Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 29, 2012

C. Perkins
University of Glasgow
M. Westerlund
Ericsson
J. Ott
Aalto University

August 28, 2011

RTP Requirements for RTC-Web
draft-perkins-rtcweb-rtp-usage-03

Abstract

This memo discusses use of RTP in the context of the RTC-Web activity. It discusses important features of RTP that need to be considered by other parts of the RTC-Web framework, describes which RTP profile to use in this environment, and outlines what RTP extensions should be supported.

This document is a candidate to become a work item of the RTCWEB working group as <WORKING GROUP DRAFT "MEDIA TRANSPORTS">.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

This memo discusses the Real-time Transport Protocol (RTP) [RFC3550] in the context of the RTC-Web activity. The work in the IETF Audio/Video Transport Working Group, and it's successors, has been about providing building blocks for real-time multimedia transport, and has not specified who should use which building blocks. The selection of building blocks and functionalities can really only be done in the context of some application, for example RTC-Web. We have selected a set of RTP features and extensions that are suitable for a number of applications that fit the RTC-Web context. Thus, applications such as VoIP, audio and video conferencing, and on-demand multimedia streaming are considered. Applications that rely on IP multicast have not been considered likely to be applicable to RTC-Web, thus extensions related to multicast have been excluded. We believe that RTC-Web will greatly benefit in interoperability if a reasonable set of RTP functionalities and extensions are selected. This memo is intended as a starting point for discussion of those features in the RTC-Web framework.

This memo is structured into different topics. For each topic, one or several recommendations from the authors are given. When it comes to the importance of extensions, or the need for implementation support, we use three requirement levels to indicate the importance of the feature to the RTC-Web specification:

REQUIRED: Functionality that is absolutely needed to make the RTC-Web solution work well, or functionality of low complexity that provides high value.

RECOMMENDED: Should be included as its brings significant benefit, but the solution can potentially work without it.

OPTIONAL: Something that is useful in some cases, but not always a benefit.

When this memo discusses RTP, it includes the RTP Control Protocol (RTCP) unless explicitly stated otherwise. RTCP is a fundamental and integral part of the RTP protocol, and is REQUIRED to be implemented.

1.1. Expected Topologies

As RTC-Web is focused on peer to peer connections established from clients in web browsers the following topologies further discussed in RTP Topologies [RFC5117] are primarily considered. The topologies are depicted and briefly explained here for ease of the reader.

```
+---+         +---+
| A |<------->| B |
+---+         +---+
```

Figure 1: Point to Point

The point to point topology (Figure 1) is going to be very common in any single user to single user applications.

```
+---+      +---+
| A |<---->| B |
+---+      +---+
  ^         ^
   \       /
    \     /
     v   v
     +---+
     | C |
     +---+
```

Figure 2: Multi-unicast

For small multiparty sessions it is practical enough to create RTP sessions by letting every participant send individual unicast RTP/UDP flows to each of the other participants. This is called multi-unicast and is unfortunately not discussed in the RTP Topologies [RFC5117]. This topology has the benefit of not requiring central nodes. The downside is that it increases the used bandwidth at each sender by requiring one copy of the media streams for each participant that are part of the same session beyond the sender itself. Thus this is limited to scenarios with few end-points unless the media is very low bandwidth.

It needs to be noted that, if this topology is to be supported by the RTC-Web framework, it needs to be possible to connect one RTP session to multiple established peer to peer flows that are individually established.

```
+---+      +------------+      +---+
| A |<---->|            |<---->| B |
+---+      |            |      +---+
           |   Mixer    |
+---+      |            |      +---+
| C |<---->|            |<---->| D |
+---+      +------------+      +---+
```

Figure 3: RTP Mixer with Only Unicast Paths

An RTP mixer (Figure 3) is a centralised point that selects or mixes content in a conference to optimise the RTP session so that each end-point only needs connect to one entity, the mixer. The mixer also reduces the bit-rate needs as the media sent from the mixer to the end-point can be optimised in different ways. These optimisations include methods like only choosing media from the currently most active speaker or mixing together audio so that only one audio stream is required in stead of 3 in the depicted scenario. The downside of the mixer is that someone is required to provide the actual mixer.

```
+---+      +------------+      +---+
| A |<---->|            |<---->| B |
+---+      |            |      +---+
           | Translator |
+---+      |            |      +---+
| C |<---->|            |<---->| D |
+---+      +------------+      +---+
```

