a=sim-send-cap" SHALL be interpreted as an offer to send Simulcast versions covering all combinations thereof, but MAY be further restricted by other information in the SDP such as for example the number of simulcast-related media descriptions in the SDP or use of max-ssrc signaling [I-D.westerlund-mmusic-max-ssrc].

An offerer that is capable of receiving Simulcast alternatives and thus includes "a=sim-recv-cap", MUST also include at least one media property parameter that it is willing to use as discriminator between received Simulcast alternatives, but MAY include more media property parameters. Including multiple media property parameters in "a=sim-recv-cap" SHALL be interpreted as an offer to receive Simulcast versions covering all combinations thereof, but MAY be further restricted by other information in the SDP such as for example the number of simulcast-related media descriptions in the SDP or use of max-ssrc signaling [I-D.westerlund-mmusic-max-ssrc].

An answerer that either lacks the capability or does not desire to use Simulcast versions based on a certain media property parameter in a specific direction MUST remove such media property parameter from "a=sim-send-cap" or "a=sim-recv-cap". The answerer MUST NOT add any media property parameters that were not included in the offer.

An answerer SHOULD take the offerer's q-values into account when choosing which media configurations (Section 6.2) to include in the answer and how to group them (Section 6.3) into the resulting Simulcast(s).

6.2. Media Configuration

Media that constitutes a Simulcast version has certain desirable characteristics that is meant to suit one category of diverse receivers (Section 3.1). A receiver that is willing to receive Simulcast streams must be given sufficient means to express what it is capable of and desires to receive. A sender that is willing to

send Simulcast streams must similarly be given sufficient means to express what it is capable of and desires to send.

An obvious candidate to express those characteristics is the media format in an SDP media description, defined by the `rtpmap` and `fmtp` attributes, which is typically mapped to an RTP Payload Type. Some of the most interesting characteristics for Simulcast purposes are however not included in `rtpmap` or `fmtp`, but are instead defined as separate attributes. Some of those individual attributes are possible to directly relate to a defined media format and could form a configuration together with the media format, but some attributes cannot be related to a specific media format and using the existing media format as a common identifier for a media configuration is not fully sufficient.

The act of Simulcast is trying to handle senders and receivers belonging to the vast multi-dimensional parameter space of "media configuration" by sub-dividing that parameter space into manageable and meaningful sub-sets. Communication between a sender and a receiver can be established successfully only when the actually sent media configuration (sub-set) fits within the receiver's available media configuration sub-set. At the same time, practical and implementation aspects often limits the size of those sub-sets. When that receiver or sender sub-set is either too small or is not known, the probability of successful communication decreases significantly. To increase the probability of finding a match between sender and receiver media configurations, it is essential that a media configuration can be a set instead of a single point in the parameter space, i.e. include parameter listings and/or ranges instead of single values.

Therefore, it is proposed to define a new media level SDP attribute, "a=config-id", which has relate the needed parameter types and the corresponding value ranges that together constitute a Simulcast media configuration. Each SDP media description MAY contain zero or more config-id attributes. The meaning of the attribute on SDP session level is undefined and MUST NOT be used.

```
configuration      = "a=config-id:" config-id WSP config-dir
                    WSP config-list
config-id          = token
config-dir         = "send"
                    / "recv"
config-list        = config-entry *(WSP config-entry)
config-entry       = "pt" "=" pt-value *(", " pt-value)
                    / image-attr
                    / "framerate" "=" fr-param
                    / "b" "=" bw-mod ":" bw-value *1("-" bw-value)
```

```

                                / ext-config-id [ "=" ext-config-value ]
                                ; for future ext
image-attr                     = "imageattr" "=" resolution-list
resolution-list                = resolution-set *("," resolution-set)
ext-config-id                  = token
ext-config-value               = non-ws-string
pt-value                       = 1*3DIGIT ; could be made more strict
resolution-set                 = "[" "x=" xyrange "," "y=" xyrange *key-values "]"
key-values                     = ( "," key-value )
key-value                     = ( "sar=" srange )
                                / ( "par=" prange )
                                / ( "q=" qvalue )
onetone                        = "1" / "2" / "3" / "4" / "5"
                                / "6" / "7" / "8" / "9"
xyvalue                        = onetone *5DIGIT
step                           = xyvalue
xyrange                        = ( "[" xyvalue ":" [ step ":" ] xyvalue "]" )
                                / ( "[" xyvalue 1*( "," xyvalue ) "]" )
                                / ( xyvalue )
spvalue                        = ( "0" "." onetone *3DIGIT )
                                / ( onetone "." 1*4DIGIT )
srange                         = ( "[" spvalue 1*( "," spvalue ) "]" )
                                / ( "[" spvalue "-" spvalue "]" )
                                / ( spvalue )
prange                         = ( "[" spvalue "-" spvalue "]" )
qvalue                         = ( "0" "." 1*2DIGIT )
                                / ( "1" "." 1*2("0") )
fr-param                       = fr-value *("," fr-value)
                                / fr-value "-" fr-value
fr-value                       = 1*3DIGIT [ "." 1*2DIGIT ]
bw-mod                         = "AS"
                                / "TIAS"
                                / token ; for future extensions
bw-value                       = 1*DIGIT
; WSP, DQUOTE and DIGIT defined in [RFC5234]
; token and non-ws-string defined in [RFC4566]

```

Figure 2: ABNF for Media Configuration

A media configuration is thus identified by:

config-id: A token that identifies the media configuration, which MUST be unique across all media configurations and media descriptions in the SDP.

config-dir: The direction for the stream(s) receiving the media configuration, as seen from the part issuing the SDP.

