

Audio/Video Transport Core
Maintenance
Internet-Draft
Intended status: Standards Track
Expires: January 15, 2014

A. Williams
Audinate
K. Gross
AVA Networks
R. van Brandenburg
H. Stokking
TNO
July 14, 2013

RTP Clock Source Signalling
draft-ietf-avtcore-clksrc-05

Abstract

NTP format timestamps are used by several RTP protocols for synchronisation and statistical measurements. This memo specifies SDP signalling identifying timestamp reference clock sources and SDP signalling identifying the media clock sources in a multimedia session.

Requirements Language

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

Status of this Memo

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

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

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

This Internet-Draft will expire on January 15, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	4
2. Applications	4
3. Definitions	5
4. Timestamp Reference Clock Source Signalling	6
4.1. Clock synchronization	6
4.2. Identifying NTP Reference Clocks	7
4.3. Identifying PTP Reference Clocks	7
4.4. Identifying Global Reference Clocks	9
4.5. Private Reference Clocks	9
4.6. Local Reference Clocks	9
4.7. Traceable Reference Clocks	9
4.8. SDP Signalling of Timestamp Reference Clock Source	9
4.8.1. Examples	12
5. Media Clock Source Signalling	13
5.1. Asynchronously Generated Media Clock	13
5.2. Direct-Referenced Media Clock	14
5.3. Stream-Referenced Media Clock	14
5.4. SDP Signalling of Media Clock Source	15
5.5. Examples	17
6. Signalling Considerations	19
6.1. Usage in Offer/Answer	19
6.1.1. Indicating Support for Clock Source Signalling	20
6.1.2. Timestamp Reference Clock	20
6.1.3. Media Clock	20
6.2. Usage Outside of Offer/Answer	21
7. Security Considerations	21
8. IANA Considerations	21
8.1. Reference Clock SDP Parameter	22
8.2. Media Clock SDP Parameter	22
8.3. Timestamp Reference Clock Source Parameters Registry	23
8.4. Media Clock Source Parameters Registry	24
8.5. Source-level Attributes	24
8.5.1. Source-level Reference Clock Attribute	24
8.5.2. Source-level Media Clock Attribute	24
9. References	25
9.1. Normative References	25
9.2. Informative References	26
Authors' Addresses	27

1. Introduction

RTP protocols use NTP format timestamps to facilitate multimedia session synchronisation and for providing estimates of round trip time (RTT) and other statistical parameters.

Information about media clock timing exchanged in NTP format timestamps may come from a clock which is synchronised to a global time reference, but this cannot be assumed nor is there a standardised mechanism available to indicate that timestamps are derived from a common reference clock. Therefore, RTP implementations typically assume that NTP timestamps are taken using unsynchronised clocks and must compensate for absolute time differences and rate differences. Without a shared reference clock, RTP can time align flows from the same source at a given receiver using relative timing, however tight synchronisation between two or more different receivers (possibly with different network paths) or between two or more senders is not possible.

High performance AV systems often use a reference media clock distributed to all devices in the system. The reference media clock is often distinct from the reference clock used to provide timestamps. A reference media clock may be provided along with an audio or video signal interface, or via a dedicated clock signal (e.g. genlock [SMPTE-318-1999] or audio word clock [AES11-2009]). If sending and receiving media clocks are known to be synchronised to a common reference clock, performance can be improved by minimising buffering and avoiding rate conversion.

This specification defines SDP signalling of timestamp reference clock sources and media reference clock sources.

2. Applications

Timestamp reference clock source and media clock signalling benefit applications requiring synchronised media capture or playout and low latency operation.

Examples include, but are not limited to:

Social TV : RTCP for inter-destination media synchronization
[I-D.ietf-avtcore-idms] defines social TV as the combination of media content consumption by two or more users at different devices and locations and real-time communication between those users. An example of Social TV, is where two or more users are watching the same television broadcast at different devices and/or locations, while communicating with each other using text, audio

and/or video. A skew in the media playout of the two or more users can have adverse effects on their experience. A well-known use case here is one friend experiencing a goal in a football match well before or after other friends.

Video Walls : A video wall consists of multiple computer monitors, video projectors, or television sets tiled together contiguously or overlapped in order to form one large screen. Each of the screens reproduces a portion of the larger picture. In some implementations, each screen or projector may be individually connected to the network and receive its portion of the overall image from a network-connected video server or video scaler. Screens are refreshed at 50 or 60 hertz or potentially faster. If the refresh is not synchronized, the effect of multiple screens acting as one is broken.

Networked Audio : Networked loudspeakers, amplifiers and analogue I/O devices transmitting or receiving audio signals via RTP can be connected to various parts of a building or campus network. Such situations can for example be found in large conference rooms, legislative chambers, classrooms (especially those supporting distance learning) and other large-scale environments such as stadiums. Since humans are more susceptible to differences in audio delay, this use case needs even more accuracy than the video wall use case. Depending on the exact application, the need for accuracy can then be in the range of microseconds [1].

Sensor Arrays : Sensor arrays contain many synchronised measurement elements producing signals which are then combined to form an overall measurement. Accurate capture of the phase relationships between the various signals arriving at each element of the array is critically important for proper operation. Examples include towed or fixed sonar arrays, seismic arrays and phased arrays used in radar applications, for instance.

3. Definitions

The following definitions are used in this draft:

media level : Media level information applies to a single SDP media stream. In an SDP description, media-level information appears after each "m"-line.

multimedia session : A set of multimedia senders and receivers as well as the data streams flowing from senders to receivers. The Session Description Protocol (SDP) [RFC4566] describes multimedia sessions.

RTP media stream : A single stream of RTP packets identified by an RTP SSRC.

RTP media sender : The device generating an associated RTP media stream

SDP media stream : An RTP session potentially containing more than one RTP source. SDP media descriptions beginning with an "m"-line define the parameters of an SDP media stream.

session level : Session level information applies to an entire multimedia session. In an SDP description, session-level information appears before the first "m"-line.

source level : Source level information applies to a RTP media stream Source-Specific Media Attributes in the Session Description Protocol (SDP) [RFC5576] defines how source-level information is included into an SDP session description.

traceable time : A clock is considered to provide traceable time if it can be proven to be synchronised to International Atomic Time (TAI). Coordinated Universal Time (UTC) is a time standard synchronized to TAI. UTC is therefore also considered traceable time once leap seconds have been taken into account. GPS [IS-GPS-200F] is commonly used to provide a TAI traceable time reference. Some network time synchronisation protocols (e.g. PTP [IEEE1588-2008], NTP) can explicitly indicate that the master clock is providing a traceable time reference over the network.

4. Timestamp Reference Clock Source Signalling

The NTP format timestamps used by RTP are taken by reading a local real-time clock at the sender or receiver. This local clock may be synchronised to another clock (time source) by some means or it may be unsynchronised. A variety of methods are available to synchronise local clocks to a reference time source, including network time protocols (e.g. NTP [RFC5905], PTP [IEEE1588-2008]) and radio clocks (e.g. GPS [IS-GPS-200F]).

The following sections describe and define SDP signalling, indicating whether and how the local timestamping clock in an RTP sender/receiver is synchronised to a reference clock.

4.1. Clock synchronization

Two or more local clocks that are sufficiently synchronised will produce timestamps for a given RTP event can be used as if they came

from the same clock. Providing they are sufficiently synchronised, timestamps produced in one RTP sender or receiver can be directly compared to a local clock in another RTP sender or receiver.

The accuracy of synchronisation required is application dependent. See Applications (Section 2) section for a discussion of applications and their corresponding requirements. To serve as a reference clock, clocks must minimally be syntonised (exactly frequency matched) to one another.

Sufficient synchronisation can typically be achieving by using a network time protocol (e.g. NTP, 802.1AS, IEEE 1588-2008) to synchronize all devices to a single master clock.

Another approach is to use clocks providing a global time reference (e.g. GPS, Galileo). This concept may be used in conjunction with network time protocols as some protocols (e.g. PTP, NTP) allow master clocks to indicate explicitly that they are providing traceable time.

4.2. Identifying NTP Reference Clocks

A single NTP server is identified by hostname (or IP address) and an optional port number. If the port number is not indicated, it is assumed to be the standard NTP port (123).

Two or more NTP servers MAY be listed at the same level in the session description to indicate that all of the listed servers deliver the same reference time and may be used interchangeably. RTP senders and receivers are assured proper synchronization regardless of which server they choose and, in support of fault tolerance, may switch servers while streaming.

4.3. Identifying PTP Reference Clocks

The IEEE 1588 Precision Time Protocol (PTP) family of clock synchronisation protocols provides a shared reference clock in an network - typically a LAN. IEEE 1588 provides sub-microsecond synchronisation between devices on a LAN and typically locks within seconds at startup. With support from Ethernet switches, IEEE 1588 protocols can achieve nanosecond timing accuracy in LANs. Network interface chips and cards supporting hardware time-stamping of timing critical protocol messages are also available.

Three flavours of IEEE 1588 are in use today:

- o IEEE 1588-2002 [IEEE1588-2002]: the original "Standard for a Precision Clock Synchronization Protocol for Networked Measurement

and Control Systems". This is also known as IEEE1588v1 or PTPv1.

- o IEEE 1588-2008 [IEEE1588-2008]: the second version of the "Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems". This is a revised version of the original IEEE1588-2002 standard and is also known as IEEE1588v2 or PTPv2. IEEE 1588-2008 is not protocol compatible with IEEE 1588-2002.
- o IEEE 802.1AS [IEEE802.1AS-2011]: "Timing and Synchronization for Time Sensitive Applications in Bridged Local Area Networks". This is a Layer-2 only profile of IEEE 1588-2008 for use in Audio/Video Bridged LANs as described in IEEE 802.1BA-2011 [IEEE802.1BA-2011].

Each IEEE 1588 clock is identified by a globally unique EUI-64 called a "ClockIdentity". A slave clock using one of the IEEE 1588 family of network time protocols acquires the ClockIdentity/EUI-64 of the grandmaster clock that is the ultimate source of timing information for the network. A boundary clock which is itself slaved to another boundary clock or the grandmaster passes the grandmaster ClockIdentity through to its slaves.

Several instances of the IEEE 1588 protocol may operate independently on a single network, forming distinct PTP domains, each of which may have a different grandmaster clock. As the IEEE 1588 standards have developed, the definition of PTP domains has changed. IEEE 1588-2002 identifies protocol subdomains by a textual name, but IEEE 1588-2008 identifies protocol domains using a numeric domain number. 802.1AS is a Layer-2 profile of IEEE 1588-2008 supporting a single numeric clock domain (0).

When PTP domains are signalled via SDP, senders and receivers SHOULD check that both grandmaster ClockIdentity and PTP domain match when determining clock equivalence.

Two or more IEEE 1588 clocks MAY be listed at the same level in the session description to indicate that all of the listed clocks are candidate grandmaster clocks for the domain or deliver the same reference time and may be used interchangeably. RTP senders and receivers are assured proper synchronization regardless of which synchronization source they choose and, in support of fault tolerance, may switch reference clock source while streaming.

The PTP protocols employ a distributed election protocol called the "Best Master Clock Algorithm" (BMCA) to determine the active clock master. The clock master choices available to BMCA can be restricted or biased by configuration parameters to influence the election process. In some systems it may be desirable to limit the number of

possible PTP clock masters to avoid the need to re-signal timestamp reference clock sources when the clock master changes.

4.4. Identifying Global Reference Clocks

Global reference clocks provide a source of traceable time, typically via a hardware radio receiver interface. Examples include GPS and Galileo. Apart from the name of the reference clock system, no further identification is required.

4.5. Private Reference Clocks

In other systems, all RTP senders and receivers may use a timestamp reference clock that is not provided by one of the methods listed above. Examples may include the reference time information provided by digital television or cellular services. These sources are identified as "private" reference clocks. All RTP senders and receivers in a session using a private reference clock are assumed to have a mechanism outside this specification for determining whether their timestamp reference clocks are equivalent.

4.6. Local Reference Clocks

RFC 3550 allows senders and receivers to either use a local wallclock reference for their NTP timestamps or, by setting the timestamp field to 0, to supply no timestamps at all. Both are common practice in embedded RTP implementations. These clocks are identified as "local" and can only be assumed to be equivalent to clocks originating from the same device.

4.7. Traceable Reference Clocks

A timestamp reference clock source may be labelled "traceable" if it is known to be delivering traceable time. Providing adjustments are made for differing epochs, timezones and leap seconds, timestamps taken using clocks synchronised to a traceable time source can be directly compared even if the clocks are synchronised to different sources or via different mechanisms.

Since all NTP and PTP servers providing traceable time can be directly compared, it is not necessary to identify traceable time servers by protocol address or other identifiers.

4.8. SDP Signalling of Timestamp Reference Clock Source

Specification of the timestamp reference clock source may be at any or all levels (session, media or source) of an SDP description (see level definitions (Section 3) earlier in this document for more

information).

Timestamp reference clock source signalling included at session-level provides default parameters for all RTP sessions and sources in the session description. More specific signalling included at the media level overrides default session level signalling. More specific signalling included at the source level overrides default media level signalling.

If timestamp reference clock source signalling is included anywhere in an SDP description, it must be properly defined for all levels in the description. This may simply be achieved by providing default signalling at the session level.

Timestamp reference clock parameters may be repeated at a given level (i.e. for a session or source) to provide information about additional servers or clock sources. If the attribute is repeated at a given level, all clocks described at that level are assumed to be equivalent. Traceable time sources **MUST NOT** be mixed with non-traceable time sources at any given level.

Note that clock source parameters may change from time to time, for example, as a result of a PTP clock master election. The SIP [RFC3261] protocol supports re-signalling of updated SDP information, however other protocols may require additional notification mechanisms. "refclk-sl" is used to describe a reference clock at the source level. "refclk" is used to describe a reference clock at session or media levels.

Figure 1 shows the ABNF [RFC5234] grammar for the SDP reference clock source information.

```

refclk-sl = "a=ssrc:" integer SP timestamp-refclk

timestamp-refclk = "a=ts-refclk:" clksrc CRLF
clksrc = ntp / ptp / gps / gal / local / private / clksrc-ext

ntp
    = "ntp=" ntp-server-addr
ntp-server-addr = host [ ":" port ]
ntp-server-addr =/ "traceable"

ptp
    = "ptp=" ptp-version ":" ptp-server
ptp-version     = "IEEE1588-2002"
ptp-version     =/ "IEEE1588-2008"
ptp-version     =/ "IEEE802.1AS-2011"
ptp-version     =/ ptp-version-ext
ptp-version-ext = token

ptp-server      = ptp-gmid [ ":" ptp-domain ] / "traceable"
ptp-gmid        = EUI64
ptp-domain      = ptp-domain-name / ptp-domain-nmbr
ptp-domain-name = "domain-name=" 16ptp-domain-char
ptp-domain-char = %x21-7E / %x00
                  ; allowed characters: 0x21-0x7E (IEEE 1588-2002)
ptp-domain-nmbr = "domain-nmbr=" %x00-7f
                  ; allowed number range: 0-127 (IEEE 1588-2008)

gps      = "gps"
gal      = "gal"
local    = "local"
private  = "private" [ ":" "traceable" ]

clksrc-ext = token

host        = hostname / IPv4address / IPv6reference
hostname    = *( domainlabel "." ) toplabel [ "." ]
toplabel    = ALPHA / ALPHA *( alphanum / "-" ) alphanum
domainlabel = alphanum
              / alphanum *( alphanum / "-" ) alphanum
IPv4address = 1*3DIGIT "." 1*3DIGIT "." 1*3DIGIT "." 1*3DIGIT
IPv6reference = "[" IPv6address "]"
IPv6address  = hexpart [ ":" IPv4address ]
hexpart      = hexseq / hexseq ":" [ hexseq ] / ":" [ hexseq ]
hexseq       = hex4 *( ":" hex4 )
hex4         = 1*4HEXDIG

port = 1*DIGIT

EUI64 = 7(2HEXDIG "-") 2HEXDIG

```

Figure 1: Timestamp Reference Clock Source Signalling

4.8.1. Examples

Figure 2 shows an example SDP description with a timestamp reference clock source defined at the session level.

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 192.0.2.1
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.example.com/seminars/sdp.pdf
e=j.doe@example.com (Jane Doe)
c=IN IP4 233.252.0.1/64
t=2873397496 2873404696
a=recvonly
a=ts-refclk:ntp=traceable
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 99
a=rtpmap:99 h263-1998/90000
```

Figure 2: Timestamp reference clock definition at the session level

Figure 3 shows an example SDP description with timestamp reference clock definitions at the media level overriding the session level defaults.

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 192.0.2.1
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.example.com/seminars/sdp.pdf
e=j.doe@example.com (Jane Doe)
c=IN IP4 233.252.0.1/64
t=2873397496 2873404696
a=recvonly
a=ts-refclk:local
m=audio 49170 RTP/AVP 0
a=ts-refclk:ntp=203.0.113.10
a=ts-refclk:ntp=198.51.100.22
m=video 51372 RTP/AVP 99
a=rtpmap:99 h263-1998/90000
a=ts-refclk:ptp=IEEE802.1AS-2011:39-A7-94-FF-FE-07-CB-D0
```

Figure 3: Timestamp reference clock definition at the media level

Figure 4 shows an example SDP description with a timestamp reference clock definition at the source level overriding the session level

default.

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 192.0.2.1
s=SDP Seminar
i=A Seminar on the session description protocol
u=http://www.example.com/seminars/sdp.pdf
e=j.doe@example.com (Jane Doe)
c=IN IP4 233.252.0.1/64
t=2873397496 2873404696
a=recvonly
a=ts-refclk:local
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 99
a=rtpmap:99 h263-1998/90000
a=ssrc:12345 ts-refclk:ptp=IEEE802.1AS-2011:39-A7-94-FF-FE-07-CB-D0
```

Figure 4: Timestamp reference clock signalling at the source level

5. Media Clock Source Signalling

The media clock source for a stream determines the timebase used to advance the RTP timestamps included in RTP packets. The media clock may be asynchronously generated by the sender, it may be generated in fixed relationship to the reference clock or it may be generated with respect to another stream on the network (which is presumably being received by the sender).

5.1. Asynchronously Generated Media Clock

In the simplest sender implementation, the sender generates media by sampling audio or video according to a free-running local clock. The RTP timestamps in media packets are advanced according to this media clock and packet transmission is typically timed to regular intervals on this timeline. The sender may or may not include an NTP timestamp in sender reports to allow mapping of this asynchronous media clock to a reference clock.

The asynchronously generated media clock is the assumed mode of operation when there is no signalling of media clock source. Alternatively, asynchronous media clock may be explicitly signalled.

```
a=mediaclock:sender
```

5.2. Direct-Referenced Media Clock

A media clock may be directly derived from a reference clock. For this case it is required that a reference clock be specified with an `a=ts-refclk` attribute (Section 4.8).

The signalling optionally indicates a media clock offset value. The offset indicates the RTP timestamp value at the epoch (time of origin) of the reference clock. If no offset is signalled, the offset can be inferred at the receiver by examining RTCP sender reports which contain NTP and RTP timestamps which combined define a mapping.

A rate modifier may be specified. The modifier is expressed as the ratio of two integers and modifies the rate specified or implied by the media description by this ratio. If omitted, the rate is assumed to be the exact rate specified or implied by the media format. For example, without a rate specification, the media clock for an 8 kHz G.711 audio stream will advance exactly 8000 units for each second advance in the reference clock from which it is derived.

The rate modifier is primarily useful for accommodating certain "oddball" audio sample rates associated with NTSC video (see Figure 7). Modified rates are not advised for video streams which generally use a 90 kHz RTP clock regardless of frame rate or sample rate used for embedded audio.

```
a=mediaclock:direct[=<offset>] [rate=<rate numerator>/<rate
denominator>]
```

5.3. Stream-Referenced Media Clock

A common synchronisation architecture for audio/visual systems involves distributing a reference media clock from a master device to a number of slave devices, typically by means of a cable. Examples include audio word clock distribution and video black burst distribution. In this case, the media clock is locally generated, often by a crystal oscillator and is not locked to a timestamp reference clock.

To support this architecture across a network, a master clock identifier is associated with an RTP media stream carrying media clock timing information from a master device. The master clock identifier represents a media clock source in the master device. Slave devices in turn associate the master media clock identifier with streams they transmit, signalling the synchronisation relationship between the master and slave devices.

Slave devices recover media clock timing from the clock master stream, using it to synchronise the slave media clock with the master. Timestamps in the master clock RTP media stream are taken using the timestamp reference clock shared by the master and slave devices. The timestamps communicate information about media clock timing (rate, phase) from the master to the slave devices. Timestamps are communicated in the usual RTP fashion via RTCP SRs, or via the RFC6051 [RFC6051] header extension. The stream media format may indicate other clock information, such as the nominal rate.

Note that slaving of a device media clock to a master device does not affect the usual RTP lip sync / time alignment algorithms. Time aligned playout of two or more RTP sources still relies upon NTP timestamps supplied via RTCP SRs or by the RFC6051 timestamp header extension.

In a given system, master clock identifiers must be unique. Such identifiers MAY be manually configured, however 17 octet string identifiers SHOULD be generated according to the "short-term persistent RTCP CNAME" algorithm as described in RFC6222 [RFC6222].

A reference stream can be an RTP stream or AVB stream based on the IEEE 1722 [IEEE1722] standard.

An RTP clock master stream SHOULD be identified at the source level by an SSRC [RFC5576] and master clock identifier. If master clock identifiers are declared at the media or session level, all RTP sources at or below the level of declaration MUST provide equivalent timing to a slave receiver.