Figure 4: RTP Translator (Relay) with Only Unicast Paths

If one wants a less complex central node it is possible to use an relay (called an Transport Translator) (Figure 4) that takes on the role of forwarding the media to the other end-points but doesn't perform any media processing. It simply forwards the media from all other to all the other. Thus one endpoint A will only need to send a media once to the relay, but it will still receive 3 RTP streams with the media if B, C and D all currently transmits.

```
           +------------+
           |            |
+---+      |            |      +---+
| A |<---->| Translator |<---->| B |
+---+      |            |      +---+
           |            |
           +------------+
```

Figure 5: Translator towards Legacy end-point

To support legacy end-point (B) that don't fulfil the requirements of RTC-Web it is possible to insert a Translator (Figure 5) that takes on the role to ensure that from A's perspective B looks like a fully compliant end-point. Thus it is the combination of the Translator and B that looks like the end-point B. The intention is that the presence of the translator is transparent to A, however it is not certain that is possible. Thus this case is include so that it can be discussed if any mechanism specified to be used for RTC-Web results in such issues and how to handle them.

2. Requirements from RTP

This section discusses some requirements RTP and RTCP [RFC3550] place on their underlying transport protocol, the signalling channel, etc.

2.1. Signalling for RTP sessions

RTP is built with the assumption of an external to RTP/RTCP signalling channel to configure the RTP sessions and its functions. The basic configuration of an RTP session consists of the following parameters:

RTP Profile: The name of the RTP profile to be used in session. The RTP/AVP [RFC3551] and RTP/AVPF [RFC4585] profiles can interoperate on basic level, as can their secure variants RTP/SAVP [RFC3711] and RTP/SAVPF [RFC5124]. The secure variants of the profiles do not directly interoperate with the non-secure variants, due to the presence of additional header fields in addition to any cryptographic transformation of the packet content.

Transport Information: Source and destination address(s) and ports for RTP and RTCP must be signalled for each RTP session. If RTP and RTCP multiplexing [RFC5761] is to be used, such that a single port is used for RTP and RTCP flows, this must be signalled.

RTP Payload Types, media formats, and media format parameters: The mapping between media type names (and hence the RTP payload formats to be used) and the RTP payload type numbers must be signalled. Each media type may also have a number of media type parameters that must also be signalled to configure the codec and RTP payload format (the "a=fmtp:" line from SDP).

RTP Extensions: The RTP extensions one intends to use need to be agreed upon, including any parameters for each respective extension. At the very least, this will help avoiding using bandwidth for features that the other end-point will ignore. But for certain mechanisms there is requirement for this to happen as interoperability failure otherwise happens.

RTCP Bandwidth: Support for exchanging RTCP Bandwidth values to the end-points will be necessary, as described in "Session Description Protocol (SDP) Bandwidth Modifiers for RTP Control Protocol (RTCP) Bandwidth" [RFC3556], or something semantically equivalent. This also ensures that the end-points have a common view of the RTCP bandwidth, this is important as too different view of the bandwidths may lead to failure to interoperate.

These parameters are often expressed in SDP messages conveyed within

an offer/answer exchange. RTP does not depend on SDP or on the offer/answer model, but does require all the necessary parameters to be agreed somehow, and provided to the RTP implementation. We note that in RTCWEB context it will depend on the signalling model and API how these parameters need to be configured but they will be need to either set in the API or explicitly signalled between the peers.

2.2. (Lack of) Signalling for Payload Format Changes

As discussed in Section 2.1, the mapping between media type name, and its associated RTP payload format, and the RTP payload type number to be used for that format must be signalled as part of the session setup. An endpoint may signal support for multiple media formats, or multiple configurations of a single format, each using a different RTP payload type number. If multiple formats are signalled by an endpoint, that endpoint is REQUIRED to be prepared to receive data encoded in any of those formats at any time. RTP does not require advance signalling for changes between formats that were signalled during the session setup. This is needed for rapid rate adaptation.

