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

C. Perkins
University of Glasgow
M. Westerlund
Ericsson
J. Ott
Aalto University
October 31, 2011

Web Real-Time Communication (WebRTC): Media Transport and Use of RTP
draft-ietf-rtcweb-rtp-usage-01

Abstract

The Web Real-Time Communication (WebRTC) framework aims to provide support for direct interactive rich communication using audio, video, collaboration, games, etc. between two peers' web-browsers. This memo describes the media transport aspects of the WebRTC framework. It specifies how the Real-time Transport Protocol (RTP) is used in the WebRTC context, and gives requirements for which RTP features, profiles, and extensions need to be supported.

Status of this Memo

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

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

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

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

Table of Contents

1. Introduction

The Real-time Transport Protocol (RTP) [RFC3550] was designed to provide a framework for delivery of audio and video teleconferencing data and other real-time media applications. This memo describes how RTP is to be used in the context of the Web Real-Time Communication (WebRTC) framework, a new activity that aims to provide support for direct, interactive, and rich communication using audio, video, collaboration, games, etc. between two peers' web-browsers.

Previous work in the IETF Audio/Video Transport Working Group, and it's successors, has been about providing a framework for real-time multimedia transport, but has not specified how the pieces of this framework should be combined. This is because the choice of building blocks and protocol features can really only be done in the context of some application. This memo proposes a set of RTP features and extensions to be implemented by applications that fit within the WebRTC application context. This includes applications such as voice over IP (VoIP), video teleconferencing, and on-demand multimedia streaming, delivered in the context of the WebRTC browser-based infrastructure.

2. Terminology

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 WebRTC specification:

REQUIRED: Functionality that is absolutely needed to make the WebRTC 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.

3. Media Transport in WebRTC

3.1. Expected Topologies

As WebRTC 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.

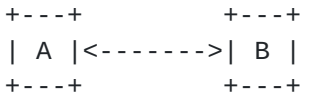

Figure 1: Point to Point

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

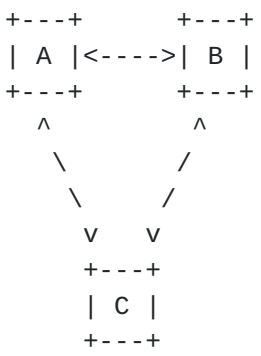

Figure 2: Multi-unicast

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 (Figure 2), 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.

This topology may be implemented as a single RTP session, spanning multiple peer to peer transport layer connections, or as several pairwise RTP sessions, one between each pair of peers. The later approach simplifies rate adaptation, but reduces the effectiveness of RTCP for debugging purposes, by limiting the ability to send third-party RTCP reports.

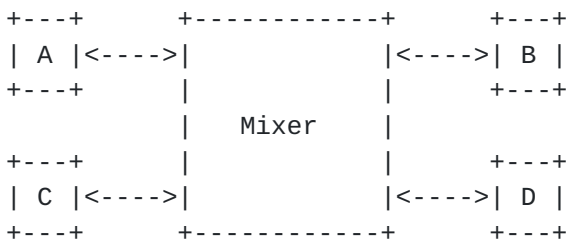

Figure 3: RTP Mixer with Only Unicast Paths

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.

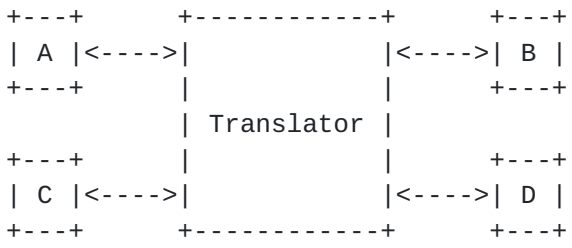

Figure 4: RTP Translator (Relay) with Only Unicast Paths

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.