```
a=ssrc:<master-media-clock-stream-ssrc> mediack:master-id=<media-  
clock-master-id>
```

An RTP media sender indicates that it is slaved to a media clock master via a clock master identifier:

```
a=mediack:master-id=<media-clock-master-id>
```

An RTP media sender indicates that it is slaved to an IEEE 1722 clock master via a stream identifier (an EUI-64):

```
a=mediack:IEEE1722=<StreamID>
```

5.4. SDP Signalling of Media Clock Source

Specification of the media clock source may be at any or all levels (session, media or source) of an SDP description (see level definitions (Section 3) earlier in this document for more

information).

Media clock source signalling included at session level provides default parameters for all RTP sessions and sources in the session description. More specific signalling included at the media level overrides default session level signalling. Further, source-level signalling overrides media clock source signalling at the enclosing media level and session level.

Media clock source signalling may be present or absent on a per-stream basis. In the absence of media clock source signals, receivers assume an asynchronous media clock generated by the sender.

Media clock source parameters may be repeated at a given level (i.e. for a session or source) to provide information about additional clock sources. If the attribute is repeated at a given level, all clocks described at that level are comparable clock sources and may be used interchangeably.

Figure 5 shows the ABNF [RFC5234] grammar for the SDP media clock source information. "mediaclock-master" is used to declare a master media clock reference at the source level. "timestamp-mediaclock-sl" is a media clock description used at the source level. "timestamp-mediaclock" is a media clock description at the session or media level.


```

mediaclock-master = "a:ssrc:" integer SP clk-master-id
clk-master-id = "mediaclock:master-id=" master-id
timestamp-mediack-s1 = "a:ssrc" integer SP timestamp-mediack
timestamp-mediack = "a=mediack:" mediack
mediack = sender / refclk / streamid / mediack-ext
sender = "sender" sender-ext
sender-ext = token
refclk = "direct" [ "=" 1*DIGIT ] [rate] [direct-ext]
rate = [ SP "rate=" integer "/" integer ]
direct-ext = token

streamid = "master-id=" master-id
streamid =/ "IEEE1722=" avb-stream-id
streamid =/ streamid-ext

master-id = EUI48
avb-stream-id = EUI64

EUI48 = 5(2HEXDIG ":") 2HEXDIG
EUI64 = 7(2HEXDIG ":") 2HEXDIG

streamid-ext = token

mediack-ext = token [SP byte-string]

```

Figure 5: Media Clock Source Signalling

5.5. Examples

Figure 6 shows an example SDP description 8 channels of 24-bit, 48 kHz audio transmitted as a multicast stream. Media clock is derived directly from an IEEE 1588-2008 reference.

```
v=0
o=- 1311738121 1311738121 IN IP4 192.0.2.1
c=IN IP4 233.252.0.1/64
s=
t=0 0
m=audio 5004 RTP/AVP 96
a=rtpmap:96 L24/48000/8
a=sendonly
a=ts-refclk:ptp=IEEE1588-2008:39-A7-94-FF-FE-07-CB-D0:0
a=mediaclk:direct=963214424
```

Figure 6: Media clock directly referenced to IEEE 1588-2008

Figure 7 shows an example SDP description 2 channels of 24-bit, 44056 kHz NTSC "pull-down" media clock derived directly from an IEEE 1588-2008 reference clock

```
v=0
o=- 1311738121 1311738121 IN IP4 192.0.2.1
c=IN IP4 233.252.0.1/64
s=
t=0 0
m=audio 5004 RTP/AVP 96
a=rtpmap:96 L24/44100/2
a=sendonly
a=ts-refclk:ptp=IEEE1588-2008:39-A7-94-FF-FE-07-CB-D0:0
a=mediaclk:direct=963214424 rate=1000/1001
```

Figure 7: "Oddball" sample rate directly referenced to IEEE 1588-2008

Figure 8 shows the same 48 kHz audio transmission from Figure 6 with media clock derived from another RTP stream.

```
v=0
o=- 1311738121 1311738121 IN IP4 192.0.2.1
c=IN IP4 233.252.0.1/64
s=
t=0 0
m=audio 5004 RTP/AVP 96
a=rtpmap:96 L24/48000/2
a=sendonly
a=ts-refclk:ptp=IEEE1588-2008:39-A7-94-FF-FE-07-CB-D0:0
a=mediaclk:master-id=00:60:2b:20:12:1f
```

Figure 8: RTP stream with media clock slaved to a master device

Figure 9 shows the same 48 kHz audio transmission from Figure 6 with media clock derived from an IEEE 1722 AVB stream.

```
v=0
o=- 1311738121 1311738121 IN IP4 192.0.2.1
c=IN IP4 233.252.0.1/64
s=
t=0 0
m=audio 5004 RTP/AVP 96
a=rtpmap:96 L24/48000/2
a=sendonly
a=ts-refclk:ptp=IEEE1588-2008:39-A7-94-FF-FE-07-CB-D0:0
a=mediaclock:IEEE1722=38-D6-6D-8E-D2-78-13-2F
```

Figure 9: RTP stream with media clock slaved to an IEEE1722 master device

6. Signalling Considerations

Signaling of timestamp reference clock source (Section 4.8) and media clock source (Section 5.4) is defined to be used either by applications that implement the SDP Offer/Answer model [RFC3264] or by applications that use SDP to describe media and transport configurations.

A description SHOULD include both reference clock signalling and media clock signalling. If no reference clock is available, this SHOULD be signalled as a local reference (Section 4.6).

When no media clock signalling is present, an asynchronous media clock (Section 5.1) MUST be assumed. When no reference clock signalling is present, a local reference clock (Section 4.6) MUST be assumed.

If a reference clock is not signalled or a local reference is specified, the corresponding media clock may be established as rate synchronised with no assurance of time synchronisation.

When the description signals a direct-referenced media clock (Section 5.2), reference clock signalling is REQUIRED. Asynchronous and stream-referenced media clocks (Section 5.3) MAY be specified with or without a reference clock signalling.

6.1. Usage in Offer/Answer

During offer/answer, clock source signalling via SDP uses a declarative model. Supported media and/or reference clocks are specified in the offered SDP description. The answerer may accept or reject the offer in an application-specific way depending on the clocks that are available and the clocks that are offered. For

example, an answerer may choose to accept an offer that lacks a common clock by falling back to a lower performance mode of operation (e.g. by assuming reference or media clocks are local rather than shared). Conversely, the answerer may choose to reject the offer when the offered clock specifications indicate that the available reference and/or media clocks are incompatible.

While negotiation of reference clock and media clock attributes is not defined in this document, negotiation MAY be accomplished using the capabilities negotiation procedures defined in [RFC5939].

6.1.1. Indicating Support for Clock Source Signalling

An offerer or answerer indicates support for media clock signalling by including a reference or media clock specification in the SDP description. An offerer or answerer without specific reference or media clocks to signal SHOULD indicate support for clock source signalling by including a local reference clock (Section 4.6) specification in the SDP description.

6.1.2. Timestamp Reference Clock

If one or more of the reference clocks specified in the offer are usable by the answerer, the answerer SHOULD respond with an answer containing the subset of reference clock specifications in the offer that are usable by the answerer. If the answerer rejects the offer because the available reference clocks are incompatible, the rejection MUST contain at least one timestamp reference clock specification usable by the answerer. If no external reference clock is available to the answerer a local reference clock (Section 4.6) specification SHOULD be included in the rejection.

In both offers and answers, multiple reference clock specifications indicate equivalent clocks from different sources which may be used interchangeably. RTP senders and receivers are assured proper synchronization regardless of which of the specified sources is chosen and, in support of fault tolerance, may switch clock sources while streaming.

6.1.3. Media Clock

If the media clock mode specified in the offer is acceptable to the answerer, the answerer SHOULD respond with an answer containing the same media clock specification as the offer. If the answerer rejects the offer because the available reference clocks are incompatible, the rejection MUST contain a media clock specification supported by the answerer. If no shared media clocks are available to the answerer an asynchronous media clock (Section 5.1) specification

SHOULD be included in the rejection.

6.2. Usage Outside of Offer/Answer

SDP can be employed outside of the Offer/Answer context, for instance for multimedia sessions that are announced through the Session Announcement Protocol (SAP) [RFC2974], or streamed through the Real Time Streaming Protocol (RTSP) [RFC2326].

Devices using published descriptions to join sessions SHOULD assess their synchronization compatibility with the described session based on the clock source signaling and SHOULD NOT attempt to join a session with incompatible reference or media clocks.

7. Security Considerations

Entities receiving and acting upon an SDP message SHOULD be aware that a session description cannot be trusted unless it has been obtained by an authenticated transport protocol from a known and trusted source. Many different transport protocols may be used to distribute session description, and the nature of the authentication will differ from transport to transport. For some transports, security features are often not deployed. In case a session description has not been obtained in a trusted manner, the endpoint SHOULD exercise care because, among other attacks, the media sessions received may not be the intended ones, the destination where media is sent to may not be the expected one, any of the parameters of the session may be incorrect.

Incorrect reference or media clock parameters may cause devices or streams to synchronize to unintended clock sources. Normally this simply results in failure to establish a session or failure to synchronize once connected. Enough devices fraudulently assigned to a specific clock source (e.g. a particular IEEE 1588 grandmaster) may, however, constitute a successful denial of service attack on that source. Devices MAY wish to validate the integrity of the clock description through some means before connecting to unfamiliar clock sources.

8. IANA Considerations

This document defines two new SDP attributes: 'ts-refclk' and 'mediacclk', within the existing Internet Assigned Numbers Authority (IANA) registry of SDP Parameters.

This document also defines a new IANA registry subordinate to the

IANA SDP Parameters registry: the Media Clock Source Parameters Registry. Within this new registry, this document defines an initial set of three media clock source parameters. Further, this document defines a second new IANA registry subordinate to the IANA SDP Parameters registry: the Timestamp Reference Clock Source Parameters Registry. Within this new registry, this document defines an initial six parameters.

8.1. Reference Clock SDP Parameter

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

SDP Attribute ("att-field (both session and media level)":

Attribute name:	ts-refclk
Long form:	Timestamp reference clock source
Type of name:	att-field
Type of attribute:	Session, media and source level
Subject to charset:	No
Purpose:	See section 4 of this document
Reference:	This document
Values:	See section 8.3 of this document

Figure 10

The attribute has an extensible parameter field and therefore a registry for these parameters is required. This new registry is defined in Section 8.3.

8.2. Media Clock SDP Parameter

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

SDP Attribute ("att-field (both session and media level)":