3. RTP Profile

The "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)" [RFC5124] is REQUIRED to be implemented. This builds on the basic RTP/AVP profile [RFC3551], the RTP/AVPF feedback profile [RFC4585], and the secure RTP/SAVP profile [RFC3711].

The RTP/AVPF part of RTP/SAVPF is required to get the improved RTCP timer model, that allows more flexible transmission of RTCP packets in response to events, rather than strictly according to bandwidth. This also saves RTCP bandwidth and will commonly only use the full amount when there is a lot of events on which to send feedback. This functionality is needed to make use of the RTP conferencing extensions discussed in Section 6.1.

The RTP/SAVP part of RTP/SAVPF is for support for Secure RTP (SRTP) [RFC3711]. This provides media encryption, integrity protection, replay protection and a limited form of source authentication. It does not contain a specific keying mechanism, so that, and the set of security transforms, will be required to be chosen. It is possible that a security mechanism operating on a lower layer than RTP can be used instead and that should be evaluated. However, the reasons for the design of SRTP should be taken into consideration in that discussion.

4. RTP and RTCP Guidelines

RTP and RTCP are two flexible and extensible protocols that allow, on the one hand, choosing from a variety of building blocks and combining those to meet application needs, and on the other hand, create extensions where existing mechanisms are not sufficient: from new payload formats to RTP extension headers to additional RTCP control packets.

Different informational documents provide guidelines to the use and particularly the extension of RTP and RTCP, including the following: Guidelines for Writers of RTP Payload Format Specifications [RFC2736] and Guidelines for Extending the RTP Control Protocol [RFC5968].

5. RTP Optimisations

This section discusses some optimisations that makes RTP/RTCP work better and more efficient and therefore are considered.

5.1. RTP and RTCP Multiplexing

Historically, RTP and RTCP have been run on separate UDP ports. With the increased use of Network Address/Port Translation (NAPT) this has become problematic, since maintaining multiple NAT bindings can be costly. It also complicates firewall administration, since multiple ports must be opened to allow RTP traffic. To reduce these costs and session setup times, support for multiplexing RTP data packets and RTCP control packets on a single port [RFC5761] is REQUIRED. Supporting this specification is generally a simplification in code, since it relaxes the tests in [RFC3550].

Note that the use of RTP and RTCP multiplexed on a single port ensures that there is occasional traffic sent on that port, even if there is no active media traffic. This may be useful to keep-alive NAT bindings.

5.2. Reduced Size RTCP

RTCP packets are usually sent as compound RTCP packets; and RFC 3550 demands that those compound packets always start with an SR or RR packet. However, especially when using frequent feedback messages, these general statistics are not needed in every packet and unnecessarily increase the mean RTCP packet size and thus limit the frequency at which RTCP packets can be sent within the RTCP bandwidth share.

RFC5506 "Support for Reduced-Size Real-Time Transport Control

Protocol (RTCP): Opportunities and Consequences" [RFC5506] specifies how to reduce the mean RTCP message and allow for more frequent feedback. Frequent feedback, in turn, is essential to make real-time application quickly aware of changing network conditions and allow them to adapt their transmission and encoding behaviour.

Support for RFC5506 is REQUIRED.

5.3. Symmetric RTP/RTCP

RTP entities choose the RTP and RTCP transport addresses, i.e., IP addresses and port numbers, to receive packets on and bind their respective sockets to those. When sending RTP packets, however, they may use a different IP address or port number for RTP, RTCP, or both; e.g., when using a different socket instance for sending and for receiving. Symmetric RTP/RTCP requires that the IP address and port number for sending and receiving RTP/RTCP packets are identical.

The reasons for using symmetric RTP is primarily to avoid issues with NAT and Firewalls by ensuring that the flow is actually bi-directional and thus kept alive and registered as flow the intended recipient actually wants. In addition it saves resources in the form of ports at the end-points, but also in the network as NAT mappings or firewall state is not unnecessary bloated. Also the number of QoS state are reduced.