The media configuration MUST contain at least one and MAY contain more of the below media configuration entries. Each entry type MUST NOT appear more than once in every media configuration.

pt: A comma-separated list of media formats, RTP payload types, which MUST be defined within the same media description as config-id. This describes the allowed set of codecs or codec configurations for this media configuration. MUST be present in every media configuration.

imageattr: An OPTIONAL listing of preferred image resolutions for this media configuration. MUST NOT be used with other than video and image media types. An imageattr media configuration entry MUST NOT conflict with any "a=imageattr" attribute present in the same media description.

framerate: An OPTIONAL range or enumeration of preferred framerates for this media configuration. MUST NOT be used with other than video media types. The high end of the range MUST be equal to or larger than the low end. An enumerating framerate media configuration entry MUST include the value of the "a=framerate" attribute, if any. A framerate range media configuration entry MUST include the "a=framerate" value in the range.

b: An acceptable bandwidth range for this media configuration. Either one of the defined bandwidth modifiers MAY be used, which MUST share semantics with corresponding bandwidth modifiers from the SDP bandwidth attribute. The bandwidth value MUST be interpreted as defined by the bandwidth modifier. The high end of the range MUST be equal to or larger than the low end. The high end of the range MUST NOT exceed the bandwidth parameter in the same media description, if any. The sum of bandwidth range low ends for all media configurations within a media description MUST NOT exceed the value of that media description's bandwidth parameter. MUST be present in every media configuration.

Media configuration entry types "pt" and "b" MUST be supported by all implementations of this specification. Otherwise, an implementation MAY ignore any media configuration entry types that are not understood. A media configuration MAY be re-used to describe more than a single Source Packet Stream.

6.2.1. Simulcast Limitations

The Session and Media level attributes and parameters outside of individual media configurations (a=config-id) provides limitations on the set of media configurations in simultaneous use. For example a media description bandwidth limitation using b=AS would apply on all

the Packet Streams sent within the scope of that media description, thus forcing the sum of the media configuration bandwidth in use to share that available bandwidth. Don't forget other Packet Streams such as RTP retransmission or FEC flows that also needs to be included.

There exist a number of different limitations, and this section does not intend to be complete. The payload formats and their configurations can offer limitations, for example video profile and levels imposes a joint limit on bit-rate, frame-rate and resolution. The bandwidth parameters on session and media description level apply according to their semantics and their level. Packetization limitations, e.g. maxptime, as well as recommendations apply to all the configurations within the scope where this parameter is defined.

It is important to note that limits, such as bandwidth expressed within a media configuration are not limited by the media description values. First of all, the sum of bit-rates across all media configurations in a media description can be greater than the media description limit as not all configurations may be in simultaneous use. For example, only a single configuration can be enabled, which is then allowed to consume the full outer limit. Secondly, the media configuration directionality needs to be taken into account, for example that SDP receiver limitations are not applied to the sender configuration.

6.2.2. Declarative Use

When used as a declarative media description, config-id with rcv parameter indicates the configured end-point's required media configuration to receive a specified set of Source Packet Streams as Simulcast streams. In the same fashion, config-id with send parameter requests the end-point to use the specified media configuration when sending a specified set of Source Packet Streams as Simulcast streams.

6.2.3. Offer/Answer Use

An offerer wanting to use Simulcast in a specific direction SHALL use config-id to describe the media configurations to use in that direction in the Offer.

An answerer receiving a config-id media configuration for a specific direction, accepting to use that media configuration SHALL include a corresponding media configuration with the reverse direction in the Answer. The config-id identification value MUST be kept between the Offer and the Answer. An answerer not accepting to use a specific media configuration SHALL remove it from the Answer.

The Answer MUST keep exactly the same media configuration types in a media configuration as were present in the corresponding media configuration in the Offer.

The answerer MAY remove values from enumerations and MAY reduce ranges of media configuration entries in the Answer. If the reduced media configuration entry relates to the answerer's send direction, negotiation is complete and no further action is needed. If the reduced media configuration relates to the answerer's receive direction, the offerer SHOULD send another Offer where that related, send direction media configuration is reduced at least to the level in the previous Answer, but MAY be reduced even more, and MAY be removed entirely.