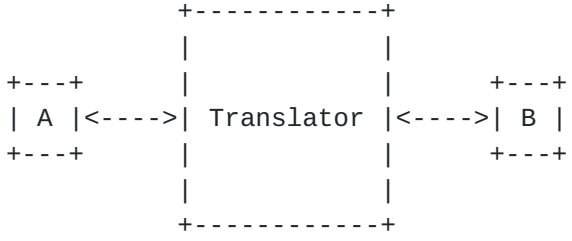

Figure 5: Translator towards Legacy end-point

To support legacy end-point (B) that don't fulfil the requirements of WebRTC 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 WebRTC results in such issues and how to handle them.

3.2. Requirements from RTP

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

3.2.1. 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 (see Section 5.1). If several RTP sessions are to be multiplexed onto a single transport layer flow, this MUST also be signalled (see Section 4.4).

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.

3.2.2. (Lack of) Signalling for Payload Format Changes

As discussed in Section 3.2.1, 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 (this is slightly modified if several RTP sessions are multiplexed onto one transport layer connection, such that an endpoint must be prepared for a source to switch between formats of the same media type at any time; see Section 4.4). RTP does not require advance signalling for changes between formats that were signalled during the session setup. This is needed for rapid rate adaptation.

4. WebRTC Use of RTP: Core Protocols

4.1. RTP and RTCP

The Real-time Transport Protocol (RTP) [RFC3550] is REQUIRED to be implemented as the media transport protocol for WebRTC. RTP itself comprises two parts: the RTP data transfer protocol, and the RTP

control protocol (RTCP). RTCP is a fundamental and integral part of the RTP protocol, and is REQUIRED to be implemented.

RTP and RTCP are flexible and extensible protocols that allow, on the one hand, choosing from a variety of building blocks and combining those to meet application needs, but on the other hand, offer the ability to create extensions where existing mechanisms are not sufficient. This memo requires a number of RTP and RTCP extensions that have been shown to be provide important functionality in the WebRTC context be implemented. It is possible that future extensions will be needed: several documents provide guidelines for the use and extension of RTP and RTCP, including Guidelines for Writers of RTP Payload Format Specifications [RFC2736] and Guidelines for Extending the RTP Control Protocol [RFC5968], and should be consulted before extending this memo.

4.2. Choice of RTP Profile

The complete specification of RTP for a particular application domain requires the choice of an RTP Profile. For WebRTC use, 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 improved RTCP timer model defined by RTP/AVPF is backwards compatible with legacy systems that implement only the RTP/AVP profile given some constraints on parameter configuration such as RTCP banwidth and "trr-int".

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. A mandatory to implement media security mechanism including keying must be required so that confidentialtiy, integrity protection and source authentication of the media stream can be

provided when desired by the user.

4.3. Choice of RTP Payload Formats

(tbd: say something about the choice of RTP Payload Format for WebRTC. If there is a mandatory to implement set of codecs, this should reference them. In any case, it should reference a discussion of signalling for the choice of codec, once that discussion reaches closure.)

4.4. RTP Session Multiplexing

An association amongst a set of participants communicating with RTP is known as an RTP session. A participant may be involved in multiple RTP sessions at the same time. In a multimedia session, each medium has typically been carried in a separate RTP session with its own RTCP packets (i.e., one RTP session for the audio, with a separate RTP session running on a different transport connection for the video; if SDP is used, this corresponds to one RTP session for each "m=" line in the SDP). WebRTC implementations of RTP are REQUIRED to implement support of multimedia sessions in this way, for compatibility with legacy systems.

In today's networks, however, with the prolific use of Network Address Translators (NAT) and Firewalls (FW), there is a desire to reduce the number of transport layer ports used by real-time media applications using RTP by combining multimedia traffic in a single RTP session. (Details of how this is to be done are tbd, but see [I-D.lennox-rtcweb-rtp-media-type-mux], [I-D.holmberg-mmusic-sdp-bundle-negotiation] and [I-D.westerlund-avtcore-multiplex-architecture].) WebRTC implementations of RTP are REQUIRED to support multiplexing of multimedia sessions onto a single RTP session according to (tbd). If such RTP session multiplexing is to be used, this MUST be negotiated during the signalling phase.