Using Symmetric RTP and RTCP [RFC4961] is REQUIRED.

5.4. Generation of the RTCP Canonical Name (CNAME)

The RTCP Canonical Name (CNAME) provides a persistent transport-level identifier for an RTP endpoint. While the Synchronisation Source (SSRC) identifier for an RTP endpoint may change if a collision is detected, or when the RTP application is restarted, it's RTCP CNAME is meant to stay unchanged, so that RTP endpoints can be uniquely identified and associated with their RTP media streams. For proper functionality, RTCP CNAMEs should be unique among the participants of an RTP session.

The RTP specification [RFC3550] includes guidelines for choosing a unique RTP CNAME, but these are not sufficient in the presence of NAT devices. In addition, some may find long-term persistent identifiers problematic from a privacy viewpoint. Accordingly, support for generating a short-term persistent RTCP CNAMEs following method (b) as specified in Section 4.2 of "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)" [RFC6222] is RECOMMENDED, since this addresses both concerns.

6. RTP Extensions

There are a number of RTP extensions that could be very useful in the RTC-Web context. One set is related to conferencing, others are more generic in nature.

6.1. RTP Conferencing Extensions

RTP is inherently defined for group communications, whether using IP multicast, multi-unicast, or based on a centralised server. In today's practice, however, overlay-based conferencing dominates, typically using one or a few so-called conference bridges or servers to connect endpoints in a star or flat tree topology. Quite diverse conferencing topologies can be created using the basic elements of RTP mixers and translators as defined in RFC 3550.

An number of conferencing topologies are defined in [RFC5117] out of the which the following ones are the more common (and most likely in practice workable) ones:

1) RTP Translator (Relay) with Only Unicast Paths (RFC 5117, section 3.3)

2) RTP Mixer with Only Unicast Paths (RFC 5117, section 3.4)

3) Point to Multipoint Using a Video Switching MCU (RFC 5117, section 3.5)

4) Point to Multipoint Using Content Modifying MCUs (RFC 5117, section 3.6)

We note that 3 and 4 are not well utilising the functions of RTP and in some cases even violates the RTP specifications. Thus we recommend that one focus on 1 and 2.

RTP protocol extensions to be used with conferencing are included because they are important in the context of centralised conferencing, where one RTP Mixer (Conference Focus) receives a participants media streams and distribute them to the other participants. These messages are defined in the Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF) [RFC4585] and the "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)" (CCM) [RFC5104] and are fully usable by the Secure variant of this profile (RTP/SAVPF) [RFC5124].

6.1.1. Full Intra Request

The Full Intra Request is defined in Sections 3.5.1 and 4.3.1 of CCM [RFC5104]. It is used to have the mixer request from a session participants a new Intra picture. This is used when switching between sources to ensure that the receivers can decode the video or other predicted media encoding with long prediction chains. It is RECOMMENDED that this feedback message is supported.

6.1.2. Picture Loss Indication

The Picture Loss Indication is defined in Section 6.3.1 of the RTP/AVPF profile [RFC4585]. It is used by a receiver to tell the encoder that it lost the decoder context and would like to have it repaired somehow. This is semantically different from the Full Intra Request above. It is RECOMMENDED that this feedback message is supported as a loss tolerance mechanism.

6.1.3. Slice Loss Indication

The Slice Loss Indicator is defined in Section 6.3.2 of the RTP/AVPF profile [RFC4585]. It is used by a receiver to tell the encoder that it has detected the loss or corruption of one or more consecutive macroblocks, and would like to have these repaired somehow. The use of this feedback message is OPTIONAL as a loss tolerance mechanism.

6.1.4. Reference Picture Selection Indication

Reference Picture Selection Indication (RPSI) is defined in Section 6.3.3 of the RTP/AVPF profile [RFC4585]. Some video coding standards allow the use of older reference pictures than the most recent one for predictive coding. If such a codec is in used, and if the encoder has learned about a loss of encoder-decoder synchronicity, a known-as-correct reference picture can be used for future coding. The RPSI message allows this to be signalled. The use of this RTCP feedback message is OPTIONAL as a loss tolerance mechanism.