```

Attribute name:      mediaclk
Long form:           Media clock source
Type of name:        att-field
Type of attribute:   session and media level
Subject to charset:  No
Purpose:             See section 5 of this document
Reference:           This document
Values:              See section 8.4 of this document

```

Figure 11

The attribute has an extensible parameter field and therefore a registry for these parameters is required. The new registry is defined in Section 8.4.

8.3. Timestamp Reference Clock Source Parameters Registry

This document creates a new IANA sub-registry called the Timestamp Reference Clock Source Parameters Registry, subordinate to the IANA SDP Parameters registry.

Initial values for the Timestamp Reference Clock Source Parameters registry are given below; future assignments are to be made through the Specification Required policy [RFC5226].

Value	Long Name	Reference
ntp	Network Time Protocol	This document, section 4
ptp	Precision Time Protocol	This document, section 4
gps	Global Position System	This document, section 4
gal	Galileo	This document, section 4
local	Local Clock	This document, section 4
private	Private Clock	This document, section 4

8.4. Media Clock Source Parameters Registry

This document creates a new IANA sub-registry called the Media Clock Source Parameters registry, subordinate to the IANA SDP Parameters registry.

Initial values for the Media Clock Source Parameters registry are given below; future assignments are to be made through the Specification Required policy [RFC5226].

Value	Long Name	Reference
sender	Asynchronously Generated Media Clock	This document, section 5
direct	Direct-Referenced Media Clock	This document, section 5
master-id	Media Clock Master Identifier	This document, section 5
IEEE1722	IEEE1722 Media Stream Identifier	This document, section 5

8.5. Source-level Attributes

[RFC5576] requires new source-level attributes to be registered with the IANA registry named "att-field (source level)".

8.5.1. Source-level Reference Clock Attribute

The source-level SDP attribute "ts-refclk" defined by this document is registered with the "att-field (source level)" IANA registry of SDP Parameters according to Figure 10.

8.5.2. Source-level Media Clock Attribute

The source-level SDP attribute "mediaclk" defined by this document is registered with the "att-field (source level)" IANA registry of SDP Parameters according to Figure 11.

The source-level SDP attribute "master-id" defined by this document is registered with the IANA registry of SDP Parameters as follows:

SDP Attribute ("att-field (source level)"):

Attribute name: master-id
Long form: Media clock source
Type of name: att-field
Type of attribute: source level
Subject to charset: No
Purpose: See section 5 of this document
Reference: This document
Values: EUI48

Figure 12

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.
- [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

- [RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.
- [RFC5939] Andreasen, F., "Session Description Protocol (SDP) Capability Negotiation", RFC 5939, September 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.

9.2. Informative References

- [AES11-2009]
Audio Engineering Society, "AES11-2009: AES recommended practice for digital audio engineering - Synchronization of digital audio equipment in studio operations", <<http://www.aes.org/standards/>>.
- [I-D.ietf-avtcore-idms]
Brandenburg, R., Stokking, H., Deventer, O., Boronat, F., Montagud, M., and K. Gross, "Inter-destination Media Synchronization using the RTP Control Protocol (RTCP)", draft-ietf-avtcore-idms-07 (work in progress), October 2012.
- [IEEE1588-2002]
Institute of Electrical and Electronics Engineers, "1588-2002 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", IEEE Std 1588-2002, 2002, <<http://standards.ieee.org/findstds/standard/1588-2002.html>>.
- [IEEE1588-2008]
Institute of Electrical and Electronics Engineers, "1588-2008 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", IEEE Std 1588-2008, 2008, <<http://standards.ieee.org/findstds/standard/1588-2008.html>>.
- [IEEE1722]
Institute of Electrical and Electronics Engineers, "IEEE Standard for Layer 2 Transport Protocol for Time Sensitive Applications in a Bridged Local Area Network", <<http://standards.ieee.org/findstds/standard/1722-2011.html>>.

- [IEEE802.1AS-2011]
Institute of Electrical and Electronics Engineers, "Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks", <<http://standards.ieee.org/findstds/standard/802.1AS-2011.html>>.
- [IEEE802.1BA-2011]
Institute of Electrical and Electronics Engineers, "Audio Video Bridging (AVB) Systems", <<http://standards.ieee.org/findstds/standard/802.1BA-2011.html>>.
- [IS-GPS-200F]
Global Positioning Systems Directorate, "Navstar GPS Space Segment/Navigation User Segment Interfaces", September 2011.
- [RFC2326] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.
- [RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, October 2000.
- [RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.
- [SMPTE-318-1999]
Society of Motion Picture & Television Engineers, "Television and Audio - Synchronization of 59.94- or 50-Hz Related Video and Audio Systems in Analog and Digital Areas - Reference Signals", <<http://standards.smpte.org/>>.

URIs

- [1] <<http://www.ieee802.org/1/files/public/docs2007/as-dolsen-time-accuracy-0407.pdf>>

Authors' Addresses

Aidan Williams
Audinate
Level 1, 458 Wattle St
Ultimo, NSW 2007
Australia

Phone: +61 2 8090 1000
Fax: +61 2 8090 1001
Email: aidan.williams@audinate.com
URI: <http://www.audinate.com/>

Kevin Gross
AVA Networks
Boulder, CO
US

Email: kevin.gross@avanw.com
URI: <http://www.avanw.com/>

Ray van Brandenburg
TNO
Brassersplein 2
Delft 2612CT
the Netherlands

Phone: +31-88-866-7000
Email: ray.vanbrandenburg@tno.nl

Hans Stokking
TNO
Brassersplein 2
Delft 2612CT
the Netherlands

Email: hans.stokking@tno.nl

AVTCORE WG
Internet-Draft
Updates: 3550, 3551 (if approved)
Intended status: Standards Track
Expires: January 11, 2014

M. Westerlund
Ericsson
C. Perkins
University of Glasgow
J. Lennox
Vidyo
July 10, 2013

Sending Multiple Types of Media in a Single RTP Session
draft-ietf-avtcore-multi-media-rtp-session-03

Abstract

This document specifies how an RTP session can contain media streams with media from multiple media types such as audio, video, and text. This has been restricted by the RTP Specification, and thus this document updates RFC 3550 and RFC 3551 to enable this behaviour for applications that satisfy the applicability for using multiple media types in a single RTP session.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 11, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

1. Introduction	2
2. Definitions	3
3. Motivation	4
3.1. NAT and Firewalls	4
3.2. No Transport Level QoS	4
3.3. Architectural Equality	5
4. Overview of Solution	5
5. Applicability	6
5.1. Usage of the RTP session	6
5.2. Signalled Support	7
5.3. Homogeneous Multi-party	7
5.4. Reduced number of Payload Types	8
5.5. Stream Differentiation	8
5.6. Non-compatible Extensions	8
6. RTP Session Specification	9
6.1. RTP Session	9
6.2. Sender Source Restrictions	11
6.3. Payload Type Applicability	12
6.4. RTCP Considerations	12
7. Extension Considerations	12
7.1. RTP Retransmission	13
7.2. Generic FEC	13
8. Signalling	14
8.1. SDP-Based Signalling	14
9. IANA Considerations	14
10. Security Considerations	14
11. Acknowledgements	15
12. References	15
12.1. Normative References	15
12.2. Informative References	16
Authors' Addresses	17

1. Introduction

When the Real-time Transport Protocol (RTP) [RFC3550] was designed, close to 20 years ago, IP networks were very different compared to the ones in 2013 when this is written. The almost ubiquitous deployment of Network Address Translators (NAT) and Firewalls has increased the cost and likely-hood of communication failure when using many different transport flows. Thus there exists a pressure to reduce the number of concurrent transport flows.

RTP [RFC3550] recommends against sending several different types of media, for example audio and video, in a single RTP session. The RTP profile for Audio and Video Conferences with Minimal Control (RTP/AVP) [RFC3551] mandates a similar restriction. The motivation for these limitations is partly to allow lower layer Quality of Service (QoS) mechanisms to be used, and partly due to limitations of the RTCP timing rules that assumes all media in a session to have similar bandwidth. The Session Description Protocol (SDP) [RFC4566], as one of the dominant signalling method for establishing RTP session, has enforced this rule, simply by not allowing multiple media types for a given receiver destination or set of ICE candidates, which is the most common method to determine which RTP session the packets are intended for.

The fact that these limitations have been in place for so long a time, in addition to RFC 3550 being written without fully considering multiple media types in an RTP session, does result in a number of considerations being needed when allowing this behaviour. This document provides such considerations regarding applicability as well as functionality, including normative specification of behaviour.

First, some basic definitions are provided. This is followed by a background that discusses the motivation in more detail. A overview of the solution of how to provide multiple media types in one RTP session is then presented. Next is the formal applicability this specification have followed by the normative specification. This is followed by a discussion how some RTP/RTCP Extensions is expected to function in the case of multiple media types in one RTP session. A specification of the requirements on signalling from this specification and a look how this is realized in SDP using Bundle [I-D.ietf-mmusic-sdp-bundle-negotiation]. The document ends with the security considerations.

2. Definitions

The following terms are used with supplied definitions:

Endpoint: A single entity sending or receiving RTP packets. It can be decomposed into several functional blocks, but as long as it behaves as a single RTP stack entity it is classified as a single endpoint.

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

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

QoS: Quality of Service, i.e. network mechanisms that intended to ensure that the packets within a flow or with a specific marking are transported with certain properties.

RTP Session: As defined by [RFC3550], the endpoints belonging to the same RTP Session are those that share a single SSRC space. That is, those endpoints can see an SSRC identifier transmitted by any one of the other endpoints. An endpoint can receive an SSRC either as SSRC or as CSRC in RTP and RTCP packets. Thus, the RTP Session scope is decided by the endpoints' network interconnection topology, in combination with RTP and RTCP forwarding strategies deployed by endpoints and any interconnecting middle nodes.

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

3. Motivation

This section discusses in more detail the main motivations why allowing multiple media types in the same RTP session is suitable.

3.1. NAT and Firewalls

The existence of NATs and Firewalls at almost all Internet access has had implications on protocols like RTP that were designed to use multiple transport flows. First of all, the NAT/FW traversal solution needs to ensure that all these transport flows are established. This has three consequences:

1. Increased delay to perform the transport flow establishment
2. The more transport flows, the more state and the more resource consumption in the NAT and Firewalls. When the resource consumption in NAT/FWs reaches their limits, unexpected behaviours usually occur.
3. More transport flows means a higher risk that some transport flow fails to be established, thus preventing the application to communicate.

Using fewer transport flows reduces the risk of communication failure, improved establishment behaviour and less load on NAT and Firewalls.

3.2. No Transport Level QoS

Many RTP-using applications don't utilize any network level Quality of Service functions. Nor do they expect or desire any separation in network treatment of its media packets, independent of whether they are audio, video or text. When an application has no such desire, it doesn't need to provide a transport flow structure that simplifies flow based QoS.

3.3. Architectural Equality

For applications that don't require different lower-layer QoS for different media types, and that have no special requirements for RTP extensions or RTCP reporting, the requirement to separate different media into different RTP sessions might seem unnecessary. Provided the application accepts that all media flows will get similar RTCP reporting, using the same RTP session for several types of media at once appears a reasonable choice. The architecture ought to be agnostic about the type of media being carried in an RTP session to the extent possible given the constraints of the protocol.

4. Overview of Solution

The goal of the solution is to enable each RTP session to contain more than just one media type. This includes having multiple RTP sessions containing a given media type, for example having three sessions containing both video and audio.

The solution is quite straightforward. The first step is to override the SHOULD and SHOULD NOT language of the RTP specification [RFC3550]. Similar change is needed to a sentence in Section 6 of [RFC3551] that states that "different media types SHALL NOT be interleaved or multiplexed within a single RTP Session". This is resolved by appropriate exception clauses given that this specification and its applicability is followed.

Within an RTP session where multiple media types have been configured for use, an SSRC can only send one type of media during its lifetime (i.e., it can switch between different audio codecs, since those are both the same type of media, but cannot switch between audio and video). Different SSRCs MUST be used for the different media sources, the same way multiple media sources of the same media type already have to do. The payload type will inform a receiver which media type the SSRC is being used for. Thus the payload type MUST be unique across all of the payload configurations independent of media type that is used in the RTP session.

Some few extra considerations within the RTP sessions also needs to be considered. RTCP bandwidth and regular reporting suppression (RTP/AVPF and RTP/SAVPF) SHOULD be configured to reduce the impact for

bit-rate variations between streams and media types. It is also clarified how timeout calculations are to be done to avoid any issues. Certain payload types like FEC also need additional rules.

The final important part of the solution to this is to use signalling and ensure that agreement on using multiple media types in an RTP session exists, and how that then is configured. This memo describes some existing requirements, while an external reference defines how this is accomplished in SDP.

5. Applicability

This specification has limited applicability, and anyone intending to use it needs to ensure that their application and usage meets the below criteria.

5.1. Usage of the RTP session

Before choosing to use this specification, an application implementer needs to ensure that they don't have a need for different RTP sessions between the media types for some reason. The main rule is that if one expects to have equal treatment of all media packets, then this specification might be suitable. The equal treatment include anything from network level up to RTCP reporting and feedback. The document Guidelines for using the Multiplexing Features of RTP [I-D.westerlund-avtcore-multiplex-architecture] gives more detailed guidance on aspects to consider when choosing how to use RTP and specifically sessions. RTP-using applications that need or would prefer multiple RTP sessions, but do not require the functionalities or behaviours that multiple transport flows give, can consider using Multiple RTP Sessions on a Single Lower-Layer Transport [I-D.westerlund-avtcore-transport-multiplexing]. It needs to be noted that some difference in treatment is still possible to achieve, for example marking based QoS, or RTCP feedback traffic for only some media streams.

The second important consideration is the resulting behaviour when media flows to be sent within a single RTP session does not have similar bandwidth. There are limitations in the RTCP timing rules, and this implies a common RTCP reporting interval across all participants in a session. If an RTP session contains flows with very different bandwidths, for example low-rate audio coupled with high-quality video, this can result in either excessive or insufficient RTCP for some flows, depending how the RTCP session bandwidth, and hence reporting interval, is configured. This is discussed further in Section 6.4.

5.2. Signalled Support

Usage of this specification is not compatible with anyone following RFC 3550 and intending to have different RTP sessions for each media type. Therefore there needs to be mutual agreement to use multiple media types in one RTP session by all participants within that RTP session. This agreement has to be determined using signalling in most cases.

This requirement can be a problem for signalling solutions that can't negotiate with all participants. For declarative signalling solutions, mandating that the session is using multiple media types in one RTP session can be a way of attempting to ensure that all participants in the RTP session follow the requirement. However, for signalling solutions that lack methods for enforcing that a receiver supports a specific feature, this can still cause issues.

5.3. Homogeneous Multi-party

In multiparty communication scenarios it is important to separate two different cases. One case is where the RTP session contains multiple participants in a common RTP session. This occurs for example in Any Source Multicast (ASM) and Transport Translator topologies as defined in RTP Topologies [RFC5117]. It can also occur in some implementations of RTP mixers that share the same SSRC/CSRC space across all participants. The second case is when the RTP session is terminated in a middlebox and the other participants sources are projected or switched into each RTP session and rewritten on RTP header level including SSRC mappings.

For the first case, with a common RTP session or at least shared SSRC/CSRC values, all participants in multiparty communication are REQUIRED to support multiple media types in an RTP session. An participant using two or more RTP sessions towards a multiparty session can't be collapsed into a single session with multiple media types. The reason is that in case of multiple RTP sessions, the same SSRC value can be use in both RTP sessions without any issues, but when collapsed to a single session there is an SSRC collision. In addition some collisions can't be represented in the multiple separate RTP sessions. For example, in a session with audio and video, an SSRC value used for video will not show up in the Audio RTP session at the participant using multiple RTP sessions, and thus not trigger any collision handling. Thus any application using this type of RTP session structure MUST have a homogeneous support for multiple media types in one RTP session, or be forced to insert a translator node between that participant and the rest of the RTP session.

For the second case of separate RTP sessions for each multiparty participant and a central node it is possible to have a mix of single RTP session users and multiple RTP session users as long as one is willing to remap the SSRCs used by a participant with multiple RTP sessions into non-used values in the single RTP session SSRC space for each of the participants using a single RTP session with multiple media types. It can be noted that this type of implementation has to understand all types of RTP/RTCP extension being used in the RTP sessions to correctly be able to translate them between the RTP sessions. It can also negatively impact the possibility for loop detection, as SSRC/CSRC can't be used to detect the loops, instead some other media stream identity name space that is common across all interconnect parts are needed.

5.4. Reduced number of Payload Types

An RTP session with multiple media types in it have only a single 7-bit Payload Type range for all its payload types. Within the 128 available values, only 96 or less if "Multiplexing RTP Data and Control Packets on a Single Port" [RFC5761] is used, all the different RTP payload configurations for all the media types need to fit in the available space. For most applications this will not be a real problem, but the limitation exists and could be encountered.

5.5. Stream Differentiation

If network level differentiation of the media streams of different media types are desired using this specification can cause severe limitations. All media streams in an RTP session, independent of the media type, will be sent over the same underlying transport flow. Any flow-based Quality of Service (QoS) mechanism will be unable to provide differentiated treatment between different media types, e.g. to prioritize audio over video. If differentiated treatment is desired using flow-based QoS, separate RTP sessions over different underlying transport flows needs to be used.

Any marking-based QoS scheme like DiffServ is not affected unless a network ingress marks based on flows, in which case the same considerations as for flow based QoS applies.

5.6. Non-compatible Extensions

There exist some RTP and RTCP extensions that rely on the existence of multiple RTP sessions. If the goal of using an RTP session with multiple media types is to have only a single RTP session, then these extensions can't be used. If one has no need to have different RTP sessions for the media types but is willing to have multiple RTP sessions, one for the main media transmission and one for the

extension, they can be used. It is to be noted that this assumes that it is possible to get the extension working when the related RTP session contains multiple media types.

Identified RTP/RTCP extensions that require multiple RTP Sessions are:

RTP Retransmission: RTP Retransmission [RFC4588] has a session multiplexed mode. It also has a SSRC multiplexed mode that can be used instead. So use the mode that is suitable for the RTP application.

XOR-Based FEC: The RTP Payload Format for Generic Forward Error Correction [RFC5109] and its predecessor [RFC2733] requires a separate RTP session unless the FEC data is carried in RTP Payload for Redundant Audio Data [RFC2198]. However, using the Generic FEC with the Redundancy payload has another set of restrictions, see Section 7.2.

Note that the Source-Specific Media Attributes [RFC5576] specification defines an SDP syntax (the "FEC" semantic of the "ssrc-group" attribute) to signal FEC relationships between multiple media streams within a single RTP session. However, this can't be used as the FEC repair packets need to have the same SSRC value as the source packets being protected. [RFC5576] does not normatively update and resolve that restriction. There is ongoing work on an ULP extension to allow it be use FEC streams within the same RTP Session as the source stream [I-D.lennox-payload-ulp-ssrc-mux].

6. RTP Session Specification

This section defines what needs to be done or avoided to make an RTP session with multiple media types function without issues.

6.1. RTP Session

Section 5.2 of "RTP: A Transport Protocol for Real-Time Applications" [RFC3550] states:

For example, in a teleconference composed of audio and video media encoded separately, each medium SHOULD be carried in a separate RTP session with its own destination transport address.

Separate audio and video streams SHOULD NOT be carried in a single RTP session and demultiplexed based on the payload type or SSRC fields.

This specification changes both of these sentences. The first sentence is changed to:

For example, in a teleconference composed of audio and video media encoded separately, each medium SHOULD be carried in a separate RTP session with its own destination transport address, unless specification [RFCXXXX] is followed and the application meets the applicability constraints.

The second sentence is changed to:

Separate audio and video streams SHOULD NOT be carried in a single RTP session and demultiplexed based on the payload type or SSRC fields, unless multiplexed based on both SSRC and payload type and usage meets what Multiple Media Types in an RTP Session [RFCXXXX] specifies.

Second paragraph of Section 6 in RTP Profile for Audio and Video Conferences with Minimal Control [RFC3551] says:

The payload types currently defined in this profile are assigned to exactly one of three categories or media types: audio only, video only and those combining audio and video. The media types are marked in Tables 4 and 5 as "A", "V" and "AV", respectively. Payload types of different media types SHALL NOT be interleaved or multiplexed within a single RTP session, but multiple RTP sessions MAY be used in parallel to send multiple media types. An RTP source MAY change payload types within the same media type during a session. See the section "Multiplexing RTP Sessions" of RFC 3550 for additional explanation.

This specifications purpose is to violate that existing SHALL NOT under certain conditions. Thus also this sentence has to be changed to allow for multiple media type's payload types in the same session. The above sentence is changed to:

Payload types of different media types SHALL NOT be interleaved or multiplexed within a single RTP session unless as specified and under the restriction in Multiple Media Types in an RTP Session [RFCXXXX]. Multiple RTP sessions MAY be used in parallel to send multiple media types.

RFC-Editor Note: Please replace RFCXXXX with the RFC number of this specification when assigned.

We can now go on and discuss the five bullets that are motivating the previous in Section 5.2 of the RTP Specification [RFC3550]. They are repeated here for the reader's convenience:

1. If, say, two audio streams shared the same RTP session and the same SSRC value, and one were to change encodings and thus acquire a different RTP payload type, there would be no general way of identifying which stream had changed encodings.
2. An SSRC is defined to identify a single timing and sequence number space. Interleaving multiple payload types would require different timing spaces if the media clock rates differ and would require different sequence number spaces to tell which payload type suffered packet loss.
3. The RTCP sender and receiver reports (see Section 6.4 of RFC 3550) can only describe one timing and sequence number space per SSRC and do not carry a payload type field.
4. An RTP mixer would not be able to combine interleaved streams of incompatible media into one stream.
5. Carrying multiple media in one RTP session precludes: the use of different network paths or network resource allocations if appropriate; reception of a subset of the media if desired, for example just audio if video would exceed the available bandwidth; and receiver implementations that use separate processes for the different media, whereas using separate RTP sessions permits either single- or multiple-process implementations.

Bullets 1 to 3 are all related to that each media source has to use one or more unique SSRCs to avoid these issues as mandated below (Section 6.2). Bullet 4 can be served by two arguments, first of all each SSRC will be associated with a specific media type, communicated through the RTP payload type, allowing a middlebox to do media type specific operations. The second argument is that in many contexts blind combining without additional contexts are anyway not suitable. Regarding bullet 5 this is understood and explicitly stated applicability limitations for the method described in this document.

6.2. Sender Source Restrictions

A SSRC in the RTP session MUST only send one media type (audio, video, text etc.) during the SSRC's lifetime. The main motivation is that a given SSRC has its own RTP timestamp and sequence number spaces. The same way that you can't send two streams of encoded audio on the same SSRC, you can't send one audio and one video encoding on the same SSRC. Each media encoding when made into an RTP stream needs to have the sole control over the sequence number and timestamp space. If not, one would not be able to detect packet loss for that particular stream. Nor can one easily determine which clock rate a particular SSRCs timestamp will increase with. For additional

arguments why RTP payload type based multiplexing of multiple media streams doesn't work see Appendix A in [I-D.westerlund-avtcore-multiplex-architecture].

6.3. Payload Type Applicability

Most Payload Types have a native media type, like an audio codec is natural belonging to the audio media type. However, there exist a number of RTP payload types that don't have a native media type. For example, transport robustness mechanisms like RTP Retransmission [RFC4588] and Generic FEC [RFC5109] inherit their media type from what they protect. RTP Retransmission is explicitly bound to the payload type it is protecting, and thus will inherit it. However Generic FEC is a excellent example of an RTP payload type that has no natural media type. The media type for what it protects is not relevant as it is the recovered RTP packets that have a particular media type, and thus Generic FEC is best categorized as an application media type.

The above discussion is relevant to what limitations exist for RTP payload type usage within an RTP session that has multiple media types. In fact this document (Section 7.2) suggest that for usage of Generic FEC (XOR-based) as defined in RFC 5109 can actually use a single media type when used with independent RTP sessions for source and repair data.

Note a particular SSRC carrying Generic FEC will clearly only protect a specific SSRC and thus that instance is bound to the SSRC's media type. For this specific case, it is possible to have one be applicable to both. However, in cases when the signalling is setup to enable fall back to using separate RTP sessions, then using a different media type, e.g. application, than the media being protected can create issues.

6.4. RTCP Considerations

Guidelines for handling RTCP when sending multiple media streams with disparate rates in a single RTP session are outlined in [I-D.ietf-avtcore-rtp-multi-stream]. These guidelines apply when sending multiple types of media in a single RTP session if the different types of media have different rates.

7. Extension Considerations

This section discusses the impact on some RTP/RTCP extensions due to usage of multiple media types in on RTP session. Only extensions where something worth noting has been included.

7.1. RTP Retransmission

SSRC-multiplexed RTP retransmission [RFC4588] is actually very straightforward. Each retransmission RTP payload type is explicitly connected to an associated payload type. If retransmission is only to be used with a subset of all payload types, this is not a problem, as it will be evident from the retransmission payload types which payload types that have retransmission enabled for them.

Session-multiplexed RTP retransmission is also possible to use where an retransmission session contains the retransmissions of the associated payload types in the source RTP session. The only difference to previously is that the source RTP session is one which contains multiple media types. Thus it is even more likely that only a subset of the source RTP session's payload types and SSRCS are actually retransmitted.

Open Issue: When using SDP to signal retransmission for one RTP session with multiple media types and one RTP session for the retransmission data will cause a situation where one will have multiple m= lines grouped using FID and the ones belonging to respective RTP session being grouped using BUNDLE. This usage might contradict both the FID semantics [RFC5888] and an assumption in the RTP retransmission specification [RFC4588].

7.2. Generic FEC

The RTP Payload Format for Generic Forward Error Correction [RFC5109], and also its predecessor [RFC2733], requires some considerations, and they are different depending on what type of configuration of usage one has.

Independent RTP Sessions, i.e. where source and repair data are sent in different RTP sessions. As this mode of configuration requires different RTP session, there has to be at least one RTP session for source data, this session can be one using multiple media types. The repair session only needs one RTP Payload type indicating repair data, i.e. x/ulpfec or x/parityfec depending if RFC 5109 or RFC 2733 is used. The media type in this session is not relevant and can in theory be any of the defined ones. It is RECOMMENDED that one uses "Application".

In stream, using RTP Payload for Redundant Audio Data [RFC2198] combining repair and source data in the same packets. This is possible to use within a single RTP session. However, the usage and configuration of the payload types can create an issue. First of all it might be necessary to have one payload type per media type for the FEC repair data payload format, i.e. one for audio/ulpfec and one

for text/ulpfec if audio and text are combined in an RTP session. Secondly each combination of source payload and its FEC repair data has to be an explicit configured payload type. This has potential for making the limitation of RTP payload types available into a real issue.

8. Signalling

The Signalling requirements

Establishing an RTP session with multiple media types requires signalling. This signalling needs to fulfil the following requirements:

1. Ensure that any participant in the RTP session is aware that this is an RTP session with multiple media types.
2. Ensure that the payload types in use in the RTP session are using unique values, with no overlap between the media types.
3. Configure the RTP session level parameters, such as RTCP RR and RS bandwidth, AVPF trr-int, underlying transport, the RTCP extensions in use, and security parameters, commonly for the RTP session.
4. RTP and RTCP functions that can be bound to a particular media type SHOULD be reused when possible also for other media types, instead of having to be configured for multiple code-points.
Note: In some cases one will not have a choice but to use multiple configurations.

8.1. SDP-Based Signalling

The signalling of multiple media types in one RTP session in SDP is specified in "Multiplexing Negotiation Using Session Description Protocol (SDP) Port Numbers"
[I-D.ietf-mmusic-sdp-bundle-negotiation].

9. IANA Considerations

This document makes no request of IANA.

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

10. Security Considerations

Having an RTP session with multiple media types doesn't change the methods for securing a particular RTP session. One possible difference is that the different media have often had different security requirements. When combining multiple media types in one session, their security requirements also have to be combined by selecting the most demanding for each property. Thus having multiple media types can result in increased overhead for security for some media types to ensure that all requirements are met.

Otherwise, the recommendations for how to configure an RTP session do not add any additional requirements compared to normal RTP, except for the need to be able to ensure that the participants are aware that it is a multiple media type session. If not that is ensured it can cause issues in the RTP session for both the unaware and the aware one. Similar issues can also be produced in an normal RTP session by creating configurations for different end-points that doesn't match each other.

11. Acknowledgements

The authors would like to thank Christer Holmberg, Gunnar Hellstroem, and Charles Eckel for the feedback on the document.

12. References

12.1. Normative References

- [I-D.ietf-avtcore-rtp-multi-stream]
Lennox, J., Westerlund, M., Wu, W., and C. Perkins, "RTP Considerations for Endpoints Sending Multiple Media Streams", draft-ietf-avtcore-rtp-multi-stream-00 (work in progress), April 2013.
- [I-D.ietf-mmusic-sdp-bundle-negotiation]
Holmberg, C., Alvestrand, H., and C. Jennings, "Multiplexing Negotiation Using Session Description Protocol (SDP) Port Numbers", draft-ietf-mmusic-sdp-bundle-negotiation-04 (work in progress), June 2013.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.

12.2. Informative References

- [I-D.lennox-payload-ulp-ssrc-mux]
Lennox, J., "Supporting Source-Multiplexing of the Real-Time Transport Protocol (RTP) Payload for Generic Forward Error Correction", draft-lennox-payload-ulp-ssrc-mux-00 (work in progress), February 2013.
- [I-D.westerlund-avtcore-multiplex-architecture]
Westerlund, M., Perkins, C., and H. Alvestrand, "Guidelines for using the Multiplexing Features of RTP", draft-westerlund-avtcore-multiplex-architecture-03 (work in progress), February 2013.
- [I-D.westerlund-avtcore-transport-multiplexing]
Westerlund, M. and C. Perkins, "Multiple RTP Sessions on a Single Lower-Layer Transport", draft-westerlund-avtcore-transport-multiplexing-05 (work in progress), February 2013.
- [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.
- [RFC2733] Rosenberg, J. and H. Schulzrinne, "An RTP Payload Format for Generic Forward Error Correction", RFC 2733, December 1999.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.
- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.
- [RFC5109] Li, A., "RTP Payload Format for Generic Forward Error Correction", RFC 5109, December 2007.

- [RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, January 2008.
- [RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, February 2008.
- [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.
- [RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.
- [RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.
- [RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.

Authors' Addresses

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

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

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

Email: csp@csp Perkins.org

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

Email: jonathan@vidyo.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 16, 2014

M. Westerlund
B. Burman
Ericsson
C. Perkins
University of Glasgow
H. Alvestrand
Google
July 15, 2013

Guidelines for using the Multiplexing Features of RTP to Support
Multiple Media Streams
draft-ietf-avtcore-multiplex-guidelines-01

Abstract

The Real-time Transport Protocol (RTP) is a flexible protocol that can be used in a wide range of applications, networks, and system topologies. That flexibility makes for wide applicability, but can complicate the application design process. One particular design question that has received much attention is how to support multiple media streams in RTP. This memo discusses the available options and design trade-offs, and provides guidelines on how to use the multiplexing features of RTP to support multiple media streams.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 16, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Definitions	4
2.1. Terminology	4
2.2. Subjects Out of Scope	6
3. Reasons for Multiplexing and Grouping RTP Media Streams	6
4. RTP Multiplexing Points	7
4.1. RTP Session	8
4.2. Synchronisation Source (SSRC)	9
4.3. Contributing Source (CSRC)	10
4.4. RTP Payload Type	11
5. RTP Topologies and Issues	12
5.1. Point to Point	12
5.2. Translators & Gateways	13
5.3. Point to Multipoint Using Multicast	13
5.4. Point to Multipoint Using an RTP Transport Translator . .	14
5.5. Point to Multipoint Using an RTP Mixer	15
6. RTP Multiplexing: When to Use Multiple RTP Sessions	15
6.1. RTP and RTCP Protocol Considerations	16
6.1.1. The RTP Specification	16
6.1.2. Multiple SSRCs in a Session	18
6.1.3. Handling Varying Sets of Senders	19
6.1.4. Cross Session RTCP Requests	19
6.1.5. Binding Related Sources	19
6.1.6. Forward Error Correction	21
6.1.7. Transport Translator Sessions	21
6.2. Interworking Considerations	21
6.2.1. Types of Interworking	22
6.2.2. RTP Translator Interworking	22
6.2.3. Gateway Interworking	22
6.2.4. Multiple SSRC Legacy Considerations	23
6.3. Network Considerations	24
6.3.1. Quality of Service	24
6.3.2. NAT and Firewall Traversal	25
6.3.3. Multicast	26
6.3.4. Multiplexing multiple RTP Session on a Single Transport	27

6.4.	Security and Key Management Considerations	27
6.4.1.	Security Context Scope	27
6.4.2.	Key Management for Multi-party session	28
6.4.3.	Complexity Implications	28
7.	Archetypes	29
7.1.	Single SSRC per Session	29
7.2.	Multiple SSRCs of the Same Media Type	31
7.3.	Multiple Sessions for one Media type	32
7.4.	Multiple Media Types in one Session	34
7.5.	Summary	35
8.	Summary considerations and guidelines	35
8.1.	Guidelines	35
9.	IANA Considerations	36
10.	Security Considerations	37
11.	References	37
11.1.	Normative References	37
11.2.	Informative References	37
Appendix A.	Dismissing Payload Type Multiplexing	41
Appendix B.	Proposals for Future Work	43
Appendix C.	Signalling considerations	43
C.1.	Signalling Aspects	43
C.1.1.	Session Oriented Properties	44
C.1.2.	SDP Prevents Multiple Media Types	45
C.1.3.	Signalling Media Stream Usage	45
Authors' Addresses		45

1. Introduction

The Real-time Transport Protocol (RTP) [RFC3550] is a commonly used protocol for real-time media transport. It is a protocol that provides great flexibility and can support a large set of different applications. RTP has several multiplexing points designed for different purposes. These enable support of multiple media streams and switching between different encoding or packetization of the media. By using multiple RTP sessions, sets of media streams can be structured for efficient processing or identification. Thus the question for any RTP application designer is how to best use the RTP session, the SSRC and the payload type to meet the application's needs.

The purpose of this document is to provide clear information about the possibilities of RTP when it comes to multiplexing. The RTP application designer needs to understand the implications that come from a particular usage of the RTP multiplexing points. The document will recommend against some usages as being unsuitable, in general or for particular purposes.

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

This document has been written due to increased interest in more advanced usage of RTP, resulting in questions regarding the most appropriate RTP usage. The limitations in some implementations, RTP/RTCP extensions, and signalling has also been exposed. It is expected that some limitations will be addressed by updates or new extensions resolving the shortcomings. The authors also hope that clarification on the usefulness of some functionalities in RTP will result in more complete implementations in the future.

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

2. Definitions

2.1. Terminology

The following terms and abbreviations are used in this document:

Endpoint: A single entity sending or receiving RTP packets. It can be decomposed into several functional blocks, but as long as it behaves a single RTP stack entity it is classified as a single endpoint.

Multiparty: A communication situation including multiple endpoints. In this document it will be used to refer to situations where more than two endpoints communicate.

Media Source: The source of a stream of data of one Media Type, It can either be a single media capturing device such as a video camera, a microphone, or a specific output of a media production function, such as an audio mixer, or some video editing function.

Sending data from a Media Source can cause multiple RTP sources to send multiple Media Streams.

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

RTP Source: The originator or source of a particular Media Stream. Identified using an SSRC in a particular RTP session. An RTP source is the source of a single media stream, and is associated with a single endpoint and a single Media Source. An RTP Source is just called a Source in RFC 3550.

RTP Sink: A recipient of a Media Stream. The Media Sink is identified using one or more SSRCs. There can be more than one RTP Sink for one RTP source.

CNAME: "Canonical name" - identifier associated with one or more RTP sources from a single endpoint. Defined in the RTP specification [RFC3550]. A CNAME identifies a synchronisation context. A CNAME is associated with a single endpoint, although some RTP nodes will use an endpoint's CNAME on that endpoints behalf. An endpoint can use multiple CNAMEs. A CNAME is intended to be globally unique and stable for the full duration of a communication session. [RFC6222][I-D.ietf-avtcore-6222bis] gives updated guidelines for choosing CNAMEs.

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

Multiplexing: The operation of taking multiple entities as input, aggregating them onto some common resource while keeping the individual entities addressable such that they can later be fully and unambiguously separated (de-multiplexed) again.

RTP Session: As defined by [RFC3550], the endpoints belonging to the same RTP Session are those that share a single SSRC space. That is, those endpoints can see an SSRC identifier transmitted by any one of the other endpoints. An endpoint can receive an SSRC either as SSRC or as CSRC in RTP and RTCP packets. Thus, the RTP Session scope is decided by the endpoints' network interconnection topology, in combination with RTP and RTCP forwarding strategies deployed by endpoints and any interconnecting middle nodes.

RTP Session Group: One or more RTP sessions that are used together to perform some function. Examples are multiple RTP sessions used to carry different layers of a layered encoding. In an RTP Session Group, CNAMEs are assumed to be valid across all RTP

sessions, and designate synchronisation contexts that can cross RTP sessions.

Source: Term that ought not be used alone. An RTP Source, as identified by its SSRC, is the source of a single Media Stream; a Media Source can be the source of multiple Media Streams.

SSRC: A 32-bit unsigned integer used as identifier for a RTP Source.

CSRC: Contributing Source, A SSRC identifier used in a context, like the RTP headers CSRC list, where it is clear that the Media Source is not the source of the media stream, instead only a contributor to the Media Stream.

Signalling: The process of configuring endpoints to participate in one or more RTP sessions.

2.2. Subjects Out of Scope

This document is focused on issues that affect RTP. Thus, issues that involve signalling protocols, such as whether SIP, Jingle or some other protocol is in use for session configuration, the particular syntaxes used to define RTP session properties, or the constraints imposed by particular choices in the signalling protocols, are mentioned only as examples in order to describe the RTP issues more precisely.

This document assumes the applications will use RTCP. While there are such applications that don't send RTCP, they do not conform to the RTP specification, and thus can be regarded as reusing the RTP packet format but not implementing the RTP protocol.

3. Reasons for Multiplexing and Grouping RTP Media Streams

The reasons why an endpoint might choose to send multiple media streams are widespread. In the below discussion, please keep in mind that the reasons for having multiple media streams vary and include but are not limited to the following:

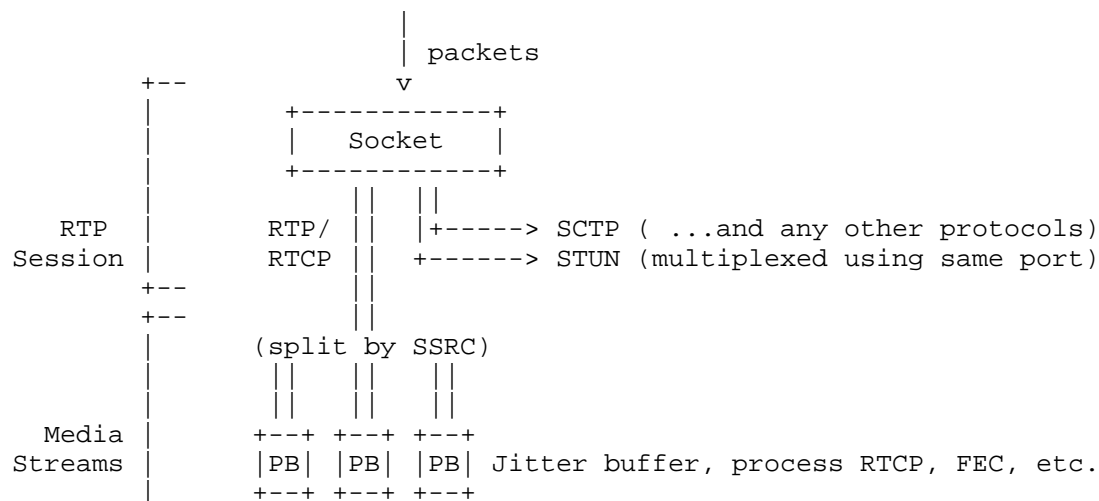
- o Multiple Media Sources
- o Multiple Media Streams might be needed to represent one Media Source (for instance when using layered encodings)
- o A Retransmission stream might repeat the content of another Media Stream

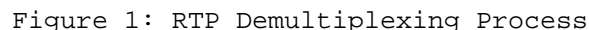
- o An FEC stream might provide material that can be used to repair another Media Stream
- o Alternative Encodings, for instance different codecs for the same audio stream
- o Alternative formats, for instance multiple resolutions of the same video stream

For each of these, it is necessary to decide if each additional media stream gets its own SSRC multiplexed within a RTP Session, or if it is necessary to use additional RTP sessions to group the media streams. The choice between these made due to one reason might not be the choice suitable for another reason. In the above list, the different items have different levels of maturity in the discussion on how to solve them. The clearest understanding is associated with multiple media sources of the same media type. However, all warrant discussion and clarification on how to deal with them. As the discussion below will show, in reality we cannot choose a single one of the two solutions. To utilise RTP well and as efficiently as possible, both are needed. The real issue is finding the right guidance on when to create RTP sessions and when additional SSRCs in an RTP session is the right choice.

4. RTP Multiplexing Points

This section describes the multiplexing points present in the RTP protocol that can be used to distinguish media streams and groups of media streams. Figure 1 outlines the process of demultiplexing incoming RTP streams:





application, transport, and signalling protocol. The RTP protocol makes no normative statements about the relationship between different RTP sessions, however the applications that use more than one RTP session will have some higher layer understanding of the relationship between the sessions they create.

4.2. Synchronisation Source (SSRC)

A synchronisation source (SSRC) identifies an RTP source or an RTP sink. Every endpoint will have at least one synchronisation source identifier, even if it does not send media (endpoints that are only RTP sinks still send RTCP, and use their synchronisation source identifier in the RTCP packets they send). An endpoint can have multiple synchronisation sources identifiers if it contains multiple RTP sources (i.e., if it sends multiple media streams). Endpoints that are both RTP sources and RTP sinks use the same synchronisation sources in both roles. At any given time, a RTP source has one and only one SSRC - although that can change over the lifetime of the RTP source or sink.

The synchronisation Source identifier is a 32-bit unsigned integer. It is present in every RTP and RTCP packet header, and in the payload of some RTCP packet types. It can also be present in SDP signalling. Unless pre-signalled using the SDP "a=ssrc:" attribute [RFC5576], the synchronisation source identifier is chosen at random. It is not dependent on the network address of the endpoint, and is intended to be unique within an RTP session. Synchronisation source identifier collisions can occur, and are handled as specified in [RFC3550] and [RFC5576], resulting in the synchronisation source identifier of the affecting RTP sources and/or sinks changing. An RTP source that changes its RTP Session identifier (e.g. source transport address) during a session has to choose a new SSRC identifier to avoid being interpreted as looped source.

Synchronisation source identifiers that belong to the same synchronisation context (i.e., that represent media streams that can be synchronised using information in RTCP SR packets) are indicated by use of identical CNAME chunks in corresponding RTCP SDES packets. SDP signalling can also be used to provide explicit grouping of synchronisation sources [RFC5576].

In some cases, the same SSRC Identifier value is used to relate streams in two different RTP Sessions, such as in Multi-Session Transmission of scalable video [RFC6190]. This is NOT RECOMMENDED since there is no guarantee of uniqueness in SSRC values across RTP sessions.

Note that RTP sequence number and RTP timestamp are scoped by the synchronisation source. Each RTP source will have a different synchronisation source, and the corresponding media stream will have a separate RTP sequence number and timestamp space.

An SSRC identifier is used by different type of sources as well as sinks:

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

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

RTP Sink: A source that does not generate any RTP media stream in itself (e.g. an endpoint or middlebox only receiving in an RTP session). It still needs a sender SSRC for use as source in RTCP reports.

Note that a endpoint that generates more than one media type, e.g. a conference participant sending both audio and video, need not (and commonly does not) use the same SSRC value across RTP sessions. RTCP Compound packets containing the CNAME SDES item is the designated method to bind an SSRC to a CNAME, effectively cross-correlating SSRCs within and between RTP Sessions as coming from the same endpoint. The main property attributed to SSRCs associated with the same CNAME is that they are from a particular synchronisation context and can be synchronised at playback.

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

4.3. Contributing Source (CSRC)

The Contributing Source (CSRC) is not a separate identifier. Rather a synchronisation source identifier is listed as a CSRC in the RTP header of a packet generated by an RTP mixer if the corresponding SSRC was in the header of one of the packets that contributed to the mix.

It is not possible, in general, to extract media represented by an individual CSRC since it is typically the result of a media mixing (merge) operation by an RTP mixer on the individual media streams corresponding to the CSRC identifiers. The exception is the case when only a single CSRC is indicated as this represent forwarding of a media stream, possibly modified. The RTP header extension for Mixer-to-Client Audio Level Indication [RFC6465] expands on the receivers information about a packet with a CSRC list. Due to these restrictions, CSRC will not be considered a fully qualified multiplexing point and will be disregarded in the rest of this document.

4.4. RTP Payload Type

Each Media Stream utilises one or more RTP payload formats. An RTP payload format describes how the output of a particular media codec is framed and encoded into RTP packets. The payload format used is identified by the payload type field in the RTP data packet header. The combination therefore identifies a specific Media Stream encoding format. The format definition can be taken from [RFC3551] for statically allocated payload types, but ought to be explicitly defined in signalling, such as SDP, both for static and dynamic Payload Types. The term "format" here includes whatever can be described by out-of-band signalling means. In SDP, the term "format" includes media type, RTP timestamp sampling rate, codec, codec configuration, payload format configurations, and various robustness mechanisms such as redundant encodings [RFC2198].

The payload type is scoped by sending endpoint within an RTP Session. All synchronisation sources sent from an single endpoint share the same payload types definitions. The RTP Payload Type is designed such that only a single Payload Type is valid at any time instant in the RTP source's RTP timestamp time line, effectively time-multiplexing different Payload Types if any change occurs. The payload type used can change on a per-packet basis for an SSRC, for example a speech codec making use of generic comfort noise [RFC3389]. If there is a true need to send multiple Payload Types for the same SSRC that are valid for the same instant, then redundant encodings [RFC2198] can be used. Several additional constraints than the ones mentioned above need to be met to enable this use, one of which is that the combined payload sizes of the different Payload Types ought not exceed the transport MTU.

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

The payload type is not a multiplexing point at the RTP layer (see Appendix A for a detailed discussion of why using the payload type as

an RTP multiplexing point does not work). The RTP payload type is, however, used to determine how to render a media stream, and so can be viewed as selecting a rendering context. The rendering context can be defined by the signalling, and the RTP payload type number is sometimes used to associate an RTP media stream with the signalling. This association is possible provided unique RTP payload type numbers are used in each context. For example, an RTP media stream can be associated with an SDP "m=" line by comparing the RTP payload type numbers used by the media stream with payload types signalled in the "a=rtpmap:" lines in the media sections of the SDP. If RTP media streams are being associated with signalling contexts based on the RTP payload type, then the assignment of RTP payload type numbers MUST be unique across signalling contexts; if the same RTP payload format configuration is used in multiple contexts, then a different RTP payload type number has to be assigned in each context to ensure uniqueness. If the RTP payload type number is not being used to associated RTP media streams with a signalling context, then the same RTP payload type number can be used to indicate the exact same RTP payload format configuration in multiple contexts.

5. RTP Topologies and Issues

The impact of how RTP multiplexing is performed will in general vary with how the RTP Session participants are interconnected, described by RTP Topology [RFC5117] and its intended successor [I-D.westerlund-avtcore-rtp-topologies-update].

5.1. Point to Point

Even the most basic use case, denoted Topo-Point-to-Point in [I-D.westerlund-avtcore-rtp-topologies-update], raises a number of considerations that are discussed in detail below (Section 6). They range over such aspects as:

- o Does my communication peer support RTP as defined with multiple SSRCS?
- o Do I need network differentiation in form of QoS?
- o Can the application more easily process and handle the media streams if they are in different RTP sessions?
- o Do I need to use additional media streams for RTP retransmission or FEC.
- o etc.

The point to point topology can contain one to many RTP sessions with one to many media sources per session, each having one or more RTP sources per media source.

5.2. Translators & Gateways

A point to point communication can end up in a situation when the peer it is communicating with is not compatible with the other peer for various reasons:

- o No common media codec for a media type thus requiring transcoding
- o Different support for multiple RTP sources and RTP sessions
- o Usage of different media transport protocols, i.e RTP or other.
- o Usage of different transport protocols, e.g. UDP, DCCP, TCP
- o Different security solutions, e.g. IPsec, TLS, DTLS, SRTP with different keying mechanisms.

In many situations this is resolved by the inclusion of a translator between the two peers, as described by Topo-PtP-Translator in [I-D.westerlund-avtcore-rtp-topologies-update]. The translator's main purpose is to make the peer look to the other peer like something it is compatible with. There can also be other reasons than compatibility to insert a translator in the form of a middlebox or gateway, for example a need to monitor the media streams. If the stream transport characteristics are changed by the translator, appropriate media handling can require thorough understanding of the application logic, specifically any congestion control or media adaptation.

5.3. Point to Multipoint Using Multicast

The Point to Multi-point topology is using Multicast to interconnect the session participants. This includes both Topo-ASM and Topo-SSM in [I-D.westerlund-avtcore-rtp-topologies-update].

Special considerations need to be made as multicast is a one to many distribution system. For example, the only practical method for adapting the bit-rate sent towards a given receiver for large groups is to use a set of multicast groups, where each multicast group represents a particular bit-rate. Otherwise the whole group gets media adapted to the participant with the worst conditions. The media encoding is either scalable, where multiple layers can be combined, or simulcast, where a single version is selected. By either selecting or combining multicast groups, the receiver can

control the bit-rate sent on the path to itself. It is also common that streams that improve transport robustness are sent in their own multicast group to allow for interworking with legacy or to support different levels of protection.

The result of this is some common behaviours for RTP multicast:

1. Multicast applications use a group of RTP sessions, not one. Each endpoint will need to be a member of a number of RTP sessions in order to perform well.
2. Within each RTP session, the number of RTP Sinks is likely to be much larger than the number of RTP sources.
3. Multicast applications need signalling functions to identify the relationships between RTP sessions.
4. Multicast applications need signalling functions to identify the relationships between SSRCs in different RTP sessions.

All multicast configurations share a signalling requirement; all of the participants will need to have the same RTP and payload type configuration. Otherwise, A could for example be using payload type 97 as the video codec H.264 while B thinks it is MPEG-2. It is to be noted that SDP offer/answer [RFC3264] is not appropriate for ensuring this property. The signalling aspects of multicast are not explored further in this memo.

Security solutions for this type of group communications are also challenging. First of all the key-management and the security protocol needs to support group communication. Source authentication requires special solutions. For more discussion on this please review Options for Securing RTP Sessions [I-D.ietf-avtcore-rtp-security-options].

5.4. Point to Multipoint Using an RTP Transport Translator

This mode is described as Topo-Translator in [I-D.westerlund-avtcore-rtp-topologies-update].

Transport Translators (Relays) result in an RTP session situation that is very similar to how an ASM group RTP session would behave.

One of the most important aspects with the simple relay is that it is only rewriting transport headers, no RTP modifications nor media transcoding occur. The most obvious downside of this basic relaying is that the translator has no control over how many streams need to be delivered to a receiver. Nor can it simply select to deliver only

certain streams, as this creates session inconsistencies: If the translator temporarily stops a stream, this prevents some receivers from reporting on it. From the sender's perspective it will look like a transport failure. Applications needing to stop or switch streams in the central node ought to consider using an RTP mixer to avoid this issue.

The Transport Translator has the same signalling requirement as multicast: All participants need to have the same payload type configuration. Most of the ASM security issues also arise here. Some alternatives when it comes to solution do exist, as there exists a central node to communicate with, one that also can enforce some security policies depending on the level of trust placed in the node.

5.5. Point to Multipoint Using an RTP Mixer

A mixer, described by Topo-Mixer in [I-D.westerlund-avtcore-rtp-topologies-update], is a centralised node that selects or mixes content in a conference to optimise the RTP session so that each endpoint only needs connect to one entity, the mixer. The media sent from the mixer to the endpoint 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 needed.

Mixers have some downsides, the first is that the mixer has to be a trusted node as they repacketize the media, and can perform media transformation operations. When using SRTP, both media operations and repacketization requires that the mixer verifies integrity, decrypts the content, performs the operation and forms new RTP packets, encrypts and integrity-protects them. This applies to all types of mixers. The second downside is that all these operations and optimisations of the session requires processing. How much depends on the implementation, as will become evident below.

A mixer, unlike a pure transport translator, is always application specific: the application logic for stream mixing or stream selection has to be embedded within the mixer, and controlled using application specific signalling. The implementation of a mixer can take several different forms, as discussed below.

A Mixer can also contain translator functionalities, like a media transcoder to adjust the media bit-rate or codec used for a particular RTP media stream.

6. RTP Multiplexing: When to Use Multiple RTP Sessions

Using multiple media streams is a well supported feature of RTP. However, it can be unclear for most implementers or people writing RTP/RTCP applications or extensions attempting to apply multiple streams when it is most appropriate to add an additional SSRC in an existing RTP session and when it is better to use multiple RTP sessions. This section tries to discuss the various considerations needed. The next section then concludes with some guidelines.

6.1. RTP and RTCP Protocol Considerations

This section discusses RTP and RTCP aspects worth considering when selecting between using an additional SSRC and Multiple RTP sessions.

6.1.1. The RTP Specification

RFC 3550 contains some recommendations and a bullet list with 5 arguments for different aspects of RTP multiplexing. Let's review Section 5.2 of [RFC3550], reproduced below:

"For efficient protocol processing, the number of multiplexing points should be minimised, as described in the integrated layer processing design principle [ALF]. In RTP, multiplexing is provided by the destination transport address (network address and port number) which is different for each RTP session. For example, in a teleconference composed of audio and video media encoded separately, each medium SHOULD be carried in a separate RTP session with its own destination transport address.

Separate audio and video streams SHOULD NOT be carried in a single RTP session and demultiplexed based on the payload type or SSRC fields. Interleaving packets with different RTP media types but using the same SSRC would introduce several problems:

1. If, say, two audio streams shared the same RTP session and the same SSRC value, and one were to change encodings and thus acquire a different RTP payload type, there would be no general way of identifying which stream had changed encodings.
2. An SSRC is defined to identify a single timing and sequence number space. Interleaving multiple payload types would require different timing spaces if the media clock rates differ and would require different sequence number spaces to tell which payload type suffered packet loss.
3. The RTCP sender and receiver reports (see Section 6.4) can only describe one timing and sequence number space per SSRC and do not carry a payload type field.

4. An RTP mixer would not be able to combine interleaved streams of incompatible media into one stream.
5. Carrying multiple media in one RTP session precludes: the use of different network paths or network resource allocations if appropriate; reception of a subset of the media if desired, for example just audio if video would exceed the available bandwidth; and receiver implementations that use separate processes for the different media, whereas using separate RTP sessions permits either single- or multiple-process implementations.

Using a different SSRC for each medium but sending them in the same RTP session would avoid the first three problems but not the last two.