5. WebRTC Use of 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 and is recommend method for application level keep-alives of RTP sessions [RFC6263].

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. WebRTC Use of RTP: Extensions

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

6.1. Conferencing Extensions

RTP is inherently a group communication protocol. Groups can be implemented using a centralised server, multi-unicast, or IP multicast. While IP multicast was popular in early deployments, in today's practice, overlay-based conferencing dominates, typically using one or more central servers to connect endpoints in a star or flat tree topology. These central servers can be implemented in a number of ways [RFC5117], of which the following are the most common:

1. RTP Translator (Relay) with Only Unicast Paths ([RFC5117], section 3.3)

2. RTP Mixer with Only Unicast Paths ([RFC5117], section 3.4)

3. Point to Multipoint Using a Video Switching MCU ([RFC5117], section 3.5)

4. Point to Multipoint Using Content Modifying MCUs ([RFC5117], section 3.6)

As discussed in [RFC5117] section 3.5, the use of a video switching MCU makes the use of RTCP for congestion control, or any type of quality reports, very problematic. Also, as discussed in [RFC5117] section 3.6, the use of a content modifying MCU with RTCP termination breaks RTP loop detection and removes the ability for receivers to identify active senders. According only the first two options are recommended.

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. 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 REQUIRED that the mechanism to encrypt header extensions [I-D.ietf-avtcore-srtp-encrypted-header-ext] is implemented when the client-to-mixer or 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.

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. Mixer Audio Level Extensions

6.4.1. 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.

6.4.2. 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. WebRTC Use of RTP: Improving 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. 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.

"WebRTC 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 same continent. In addition, some applications 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 varies depending on the amount and pattern of packet loss to be recovered, and on the mechanism used 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 a 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 Format Support: Some RTP payload format do support basic redundancy within the RTP paylaod format itself. Examples are AMR-WB [RFC4867] and G.719 [RFC5404].

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 audio and text 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 FEC

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. WebRTC Use of RTP: Rate Control and Media Adaptation

WebRTC will be used in very varied network environment with a hetrogenous set of link technologies, including wired and wireless, interconnecting peers at different topological locations resulting in network paths with widely varying one way delays, bit-rate capacity, load levels and traffic mixes. In addition individual end-points will open one or more WebRTC sessions between one or more peers. Each of these session may contain different mixes of media and data flows. Assymetric usage of media bit-rates and number of media streams is also to be expected. A single end-point may receive zero to many simultanous media streams while itself transmitting one or more streams.

The WebRTC application is very dependent from a quality perspective on the media adapation working well so that an end-point doesn't transmit significantly more than the path is capable of handling. If it would, the result would be high levels of packet loss or delay spikes causing media degradations.

WebRTC applications using more than a single media stream of any media type or data flows has an additional concern. In this case the different flows should try to avoid affecting each other negatively. In addition in case there is a resource limiation, the available resources needs to be shared. How to share them is something the application should prioritize so that the limiation in quality or capabilities are the ones that provide the least affect on the application.

This hetrogenous situation results in a requirement to have functionality that adapts to the available capacity and that competes fairly with other network flows. If it would not compete fairly enough WebRTC could be used as an attack method for starving out other traffic on specific links as long as the attacker is able to create traffic across a specific link. This is not far-fetched for a web-service capable of attracting large number of end-points and use the service, combined with BGP routing state a server could pick client pairs to drive traffic to specific paths.

The above estalish a clear need based on several reasons why there need to be a well working media adaptation mechanism. This mechanism also have a number of requirements on what services it should provide and what performance it needs to provide.