6.1.5. Temporary Maximum Media Stream Bit Rate Request

This feedback message is defined in Section 3.5.4 and 4.2.1 in CCM [RFC5104]. This message and its notification message is used by a media receiver, to inform the sending party that there is a current limitation on the amount of bandwidth available to this receiver. This can be for various reasons, and can for example be used by an RTP mixer to limit the media sender being forwarded by the mixer (without doing media transcoding) to fit the bottlenecks existing towards the other session participants. It is RECOMMENDED that this feedback message is supported.

6.2. RTP Header Extensions

The RTP specification [RFC3550] provides a capability to extend the RTP header with in-band data, but the format and semantics of the extensions are poorly specified. Accordingly, if header extensions are to be used, it is REQUIRED that they be formatted and signalled according to the general mechanism of RTP header extensions defined in [RFC5285].

As noted in [RFC5285], the requirement from the RTP specification that header extensions are "designed so that the header extension may be ignored" [RFC3550] stands. To be specific, header extensions must only be used for data that can safely be ignored by the recipient without affecting interoperability, and must not be used when the presence of the extension has changed the form or nature of the rest of the packet in a way that is not compatible with the way the stream is signalled (e.g., as defined by the payload type). Valid examples might include metadata that is additional to the usual RTP information.

The RTP rapid synchronisation header extension [RFC6051] is recommended, as discussed in Section 6.3 we also recommend the client to mixer audio level [I-D.ietf-avtext-client-to-mixer-audio-level], and consider the mixer to client audio level [I-D.ietf-avtext-mixer-to-client-audio-level] as optional feature.

It is RECOMMENDED that the mechanism to encrypt header extensions [I-D.ietf-avtcore-srtp-encrypted-header-ext] is implemented when the client-to-mixer and mixer-to-client audio level indications are in use in SRTP encrypted sessions, since the information contained in these header extensions may be considered sensitive.

Currently the other header extensions are not recommended to be included at this time. But we do include a list of the available ones for information below:

Transmission Time offsets: [RFC5450] defines a format for including an RTP timestamp offset of the actual transmission time of the RTP packet in relation to capture/display timestamp present in the RTP header. This can be used to improve jitter determination and buffer management.

Associating Time-Codes with RTP Streams: [RFC5484] defines how to associate SMPTE times codes with the RTP streams.

6.3. Rapid Synchronisation Extensions

Many RTP sessions require synchronisation between audio, video, and other content. This synchronisation is performed by receivers, using information contained in RTCP SR packets, as described in the RTP specification [RFC3550]. This basic mechanism can be slow, however, so it is RECOMMENDED that the rapid RTP synchronisation extensions described in [RFC6051] be implemented. The rapid synchronisation extensions use the general RTP header extension mechanism [RFC5285], which requires signalling, but are otherwise backwards compatible.

6.4. Client to Mixer Audio Level

The Client to Mixer Audio Level [I-D.ietf-avtext-client-to-mixer-audio-level] is an RTP header extension used by a client to inform a mixer about the level of audio activity in the packet the header is attached to. This enables a central node to make mixing or selection decisions without decoding or detailed inspection of the payload. Thus reducing the needed complexity in some types of central RTP nodes.

Assuming that the Client to Mixer Audio Level [I-D.ietf-avtext-client-to-mixer-audio-level] is published as a finished specification prior to RTCWEB's first RTP specification then it is RECOMMENDED that this extension is included.

6.5. Mixer to Client Audio Level

The Mixer to Client Audio Level header extension [I-D.ietf-avtext-mixer-to-client-audio-level] provides the client with the audio level of the different sources mixed into a common mix from the RTP mixer. Thus enabling a user interface to indicate the relative activity level of a session participant, rather than just being included or not based on the CSRC field. This is a pure optimisations of non critical functions and thus optional functionality.

Assuming that the Mixer to Client Audio Level [I-D.ietf-avtext-client-to-mixer-audio-level] is published as a finished specification prior to RTCWEB's first RTP specification then it is OPTIONAL that this extension is included.