6.3. Grouping Simulcast Configurations

A set of media configurations (Section 6.2) is needed to describe a Simulcast. Each Source Packet Stream in the Simulcast share the same Media Source, but have different media configurations. Thus, the actual grouping of media configurations is what defines a specific Simulcast. It is proposed to define two new media level and session level SDP attributes, "a=sim-send" and "a=sim-recv", which uses config-id values to group media configurations for the purpose of Simulcast transmission and reception, respectively. "a=sim-send" and "a=sim-recv" MAY be used independently and simultaneously. They MAY be used on session level to group media configurations when different Simulcast encodings of a Media Source are to be sent in different Media Transports and RTP sessions. They MAY also be used on media level to group media configurations when different Simulcast encodings of a Media Source are to be sent based on the same media description and thus use the same Media Transport and RTP session. When used on media level, the Simulcast direction MAY conflict with the general media description direction, but a conflict MUST be interpreted as the Simulcast being effectively inhibited. For example, sim-send in a recvonly media description means that no Simulcast Source Packet Streams are sent.

```

simulcast          = "a="( "sim-send:" / "sim-recv:" ) config-id-list
config-id-list     = config-item *(WSP config-item)
config-item        = config-id [":" config-param-list]
config-id          = token
config-param-list  = config-param *("," config-param)
config-param       = "inactive"
                  / token ["=" param-value] ; for future extension
param-value        = 1*(value-char)
                  / DQUOTE non_ws_string DQUOTE
value-char         = token-char / %x28 / %x29 / %x2F / %x3A-3C
                  / %x3E-40 / %x5B-5D ; VCHAR except "=" and ","

```



```
; WSP and VCHAR defined in [RFC5234]
; token, token-char and non_ws_string defined in [RFC4566]
```

Figure 3: ABNF for Simulcast Configuration Grouping

The config-id identification of a media configuration MUST be defined by a "config-id" attribute in any of the media descriptions that are part of the SDP.

6.3.1. Declarative Use

When used as a declarative media description, sim-recv indicates the configured end-point's required ability to receive Source Packet Streams with the specified set of media configurations as Simulcast streams. In the same fashion, sim-send requests the end-point to send Source Packet Streams with the specified set of media configurations as Simulcast streams.

The configuration parameter "inactive" SHALL be interpreted as the related Source Packet Stream is in PAUSED state [I-D.westerlund-avtext-rtp-stream-pause] at the start of the session, and applicable RTP level procedures from that specification SHALL be applied.

6.3.2. Offer/Answer Use

An offerer wanting to send a set of Source Packet Streams as Simulcast streams includes sim-send in the Offer to describe which media configurations to use for that Simulcast. Similarly, an offerer wanting to receive a set of Source Packet Streams as Simulcast streams includes sim-recv in the Offer to describe which media configurations to use for that Simulcast.

An answerer receiving sim-send, accepting to receive those media configurations as Simulcasted Source Packet Streams SHALL include sim-recv with the accepted media configurations in the Answer. Similarly, an answerer receiving sim-recv, accepting to send those media configurations as Simulcasted Source Packet Streams SHALL include sim-send with the accepted media configurations in the Answer. An answerer MAY remove media configurations from sim-send or sim-recv included in the Answer compared to the ones included in the sim-send or sim-recv in the Offer. The answerer MUST NOT add any media configurations to sim-send or sim-recv in the Answer that were not in the corresponding ones in the Offer.

An "inactive" parameter present in the Offer MUST be kept in the Answer. The Answer MAY add an "inactive" parameter to any of the

media configurations. An "inactive" parameter on a media configuration in "sim-recv" is equivalent to a PAUSE (or in some cases, an equivalent TMMBR 0) message [I-D.westerlund-avtext-rtp-stream-pause] being sent for the received Source Packet Stream at the start of the session, and applicable RTP level procedures from that specification SHALL be applied. An "inactive" parameter on a media configuration in "sim-send" is equivalent to the related Source Packet Stream being in PAUSED state at the start of the session, and applicable RTP level procedures SHALL be applied.

The number of different Source Packet Streams used for a Simulcast related to a single media description MUST NOT exceed the number of listed media configurations in the corresponding sim-recv in that media description sent by the media receiver.

6.4. Relating Simulcast Versions

To ensure that Simulcast Packet Streams can be related correctly on RTP level, SDES SRCNAME [I-D.westerlund-avtext-rtcp-sdes-srname] MUST be used to label Simulcast versions belonging to the same Media Source. The RTP Header Extension option of that specification MAY be used with Simulcast.

The SRCNAME identifier for Simulcast MUST contain a first part that uniquely identifies the Media Source within a given CNAME, followed by a single "." (period) and the config-id as defined above (Section 6.2).

The SRCNAME parameter to source-specific signaling [RFC5576] ("a=ssrc") MAY be used for Source Packet Streams in the send direction to relate SRCNAME to SSRC already in the SDP.