On the other hand, multiplexing multiple related sources of the same medium in one RTP session using different SSRC values is the norm for multicast sessions. The problems listed above don't apply: an RTP mixer can combine multiple audio sources, for example, and the same treatment is applicable for all of them. It might also be appropriate to multiplex streams of the same medium using different SSRC values in other scenarios where the last two problems do not apply."

Let's consider one argument at a time. The first is an argument for using different SSRC for each individual media stream, which is very applicable.

The second argument is advocating against using payload type multiplexing, which still stands as can be seen by the extensive list of issues found in Appendix A.

The third argument is yet another argument against payload type multiplexing.

The fourth is an argument against multiplexing media streams that require different handling into the same session. As we saw in the discussion of RTP mixers, the RTP mixer has to embed application logic in order to handle streams anyway; the separation of streams according to stream type is just another piece of application logic, which might or might not be appropriate for a particular application. A type of application that can mix different media sources "blindly" is the audio only "telephone" bridge; most other type of application needs application-specific logic to perform the mix correctly.

The fifth argument discusses network aspects that we will discuss more below in Section 6.3. It also goes into aspects of implementation, like decomposed endpoints where different processes

or inter-connected devices handle different aspects of the whole multi-media session.

A summary of RFC 3550's view on multiplexing is to use unique SSRCs for anything that is its own media/packet stream, and to use different RTP sessions for media streams that don't share a media type. This document supports the first point; it is very valid. The later is one thing which is further discussed in this document as something the application developer needs to make a conscious choice for, but where imposing a single solution on all usages of RTP is inappropriate.

6.1.1.1. Different Media Types: Recommendations

The above quote from RTP [RFC3550] includes a strong recommendation:

"For example, in a teleconference composed of audio and video media encoded separately, each medium SHOULD be carried in a separate RTP session with its own destination transport address."

It was identified in "Why RTP Sessions Should Be Content Neutral" [I-D.alvestrand-rtp-sess-neutral] that the above statement is poorly supported by any of the motivations provided in the RTP specification. This has resulted in the creation of a specification Multiple Media Types in an RTP Session specification [I-D.ietf-avtcore-multi-media-rtp-session] which intends to update this recommendation. That document has a detailed analysis of the potential issues in having multiple media types in the same RTP session. This document tries to provide an moreover arching consideration regarding the usage of RTP session and considers multiple media types in one RTP session as possible choice for the RTP application designer.

6.1.2. Multiple SSRCs in a Session

Using multiple SSRCs in an RTP session at one endpoint requires resolving some unclear aspects of the RTP specification. These could potentially lead to some interoperability issues as well as some potential significant inefficiencies. These are further discussed in "RTP Considerations for Endpoints Sending Multiple Media Streams" [I-D.lennox-avtcore-rtp-multi-stream]. A application designer needs to consider these issues and the impact availability or lack of the optimization in the endpoints has on their application.

If an application will become affected by the issues described, using Multiple RTP sessions can mitigate these issues.

6.1.3. Handling Varying Sets of Senders

In some applications, the set of simultaneously active sources varies within a larger set of session members. A receiver can then possibly try to use a set of decoding chains that is smaller than the number of senders, switching the decoding chains between different senders. As each media decoding chain can contain state, either the receiver needs to either be able to save the state of swapped-out senders, or the sender needs to be able to send data that permits the receiver to reinitialise when it resumes activity.

This behaviour will cause similar issues independent of Additional SSRC or Multiple RTP session.

6.1.4. Cross Session RTCP Requests

There currently exists no functionality to make truly synchronised and atomic RTCP messages with some type of request semantics across multiple RTP Sessions. Instead, separate RTCP messages will have to be sent in each session. This gives streams in the same RTP session a slight advantage as RTCP messages for different streams in the same session can be sent in a compound RTCP packet, thus providing an atomic operation if different modifications of different streams are requested at the same time.

When using multiple RTP sessions, the RTCP timing rules in the sessions and the transport aspects, such as packet loss and jitter, prevents a receiver from relying on atomic operations, forcing it to use more robust and forgiving mechanisms.

6.1.5. Binding Related Sources

A common problem in a number of various RTP extensions has been how to bind related RTP sources and their media streams together. This issue is common to both using additional SSRCs and Multiple RTP sessions.

The solutions can be divided into some groups, RTP/RTCP based, Signalling based (SDP), grouping related RTP sessions, and grouping SSRCs within an RTP session. Most solutions are explicit, but some implicit methods have also been applied to the problem.

The SDP-based signalling solutions are:

SDP Media Description Grouping: The SDP Grouping Framework [RFC5888] uses various semantics to group any number of media descriptions. These has previously been considered primarily as grouping RTP sessions, but this might change.

SDP SSRC grouping: Source-Specific Media Attributes in SDP [RFC5576] includes a solution for grouping SSRCs the same way as the Grouping framework groups Media Descriptions.

SDP MSID grouping: Media Stream Identifiers [I-D.ietf-mmusic-msid] includes a solution for grouping SSRCs that is independent of their allocation to RTP sessions.

This supports a lot of use cases. All these solutions have shortcomings in cases where the session's dynamic properties are such that it is difficult or resource consuming to keep the list of related SSRCs up to date.

Within RTP/RTCP based solutions when binding to a endpoint or synchronization context, i.e. the CNAME has not be sufficient and one has multiple RTP sessions has been to using the same SSRC value across all the RTP sessions. RTP Retransmission [RFC4588] is multiple RTP session mode, Generic FEC [RFC5109], as well as the RTP payload format for Scalable Video Coding [RFC6190] in Multi Session Transmission (MST) mode uses this method. This method clearly works but might have some downside in RTP sessions with many participating SSRCs. The birthday paradox ensures that if you populate a single session with 9292 SSRCs at random, the chances are approximately 1% that at least one collision will occur. When a collision occur this will force one to change SSRC in all RTP sessions and thus resynchronizing all of them instead of only the single media stream having the collision.

It can be noted that Section 8.3 of the RTP Specification [RFC3550] recommends using a single SSRC space across all RTP sessions for layered coding.

Another solution that has been applied to binding SSRCs has been an implicit method used by RTP Retransmission [RFC4588] when doing retransmissions in the same RTP session as the source RTP media stream. This issues an RTP retransmission request, and then await a new SSRC carrying the RTP retransmission payload and where that SSRC is from the same CNAME. This limits a requestor to having only one outstanding request on any new source SSRCs per endpoint.

There exists no RTP/RTCP based mechanism capable of supporting explicit association accross multiple RTP sessions as well within an RTP session. A proposed solution for handling this issue is [I-D.westerlund-avtext-rtcp-sdes-srcname]. If accepted, this can potentially also be part of an SDP based solution also by reusing the same identifiers and name space.

6.1.6. Forward Error Correction

There exist a number of Forward Error Correction (FEC) based schemes for how to reduce the packet loss of the original streams. Most of the FEC schemes will protect a single source flow. The protection is achieved by transmitting a certain amount of redundant information that is encoded such that it can repair one or more packet losses over the set of packets they protect. This sequence of redundant information also needs to be transmitted as its own media stream, or in some cases instead of the original media stream. Thus many of these schemes create a need for binding related flows as discussed above. Looking at the history of these schemes, there are schemes using multiple SSRCs and schemes using multiple RTP sessions, and some schemes that support both modes of operation.

Using multiple RTP sessions supports the case where some set of receivers might not be able to utilise the FEC information. By placing it in a separate RTP session, it can easily be ignored.

In usages involving multicast, having the FEC information on its own multicast group, and therefore in its own RTP session, allows for flexibility. This is especially useful when receivers see very heterogeneous packet loss rates. Those receivers that are not seeing packet loss don't need to join the multicast group with the FEC data, and so avoid the overhead of receiving unnecessary FEC packets, for example.

6.1.7. Transport Translator Sessions

A basic Transport Translator relays any incoming RTP and RTCP packets to the other participants. The main difference between Additional SSRCs and Multiple RTP Sessions resulting from this use case is that with Additional SSRCs it is not possible for a particular session participant to decide to receive a subset of media streams. When using separate RTP sessions for the different sets of media streams, a single participant can choose to leave one of the sessions but not the other.

6.2. Interworking Considerations

There are several different kinds of interworking, and this section discusses two related ones. The interworking between different applications and the implications of potentially different choices of usage of RTP's multiplexing points. The second topic relates to what limitations have to be considered working with some legacy applications.

6.2.1. Types of Interworking

It is not uncommon that applications or services of similar usage, especially the ones intended for interactive communication, encounter a situation where one want to interconnect two or more of these applications.

In these cases one ends up in a situation where one might use a gateway to interconnect applications. This gateway then needs to change the multiplexing structure or adhere to limitations in each application.

There are two fundamental approaches to gatewaying: RTP Translator interworking (RTP bridging), where the gateway acts as an RTP Translator, and the two applications are members of the same RTP session, and Gateway Interworking (with RTP termination), where there are independent RTP sessions running from each interconnected application to the gateway.

6.2.2. RTP Translator Interworking

From an RTP perspective the RTP Translator approach could work if all the applications are using the same codecs with the same payload types, have made the same multiplexing choices, have the same capabilities in number of simultaneous media streams combined with the same set of RTP/RTCP extensions being supported. Unfortunately this might not always be true.

When one is gatewaying via an RTP Translator, a natural requirement is that the two applications being interconnected need to use the same approach to multiplexing. Furthermore, if one of the applications is capable of working in several modes (such as being able to use Additional SSRCs or Multiple RTP sessions at will), and the other one is not, successful interconnection depends on locking the more flexible application into the operating mode where interconnection can be successful, even if no participants using the less flexible application are present when the RTP sessions are being created.

6.2.3. Gateway Interworking

When one terminates RTP sessions at the gateway, there are certain tasks that the gateway has to carry out:

- o Generating appropriate RTCP reports for all media streams (possibly based on incoming RTCP reports), originating from SSRCs controlled by the gateway.

- o Handling SSRC collision resolution in each application's RTP sessions.
- o Signalling, choosing and policing appropriate bit-rates for each session.

If either of the applications has any security applied, e.g. in the form of SRTP, the gateway needs to be able to decrypt incoming packets and re-encrypt them in the other application's security context. This is necessary even if all that's needed is a simple remapping of SSRC numbers. If this is done, the gateway also needs to be a member of the security contexts of both sides, of course.

Other tasks a gateway might need to apply include transcoding (for incompatible codec types), rescaling (for incompatible video size requirements), suppression of content that is known not to be handled in the destination application, or the addition or removal of redundancy coding or scalability layers to fit the need of the destination domain.

From the above, we can see that the gateway needs to have an intimate knowledge of the application requirements; a gateway is by its nature application specific, not a commodity product.

This fact reveals the potential for these gateways to block evolution of the applications by blocking unknown RTP and RTCP extensions that the regular application has been extended with.

If one uses security functions, like SRTP, they can as seen above incur both additional risk due to the gateway needing to be in security association between the endpoints, unless the gateway is on the transport level, and additional complexities in form of the decrypt-encrypt cycles needed for each forwarded packet. SRTP, due to its keying structure, also requires that each RTP session needs different master keys, as use of the same key in two RTP sessions can result in two-time pads that completely breaks the confidentiality of the packets.

6.2.4. Multiple SSRC Legacy Considerations

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

When establishing RTP sessions that can contain endpoints that aren't updated to handle multiple streams following these recommendations, a particular application can have issues with multiple SSRCs within a single session. These issues include:

1. Need to handle more than one stream simultaneously rather than replacing an already existing stream with a new one.
2. Be capable of decoding multiple streams simultaneously.
3. Be capable of rendering multiple streams simultaneously.

This indicates that gateways attempting to interconnect to this class of devices has to make sure that only one media stream of each type gets delivered to the endpoint if it's expecting only one, and that the multiplexing format is what the device expects. It is highly unlikely that RTP translator-based interworking can be made to function successfully in such a context.

6.3. Network Considerations

The multiplexing choice has impact on network level mechanisms that need to be considered by the implementor.

6.3.1. Quality of Service

When it comes to Quality of Service mechanisms, they are either flow based or marking based. RSVP [RFC2205] is an example of a flow based mechanism, while Diff-Serv [RFC2474] is an example of a Marking based one. For a marking based scheme, the method of multiplexing will not affect the possibility to use QoS.

However, for a flow based scheme there is a clear difference between the methods. Additional SSRC will result in all media streams being part of the same 5-tuple (protocol, source address, destination address, source port, destination port) which is the most common selector for flow based QoS. Thus, separation of the level of QoS between media streams is not possible. That is however possible when

using multiple RTP sessions, where each media stream for which a separate QoS handling is desired can be in a different RTP session that can be sent over different 5-tuples.

6.3.2. NAT and Firewall Traversal

In today's network there exist a large number of middleboxes. The ones that normally have most impact on RTP are Network Address Translators (NAT) and Firewalls (FW).

Below we analyze and comment on the impact of requiring more underlying transport flows in the presence of NATs and Firewalls:

End-Point Port Consumption: A given IP address only has 65536 available local ports per transport protocol for all consumers of ports that exist on the machine. This is normally never an issue for an end-user machine. It can become an issue for servers that handle large number of simultaneous streams. However, if the application uses ICE to authenticate STUN requests, a server can serve multiple endpoints from the same local port, and use the whole 5-tuple (source and destination address, source and destination port, protocol) as identifier of flows after having securely bound them to the remote endpoint address using the STUN request. In theory the minimum number of media server ports needed are the maximum number of simultaneous RTP Sessions a single endpoint can use. In practice, implementation will probably benefit from using more server ports to simplify implementation or avoid performance bottlenecks.

NAT State: If an endpoint sits behind a NAT, each flow it generates to an external address will result in a state that has to be kept in the NAT. That state is a limited resource. In home or Small Office/Home Office (SOHO) NATs, memory or processing are usually the most limited resources. For large scale NATs serving many internal endpoints, available external ports are likely the scarce resource. Port limitations is primarily a problem for larger centralised NATs where endpoint independent mapping requires each flow to use one port for the external IP address. This affects the maximum number of internal users per external IP address. However, it is worth pointing out that a real-time video conference session with audio and video is likely using less than 10 UDP flows, compared to certain web applications that can use 100+ TCP flows to various servers from a single browser instance.

NAT Traversal Excess Time: Making the NAT/FW traversal takes a certain amount of time for each flow. It also takes time in a phase of communication between accepting to communicate and the media path being established which is fairly critical. The best

case scenario for how much extra time it takes after finding the first valid candidate pair following the specified ICE procedures are: $1.5 \cdot \text{RTT} + T_a \cdot (\text{Additional_Flows} - 1)$, where T_a is the pacing timer, which ICE specifies to be no smaller than 20 ms. That assumes a message in one direction, and then an immediate triggered check back. The reason it isn't more, is that ICE first finds one candidate pair that works prior to attempting to establish multiple flows. Thus, there is no extra time until one has found a working candidate pair. Based on that working pair the needed extra time is to in parallel establish the, in most cases 2-3, additional flows. However, packet loss causes extra delays, at least 100 ms, which is the minimal retransmission timer for ICE.

NAT Traversal Failure Rate: Due to the need to establish more than a single flow through the NAT, there is some risk that establishing the first flow succeeds but that one or more of the additional flows fail. The risk that this happens is hard to quantify, but ought to be fairly low as one flow from the same interfaces has just been successfully established. Thus only rare events such as NAT resource overload, or selecting particular port numbers that are filtered etc, ought to be reasons for failure.

Deep Packet Inspection and Multiple Streams: Firewalls differ in how deeply they inspect packets. There exist some potential that deeply inspecting firewalls will have similar legacy issues with multiple SSRCs as some stack implementations.

Additional SSRC keeps the additional media streams within one RTP Session and transport flow and does not introduce any additional NAT traversal complexities per media stream. This can be compared with normally one or two additional transport flows per RTP session when using multiple RTP sessions. Additional lower layer transport flows will be needed, unless an explicit de-multiplexing layer is added between RTP and the transport protocol. A proposal for how to multiplex multiple RTP sessions over the same single lower layer transport exist in [I-D.westerlund-avtcore-transport-multiplexing].

6.3.3. Multicast

Multicast groups provides a powerful semantics for a number of real-time applications, especially the ones that desire broadcast-like behaviours with one endpoint transmitting to a large number of receivers, like in IPTV. But that same semantics do result in a certain number of limitations.

One limitation is that for any group, sender side adaptation to the actual receiver properties causes degradation for all participants to

what is supported by the receiver with the worst conditions among the group participants. In most cases this is not acceptable. Instead various receiver based solutions are employed to ensure that the receivers achieve best possible performance. By using scalable encoding and placing each scalability layer in a different multicast group, the receiver can control the amount of traffic it receives. To have each scalability layer on a different multicast group, one RTP session per multicast group is used.

In addition, the transport flow considerations in multicast are a bit different from unicast; NATs are not useful in the multicast environment, meaning that the entire port range of each multicast address is available for distinguishing between RTP sessions.

Thus it appears easiest and most straightforward to use multiple RTP sessions for sending different media flows used for adapting to network conditions.

6.3.4. Multiplexing multiple RTP Session on a Single Transport

For applications that don't need flow based QoS and like to save ports and NAT/FW traversal costs and where usage of multiple media types in one RTP session is not suitable, there is a proposal for how to achieve multiplexing of multiple RTP sessions over the same lower layer transport [I-D.westerlund-avtcore-transport-multiplexing]. Using such a solution would allow Multiple RTP session without most of the perceived downsides of Multiple RTP sessions creating a need for additional transport flows, but this solution would require support from all functions that handle RTP packets, including firewalls.

6.4. Security and Key Management Considerations

When dealing with point-to-point, 2-member RTP sessions only, there are few security issues that are relevant to the choice of having one RTP session or multiple RTP sessions. However, there are a few aspects of multiparty sessions that might warrant consideration. For general information of possible methods of securing RTP, please review RTP Security Options [I-D.ietf-avtcore-rtp-security-options].

6.4.1. Security Context Scope

When using SRTP [RFC3711] the security context scope is important and can be a necessary differentiation in some applications. As SRTP's crypto suites (so far) are built around symmetric keys, the receiver will need to have the same key as the sender. This results in that no one in a multi-party session can be certain that a received packet really was sent by the claimed sender or by another party having

access to the key. In most cases this is a sufficient security property, but there are a few cases where this does create issues.

The first case is when someone leaves a multi-party session and one wants to ensure that the party that left can no longer access the media streams. This requires that everyone re-keys without disclosing the keys to the excluded party.

A second case is when using security as an enforcing mechanism for differentiation. Take for example a scalable layer or a high quality simulcast version which only premium users are allowed to access. The mechanism preventing a receiver from getting the high quality stream can be based on the stream being encrypted with a key that user can't access without paying premium, having the key-management limit access to the key.

SRTP [RFC3711] has no special functions for dealing with different sets of master keys for different SSRCS. The key-management functions have different capabilities to establish different set of keys, normally on a per endpoint basis. For example, DTLS-SRTP [RFC5764] and Security Descriptions [RFC4568] establish different keys for outgoing and incoming traffic from an endpoint. This key usage has to be written into the cryptographic context, possibly associated with different SSRCS.

6.4.2. Key Management for Multi-party session

Performing key-management for multi-party session can be a challenge. This section considers some of the issues.

Multi-party sessions, such as transport translator based sessions and multicast sessions, cannot use Security Description [RFC4568] nor DTLS-SRTP [RFC5764] without an extension as each endpoint provides its set of keys. In centralised conferences, the signalling counterpart is a conference server and the media plane unicast counterpart (to which DTLS messages would be sent) is the transport translator. Thus an extension like Encrypted Key Transport [I-D.ietf-avt-srtp-ekt] is needed or a MIKEY [RFC3830] based solution that allows for keying all session participants with the same master key.

6.4.3. Complexity Implications

The usage of security functions can surface complexity implications of the choice of multiplexing and topology. This becomes especially evident in RTP topologies having any type of middlebox that processes or modifies RTP/RTCP packets. Where there is very small overhead for an RTP translator or mixer to rewrite an SSRC value in the RTP packet

of an unencrypted session, the cost of doing it when using cryptographic security functions is higher. For example if using SRTP [RFC3711], the actual security context and exact crypto key are determined by the SSRC field value. If one changes it, the encryption and authentication tag needs to be performed using another key. Thus changing the SSRC value implies a decryption using the old SSRC and its security context followed by an encryption using the new one.

7. Archetypes

This section discusses some archetypes of how RTP multiplexing can be used in applications to achieve certain goals and a summary of their implications. For each archetype there is discussion of benefits and downsides.

7.1. Single SSRC per Session

In this archetype each endpoint in a point-to-point session has only a single SSRC, thus the RTP session contains only two SSRCs, one local and one remote. This session can be used both unidirectional, i.e. only a single media stream or bi-directional, i.e. both endpoints have one media stream each. If the application needs additional media flows between the endpoints, they will have to establish additional RTP sessions.

The Pros:

1. This archetype has great legacy interoperability potential as it will not tax any RTP stack implementations.
2. The signalling has good possibilities to negotiate and describe the exact formats and bit-rates for each media stream, especially using today's tools in SDP.
3. It does not matter if usage or purpose of the media stream is signalled on media stream level or session level as there is no difference.
4. It is possible to control security association per RTP media stream with current key-management, since each media stream is directly related to an RTP session, and the keying operates on a per-session basis.

The Cons:

- a. The number of RTP sessions grows directly in proportion with the number of media streams, which has the implications:

- * Linear growth of the amount of NAT/FW state with number of media streams.
 - * Increased delay and resource consumption from NAT/FW traversal.
 - * Likely larger signalling message and signalling processing requirement due to the amount of session related information.
 - * Higher potential for a single media stream to fail during transport between the endpoints.
- b. When the number of RTP sessions grows, the amount of explicit state for relating media stream also grows, linearly or possibly exponentially, depending on how the application needs to relate media streams.
 - c. The port consumption might become a problem for centralised services, where the central node's port consumption grows rapidly with the number of sessions.
 - d. For applications where the media streams are highly dynamic in their usage, i.e. entering and leaving, the amount of signalling can grow high. Issues arising from the timely establishment of additional RTP sessions can also arise.
 - e. Cross session RTCP requests might be needed, and the fact that they're impossible can cause issues.
 - f. If the same SSRC value is reused in multiple RTP sessions rather than being randomly chosen, interworking with applications that uses another multiplexing structure than this application will require SSRC translation.
 - g. Cannot be used with Any Source Multicast (ASM) as one cannot guarantee that only two endpoints participate as packet senders. Using SSM, it is possible to restrict to these requirements if no RTCP feedback is injected back into the SSM group.
 - h. For most security mechanisms, each RTP session or transport flow requires individual key-management and security association establishment thus increasing the overhead.

RTP applications that need to inter-work with legacy RTP applications, like most deployed VoIP and video conferencing solutions, can potentially benefit from this structure. However, a large number of media descriptions in SDP can also run into issues with existing implementations. For any application needing a larger

number of media flows, the overhead can become very significant. This structure is also not suitable for multi-party sessions, as any given media stream from each participant, although having same usage in the application, needs its own RTP session. In addition, the dynamic behaviour that can arise in multi-party applications can tax the signalling system and make timely media establishment more difficult.

7.2. Multiple SSRCs of the Same Media Type

In this archetype, each RTP session serves only a single media type. The RTP session can contain multiple media streams, either from a single endpoint or from multiple endpoints. This commonly creates a low number of RTP sessions, typically only one for audio and one for video, with a corresponding need for two listening ports when using RTP/RTCP multiplexing.

The Pros:

1. Low number of RTP sessions needed compared to single SSRC case. This implies:
 - * Reduced NAT/FW state
 - * Lower NAT/FW Traversal Cost in both processing and delay.
2. Allows for early de-multiplexing in the processing chain in RTP applications where all media streams of the same type have the same usage in the application.
3. Works well with media type de-composite endpoints.
4. Enables Flow-based QoS with different prioritisation between media types.
5. For applications with dynamic usage of media streams, i.e. they come and go frequently, having much of the state associated with the RTP session rather than an individual SSRC can avoid the need for in-session signalling of meta-information about each SSRC.
6. Low overhead for security association establishment.

The Cons:

- a. May have some need for cross session RTCP requests for things that affect both media types in an asynchronous way.

- b. Some potential for concern with legacy implementations that does not support the RTP specification fully when it comes to handling multiple SSRC per endpoint.
- c. Will not be able to control security association for sets of media streams within the same media type with today's key-management mechanisms, unless these are split into different RTP sessions.

For RTP applications where all media streams of the same media type share same usage, this structure provides efficiency gains in amount of network state used and provides more fate sharing with other media flows of the same type. At the same time, it is still maintaining almost all functionalities when it comes to negotiation in the signalling of the properties for the individual media type and also enabling flow based QoS prioritisation between media types. It handles multi-party session well, independently of multicast or centralised transport distribution, as additional sources can dynamically enter and leave the session.

7.3. Multiple Sessions for one Media type

In this archetype one goes one step further than in the above (Section 7.2) by using multiple RTP sessions also for a single media type, but still not as far as having a single SSRC per RTP session. The main reason for going in this direction is that the RTP application needs separation of the media streams due to their usage. Some typical reasons for going to this archetype are scalability over multicast, simulcast, need for extended QoS prioritisation of media streams due to their usage in the application, or the need for fine-grained signalling using today's tools.

The Pros:

1. More suitable for Multicast usage where receivers can individually select which RTP sessions they want to participate in, assuming each RTP session has its own multicast group.
2. Indication of the application's usage of the media stream, where multiple different usages exist.
3. Less need for SSRC specific explicit signalling for each media stream and thus reduced need for explicit and timely signalling.
4. Enables detailed QoS prioritisation for flow based mechanisms.
5. Works well with de-composite endpoints.

6. Handles dynamic usage of media streams well.
7. For transport translator based multi-party sessions, this structure allows for improved control of which type of media streams an endpoint receives.
8. The scope for who is included in a security association can be structured around the different RTP sessions, thus enabling such functionality with existing key-management.

The Cons:

- a. Increases the amount of RTP sessions compared to Multiple SSRCs of the Same Media Type.
- b. Increased amount of session configuration state.
- c. May need synchronised cross-session RTCP requests and require some consideration due to this.
- d. For media streams that are part of scalability, simulcast or transport robustness it will be needed to bind sources, which need to support multiple RTP sessions.
- e. Some potential for concern with legacy implementations that does not support the RTP specification fully when it comes to handling multiple SSRC per endpoint.
- f. Higher overhead for security association establishment.
- g. If the applications need finer control than on media type level over which session participants that are included in different sets of security associations, most of today's key-management will have difficulties establishing such a session.

For more complex RTP applications that have several different usages for media streams of the same media type and / or uses scalability or simulcast, this solution can enable those functions at the cost of increased overhead associated with the additional sessions. This type of structure is suitable for more advanced applications as well as multicast based applications requiring differentiation to different participants.

7.4. Multiple Media Types in one Session

This archetype is to use a single RTP session for multiple different media types, like audio and video, and possibly also transport robustness mechanisms like FEC or Retransmission. Each media stream will use its own SSRC and a given SSRC value from a particular endpoint will never use the SSRC for more than a single media type.

The Pros:

1. Single RTP session which implies:
 - * Minimal NAT/FW state.
 - * Minimal NAT/FW Traversal Cost.
 - * Fate-sharing for all media flows.
2. Enables separation of the different media types based on the payload types so media type specific endpoint or central processing can still be supported despite single session.
3. Can handle dynamic allocations of media streams well on an RTP level. Depends on the application's needs for explicit indication of the stream usage and how timely that can be signalled.
4. Minimal overhead for security association establishment.

The Cons:

- a. Less suitable for interworking with other applications that uses individual RTP sessions per media type or multiple sessions for a single media type, due to need of SSRC translation.
- b. Negotiation of bandwidth for the different media types is currently not possible in SDP. This requires SDP extensions to enable payload or source specific bandwidth. Likely to be a problem due to media type asymmetry in needed bandwidth.
- c. Not suitable for de-composite endpoints.
- d. Flow based QoS cannot provide separate treatment to some media streams compared to others in the single RTP session.

- e. If there is significant asymmetry between the media streams' RTCP reporting needs, there are some challenges in configuration and usage to avoid wasting RTCP reporting on the media stream that does not need that frequent reporting.
- f. Not suitable for applications where some receivers like to receive only a subset of the media streams, especially if multicast or transport translator is being used.
- g. Additional concern with legacy implementations that do not support the RTP specification fully when it comes to handling multiple SSRC per endpoint, as also multiple simultaneous media types needs to be handled.
- h. If the applications need finer control over which session participants that are included in different sets of security associations, most key-management will have difficulties establishing such a session.

7.5. Summary

There are some clear relations between these archetypes. Both the "single SSRC per RTP session" and the "multiple media types in one session" are cases which require full explicit signalling of the media stream relations. However, they operate on two different levels where the first primarily enables session level binding, and the second needs to do it all on SSRC level. From another perspective, the two solutions are the two extreme points when it comes to number of RTP sessions needed.

The two other archetypes "Multiple SSRCs of the Same Media Type" and "Multiple Sessions for one Media Type" are examples of two other cases that first of all allows for some implicit mapping of the role or usage of the media streams based on which RTP session they appear in. It thus potentially allows for less signalling and in particular reduced need for real-time signalling in dynamic sessions. They also represent points in between the first two when it comes to amount of RTP sessions established, i.e. representing an attempt to reduce the amount of sessions as much as possible without compromising the functionality the session provides both on network level and on signalling level.

8. Summary considerations and guidelines

8.1. Guidelines

This section contains a number of recommendations for implementors or specification writers when it comes to handling multi-stream.

Do not Require the same SSRC across Sessions: As discussed in Section 6.1.5 there exist drawbacks in using the same SSRC in multiple RTP sessions as a mechanism to bind related media streams together. It is instead suggested that a mechanism to explicitly signal the relation is used, either in RTP/RTCP or in the used signalling mechanism that establishes the RTP session(s).

Use additional SSRCs additional Media Sources: In the cases where an RTP endpoint needs to transmit additional media streams of the same media type in the application, with the same processing requirements at the network and RTP layers, it is suggested to send them as additional SSRCs in the same RTP session. For example a telepresence room where there are three cameras, and each camera captures 2 persons sitting at the table, sending each camera as its own SSRC within a single RTP session is suggested.

Use additional RTP sessions for streams with different requirements: When media streams have different processing requirements from the network or the RTP layer at the endpoints, it is suggested that the different types of streams are put in different RTP sessions. This includes the case where different participants want different subsets of the set of RTP streams.

When using multiple RTP Sessions use grouping: When using Multiple RTP session solutions, it is suggested to explicitly group the involved RTP sessions when needed using the signalling mechanism, for example The Session Description Protocol (SDP) Grouping Framework. [RFC5888], using some appropriate grouping semantics.

RTP/RTCP Extensions May Support Additional SSRCs as well as Multiple RTP sessions:

When defining an RTP or RTCP extension, the creator needs to consider if this extension is applicable to usage with additional SSRCs and Multiple RTP sessions. Any extension intended to be generic is suggested to support both. Applications that are not as generally applicable will have to consider if interoperability is better served by defining a single solution or providing both options.

Transport Support Extensions: When defining new RTP/RTCP extensions intended for transport support, like the retransmission or FEC mechanisms, they are expected to include support for both additional SSRCs and multiple RTP sessions so that application developers can choose freely from the set of mechanisms without concerning themselves with which of the multiplexing choices a particular solution supports.

9. IANA Considerations

This document makes no request of IANA.

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

10. Security Considerations

There is discussion of the security implications of choosing SSRC vs Multiple RTP session in Section 6.4.

11. References

11.1. Normative References

- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

11.2. Informative References

- [ALF] Clark, D. and D. Tennenhouse, "Architectural Considerations for a New Generation of Protocols", SIGCOMM Symposium on Communications Architectures and Protocols (Philadelphia, Pennsylvania), pp. 200--208, IEEE Computer Communications Review, Vol. 20(4), September 1990.
- [I-D.alvestrand-rtp-sess-neutral] Alvestrand, H., "Why RTP Sessions Should Be Content Neutral", draft-alvestrand-rtp-sess-neutral-01 (work in progress), June 2012.
- [I-D.ietf-avt-srtp-ekt] Wing, D., McGrew, D., and K. Fischer, "Encrypted Key Transport for Secure RTP", draft-ietf-avt-srtp-ekt-03 (work in progress), October 2011.
- [I-D.ietf-avtcore-6222bis] Begen, A., Perkins, C., Wing, D., and E. Rescorla, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", draft-ietf-avtcore-6222bis-06 (work in progress), July 2013.
- [I-D.ietf-avtcore-multi-media-rtp-session] Westerlund, M., Perkins, C., and J. Lennox, "Sending Multiple Types of Media in a Single RTP Session", draft-ietf-avtcore-multi-media-rtp-session-03 (work in progress), July 2013.

- [I-D.ietf-avtcore-rtp-security-options]
Westerlund, M. and C. Perkins, "Options for Securing RTP Sessions", draft-ietf-avtcore-rtp-security-options-03 (work in progress), May 2013.
- [I-D.ietf-avtext-multiple-clock-rates]
Petit-Huguenin, M. and G. Zorn, "Support for Multiple Clock Rates in an RTP Session", draft-ietf-avtext-multiple-clock-rates-09 (work in progress), April 2013.
- [I-D.ietf-mmusic-msid]
Alvestrand, H., "Cross Session Stream Identification in the Session Description Protocol", draft-ietf-mmusic-msid-00 (work in progress), February 2013.
- [I-D.ietf-mmusic-sdp-bundle-negotiation]
Holmberg, C., Alvestrand, H., and C. Jennings, "Multiplexing Negotiation Using Session Description Protocol (SDP) Port Numbers", draft-ietf-mmusic-sdp-bundle-negotiation-04 (work in progress), June 2013.
- [I-D.ietf-payload-rtp-howto]
Westerlund, M., "How to Write an RTP Payload Format", draft-ietf-payload-rtp-howto-04 (work in progress), June 2013.
- [I-D.lennox-avtcore-rtp-multi-stream]
Lennox, J., Westerlund, M., Wu, W., and C. Perkins, "RTP Considerations for Endpoints Sending Multiple Media Streams", draft-lennox-avtcore-rtp-multi-stream-02 (work in progress), February 2013.
- [I-D.lennox-mmusic-sdp-source-selection]
Lennox, J. and H. Schulzrinne, "Mechanisms for Media Source Selection in the Session Description Protocol (SDP)", draft-lennox-mmusic-sdp-source-selection-05 (work in progress), October 2012.
- [I-D.westerlund-avtcore-max-ssrc]
Westerlund, M., Burman, B., and F. Jansson, "Multiple Synchronization sources (SSRC) in RTP Session Signaling", draft-westerlund-avtcore-max-ssrc-02 (work in progress), July 2012.
- [I-D.westerlund-avtcore-rtp-topologies-update]
Westerlund, M. and S. Wenger, "RTP Topologies", draft-westerlund-avtcore-rtp-topologies-update-02 (work in progress), February 2013.

- [I-D.westerlund-avtcore-transport-multiplexing]
Westerlund, M. and C. Perkins, "Multiple RTP Sessions on a Single Lower-Layer Transport", draft-westerlund-avtcore-transport-multiplexing-05 (work in progress), February 2013.
- [I-D.westerlund-avtext-rtcp-sdes-srcname]
Westerlund, M., Burman, B., and P. Sandgren, "RTCP SDES Item SRCNAME to Label Individual Sources", draft-westerlund-avtext-rtcp-sdes-srcname-02 (work in progress), October 2012.
- [RFC2198] Perkins, C., Kouvelas, I., Hodson, O., Hardman, V., Handley, M., Bolot, J., Vega-Garcia, A., and S. Fosse-Parisis, "RTP Payload for Redundant Audio Data", RFC 2198, September 1997.
- [RFC2205] Braden, B., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.
- [RFC2326] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D.L. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.
- [RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, October 2000.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.
- [RFC3389] Zopf, R., "Real-time Transport Protocol (RTP) Payload for Comfort Noise (CN)", RFC 3389, September 2002.
- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.

- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.
- [RFC3830] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and K. Norrman, "MIKEY: Multimedia Internet KEYing", RFC 3830, August 2004.
- [RFC4103] Hellstrom, G. and P. Jones, "RTP Payload for Text Conversation", RFC 4103, June 2005.
- [RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.
- [RFC4568] Andreassen, F., Baugher, M., and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, July 2006.
- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.
- [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.
- [RFC5117] Westerlund, M. and S. Wenger, "RTP Topologies", RFC 5117, January 2008.
- [RFC5576] Lennox, J., Ott, J., and T. Schierl, "Source-Specific Media Attributes in the Session Description Protocol (SDP)", RFC 5576, June 2009.
- [RFC5583] Schierl, T. and S. Wenger, "Signaling Media Decoding Dependency in the Session Description Protocol (SDP)", RFC 5583, July 2009.
- [RFC5761] Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", RFC 5761, April 2010.
- [RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, May 2010.

- [RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.
- [RFC6190] Wenger, S., Wang, Y.-K., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, May 2011.
- [RFC6222] Begen, A., Perkins, C., and D. Wing, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", RFC 6222, April 2011.
- [RFC6285] Ver Steeg, B., Begen, A., Van Caenegem, T., and Z. Vax, "Unicast-Based Rapid Acquisition of Multicast RTP Sessions", RFC 6285, June 2011.
- [RFC6465] Ivov, E., Marocco, E., and J. Lennox, "A Real-time Transport Protocol (RTP) Header Extension for Mixer-to-Client Audio Level Indication", RFC 6465, December 2011.

Appendix A. Dismissing Payload Type Multiplexing

This section documents a number of reasons why using the payload type as a multiplexing point for most things related to multiple streams is unsuitable. If one attempts to use Payload type multiplexing beyond it's defined usage, that has well known negative effects on RTP. To use Payload type as the single discriminator for multiple streams implies that all the different media streams are being sent with the same SSRC, thus using the same timestamp and sequence number space. This has many effects:

1. Putting restraint on RTP timestamp rate for the multiplexed media. For example, media streams that use different RTP timestamp rates cannot be combined, as the timestamp values need to be consistent across all multiplexed media frames. Thus streams are forced to use the same rate. When this is not possible, Payload Type multiplexing cannot be used.
2. Many RTP payload formats can fragment a media object over multiple packets, like parts of a video frame. These payload formats need to determine the order of the fragments to correctly decode them. Thus it is important to ensure that all fragments related to a frame or a similar media object are transmitted in sequence and without interruptions within the object. This can relatively simple be solved on the sender side by ensuring that the fragments of each media stream are sent in sequence.

3. Some media formats require uninterrupted sequence number space between media parts. These are media formats where any missing RTP sequence number will result in decoding failure or invoking of a repair mechanism within a single media context. The text/T140 payload format [RFC4103] is an example of such a format. These formats will need a sequence numbering abstraction function between RTP and the individual media stream before being used with Payload Type multiplexing.
4. Sending multiple streams in the same sequence number space makes it impossible to determine which Payload Type and thus which stream a packet loss relates to.
5. If RTP Retransmission [RFC4588] is used and there is a loss, it is possible to ask for the missing packet(s) by SSRC and sequence number, not by Payload Type. If only some of the Payload Type multiplexed streams are of interest, there is no way of telling which missing packet(s) belong to the interesting stream(s) and all lost packets need be requested, wasting bandwidth.
6. The current RTCP feedback mechanisms are built around providing feedback on media streams based on stream ID (SSRC), packet (sequence numbers) and time interval (RTP Timestamps). There is almost never a field to indicate which Payload Type is reported, so sending feedback for a specific media stream is difficult without extending existing RTCP reporting.
7. The current RTCP media control messages [RFC5104] specification is oriented around controlling particular media flows, i.e. requests are done addressing a particular SSRC. Such mechanisms would need to be redefined to support Payload Type multiplexing.
8. The number of payload types are inherently limited. Accordingly, using Payload Type multiplexing limits the number of streams that can be multiplexed and does not scale. This limitation is exacerbated if one uses solutions like RTP and RTCP multiplexing [RFC5761] where a number of payload types are blocked due to the overlap between RTP and RTCP.
9. At times, there is a need to group multiplexed streams and this is currently possible for RTP Sessions and for SSRC, but there is no defined way to group Payload Types.
10. It is currently not possible to signal bandwidth requirements per media stream when using Payload Type Multiplexing.

11. Most existing SDP media level attributes cannot be applied on a per Payload Type level and would require re-definition in that context.
12. A legacy endpoint that doesn't understand the indication that different RTP payload types are different media streams might be slightly confused by the large amount of possibly overlapping or identically defined RTP Payload Types.

Appendix B. Proposals for Future Work

The above discussion and guidelines indicates that a small set of extension mechanisms could greatly improve the situation when it comes to using multiple streams independently of Multiple RTP session or Additional SSRC. These extensions are:

Media Source Identification: A Media source identification that can be used to bind together media streams that are related to the same media source. A proposal [I-D.westerlund-avtext-rtcp-sdes-srcname] exist for a new SDES item SRCNAME that also can be used with the a=ssrc SDP attribute to provide signalling layer binding information.

MSID: A Media Stream identification scheme that can be used to signal relationships between SSRCS that can be in the same or in different RTP sessions. Described in [I-D.ietf-mmusic-msid]

SSRC limitations within RTP sessions: By providing a signalling solution that allows the signalling peers to explicitly express both support and limitations on how many simultaneous media streams an endpoint can handle within a given RTP Session. That ensures that usage of Additional SSRC occurs when supported and without overloading an endpoint. This extension is proposed in [I-D.westerlund-avtcore-max-ssrc].

Appendix C. Signalling considerations

Signalling is not an architectural consideration for RTP itself, so this discussion has been moved to an appendix. However, it is hugely important for anyone building complete applications, so it is deserving of discussion.

The issues raised here need to be addressed in the WGs that deal with signalling; they cannot be addressed by tweaking, extending or profiling RTP.

C.1. Signalling Aspects

There exist various signalling solutions for establishing RTP sessions. Many are SDP [RFC4566] based, however SDP functionality is also dependent on the signalling protocols carrying the SDP. Where RTSP [RFC2326] and SAP [RFC2974] both use SDP in a declarative fashion, while SIP [RFC3261] uses SDP with the additional definition of Offer/Answer [RFC3264]. The impact on signalling and especially SDP needs to be considered as it can greatly affect how to deploy a certain multiplexing point choice.

C.1.1. Session Oriented Properties

One aspect of the existing signalling is that it is focused around sessions, or at least in the case of SDP the media description. There are a number of things that are signalled on a session level/media description but those are not necessarily strictly bound to an RTP session and could be of interest to signal specifically for a particular media stream (SSRC) within the session. The following properties have been identified as being potentially useful to signal not only on RTP session level:

- o Bitrate/Bandwidth exist today only at aggregate or a common any media stream limit, unless either codec-specific bandwidth limiting or RTCP signalling using TMMBR is used.
- o Which SSRC that will use which RTP Payload Types (this will be visible from the first media packet, but is sometimes useful to know before packet arrival).

Some of these issues are clearly SDP's problem rather than RTP limitations. However, if the aim is to deploy an solution using additional SSRCs that contains several sets of media streams with different properties (encoding/packetization parameter, bit-rate, etc), putting each set in a different RTP session would directly enable negotiation of the parameters for each set. If insisting on additional SSRC only, a number of signalling extensions are needed to clarify that there are multiple sets of media streams with different properties and that they need in fact be kept different, since a single set will not satisfy the application's requirements.

For some parameters, such as resolution and framerate, a SSRC-linked mechanism has been proposed:
[I-D.lennox-mmusic-sdp-source-selection].

C.1.2. SDP Prevents Multiple Media Types

SDP chose to use the m= line both to delineate an RTP session and to specify the top level of the MIME media type; audio, video, text, image, application. This media type is used as the top-level media type for identifying the actual payload format bound to a particular payload type using the rtpmap attribute. This binding has to be loosened in order to use SDP to describe RTP sessions containing multiple MIME top level types.

There is an accepted WG item in the MMUSIC WG to define how multiple media lines describe a single underlying transport [I-D.ietf-mmusic-sdp-bundle-negotiation] and thus it becomes possible in SDP to define one RTP session with media types having different MIME top level types.

C.1.3. Signalling Media Stream Usage

Media streams being transported in RTP has some particular usage in an RTP application. This usage of the media stream is in many applications so far implicitly signalled. For example, an application might choose to take all incoming audio RTP streams, mix them and play them out. However, in more advanced applications that use multiple media streams there will be more than a single usage or purpose among the set of media streams being sent or received. RTP applications will need to signal this usage somehow. The signalling used will have to identify the media streams affected by their RTP-level identifiers, which means that they have to be identified either by their session or by their SSRC + session.

In some applications, the receiver cannot utilise the media stream at all before it has received the signalling message describing the media stream and its usage. In other applications, there exists a default handling that is appropriate.

If all media streams in an RTP session are to be treated in the same way, identifying the session is enough. If SSRCS in a session are to be treated differently, signalling needs to identify both the session and the SSRC.

If this signalling affects how any RTP central node, like an RTP mixer or translator that selects, mixes or processes streams, treats the streams, the node will also need to receive the same signalling to know how to treat media streams with different usage in the right fashion.

Authors' Addresses

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

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

Bo Burman
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

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

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

Email: csp@csp Perkins.org

Harald Tveit Alvestrand
Google
Kungsbron 2
Stockholm 11122
Sweden

Email: harald@alvestrand.no

AVTCORE Working Group
Internet-Draft
Updates: 3550 (if approved)
Intended status: Standards Track
Expires: January 16, 2014

C. S. Perkins
University of Glasgow
V. Singh
Aalto University
July 15, 2013

Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions
draft-ietf-avtccore-rtp-circuit-breakers-03

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on January 16, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Terminology	3
3. Background	3
4. RTP Circuit Breakers for Systems Using the RTP/AVP Profile .	6
4.1. RTP/AVP Circuit Breaker #1: Media Timeout	7
4.2. RTP/AVP Circuit Breaker #2: RTCP Timeout	8
4.3. RTP/AVP Circuit Breaker #3: Congestion	9
4.4. Ceasing Transmission	12
5. RTP Circuit Breakers for Systems Using the RTP/AVPF Profile .	12
6. Impact of RTCP XR	13
7. Impact of RTCP Reporting Groups	14
8. Impact of Explicit Congestion Notification (ECN)	14
9. Security Considerations	14
10. IANA Considerations	15
11. Acknowledgements	15
12. References	15
12.1. Normative References	15
12.2. Informative References	15
Authors' Addresses	17

1. Introduction

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

This memo does not attempt to propose a new RTP congestion control algorithm. Rather, it proposes a minimal set of "circuit breakers";

conditions under which there is general agreement that an RTP flow is causing serious congestion, and ought to cease transmission. It is expected that future standards-track congestion control algorithms for RTP will operate within the envelope defined by this memo.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119]. This interpretation of these key words applies only when written in ALL CAPS. Mixed- or lower-case uses of these key words are not to be interpreted as carrying special significance in this memo.

3. Background

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

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

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

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

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

- o The base RTP specification [RFC3550] defines RTCP Reception Report (RR) packets to convey reception quality feedback information, and Sender Report (SR) packets to convey information about the media transmission. RTCP SR packets contain data that can be used to reconstruct media timing at a receiver, along with a count of the total number of octets and packets sent. RTCP RR packets report on the fraction of packets lost in the last reporting interval, the cumulative number of packets lost, the highest sequence number received, and the inter-arrival jitter. The RTCP RR packets also contain timing information that allows the sender to estimate the network round trip time (RTT) to the receivers. RTCP reports are sent periodically, with the reporting interval being determined by the number of SSRCs used in the session and a configured session bandwidth estimate (the number of SSRCs used is usually two in a unicast session, one for each participant, but can be greater if the participants send multiple media streams). The interval between reports sent from each receiver tends to be on the order of a few seconds on average, and it is randomised to avoid synchronisation of reports from multiple receivers. RTCP RR packets allow a receiver to report ongoing network congestion to the sender. However, if a receiver detects the onset of congestion partway through a reporting interval, the base RTP specification contains no provision for sending the RTCP RR packet early, and the receiver has to wait until the next scheduled reporting interval.
- o The RTCP Extended Reports (XR) [RFC3611] allow reporting of more complex and sophisticated reception quality metrics, but do not change the RTCP timing rules. RTCP extended reports of potential

interest for congestion control purposes are the extended packet loss, discard, and burst metrics [RFC3611], [I-D.ietf-xrblock-rtcp-xr-discard], [I-D.ietf-xrblock-rtcp-xr-discard-rle-metrics], [I-D.ietf-xrblock-rtcp-xr-burst-gap-discard], [I-D.ietf-xrblock-rtcp-xr-burst-gap-loss]; and the extended delay metrics [RFC6843], [RFC6798]. Other RTCP Extended Reports that could be helpful for congestion control purposes might be developed in future.

- o Rapid feedback about the occurrence of congestion events can be achieved using the Extended RTP Profile for RTCP-Based Feedback (RTP/AVPF) [RFC4585] in place of the more common RTP/AVP profile [RFC3551]. This modifies the RTCP timing rules to allow RTCP reports to be sent early, in some cases immediately, provided the average RTCP reporting interval remains unchanged. It also defines new transport-layer feedback messages, including negative acknowledgements (NACKs), that can be used to report on specific congestion events. The use of the RTP/AVPF profile is dependent on signalling, but is otherwise generally backwards compatible with the RTP/AVP profile, as it keeps the same average RTCP reporting interval as the base RTP specification. The RTP Codec Control Messages [RFC5104] extend the RTP/AVPF profile with additional feedback messages that can be used to influence that way in which rate adaptation occurs. The dynamics of how rapidly feedback can be sent are unchanged.
- o Finally, Explicit Congestion Notification (ECN) for RTP over UDP [RFC6679] can be used to provide feedback on the number of packets that received an ECN Congestion Experienced (CE) mark. This RTCP extension builds on the RTP/AVPF profile to allow rapid congestion feedback when ECN is supported.

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

Taken together, these various mechanisms allow receivers to provide feedback on the senders when congestion events occur, with varying degrees of timeliness and accuracy. The key distinction is between systems that use only the basic RTCP mechanisms, without RTP/AVPF rapid feedback, and those that use the RTP/AVPF extensions to respond to congestion more rapidly.

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

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

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

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

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

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

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

caused by congestion. Accordingly, we cannot use the RTT estimate alone as an RTP circuit breaker.

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

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

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

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

If RTP data packets are being sent, but the RTCP SR or RR packets reporting on that SSRC indicate a non-increasing extended highest sequence number received, this is an indication that those RTP data packets are not reaching the receiver. This could be a short-term issue affecting only a few packets, perhaps caused by a slow-to-open firewall or a transient connectivity problem, but if the issue persists, it is a sign of a more ongoing and significant problem. Accordingly, if a sender of RTP data packets receives two or more consecutive RTCP SR or RR packets from the same receiver, and those packets correspond to its transmission and have a non-increasing

extended highest sequence number received field (i.e., the sender receivers at least three RTCP SR or RR packets that report the same value in the extended highest sequence number received field for an SSRC, but the sender has sent RTP data packets for that SSRC that would have caused an increase in the reported value of the extended highest sequence number received if they had reached the receiver), then that sender SHOULD cease transmission (see Section 4.4).

The reason for waiting for two or more consecutive RTCP packets with a non-increasing extended highest sequence number is to give enough time for transient reception problems to resolve themselves, but to stop problem flows quickly enough to avoid causing serious ongoing network congestion. A single RTCP report showing no reception could be caused by a transient fault, and so will not cease transmission. Waiting for more than two consecutive RTCP reports before stopping a flow might avoid some false positives, but could lead to problematic flows running for a long time period (potentially tens of seconds, depending on the RTCP reporting interval) before being cut off.

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