7. Improving RTP Transport Robustness

There are some tools that can make RTP flows robust against Packet loss and reduce the impact on media quality. However they all add extra bits compared to a non-robust stream. These extra bits needs

to be considered and the aggregate bit-rate needs to be rate controlled. Thus improving robustness might require a lower base encoding quality but has the potential to give that quality with fewer errors in it.

7.1. RTP Retransmission

Support for RTP retransmission as defined by "RTP Retransmission Payload Format" [RFC4588] is RECOMMENDED.

The retransmission scheme in RTP allows flexible application of retransmissions. Only selected missing packets can be requested by the receiver. It also allows for the sender to prioritise between missing packets based on senders knowledge about their content. Compared to TCP, RTP retransmission also allows one to give up on a packet that despite retransmission(s) still has not been received within a time window.

"RTC-Web Media Transport Requirements" [I-D.cbran-rtcweb-data] raises two issues that they think makes RTP Retransmission unsuitable for RTCWEB. We here consider these issues and explain why they are in fact not a reason to exclude RTP retransmission from the tool box available to RTCWEB media sessions.

The additional latency added by [RFC4588] will exceed the latency threshold for interactive voice and video: RTP Retransmission will require at least one round trip time for a retransmission request and repair packet to arrive. Thus the general suitability of using retransmissions will depend on the actual network path latency between the end-points. In many of the actual usages the latency between two end-points will be low enough for RTP retransmission to be effective. Interactive communication with end-to-end delays of 400 ms still provide a fair quality. Even removing half of that in end-point delays allows functional retransmission between end-points on the continent. In addition in some applications one may accept temporary delay spikes to allow for retransmission of crucial codec information such an parameter sets, intra picture etc, rather than getting no media at all.

The undesirable increase in packet transmission at the point when congestion occurs: Congestion loss will impact the rate controls view of available bit-rate for transmission. When using retransmission one will have to prioritise between performing retransmissions and the quality one can achieve with ones adaptable codecs. In many use cases one prefer error free or low rates of error with reduced base quality over high degrees of error at a higher base quality.

The RTCWEB end-point implementations will need to both select when to enable RTP retransmissions based on API settings and measurements of the actual round trip time. In addition for each NACK request that a media sender receives it will need to make a prioritisation based on the importance of the requested media, the probability that the packet will reach the receiver in time for being usable, the consumption of available bit-rate and the impact of the media quality for new encodings.

To conclude, the issues raised are implementation concerns that an implementation needs to take into consideration, they are not arguments against including a highly versatile and efficient packet loss repair mechanism.

7.2. Forward Error Correction (FEC)

Support of some type of FEC to combat the effects of packet loss is beneficial, but is heavily application dependent. However, some FEC mechanisms are encumbered.

The main benefit from FEC is the relatively low additional delay needed to protect against packet losses. The transmission of any repair packets should preferably be done with a time delay that is just larger than any loss events normally encountered. That way the repair packet isn't also lost in the same event as the source data.

The amount of repair packets needed are also highly dynamically and depends on two main factors, the amount and pattern of lost packets to be recovered and the mechanism one use to derive repair data. The later choice also effects the the additional delay required to both encode the repair packets and in the receiver to be able to recover the lost packet(s).

7.2.1. Basic Redundancy

The method for providing basic redundancy is to simply retransmit an some time earlier sent packet. This is relatively simple in theory, i.e. one saves any outgoing source (original) packet in a buffer marked with a timestamp of actual transmission, some X ms later one transmit this packet again. Where X is selected to be longer than the common loss events. Thus any loss events shorter than X can be recovered assuming that one doesn't get an another loss event before all the packets lost in the first event has been received.

The downside of basic redundancy is the overhead. To provide each packet with once chance of recovery, then the transmission rate increases with 100% as one needs to send each packet twice. It is possible to only redundantly send really important packets thus

reducing the overhead below 100% for some other trade-off is overhead.

In addition the basic retransmission of the same packet using the same SSRC in the same RTP session is not possible in RTP context. The reason is that one would then destroy the RTCP reporting if one sends the same packet twice with the same sequence number. Thus one needs more elaborate mechanisms.