6.5. Two-Phase Negotiation

The new "a=sim-send-cap" and "a=sim-recv-cap" attributes MAY be included in the SDP as an optional pre-stage in a two-phased approach, where the pre-stage involves a first SDP Offer/Answer procedure that only establishes Simulcast capability at both the offerer and the answerer. This has the additional advantage to avoid sending media descriptions related to Simulcast to an endpoint that does not support simulcast. In case two Offer/Answer procedures are already used for other reasons, it will not incur any significant extra signaling round-trips. Such other two-phase techniques include use of SIP OPTIONS, SIP UPDATE [RFC3311] with reliable provisional responses, and BUNDLE [I-D.ietf-mmusic-sdp-bundle-negotiation].

Thus, when using the pre-stage Offer/Answer, it SHOULD NOT include any simulcast-grouped media descriptions, which SHOULD then instead be added in a main Offer/Answer phase. When using the pre-stage Offer/Answer, half a signaling round-trip time can sometimes be saved if main phase is initiated by the Simulcast receiver, meaning that the endpoint that included "a=sim-recv" in the pre-stage SDP is the offerer in the main phase. If both endpoints are Simulcast receivers, it does not matter which endpoint sends the main Offer, using regular Offer/Answer rules to handle any race conditions.

It is not possible to use any pre-stage to establish capability with declarative SDP, in which case it SHALL be by-passed, using only the main phase directly.

6.6. Signaling Examples

These examples are for a case of client to video conference service using a centralized media topology with an RTP mixer.

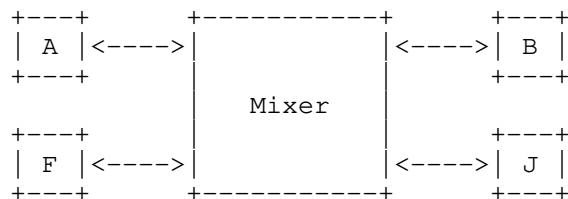


Figure 4: Four-party Mixer-based Conference

6.6.1. Unified Plan Client

Alice is calling in to the mixer with a Simulcast-enabled Unified Plan client capable of a single Media Source per media type. The only difference to a non-Simulcast client is capability to send video resolution [RFC6236] ("imageattr") and framerate based Simulcast. Alice uses a pre-stage Offer, which looks like:

```
v=0
o=alice 2362969037 2362969040 IN IP4 192.0.2.156
s=Simulcast Enabled Unified Plan Client
t=0 0
c=IN IP4 192.0.2.156
b=AS:665
a=sim-send-cap:imageattr framerate
m=audio 49200 RTP/AVP 96 8
b=AS:145
a=rtpmap:96 G719/48000/2
a=rtpmap:8 PCMA/8000
```

```
m=video 49300 RTP/AVP 97
b=AS:520
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=42c01e
a=imageattr:97 send [x=640,y=360] [x=320,y=180] \
    recv [x=640,y=360] [x=320,y=180]
```

Figure 5: Unified Plan Simulcast Pre-Stage Offer

In this pre-stage, the only thing in the SDP that indicates Simulcast capability is the line in the video media description containing the "sim-send-cap" attribute, which also indicates that sent Simulcast versions can differ in video resolution and/or framerate.

The Answer from the server indicates both that it too is Simulcast capable and that it would prefer to use video resolution ("imageattr") based Simulcast, but that it supports both video resolution and framerate. Should it not have been Simulcast capable, the "a=sim-recv-cap" line would not have been present and communication would have started with the media negotiated in the SDP.

```
v=0
o=server 823479283 1209384938 IN IP4 192.0.2.2
s=Answer to Simulcast Enabled Unified Plan Client
t=0 0
c=IN IP4 192.0.2.43
b=AS:665
a=sim-recv-cap:imageattr=1.0 framerate=0.8
m=audio 49200 RTP/AVP 96
b=AS:145
a=rtpmap:96 G719/48000/2
m=video 49300 RTP/AVP 97
b=AS:520
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=42c01e
a=imageattr:97 send [x=640,y=360] [x=320,y=180] \
    recv [x=640,y=360] [x=320,y=180]
```

Figure 6: Unified Plan Simulcast Pre-Stage Answer

Since the server is the Simulcast media receiver, it immediately initiates another Offer/Answer including details on the Simulcast versions. The server also keeps the "sim-recv-cap" as explicit Simulcast capability indication in this main Offer/Answer. Note that the "non-simulcast" media can be started already now, before the main

Offer/Answer, with the only restriction that the Simulcast functionality is not yet established.

```
v=0
o=server 823479283 1209384938 IN IP4 192.0.2.2
s=Server Inviting Simulcast Enabled Unified Plan Client
t=0 0
c=IN IP4 192.0.2.43
b=AS:825
a=sim-recv-cap:imageattr=1.0 framerate=0.8
m=audio 49200 RTP/AVP 96
b=AS:145
a=rtpmap:96 G719/48000/2
m=video 49300 RTP/AVP 97
b=AS:2200
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=42c01e
a=config-id:a recv pt=97 imageattr=[x=640,y=360],[x=1280,y=720] \
    framerate=25-60 b=AS:500-2500
a=config-id:b recv pt=97 imageattr=[x=320,y=180],[x=640,y=360] \
    framerate=25-60 b=AS:150-500
a=config-id:c recv pt=97 imageattr=[x=256,y=144],[x=320,y=180] \
    framerate=10-30 b=AS:100-250
a=sim-recv:a b c
```

Figure 7: Unified Plan Simulcast Main Offer

The server chooses to structure the Answer according to Unified Plan and has added three config-id lines in the video media description, one for each Simulcast media configuration that it is prepared to receive. Each media configuration refers to a defined media format, and lists a set of preferred video resolutions as well as a range of acceptable framerates, concluded by a bandwidth range. It also includes the sim-recv attribute for those three media configurations, indicating that the Simulcast it is prepared to receive in this media description can include one or more of those media configurations.