In addition to media timeouts, as were discussed in Section 4.1, an RTP session has the possibility of an RTCP timeout. This can occur when RTP data packets are being sent, but there are no RTCP reports returned from the receiver. This is either due to a failure of the receiver to send RTCP reports, or a failure of the return path that is preventing those RTCP reporting from being delivered. In either case, it is not safe to continue transmission, since the sender has no way of knowing if it is causing congestion. Accordingly, an RTP sender that has not received any RTCP SR or RTCP RR packets reporting on the SSRC it is using for three or more RTCP reporting intervals SHOULD cease transmission (see Section 4.4). When calculating the timeout, the fixed minimum RTCP reporting interval SHOULD be used (based on the rationale in Section 6.2 of RFC 3550 [RFC3550]).

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

If a sender is transmitting a large number of RTP media streams, such that the corresponding RTCP SR or RR packets are too large to fit

into the network MTU, this will force the receiver to generate RTCP SR or RR packets in a round-robin manner. In this case, the sender MAY treat receipt of an RTCP SR or RR packet corresponding to an SSRC it sent using the same 5-tuple of source and destination IP address, port, and protocol, as an indication that the receiver and return path are working to prevent the RTCP timeout circuit breaker from triggering.

4.3. RTP/AVP Circuit Breaker #3: Congestion

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

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

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

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

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

where:

X is the transmit rate in bytes/second.

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

R is the round trip time in seconds.

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

t_RTO is the TCP retransmission timeout value in seconds, approximated by setting $t_RTO = 4 * R$.

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

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

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

It is RECOMMENDED that this simplified throughput equation be used, since the reduction in accuracy is small, and it is much simpler to calculate than the full equation.

Given this TCP equation, two parameters need to be estimated and reported to the sender in order to calculate the throughput: the round trip time, R, and the loss event rate, p (the packet size, s, is known to the sender). The round trip time can be estimated from RTCP SR and RR packets. This is done too infrequently for accurate statistics, but is the best that can be done with the standard RTCP mechanisms.

Report blocks in RTCP SR or RR packets contain the packet loss fraction, rather than the loss event rate, so p cannot be reported (TCP typically treats the loss of multiple packets within a single RTT as one loss event, but RTCP RR packets report the overall fraction of packets lost, not caring about when the losses occurred). Using the loss fraction in place of the loss event rate can overestimate the loss. We believe that this overestimate will not be significant, given that we are only interested in order of magnitude

comparison ([Floyd] section 3.2.1 shows that the difference is small for steady-state conditions and random loss, but using the loss fraction is more conservative in the case of bursty loss).

The congestion circuit breaker is therefore: when a sender receives an RTCP SR or RR packet that contains a report block for an SSRC it is using, that sender has to check the fraction lost field in that report block to determine if there is a non-zero packet loss rate. If the fraction lost field is zero, then continue sending as normal. If the fraction lost is greater than zero, then estimate the TCP throughput using the simplified equation above, and the measured R , p (approximated by the fraction lost), and s . Compare this with the actual sending rate. If the actual sending rate is more than ten times the estimated sending rate derived from the TCP throughput equation for two consecutive RTCP reporting intervals, the sender SHOULD cease transmission (see Section 4.4). Systems that usually send at a high data rate, but that can reduce their data rate significantly (i.e., by at least a factor of ten), MAY first reduce their sending rate to this lower value to see if this resolves the congestion, but MUST then cease transmission if the problem does not resolve itself within a further two RTCP reporting intervals (see Section 4.4). An example of this might be a video conferencing system that backs off to sending audio only, before completely dropping the call. If such a reduction in sending rate resolves the congestion problem, the sender MAY gradually increase the rate at which it sends data after a reasonable amount of time has passed, provided it takes care not to cause the problem to recur ("reasonable" is intentionally not defined here).

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

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

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

4.4. Ceasing Transmission

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

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

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

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

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

congestion control algorithm chosen, they cannot be used as part of the generic RTP circuit breaker.

If the extension for Reduced-Size RTCP [RFC5506] is not used, early RTCP feedback packets sent according to the RTP/AVPF profile will be compound RTCP packets that include an RTCP SR/RR packet. That RTCP SR/RR packet MUST be processed as if it were sent as a regular RTCP report and counted towards the circuit breaker conditions specified in Section 4 of this memo. This will potentially make the RTP circuit breaker fire earlier than it would if the RTP/AVPF profile was not used.

Reduced-size RTCP reports sent under the RTP/AVPF early feedback rules that do not contain an RTCP SR or RR packet MUST be ignored by the RTP circuit breaker (they do not contain the information used by the circuit breaker algorithm). Reduced-size RTCP reports sent under the RTP/AVPF early feedback rules that contain RTCP SR or RR packets MUST be processed as if they were sent as regular RTCP reports, and counted towards the circuit breaker conditions specified in Section 4 of this memo. This will potentially make the RTP circuit breaker fire earlier than it would if the RTP/AVPF profile was not used.

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

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

6. Impact of RTCP XR

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

SR packets, RTCP RR packets, or RTCP XR ECN Summary Report packets, are used by the RTP circuit breaker algorithm.

7. Impact of RTCP Reporting Groups

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

8. Impact of Explicit Congestion Notification (ECN)

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

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

9. Security Considerations

The security considerations of [RFC3550] apply.

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

If non-authenticated RTCP reports are used, an on-path attacker can trivially generate fake RTCP packets that indicate high packet loss rates, causing the circuit breaker to trigger and disrupting an RTP session. This is somewhat more difficult for an off-path attacker, due to the need to guess the randomly chosen RTP SSRC value and the RTP sequence number. This attack can be avoided if RTCP packets are authenticated, for example using the Secure RTP profile [RFC3711].

10. IANA Considerations

There are no actions for IANA.

11. Acknowledgements

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

12. References

12.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3448] Handley, M., Floyd, S., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 3448, January 2003.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC3551] Schulzrinne, H. and S. Casner, "RTP Profile for Audio and Video Conferences with Minimal Control", STD 65, RFC 3551, July 2003.
- [RFC3611] Friedman, T., Caceres, R., and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, November 2003.
- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.

12.2. Informative References

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

Lennox, J., Westerlund, M., Wu, W., and C. Perkins,
"Sending Multiple Media Streams in a Single RTP Session:
Grouping RTCP Reception Statistics and Other Feedback",
draft-ietf-avtcore-rtp-multi-stream-optimisation-00 (work
in progress), July 2013.

[I-D.ietf-xrblock-rtcp-xr-burst-gap-discard]

Clark, A., Huang, R., and W. Wu, "RTP Control
Protocol(RTCP) Extended Report (XR) Block for Burst/Gap
Discard metric Reporting", draft-ietf-xrblock-rtcp-xr-
burst-gap-discard-14 (work in progress), April 2013.

[I-D.ietf-xrblock-rtcp-xr-burst-gap-loss]

Clark, A., Zhang, S., Zhao, J., and W. Wu, "RTP Control
Protocol (RTCP) Extended Report (XR) Block for Burst/Gap
Loss metric Reporting", draft-ietf-xrblock-rtcp-xr-burst-
gap-loss-12 (work in progress), April 2013.

[I-D.ietf-xrblock-rtcp-xr-discard-rle-metrics]

Ott, J., Singh, V., and I. Curcio, "RTP Control Protocol
(RTCP) Extended Reports (XR) for Run Length Encoding (RLE)
of Discarded Packets", draft-ietf-xrblock-rtcp-xr-discard-
rle-metrics-06 (work in progress), July 2013.

[I-D.ietf-xrblock-rtcp-xr-discard]

Clark, A., Zorn, G., and W. Wu, "RTP Control Protocol
(RTCP) Extended Report (XR) Block for Discard Count metric
Reporting", draft-ietf-xrblock-rtcp-xr-discard-15 (work in
progress), June 2013.

[Padhye]

Padhye, J., Firoiu, V., Towsley, D., and J. Kurose,
"Modeling TCP Throughput: A Simple Model and its Empirical
Validation", Proc. ACM SIGCOMM 1998, DOI 10.1145/
285237.285291, August 1998.

[RFC3168]

Ramakrishnan, K., Floyd, S., and D. Black, "The Addition
of Explicit Congestion Notification (ECN) to IP", RFC
3168, September 2001.

[RFC3711]

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

[RFC5104]

Wenger, S., Chandra, U., Westerlund, M., and B. Burman,
"Codec Control Messages in the RTP Audio-Visual Profile
with Feedback (AVPF)", RFC 5104, February 2008.

- [RFC5450] Singer, D. and H. Desineni, "Transmission Time Offsets in RTP Streams", RFC 5450, March 2009.
- [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.
- [RFC6051] Perkins, C. and T. Schierl, "Rapid Synchronisation of RTP Flows", RFC 6051, November 2010.
- [RFC6679] Westerlund, M., Johansson, I., Perkins, C., O'Hanlon, P., and K. Carlberg, "Explicit Congestion Notification (ECN) for RTP over UDP", RFC 6679, August 2012.
- [RFC6798] Clark, A. and Q. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Packet Delay Variation Metric Reporting", RFC 6798, November 2012.
- [RFC6843] Clark, A., Gross, K., and Q. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Delay Metric Reporting", RFC 6843, January 2013.

Authors' Addresses

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

Email: csp@csp Perkins.org

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

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

AVTCORE
Internet-Draft
Updates: 3550 (if approved)
Intended status: Standards Track
Expires: January 12, 2014

J. Lennox
Vidyo
M. Westerlund
Ericsson
Q. Wu
Huawei
C. Perkins
University of Glasgow
July 11, 2013

Sending Multiple Media Streams in a Single RTP Session
draft-ietf-avtcore-rtp-multi-stream-01

Abstract

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

This document updates RFC 3550 in regards to handling of multiple SSRCs per endpoint in RTP sessions.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 12, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Terminology	3
3. Use Cases For Multi-Stream Endpoints	3
3.1. Multiple-Capturer Endpoints	3
3.2. Multi-Media Sessions	3
3.3. Multi-Stream Mixers	4
4. Multi-Stream Endpoint RTP Media Recommendations	4
5. Multi-Stream Endpoint RTCP Recommendations	4
5.1. RTCP Reporting Requirement	5
5.2. Initial Reporting Interval	5
5.3. Compound RTCP Packets	5
6. RTCP Considerations for Streams with Disparate Rates	7
6.1. Timing out SSRCS	8
6.2. Tuning RTCP transmissions	9
7. Security Considerations	11
8. Open Issues	12
9. IANA Considerations	12
10. References	12
10.1. Normative References	12
10.2. Informative References	13
Appendix A. Changes From Earlier Versions	14
A.1. Changes From WG Draft -00	14
A.2. Changes From Individual Draft -02	14
A.3. Changes From Individual Draft -01	14
A.4. Changes From Individual Draft -00	14
Authors' Addresses	15

1. Introduction

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

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

The purpose of this document is to expand and clarify [RFC3550]'s language for these use cases. The authors believe this does not result in any major normative changes to the RTP specification, however this document defines how the RTP specification is to be interpreted. In these cases, this document updates RFC3550.

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

2. Terminology

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

3. Use Cases For Multi-Stream Endpoints

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

3.1. Multiple-Capturer Endpoints

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

3.2. Multi-Media Sessions

Recent work has been done in RTP [I-D.ietf-avtcore-multi-media-rtp-session] and SDP

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

3.3. Multi-Stream Mixers

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

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

More complicated is the Source Projecting Mixer, see Section 3.6 of [I-D.ietf-avtcore-rtp-topologies-update]. This is a central box that receives media streams from several endpoints, and then selectively forwards modified versions of some of the streams toward the other endpoints it is connected to. Toward one destination, a separate media source appears in the session for every other source connected to the mixer, "projected" from the original streams, but at any given time many of them can appear to be inactive (and thus are receivers, not senders, in RTP). This sort of device is closer to being an RTP mixer than an RTP translator, in that it terminates RTCP reporting about the mixed streams, and it can re-write SSRCs, timestamps, and sequence numbers, as well as the contents of the RTP payloads, and can turn sources on and off at will without appearing to be generating packet loss. Each projected stream will typically preserve its original RTCP source description (SDS) information.

4. Multi-Stream Endpoint RTP Media Recommendations

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

5. Multi-Stream Endpoint RTCP Recommendations

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

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

(tbd: rather than think in terms of media streams, it might be clearer to refer to SSRC values, where a participant with multiple active SSRC values counts as multiple participants, once per SSRC)

5.1. RTCP Reporting Requirement

For each of an endpoint's media streams, whether or not it is currently sending media, SR/RR and SDES packets MUST be sent at least once per RTCP report interval. (For discussion of the content of SR or RR packets' reception statistic reports, see [I-D.ietf-avtcore-rtp-multi-stream-optimisation].)

5.2. Initial Reporting Interval

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

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

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

5.3. Compound RTCP Packets

Section 6.1 gives the following advice to RTP translators and mixers:

It is RECOMMENDED that translators and mixers combine individual RTCP packets from the multiple sources they are forwarding into one compound packet whenever feasible in order to amortize the packet overhead (see Section 7). An example RTCP compound packet as might be produced by a mixer is shown in Fig. 1. If the overall length of a compound packet would exceed the MTU of the network path, it SHOULD be segmented into multiple shorter compound packets to be transmitted in separate packets of the underlying protocol. This does not impair the RTCP bandwidth estimation because each compound packet represents at least one distinct participant. Note that each of the compound packets MUST begin with an SR or RR packet.

Note: To avoid confusion, an RTCP packet is an individual item, such as a Sender Report (SR), Receiver Report (RR), Source Description (SDS), Goodbye (BYE), Application Defined (APP), Feedback [RFC4585] or Extended Report (XR) [RFC3611] packet. A compound packet is the combination of two or more such RTCP packets where the first packet has to be an SR or an RR packet, and which contains a SDS packet containing an CNAME item. Thus the above results in compound RTCP packets that contain multiple SR or RR packets from different sources as well as any of the other packet types. There are no restrictions on the order in which the packets can occur within the compound packet, except the regular compound rule, i.e., starting with an SR or RR.

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

Due to RTCP's randomization of reporting times, there is a fair bit of tolerance in precisely when an endpoint schedules RTCP to be sent. Thus, one potential way of implementing this recommendation would be to randomize all of an endpoint's sources together, with a single randomization schedule, so an MTU's worth of RTCP all comes out simultaneously.

(tbd: Multiplexing RTCP packets from multiple different sources might require some adjustment to the calculation of RTCP's avg_rtcp_size, as the RTCP group interval is proportional to avg_rtcp_size times the group size).

6. RTCP Considerations for Streams with Disparate Rates

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

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

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

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

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

6.1. Timing out SSRCS

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

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

Note: If endpoints implementing the RTP/AVP and RTP/AVPF profiles (or their secure variants) are combined in a single RTP session, and the RTP/AVPF endpoints use a non-zero `T_rr_interval` that is significantly lower than 5 seconds, then there is a risk that the RTP/AVP endpoints will prematurely timeout the RTP/AVPF endpoints due to their different RTCP timeout intervals. Since an RTP session can only use a single RTP profile, this issue ought never occur. If such mixed RTP profiles are used, however, the RTP/AVPF session MUST NOT use a non-zero `T_rr_interval` that is smaller than 5 seconds.

(tbd: it has been suggested that a minimum non-zero $T_{rr_interval}$ of 4 seconds is more appropriate, due to the nature of the timing rules).

6.2. Tuning RTCP transmissions

This sub-section discusses what tuning can be done to reduce the downsides of the shared RTCP packet intervals.

When using the RTP/AVP or RTP/SAVP profiles the tuning one can do is very limited. The controls one has are limited to the RTCP bandwidth values and whether the minimum RTCP interval is scaled according to the bandwidth. As the scheduling algorithm includes both random factors and reconsideration, one can't simply calculate the expected average transmission interval using the formula for T_d . But it does indicate the important factors affecting the transmission interval, namely the RTCP bandwidth available for the role (Active Sender or Participant), the average RTCP packet size, and the number of SSRCs classified in the relevant role. Note that if the ratio of senders to total number of session participants is larger than the ratio of RTCP bandwidth for senders in relation to the total RTCP bandwidth, then senders and receivers are treated together.

Let's start with some basic observations:

- a. Unless the scaled minimum RTCP interval is used, then T_d prior to randomization and reconsideration can never be less than 5 seconds (assuming default T_{min} of 5 seconds).
- b. If the scaled minimum RTCP interval is used, T_d can become as low as 360 divided by RTP Session bandwidth in kilobits. In SDP the RTP session bandwidth is signalled using $b=AS$. An RTP Session bandwidth of 72 kbps results in T_{min} being 5 seconds. An RTP session bandwidth of 360 kbps of course gives a T_{min} of 1 second, and to achieve a T_{min} equal to once every frame for a 25 Hz video stream requires an RTP session bandwidth of 9 Mbps! (The use of the RTP/AVPF or RTP/SAVPF profile allows a smaller T_{min} , and hence more frequent RTCP reports, as discussed below).
- c. Let's calculate the number (n) of SSRCs in the RTP session that 5% of the session bandwidth can support to yield a T_d value equal to T_{min} with minimal scaling. For this calculation we have to make two assumptions. The first is that we will consider most or all SSRC being senders, resulting in everyone sharing the available bandwidth. Secondly we will select an average RTCP packet size. This packet will consist of an SR, containing ($n-1$) report blocks up to 31 report blocks, and an SDES item with at least a CNAME (17 bytes in size) in it. Such a basic packet will

be 800 bytes for $n \geq 32$. With these parameters, and as the bandwidth goes up the time interval is proportionally decreased (due to minimal scaling), thus all the example bandwidths 72 kbps, 360 kbps and 9 Mbps all support 9 SSRCS.

- d. The actual transmission interval for a T_d value is $[0.5 \cdot T_d / 1.21828, 1.5 \cdot T_d / 1.21828]$, which means that for $T_d = 5$ seconds, the interval is actually $[2.052, 6.156]$ and the distribution is not uniform, but rather exponentially-increasing. The probability for sending at time X , given it is within the interval, is probability of picking X in the interval times the probability to randomly picking a number that is $\leq X$ within the interval with an uniform probability distribution. This results in that the majority of the probability mass is above the T_d value.

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

When using RTP/AVPF or RTP/SAVPF we get a quite powerful additional tool, the setting of the $T_{rr_interval}$ which has several effects on the RTCP reporting. First of all as T_{min} is set to 0 after the initial transmission, the regular reporting interval is instead determined by the regular bandwidth based calculation and the $T_{rr_interval}$. This has the effect that we are no longer restricted by the minimal interval or even the scaling rule for the minimal rule. Instead the RTCP bandwidth and the $T_{rr_interval}$ are the governing factors. Now it also becomes important to separate between the application's need for regular reports and RTCP feedback packet types. In both regular RTCP mode, as in Early RTCP Mode, the usage of the $T_{rr_interval}$ prevents regular RTCP packets, i.e. packets without any Feedback packets, to be sent more often than $T_{rr_interval}$. This value is as hard as no regular RTCP packet can be sent less than $T_{rr_interval}$ after the previous regular packet.

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

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

When using `T_rr_interval`, i.e. having it be non zero, there are configurations that have to be avoided. If the resulting `Td` value is smaller but close to `T_rr_interval` then the interval in which the actual regular RTCP packet transmission falls into becomes very large, from 0.5 times `T_rr_interval` up to 2.73 times the `T_rr_interval`. Therefore for configuration where one intends to have `Td` smaller than `T_rr_interval`, then `Td` is RECOMMENDED to be targeted at values less than 1/4th of `T_rr_interval` which results in that the range becomes $[0.5 * T_rr_interval, 1.81 * T_rr_interval]$.

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

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

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

7. Security Considerations

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

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

8. Open Issues

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

1. Further clarifications on how to handle the RTCP scheduler when sending multiple sources in one compound packet.
2. How is the RTCP avg_rtcp_size be calculated when RTCP packets are routinely multiplexed among multiple RTCP senders?
3. Do we need to provide a recommendation for unicast session joiners with many sources to not use 0 initial minimal interval from bit-rate burst perspective?

9. IANA Considerations

No IANA actions needed.

10. References

10.1. Normative References

- [I-D.ietf-avtcore-6222bis]
Begen, A., Perkins, C., Wing, D., and E. Rescorla,
"Guidelines for Choosing RTP Control Protocol (RTCP)
Canonical Names (CNAMEs)", draft-ietf-avtcore-6222bis-04
(work in progress), June 2013.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V.
Jacobson, "RTP: A Transport Protocol for Real-Time
Applications", STD 64, RFC 3550, July 2003.

- [RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.
- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.
- [RFC5124] Ott, J. and E. Carrara, "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)", RFC 5124, February 2008.
- [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.

10.2. Informative References

- [I-D.ietf-avtcore-multi-media-rtp-session]
Westerlund, M., Perkins, C., and J. Lennox, "Multiple Media Types in an RTP Session", draft-ietf-avtcore-multi-media-rtp-session-02 (work in progress), February 2013.
- [I-D.ietf-avtcore-rtp-multi-stream-optimisation]
Lennox, J., Westerlund, M., Wu, Q., and C. Perkins, "Sending Multiple Media Streams in a Single RTP Session: Grouping RTCP Reception Statistics and Other Feedback ", draft-ietf-avtcore-rtp-multi-stream-optimisation-00 (work in progress), July 2013.
- [I-D.ietf-avtcore-rtp-topologies-update]
Westerlund, M. and S. Wenger, "RTP Topologies", draft-ietf-avtcore-rtp-topologies-update-00 (work in progress), April 2013.
- [I-D.ietf-clue-framework]
Duckworth, M., Pepperell, A., and S. Wenger, "Framework for Telepresence Multi-Streams", draft-ietf-clue-framework-10 (work in progress), May 2013.
- [I-D.ietf-mmusic-sdp-bundle-negotiation]
Holmberg, C., Alvestrand, H., and C. Jennings, "Multiplexing Negotiation Using Session Description Protocol (SDP) Port Numbers", draft-ietf-mmusic-sdp-bundle-negotiation-04 (work in progress), June 2013.

- [RFC3611] Friedman, T., Caceres, R., and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, November 2003.
- [RFC3830] Arkko, J., Carrara, E., Lindholm, F., Naslund, M., and K. Norrman, "MIKEY: Multimedia Internet KEYing", RFC 3830, August 2004.

Appendix A. Changes From Earlier Versions

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

A.1. Changes From WG Draft -00

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

A.2. Changes From Individual Draft -02

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

A.3. Changes From Individual Draft -01

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

A.4. Changes From Individual Draft -00

- o Added the Reporting Group semantic to explicitly indicate which sources come from a single endpoint, rather than leaving it implicit.

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

Authors' Addresses

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

Email: jonathan@vidyo.com

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

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

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

Email: sunseawq@huawei.com

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

Email: csp@csp Perkins.org

AVTCORE WG
Internet-Draft
Updates: 3550 (if approved)
Intended status: Standards Track
Expires: January 12, 2014

J. Lennox
Vidyo
M. Westerlund
Ericsson
Q. Wu
Huawei
C. Perkins
University of Glasgow
July 11, 2013

Sending Multiple Media Streams in a Single RTP Session: Grouping RTCP
Reception Statistics and Other Feedback
draft-ietf-avtccore-rtp-multi-stream-optimisation-00

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on January 12, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
2. Terminology	3
3. Grouping of RTCP Reception Statistics and Other Feedback . .	3
3.1. Semantics and Behavior of Reporting Groups	3
3.2. Determine the Report Group	4
3.3. RTCP Packet Reporting Group's Reporting Sources	5
3.4. RTCP Source Description (SDS) item for Reporting Groups	6
3.5. Middlebox Considerations	6
3.6. SDP signaling for Reporting Groups	6
3.7. Bandwidth Benefits of RTCP Reporting Groups	6
3.8. Consequences of RTCP Reporting Groups	7
4. Security Considerations	8
5. IANA Considerations	8
6. References	8
6.1. Normative References	8
6.2. Informative References	9
Authors' Addresses	9

1. Introduction

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

An RTP endpoint will have one or more synchronisation sources (SSRCs) that send and receive media streams (it will have one SSRC for each media stream it sends). Each SSRC has to send RTCP sender reports corresponding to the RTP packets it sends, and receiver reports for traffic it receives. That is, every SSRC will send RTCP packets to

report on every other SSRC. This rule is simple, but can be quite inefficient for endpoints that send large numbers of media streams in a single RTP session. Consider a session comprising ten participants, each sending three media streams with their own SSRC. There will be 30 SSRCs in such an RTP session, and 30 RTCP reception reports will be sent per reporting interval as each SSRC reports on all the others. However, the three SSRCs comprising each participant will almost certainly see identical reception quality, since they are co-located. If there was a way to indicate that several SSRCs are co-located, and see the same reception quality, then two-thirds of those RTCP reports could be suppressed.

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

2. Terminology

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

3. Grouping of RTCP Reception Statistics and Other Feedback

3.1. Semantics and Behavior of Reporting Groups

An RTCP Reporting Group indicates that a set of sources (SSRCs) that originate from a single entity (endpoint or middlebox) in an RTP session, and therefore all the sources in the group have an identical view of the network. If reporting groups are in use, two sources SHOULD be put into the same reporting group if their view of the network is identical; i.e., if they report on traffic received at the same interface of an RTP endpoint. Sources with different views of the network MUST NOT be put into the same reporting group.

If reporting groups are in use, an endpoint MUST NOT send reception reports from one source in a reporting group about another one in the same group ("self-reports"). Similarly, an endpoint MUST NOT send reception reports about a remote media source from more than one source in a reporting group ("cross-reports"). Instead, it MUST pick one of its local media sources as the "reporting" source for each remote media source, and use it to send reception reports about that remote source; all the other media sources in the reporting group MUST NOT send any reception reports about that remote media source.

This reporting source MUST also be the source for any RTP/AVPF Feedback [RFC4585] or Extended Report (XR) [RFC3611] packets about

the corresponding remote sources as well. If a reporting source leaves the session (i.e., if it sends a BYE, or leaves the group without sending BYE under the rules of [RFC3550] section 6.3.7), another reporting source MUST be chosen if any members of the group still exist.

An endpoint or middlebox MAY use multiple sources as reporting sources; however, each reporting source MUST have non-overlapping sets of remote SSRCs it reports on. This is primarily to be done when the reporting source's number of reception report blocks is so large that it would be forced to round-robin around the sources. Thus, by splitting the reports among several reporting SSRCs, more consistent reporting can be achieved.

If RTP/AVPF feedback is in use, a reporting source MAY send immediate or early feedback at any point when any member of the reporting group could validly do so.

An endpoint SHOULD NOT create single-source reporting groups, unless it is anticipated that the group might have additional sources added to it in the future.

3.2. Determine the Report Group

A remote RTP entity, such as an endpoint or a middlebox needs to be able to determine the report group used by another RTP entity. To achieve this goal two RTCP extensions have been defined. For the SSRCs that are reporting on behalf of the reporting group, an SDES item RGRP has been defined for providing the report group with an identifier. For SSRCs that aren't reporting on any peer SSRC a new RTCP packet type is defined. This RTCP packet type "Reporting Sources" lists the SSRC that are reporting on this SSRC's behalf.

This divided approach has been selected for the following reasons:

1. To enable an explicit indication of who reports on this SSRC's behalf. Being explicit prevents the remote entity from detecting that is missing the reports if there issues with the reporting SSRC's RTCP packets.
2. To enable explicit identification of the SSRCs that are actively reporting as one entity.

3.3. RTCP Packet Reporting Group's Reporting Sources

This section defines a new RTCP packet type called "Reporting Group's Reporting Sources" (RGRS). It identifies the SSRC(s) that report on behalf of the SSRC that is the sender of the RGRS packet.

This packet consists of the fixed RTCP packet header which indicates the packet type, the number of reporting sources included and the SSRC which behalf the reporting SSRCs report on. This is followed by the list of reporting SSRCs.