RTP Payload for Redundant Audio Data: This audio and text redundancy format defined in [RFC2198] allows for multiple levels of redundancy with different delay in their transmissions, as long as the source plus payload parts to be redundantly transmitted together fits into one MTU. This should work fine for most interactive use cases as both the codec bit-rates and the framing intervals normally allow for this requirement to hold. This payload format also don't increase the packet rate, as original data and redundant data are sent together. This format does not allow perfect recovery, only recovery of information deemed necessary for audio, for example the sequence number of the original data is lost.

RTP Retransmission Format: The RTP Retransmission Payload format [RFC4588] can be used to pro-actively send redundant packets using either SSRC or session multiplexing. By using different SSRCs or a different session for the redundant packets the RTCP receiver reports will be correct. The retransmission payload format is used to recover the packets original data thus enabling a perfect recovery.

Duplication Grouping Semantics in the Session Description Protocol: This [I-D.begen-mmusic-redundancy-grouping] is proposal for new SDP signalling to indicate media stream duplication using different RTP sessions, or different SSRCs to separate the source and the redundant copy of the stream.

7.2.2. Block Based

Block based redundancy collects a number of source packets into a data block for processing. The processing results in some number of repair packets that is then transmitted to the other end allowing the receiver to attempt to recover some number of lost packets in the block. The benefit of block based approaches is the overhead which can be lower than 100% and still recover one or more lost source packet from the block. The optimal block codes allows for each received repair packet to repair a single loss within the block. Thus 3 repair packets that are received should allow for any set of 3 packets within the block to be recovered. In reality one commonly

don't reach this level of performance for any block sizes and number of repair packets, and taking the computational complexity into account there are even more trade-offs to make among the codes.

One result of the block based approach is the extra delay, as one needs to collect enough data together before being able to calculate the repair packets. In addition sufficient amount of the block needs to be received prior to recovery. Thus additional delay are added on both sending and receiving side to ensure possibility to recover any packet within the block.

The redundancy overhead and the transmission pattern of source and repair data can be altered from block to block, thus allowing a adaptive process adjusting to meet the actual amount of loss seen on the network path and reported in RTCP.

The alternatives that exist for block based FEC with RTP are the following:

RTP Payload Format for Generic Forward Error Correction: This RTP payload format [RFC5109] defines an XOR based recovery packet. This is the simplest processing wise that an block based FEC scheme can be. It also results in some limited properties, as each repair packet can only repair a single loss. To handle multiple close losses a scheme of hierarchical encodings are need. Thus increasing the overhead significantly.

Forward Error Correction (FEC) Framework: This framework [I-D.ietf-fecframe-framework] defines how not only RTP packets but how arbitrary packet flows can be protected. Some solutions produced or under development in FECFRAME WG are RTP specific. There exist alternatives supporting block codes such as Reed-Salomon and Raptor.

7.2.3. Recommendations for FEC

(tbd)

8. RTP Rate Control and Media Adaptation

It is REQUIRED to have an RTP Rate Control mechanism using Media adaptation to ensure that the generated RTP flows are network friendly, and maintain the user experience in the presence of network problems.

The biggest issue is that there are no standardised and ready to use mechanism that can simply be included in RTC-Web. Thus there will be

need for the IETF to produce such a specification. A potential starting point for defining a solution is "RTP with TCP Friendly Rate Control" [rtp-tfrc].

9. RTP Performance Monitoring

RTCP does contains a basic set of RTP flow monitoring points like packet loss and jitter. There exist a number of extensions that could be included in the set to be supported. However, in most cases which RTP monitoring that is needed depends on the application, which makes it difficult to select which to include when the set of applications is very large.

10. IANA Considerations

This memo makes no request of IANA.

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

11. Security Considerations

RTP and its various extensions each have their own security considerations. These should be taken into account when considering the security properties of the complete suite. We currently don't think this suite creates any additional security issues or properties. The use of SRTP [RFC3711] will provide protection or mitigation against all the fundamental issues by offering confidentiality, integrity and partial source authentication. The guidelines in [I-D.ietf-avtcore-srtp-vbr-audio] apply when using variable bit rate (VBR) audio codecs, for example Opus.