Alice's Answer is:

```
v=0
o=alice 2362969037 2362969040 IN IP4 192.0.2.156
s=Final answer from Simulcast Enabled Unified Plan Client
t=0 0
c=IN IP4 192.0.2.156
b=AS:825
a=sim-send-cap:imageattr framerate
m=audio 49200 RTP/AVP 96
```

```

b=AS:145
a=rtpmap:96 G719/48000/2
m=video 49300 RTP/AVP 97
b=AS:520
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=42c01e
a=config-id:b send pt=97 imageattr=[x=640,y=360] \
    framerate=25-30 b=AS:150-400
a=config-id:c send pt=97 imageattr=[x=320,y=180] \
    framerate=10-12.5 b=AS:100-150
a=sim-send:b c:inactive
a=ssrc:31053821 cname=SDIe93850aQFid9P srcname=l.b
a=ssrc:43298172 cname=SDIe93850aQFid9P srcname=l.c
a=imageattr:97 send [x=640,y=360] [x=320,y=180] \
    recv [x=640,y=360] [x=320,y=180]

```

Figure 8: Unified Plan Simulcast Main Answer

The Simulcast capability, `sim-send-cap`, is kept from Alice's previous Offer. One of the media configurations from the server Offer, `config-id:a`, is not acceptable to Alice's client for some reason and is removed from the Answer. The resulting Simulcast, described by `sim-send`, thus contains two media configurations, `b` and `c`, where `c` is initially set to "inactive" that effectively means it is paused from the start of the session. The media configuration parameter value ranges are in some cases reduced, which makes a more precise definition of what will actually be sent. This Answer SDP also includes a specification of the SSRC values that will be sent and what media configurations those SSRC will carry, by including the `srcname` parameter. The first part of `srcname`, before the ".", is the Media Source identification. Both SSRC share the same Media Source identification, since they are part of the same Simulcast. The second part, after the ".", is the `config-id` of the media configuration sent with that SSRC.

6.6.2. Multi-Transport Client

Bob is calling in to the mixer with a Simulcast-enabled client, like Alice's capable of a single Media Source per media type, but also capable of sending Source Packet Streams as Simulcast versions on separate Media Transports. In this example, Bob's client knows that the server is capable of Simulcast and does not use any pre-stage Offer, but goes straight to the main Offer.

```

v=0
o=bob 94572932847 3429478298 IN IP4 192.0.2.93
s=Offer from Simulcast Enabled Multi-Transport Client

```

```

t=0 0
c=IN IP4 192.0.2.93
b=AS:825
a=sim-send-cap:imageattr=1.0 framerate=0.9
a=sim-send:x y
m=audio 50138 RTP/AVP 101
b=AS:145
a=rtpmap:101 G719/48000/2
m=video 50226 RTP/AVP 118
b=AS:500
a=rtpmap:118 H264/90000
a=fmtp:118 profile-level-id=42c01e
a=config-id:x send pt=118 imageattr=[x=320,y=180],[x=640,y=360] \
    framerate=25-50 b=AS:200-500
a=ssrc:3929384298 cname=Nsdko390en828FKn srcname=M.x
a=imageattr:118 send [x=640,y=360] [x=320,y=180] \
    recv [x=640,y=360] [x=320,y=180]
m=video 50228 RTP/AVP 119
b=AS:150
a=config-id:y send pt=119 imageattr=[x=256,y=144],[x=320,y=180] \
    framerate=12.5-25 b=AS:100-200
a=ssrc:1923419284 cname=Nsdko390en828FKn srcname=M.y
a=imageattr:119 send [x=320,y=180] [x=256,y=144]
a=sendonly

```

Figure 9: Multi-Transport Simulcast Main Offer

As can be seen from above, this Offer uses `sim-send` on session level and has split the Simulcast media configurations on two media descriptions, in order to be able to use separate Media Transports and enable differentiated treatment of the two Simulcast streams.