The biggest issue is that there are no standardised and ready to use mechanism that can simply be included in WebRTC Thus there will be need for the IETF to produce such a specification. Therefore the suggested way forward is to specify requirements on any solution for the media adaptation. These requirements is for now proposed to be documented in this specification. In addition a proposed detailed solution will be developed, but is expected to take longer time to finalize than this document.

8.1. Rate Control Requirements

Note: This section does not yet have WG consensus.

This section provides a number of requirements on an media adaptation/congestion control solution for WebRTC.

1. All WebRTC media streams MUST be congestion-controlled. (The same requirement apply to data streams)

2. The congestion algorithms used MUST cause WebRTC streams to act reasonably fairly with TCP and other congestion-controlled flows, such as DCCP and TFRC, and other WebRTC flows. Note that WebRTC involves multiple data flows which "normally" would be separately congestion-controlled.

3. The congestion control mechanism MUST be possible to realize between two indendently implemented WebRTC end-points.

4. The congestion control algorithm SHOULD attempt to minimize the media-stream end-to-end delays between the participants, by controlling bandwidth appropriately.

5. The congestion control SHOULD allow for prioritization and shifting of banwidth between media flows. In other words, if one flow on the same path as another has to adjust its bit-rate the other flow can perform that adjustment instead, or divided between the flows.

Thus it is REQUIRED to have an implementation of an RTP Rate Control mechanism fulfilling the above requirements.

8.2. RTCP Limiations

Experience with the congestion control algorithms of TCP [RFC5681], TFRC [RFC5348], and DCCP [RFC4341], [RFC4342], [RFC4828], has shown that feedback on packet arrivals needs to be sent roughly once per round trip time. We note that the capabilities of real-time media traffic to adapt to changing path conditions may be less rapid than

for the elastic applications TCP was designed for, but frequent feedback is still required to allow the congestion control algorithm to track the path dynamics.

The total RTCP bandwidth is limited in its transmission rate to a fraction of the RTP traffic (by default 5%). RTCP packets are larger than, e.g., TCP ACKs (even when non-compound RTCP packets are used). The media stream bit rate thus limits the maximum feedback rate as a function of the mean RTCP packet size.

Interactive communication may not be able to afford waiting for packet losses to occur to indicate congestion, because an increase in playout delay due to queuing (most prominent in wireless networks) may easily lead to packets being dropped due to late arrival at the receiver. Therefore, more sophisticated cues may need to be reported -- to be defined in a suitable congestion control framework as noted above -- which, in turn, increase the report size again. For example, different RTCP XR report blocks (jointly) provide the necessary details to implement a variety of congestion control algorithms, but the (compound) report size grows quickly.

In group communication, the share of RTCP bandwidth needs to be shared by all group members, reducing the capacity and thus the reporting frequency per node.

Example: assuming 512 kbit/s video yields 3200 bytes/s RTCP bandwidth, split across two entities in a point-to-point session. An endpoint could thus send a report of 100 bytes about every 70ms or for every other frame in a 30 fps video.

8.3. Legacy Interop Limitations

Congestion control interoperability with most type of legacy devices, even using an translator could be difficult. There are numerous reasons for this:

No RTCP Support: There exist legacy implementations that does not even implement RTCP at all. Thus no feedback at all is provided.

RTP/AVP Minimal RTCP Interval of 5s: RTP [RFC3550] under the RTP/AVP profile specifies a recommended minimal fixed interval of 5 seconds. Sending RTCP report blocks as seldom as 5 seconds makes it very difficult for a sender to use these reports and react to any congestion event.

RTP/AVP Scaled Minimal Interval: If a legacy device uses the scaled minimal RTCP compound interval, the "RECOMMENDED value for the reduced minimum in seconds is 360 divided by the session bandwidth in kilobits/second" ([RFC3550], section 6.2). The minimal interval drops below a second, still several times the RTT in almost all paths in the Internet, when the session bandwidht becomes 360 kbps. A session bandwidth of 1 Mbps still has a minimal interval of 360 ms. Thus, with the exception for rather high bandwidth sessions, getting frequent enough RTCP Report Blocks to report on the order of the RTT is very difficult as long as the legacy device uses the RTP/AVP profile.