```

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

```

The RTCP Packets field has the following definition.

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

padding (P): 1 bit If set, the padding bit indicates that the packet contains additional padding octets at the end that are not part of the control information but are included in the length field. See [RFC3550].

Source Count (SC): 5 bits Indicating the number of reporting SSRCs (1-31) that are included in this RTCP packet type.

Payload type (PT): 8 bits This is the RTCP packet type that identifies the packet as being an RTCP FB message. The RGRS RTCP packet has the value [TBA].

Length: 16 bits The length of this packet in 32-bit words minus one, including the header and any padding. This is in line with the definition of the length field used in RTCP sender and receiver reports [RFC3550].

SSRC of packet sender: 32 bits. The SSRC of the sender of this packet which indicates which SSRCs that reports on its behalf, instead of reporting itself.

SSRC for Reporting Source: A variable number (as indicated by Source Count) of 32-bit SSRC values. Each SSRC is an reporting SSRC belonging to the same Report Group.

Each RGRS packet MUST contain at least one reporting SSRC. In case the reporting SSRC field is insufficient to list all the SSRCs that are reporting in this report group, the SSRC SHALL round robin around the reporting sources.

Any RTP mixer or translator which forwards SR or RR packets from members of a reporting group MUST forward the corresponding RGRS RTCP packet as well.

3.4. RTCP Source Description (SDS) item for Reporting Groups

A new RTCP Source Description (SDS) item is defined for the purpose of identifying reporting groups.

The Source Description (SDS) item "RGRP" is sent by a reporting group's reporting SSRC. Syntactically, its format is the same as the RTCP SDS CNAME item [RFC6222], and MUST be chosen with the same global-uniqueness and privacy considerations as CNAME. This name MUST be stable across the lifetime of the reporting group, even if the SSRC of a reporting source changes.

Every source which belongs to a reporting group MUST either include an RGRP SDS item in an SDS packet (if it is a reporting source), or an RGRS packet (if it is not), in every compound RTCP packet in which it sends an RR or SR packet (i.e., in every RTCP packet it sends, unless Reduced-Size RTCP [RFC5506] is in use).

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

3.5. Middlebox Considerations

This section discusses middlebox considerations for Reporting groups.

To be expanded.

3.6. SDP signaling for Reporting Groups

TBD

3.7. Bandwidth Benefits of RTCP Reporting Groups

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

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

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

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

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

3.8. Consequences of RTCP Reporting Groups

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

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

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

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

4. Security Considerations

The security considerations of [RFC3550] and [I-D.ietf-avtcore-rtp-multi-stream] apply.

(tbd: any security considerations due to these extensions?)

5. IANA Considerations

(Note to the RFC-Editor: please replace "TBA" with the IANA-assigned value, and "XXXX" with the number of this document, prior to publication as an RFC.)

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

Value	Abbrev	Name	Reference
TBA	RGRP	Reporting Group	[RFCXXXX]

Figure 1: Item for the IANA Source Attribute Registry

The IANA is also requested to register one new RTCP packet type as follows:

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

Figure 2: Item for the IANA RTCP Control Packet Types (PT) Registry

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.
- [RFC6222] Begen, A., Perkins, C., and D. Wing, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMES)", RFC 6222, April 2011.

6.2. Informative References

- [I-D.ietf-avtcore-rtp-multi-stream]
Lennox, J., Westerlund, M., Wu, W., and C. Perkins, "RTP Considerations for Endpoints Sending Multiple Media Streams", draft-ietf-avtcore-rtp-multi-stream-00 (work in progress), April 2013.
- [RFC3611] Friedman, T., Caceres, R., and A. Clark, "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, November 2003.
- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.
- [RFC5506] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.

Authors' Addresses

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

Email: jonathan@vidyo.com

Magnus Westerlund
Ericsson
Farogatan 6
SE-164 80 Kista
Sweden

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

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

Email: sunseawq@huawei.com

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

Email: csp@csp Perkins.org