The server accepts this structure to the Answer:

```

v=0
o=server 283479882 9384298374 IN IP4 192.0.2.2
s=Server Answering Simulcast Enabled Multi-Transport Client
t=0 0
c=IN IP4 192.0.2.45
b=AS:825
a=sim-recv-cap:imageattr framerate
a=sim-recv:x y
m=audio 49200 RTP/AVP 96
b=AS:145
a=rtpmap:96 G719/48000/2
m=video 49300 RTP/AVP 118
b=AS:500

```

```

a=rtpmap:118 H264/90000
a=fmtp:118 profile-level-id=42c01e
a=config-id:x recv pt=118 imageattr=[x=640,y=360] \
    framerate=25-50 b=AS:350-500
a=imageattr:118 send [x=640,y=360] [x=320,y=180] \
    recv [x=640,y=360] [x=320,y=180]
m=video 49300 RTP/AVP 119
b=AS:150
a=rtpmap:119 H264/90000
a=fmtp:119 profile-level-id=42c01e
a=config-id:y recv pt=119 imageattr=[x=256,y=144] \
    framerate=12.5-25 b=AS:120-150
a=imageattr:119 recv [x=320,y=180] [x=256,y=144]
a=recvonly

```

Figure 10: Multi-Transport Simulcast Main Answer

6.6.3. Multi-Source Client

Fred is calling in to the same conference as in the examples above with a three-camera, three-display system, thus capable of handling three separate Media Sources in each direction, where each Media Source is also Simulcast-enabled in the send direction. Fred's client is a Unified Plan client, restricted to a single Media Source per media description.

```

v=0
o=fred 238947129 823479223 IN IP4 192.0.2.125
s=Offer from Simulcast Enabled Multi-Source Client
t=0 0
c=IN IP4 192.0.2.125
b=AS:825
a=sim-send-cap:imageattr=1.0 framerate=0.5

m=audio 49200 RTP/AVP 98
b=AS:145
a=rtpmap:98 G719/48000/2

m=video 49600 RTP/AVP 100
b=AS:3500
a=rtpmap:100 H264/90000
a=fmtp:100 profile-level-id=42c02a
a=config-id:lh send pt=100 imageattr=[x=1920,y=1080] \
    framerate=30-60 b=AS:2000-3500
a=config-id:lm send pt=100 imageattr=[x=1280,y=720] \
    framerate=15-60 b=AS:1000-2000
a=config-id:ll send pt=100 imageattr=[x=640,y=360] \

```



```
    framerate=10-60 b=AS:200-1000
a=sim-send:1h 1m 1l
a=ssrc:2397234521 cname=EkeS32892Fe029DK srcname=1.1h
a=ssrc:1023894789 cname=EkeS32892Fe029DK srcname=1.1m
a=ssrc:4029284928 cname=EkeS32892Fe029DK srcname=1.1l
a=imageattr:100 send [x=1920,y=1080] [x=1280,y=720] [x=640,y=360] \
    recv [x=1920,y=1080] [x=1280,y=720] [x=640,y=360]

m=video 49600 RTP/AVP 100
b=AS:3500
a=rtpmap:100 H264/90000
a=fmtp:100 profile-level-id=42c02a
a=config-id:2h send pt=100 imageattr=[x=1920,y=1080] \
    framerate=30-60 b=AS:2000-3500
a=config-id:2m send pt=100 imageattr=[x=1280,y=720] \
    framerate=15-60 b=AS:1000-2000
a=config-id:2l send pt=100 imageattr=[x=640,y=360] \
    framerate=10-60 b=AS:200-1000
a=sim-send:2h 2m 2l
a=ssrc:2301017618 cname=EkeS32892Fe029DK srcname=2.2h
a=ssrc:639711316 cname=EkeS32892Fe029DK srcname=2.2m
a=ssrc:3293473905 cname=EkeS32892Fe029DK srcname=2.2l
a=imageattr:100 send [x=1920,y=1080] [x=1280,y=720] [x=640,y=360] \
    recv [x=1920,y=1080] [x=1280,y=720] [x=640,y=360]

m=video 49600 RTP/AVP 100
b=AS:3500
a=rtpmap:100 H264/90000
a=fmtp:100 profile-level-id=42c02a
a=config-id:3h send pt=100 imageattr=[x=1920,y=1080] \
    framerate=30-60 b=AS:2000-3500
a=config-id:3m send pt=100 imageattr=[x=1280,y=720] \
    framerate=15-60 b=AS:1000-2000
a=config-id:3l send pt=100 imageattr=[x=640,y=360] \
    framerate=10-60 b=AS:200-1000
a=sim-send:3h 3m 3l
a=ssrc:4115355057 cname=EkeS32892Fe029DK srcname=3.3h
a=ssrc:3196538337 cname=EkeS32892Fe029DK srcname=3.3m
a=ssrc:3757973912 cname=EkeS32892Fe029DK srcname=3.3l
a=imageattr:100 send [x=1920,y=1080] [x=1280,y=720] [x=640,y=360] \
    recv [x=1920,y=1080] [x=1280,y=720] [x=640,y=360]
```

Figure 11: Fred's Multi-Source Simulcast Main Offer

The three media descriptions for video are essentially the same, except values that needs to be unique are provided unique values. The above also assumes that BUNDLE will be used across these three video media description to create a common RTP session.