RTP/AVPF Supporting Legacy Device: If a legacy device supports RTP/AVPF, then that enables negotation of important parameters for frequent reporting, such as the "trr-int" parameter, and the possibility that the end-point supports some useful feedback format for congestion control purpose such as TMMBR [RFC5104].

It has been suggested on the RTCWEB mailing list that if interoperating with really limited legacy devices an WebRTC end-point may not send more than 64 kbps of media streams, to avoid it causing massive congestion on most paths in the Internet when communicating with a legacy node not providing sufficient feedback for effective congestion control. This warrants further discussion as there is clearly a number of link layers that don't even provide that amount of bit-rate consistently, and that assumes no competing traffic.

9. WebRTC Use of 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.

Exposing some metrics in the WebRTC API should be considered allowing the application to gather the measurements of interest. However, security implications for the different data sets exposed will need to be considered in this.

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. A mandatory to implement media security solution will be required to be picked. We currently don't discuss the key-management aspect of SRTP in this memo, that needs to be done taking the WebRTC communication model into account.

The guidelines in [I-D.ietf-avtcore-srtp-vbr-audio] apply when using variable bit rate (VBR) audio codecs, for example Opus or the Mixer audio level header extensions.

Security considerations for the WebRTC work are discussed in [I-D.ietf-rtcweb-security].

12. Acknowledgements

The authors would like to thank Harald Alvestrand, Cary Bran, and Cullen Jennings for valuable feedback.

13. References

13.1. Normative References

[I-D.holmberg-mmusic-sdp-bundle-negotiation]
Holmberg, C. and H. Alvestrand, "Multiplexing Negotiation Using Session Description Protocol (SDP) Port Numbers", draft-holmberg-mmusic-sdp-bundle-negotiation-00 (work in progress), October 2011.

[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.

[I-D.ietf-rtcweb-security] Rescorla, E., "Security Considerations for RTC-Web", draft-ietf-rtcweb-security-01 (work in progress), October 2011.

[I-D.lennox-rtcweb-rtp-media-type-mux] Lennox, J. and J. Rosenberg, "Multiplexing Multiple Media Types In a Single Real-Time Transport Protocol (RTP) Session", draft-lennox-rtcweb-rtp-media-type-mux-00 (work in progress), October 2011.

[I-D.westerlund-avtcore-multiplex-architecture] Westerlund, M., Burman, B., and C. Perkins, "RTP Multiplexing Architecture", draft-westerlund-avtcore-multiplex-architecture-00 (work in progress), October 2011.

[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.

[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.

[RFC4341] Floyd, S. and E. Kohler, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control", RFC 4341, March 2006.

[RFC4342] Floyd, S., Kohler, E., and J. Padhye, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC)", RFC 4342, March 2006.

[RFC4828] Floyd, S. and E. Kohler, "TCP Friendly Rate Control (TFRC): The Small-Packet (SP) Variant", RFC 4828, April 2007.

[RFC4867] Sjoberg, J., Westerlund, M., Lakaniemi, A., and Q. Xie, "RTP Payload Format and File Storage Format for the Adaptive Multi-Rate (AMR) and Adaptive Multi-Rate Wideband (AMR-WB) Audio Codecs", RFC 4867, April 2007.

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

[RFC5348] Floyd, S., Handley, M., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 5348, September 2008.

[RFC5404] Westerlund, M. and I. Johansson, "RTP Payload Format for G.719", RFC 5404, January 2009.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, September 2009.

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

[RFC6263] Marjou, X. and A. Sollaud, "Application Mechanism for Keeping Alive the NAT Mappings Associated with RTP / RTP Control Protocol (RTCP) Flows", RFC 6263, June 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