We don't discuss the key-management aspect of SRTP in this memo, that needs to be done taking the RTC-Web communication model into account.

In the context of RTC-Web the actual security properties required from RTP are currently not fully understood. Until security goals and requirements are specified it will be difficult to determine what security features in addition to SRTP and a suitable key-management, if any, that are needed.

12. Acknowledgements

13. References

13.1. Normative References

[I-D.ietf-avtcore-srtp-encrypted-header-ext]
Lennox, J., "Encryption of Header Extensions in the Secure Real-Time Transport Protocol (SRTP)", draft-ietf-avtcore-srtp-encrypted-header-ext-00 (work in progress), June 2011.

[I-D.ietf-avtcore-srtp-vbr-audio]
Perkins, C. and J. Valin, "Guidelines for the use of Variable Bit Rate Audio with Secure RTP", draft-ietf-avtcore-srtp-vbr-audio-03 (work in progress), July 2011.

[I-D.ietf-avtext-client-to-mixer-audio-level]
Lennox, J., Ivov, E., and E. Marocco, "A Real-Time Transport Protocol (RTP) Header Extension for Client-to-Mixer Audio Level Indication", draft-ietf-avtext-client-to-mixer-audio-level-03 (work in progress), July 2011.

[I-D.ietf-avtext-mixer-to-client-audio-level]
Ivov, E., Marocco, E., and J. Lennox, "A Real-Time Transport Protocol (RTP) Header Extension for Mixer-to-Client Audio Level Indication", draft-ietf-avtext-mixer-to-client-audio-level-03 (work in progress), July 2011.

[RFC2736] Handley, M. and C. Perkins, "Guidelines for Writers of RTP Payload Format Specifications", BCP 36, RFC 2736, December 1999.

[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.

[RFC3556] Casner, S., "Session Description Protocol (SDP) Bandwidth Modifiers for RTP Control Protocol (RTCP) Bandwidth", RFC 3556, 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.

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

[RFC4961] Wing, D., "Symmetric RTP / RTP Control Protocol (RTCP)", BCP 131, RFC 4961, July 2007.

[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.

[RFC5109] Li, A., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, December 2007.

[RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, February 2008.

[RFC5285] Singer, D. and H. Desineni, "A General Mechanism for RTP Header Extensions", RFC 5285, July 2008.

[RFC5450] Singer, D. and H. Desineni, "Transmission Time Offsets in RTP Streams", RFC 5450, March 2009.

[RFC5484] Singer, D., "Associating Time-Codes with RTP Streams", RFC 5484, 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.

[RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.

[RFC6051] Perkins, C. and T. Schierl, "Rapid Synchronisation of RTP Flows", RFC 6051, November 2010.

[RFC6222] Begen, A., Perkins, C., and D. Wing, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", RFC 6222, April 2011.

13.2. Informative References

[I-D.begen-mmusic-redundancy-grouping]
Begen, A., Cai, Y., and H. Ou, "Duplication Grouping Semantics in the Session Description Protocol", draft-begen-mmusic-redundancy-grouping-01 (work in progress), June 2011.

[I-D.cbran-rtcweb-data]
Bran, C. and C. Jennings, "RTC-Web Non-Media Data Transport Requirements", draft-cbran-rtcweb-data-00 (work in progress), July 2011.

[I-D.ietf-fecframe-framework]
Watson, M., Begen, A., and V. Roca, "Forward Error Correction (FEC) Framework", draft-ietf-fecframe-framework-15 (work in progress), June 2011.

[RFC2198] Perkins, C., Kouvelas, I., Hodson, O., Hardman, V., Handley, M., Bolot, J., Vega-Garcia, A., and S. Fosse-Parisis, "RTP Payload for Redundant Audio Data", RFC 2198, September 1997.

[RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, January 2008.

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

[rtp-tfrc]
Gharai, L., "RTP with TCP Friendly Rate Control (draft-gharai-avtcore-rtp-tfrc-00)", March 2011.

Authors' Addresses

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

Email: csp@csperkins.org

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

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

Joerg Ott
Aalto University
School of Electrical Engineering
Espoo 02150
Finland

Email: jorg.ott@aalto.fi