7. Network Aspects

Simulcast is in defined as the act of sending multiple alternative encodings of the same underlying media source. When transmitting multiple independent streams that originate from the same source, it could potentially be done in several different ways using RTP. A general discussion on considerations for use of the different RTP multiplexing alternatives can be found in Guidelines for Multiplexing in RTP [I-D.ietf-avtcore-multiplex-guidelines]. Discussion and clarification on how to handle multiple streams in an RTP session can be found in [I-D.ietf-avtcore-rtp-multi-stream].

The network aspects that are relevant for Simulcast are:

Quality of Service: When using Simulcast it might be of interest to prioritize a particular Simulcast version, rather than applying equal treatment to all versions. For example, lower bit-rate versions may be prioritized over higher bit-rate versions to minimize congestion or packet losses in the low bit-rate versions. Thus, there is a benefit to use a Simulcast solution that supports QoS as good as possible. By separating Simulcast versions into different RTP sessions and send those RTP sessions over different Media Transports, a Simulcast version can be prioritized by existing flow based QoS mechanisms. When using unicast, QoS mechanisms based on individual packet marking are also feasible, which do not require separation of Simulcast versions into different RTP sessions to apply different QoS.

NAT/FW Traversal: Using multiple RTP sessions will incur more cost for NAT/FW traversal unless they can re-use the same transport flow, which can be achieved by either one of multiplexing multiple RTP sessions on a single lower layer transport [I-D.westerlund-avtcore-transport-multiplexing] or Multiplexing Negotiation Using SDP Port Numbers [I-D.ietf-mmusic-sdp-bundle-negotiation]. If flow based QoS with any differentiation is desirable, the cost for additional transport flows is likely necessary.

Multicast: Multiple RTP sessions will be required to enable combining Simulcast with multicast. Different Simulcast versions have to be separated to different multicast groups to allow a multicast receiver to pick the version it wants, rather than receive all of them. In this case, the only reasonable

implementation is to use different RTP sessions for each multicast group so that reporting and other RTCP functions operate as intended.

8. IANA Considerations

This document requests that five new attributes, `sim-send-cap`, `sim-recv-cap`, `sim-send`, `sim-recv`, and `config-id`. It is also requested to make a new registry of defined parameters taken from existing SDP attributes for `sim-send-cap`, `sim-recv-cap`, and `config-id`.

Formal registrations to be written.

9. Security Considerations

The Simulcast capability and configuration attributes and parameters are vulnerable to attacks in signaling.

A false inclusion of Simulcast attributes may result in generation of a second phase SDP that potentially contains a large number of non-supported media descriptions expressing Simulcast alternatives. A correct SDP implementation will however be able to reject any non-supported media descriptions and the effect from that should be limited.

A hostile removal of the Simulcast attributes will result in skipping any second phase Offer/Answer and that Simulcast is not used.

The Simulcast grouping semantics are vulnerable to attacks in the signalling. Changing the set of media configurations that are used in a Simulcast will impact the number of Source Packet Streams.

A hostile removal of Simulcast grouping will prevent streams from being interpreted as Simulcast, which obviously prevents use of the Simulcast functionality. It will also risk that intended Simulcast streams are instead presented as separate, independent streams to a receiver.

Neither of the above will likely have any major consequences and can be mitigated by signaling that is at least integrity and source authenticated to prevent an attacker to change it.

10. Contributors

Morgan Lindqvist and Fredrik Jansson, both from Ericsson, have contributed with important material to the first versions of this document.

11. Acknowledgements

12. References

12.1. Normative References

- [I-D.westerlund-avtext-rtcp-sdes-srcname]
Westerlund, M., "RTCP Source Description Item SRCNAME to Label Individual Media Sources", draft-westerlund-avtext-rtcp-sdes-srcname-03 (work in progress), October 2013.
- [I-D.westerlund-avtext-rtp-stream-pause]
Akram, A., Burman, B., Grondal, D., and M. Westerlund, "RTP Media Stream Pause and Resume", draft-westerlund-avtext-rtp-stream-pause-03 (work in progress), October 2012.
- [I-D.westerlund-mmusic-max-ssrc]
Holmberg, C., Westerlund, M., and F. Jansson, "Multiple Synchronization Sources (SSRC) in SDP Media Descriptions", draft-westerlund-mmusic-max-ssrc-02 (work in progress), September 2013.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3311] Rosenberg, J., "The Session Initiation Protocol (SIP) UPDATE Method", RFC 3311, October 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.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC4568] Andreasen, F., Baugher, M., and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, July 2006.
- [RFC5109] Li, A., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, December 2007.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.
- [RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 2008.

- [RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.
- [RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.
- [RFC6236] Johansson, I. and K. Jung, "Negotiation of Generic Image Attributes in the Session Description Protocol (SDP)", RFC 6236, May 2011.

12.2. Informative References

- [I-D.ietf-avtcore-multiplex-guidelines]
Westerlund, M., Perkins, C., and H. Alvestrand,
"Guidelines for using the Multiplexing Features of RTP to Support Multiple Media Streams", draft-ietf-avtcore-multiplex-guidelines-01 (work in progress), July 2013.
- [I-D.ietf-avtcore-rtp-multi-stream]
Lennox, J., Westerlund, M., Wu, W., and C. Perkins,
"Sending Multiple Media Streams in a Single RTP Session", draft-ietf-avtcore-rtp-multi-stream-01 (work in progress), July 2013.
- [I-D.ietf-avtcore-rtp-topologies-update]
Westerlund, M. and S. Wenger, "RTP Topologies", draft-ietf-avtcore-rtp-topologies-update-00 (work in progress), April 2013.
- [I-D.ietf-mmusic-sdp-bundle-negotiation]
Holmberg, C., Alvestrand, H., and C. Jennings,
"Multiplexing Negotiation Using Session Description Protocol (SDP) Port Numbers", draft-ietf-mmusic-sdp-bundle-negotiation-05 (work in progress), October 2013.
- [I-D.lennox-raiarea-rtp-grouping-taxonomy]
Lennox, J., Gross, K., Nandakumar, S., and G. Salgueiro,
"A Taxonomy of Grouping Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", draft-lennox-raiarea-rtp-grouping-taxonomy-03 (work in progress), October 2013.
- [I-D.westerlund-avtcore-transport-multiplexing]
Westerlund, M. and C. Perkins, "Multiple RTP Sessions on a Single Lower-Layer Transport", draft-westerlund-avtcore-transport-multiplexing-06 (work in progress), August 2013.

- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.
- [RFC3569] Bhattacharyya, S., "An Overview of Source-Specific Multicast (SSM)", RFC 3569, July 2003.
- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.
- [RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, January 2008.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.
- [RFC6190] Wenger, S., Wang, Y., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, May 2011.

Appendix A. Discussion on Receiver Diversity

Receiver diversity can be handled in a number of different ways, each with its own advantages and disadvantages. In that, there are relations between RTP Mixer processing requirement, bandwidth usage on uplink from sending Participant to RTP Mixer, bandwidth usage on downlink from RTP Mixer to receiving Participant, and media Quality of Experience at the receiving Participant.

The following is a listing of possible approaches:

1. Lowest Common Denominator: Create a single Source Packet Stream per Media Source and, assuming that everyone can receive a "simple" stream, adapt the characteristics of that Source Packet Stream already at the sending Participant to the lowest common denominator among all receiving Participants. Let the RTP Mixer forward this single Source Packet Stream to all receiving Participants. The advantages are low bandwidth usage on both uplink and downlink and low RTP Mixer processing requirements. The disadvantage is that the least capable receiver and/or network path dictates the (low) QoE for everyone else.
2. Individual Transcoding: Create a single Source Packet Stream per Media Source with characteristics governed by resources available to the sending Participant and the network path to the RTP Mixer.

Let the RTP Mixer transcode (decode and re-encode) that into individual Source Packet Streams for each receiving Participant, governed by the RTP Mixer resources, receiving Participant resources, and the network path to that Participant. The advantages are adapted although overall slightly lowered QoE (due to transcoding) to each Participant and optimised bandwidth usage on both uplink and downlink. The disadvantage is (very) high RTP Mixer processing requirements.

3. Individual Simulcast: Create individual Source Packet Streams of each Media Source to each receiving Participant, constituting a complete individual Simulcast. Let the RTP Mixer forward each individual Source Packet Stream to the targeted receiving Participant. The advantages are low RTP Mixer processing and optimised downlink bandwidth. The disadvantage is (very) high uplink bandwidth.
4. Grouped Simulcast: For each Media Source, create a "suitable" logical grouping of receiving Participants in sub-groups with respect to available receiver resources, for example the resources listed above (Section 3.1). Create a set of Source Packet Streams for this Media Source with well-chosen characteristics, where each Source Packet Stream in the set is a good-enough fit to the receiving sub-group of Participants. This set of Source Packet Streams constitutes a Simulcast of the Media Source. The size of the set and the characteristics of each Source Packet Stream can be adjusted to cater for various restrictions in the sending Participant, receiving Participants in the sub-group, and network path(s) to the Participants in the sub-group. Let the RTP Mixer forward the same Source Packet Stream to all Participants in a sub-group, for all Source Packet Streams and sub-groups. The advantages are low RTP Mixer processing, near optimum QoE, and near optimum downlink bandwidth. The disadvantages are high uplink bandwidth and arguably that downlink bandwidth and QoE are optimum only for a sub-group and not per individual receiving Participant.

A summary of the advantages and disadvantages of the above four principle alternatives is given below (Table 1):

Method	Mixer CPU	Uplink	Downlink	QoE
1	Low	Low	Low	Low
2	Very high	Optimum	Optimum	Near optimum
3	Low	Very high	Optimum	Optimum
4	Low	High	Near optimum	Near optimum

Table 1: Receiver Diversity Handling Comparison

The authors of this document believes that alternative 4, the Grouped Simulcast, can be a good tradeoff whenever supported by sufficient uplink resources.

Authors' Addresses

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

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

Bo Burman
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

Phone: +46 10 714 13 11
Email: bo.burman@ericsson.com

Suhas Nandakumar
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Email: snandaku@cisco.com