AVT Core Working Group
Internet-Draft
Intended status: Experimental
Expires: August 30, 2012

V. Singh
T. Karkkainen
J. Ott
S. Ahsan
Aalto University
L. Eggert
NetApp

February 27, 2012

Multipath RTP (MPRTP)
draft-singh-avtcore-mprtp-04

Abstract

The Real-time Transport Protocol (RTP) is used to deliver real-time content and, along with the RTP Control Protocol (RTCP), forms the control channel between the sender and receiver. However, RTP and RTCP assume a single delivery path between the sender and receiver and make decisions based on the measured characteristics of this single path. Increasingly, endpoints are becoming multi-homed, which means that they are connected via multiple Internet paths. Network utilization can be improved when endpoints use multiple parallel paths for communication. The resulting increase in reliability and throughput can also enhance the user experience. This document extends the Real-time Transport Protocol (RTP) so that a single session can take advantage of the availability of multiple paths between two endpoints.

Status of this Memo

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

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

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

This Internet-Draft will expire on August 30, 2012.

Table of Contents

1. Introduction

Multi-homed endpoints are becoming common in today's Internet, e.g., devices that support multiple wireless access technologies such as 3G and Wireless LAN. This means that there is often more than one network path available between two endpoints. Transport protocols, such as RTP, have not been designed to take advantage of the availability of multiple concurrent paths and therefore cannot benefit from the increased capacity and reliability that can be achieved by pooling their respective capacities.

Multipath RTP (MPRTP) is an OPTIONAL extension to RTP [1] that allows splitting a single RTP stream into multiple subflows that are transmitted over different paths. In effect, this pools the resource capacity of multiple paths. Multipath RTCP (MPRTCP) is an extension to RTCP, it is used along with MPRTP to report per-path sender and receiver characteristics.

Other IETF transport protocols that are capable of using multiple paths include SCTP [11], MPTCP [12] and SHIM6 [13]. However, these protocols are not suitable for realtime communications.

1.1. 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 [2].

1.2. Terminology

- Endpoint: host either initiating or terminating an RTP flow.
- Interface: logical or physical component that is capable of acquiring a unique IP address.
- Path: sequence of links between a sender and a receiver. Typically, defined by a set of source and destination addresses.
- Subflow: flow of RTP packets along a specific path, i.e., a subset of the packets belonging to an RTP stream. The combination of all RTP subflows forms the complete RTP stream. Typically, a subflow would map to a unique path, i.e., each combination of IP addresses and port pairs (5-tuple, including protocol) is a unique subflow.

1.3. Use-cases

The primary use-case for MPRTP is transporting high bit-rate streaming multimedia content between endpoints, where at least one is multi-homed. Such endpoints could be residential IPTV devices that connect to the Internet through two different Internet service providers (ISPs), or mobile devices that connect to the Internet through 3G and WLAN interfaces. By allowing RTP to use multiple paths for transmission, the following gains can be achieved:

- Higher quality: Pooling the resource capacity of multiple Internet paths allows higher bit-rate and higher quality codecs to be used. From the application perspective, the available bandwidth between the two endpoints increases.

- Load balancing: Transmitting an RTP stream over multiple paths reduces the bandwidth usage on a single path, which in turn reduces the impact of the media stream on other traffic on that path.

- Fault tolerance: When multiple paths are used in conjunction with redundancy mechanisms (FEC, re-transmissions, etc.), outages on one path have less impact on the overall perceived quality of the stream.

A secondary use-case for MPRTP is transporting Voice over IP (VoIP) calls to a device with multiple interfaces. Again, such an endpoint could be a mobile device with multiple wireless interfaces. In this case, little is to be gained from resource pooling, i.e., higher capacity or load balancing, because a single path should be easily capable of handling the required load. However, using multiple concurrent subflows can improve fault tolerance, because traffic can shift between the subflows when path outages occur. This results in very fast transport-layer handovers that do not require support from signaling.

2. Goals

This section outlines the basic goals that multipath RTP aims to meet. These are broadly classified as Functional goals and Compatibility goals.

2.1. Functional goals

Allow unicast RTP session to be split into multiple subflows in order to be carried over multiple paths. This may prove beneficial in case of video streaming.

- Increased Throughput: Cumulative capacity of the two paths may meet the requirements of the multimedia session. Therefore, MPRTP MUST support concurrent use of the multiple paths.

- Improved Reliability: MPRTP SHOULD be able to send redundant packets or re-transmit packets along any available path to increase reliability.

The protocol SHOULD be able to open new subflows for an existing session when new paths appear and MUST be able to close subflows when paths disappear.

2.2. Compatibility goals

MPRTP MUST be backwards compatible; an MPRTP stream needs to fall back to be compatible with legacy RTP stacks if MPRTP support is not successfully negotiated.

- Application Compatibility: MPRTP service model MUST be backwards compatible with existing RTP applications, i.e., an MPRTP stack MUST be able to work with legacy RTP applications and not require changes to them. Therefore, the basic RTP APIs MUST remain unchanged, but an MPRTP stack MAY provide extended APIs so that the application can configure any additional features provided by the MPRTP stack.

- Network Compatibility: individual RTP subflows MUST themselves be well-formed RTP flows, so that they are able to traverse NATs and firewalls. This MUST be the case even when interfaces appear after session initiation. Interactive Connectivity Establishment (ICE) [3] MAY be used for discovering new interfaces or performing connectivity checks.

3. RTP Topologies

RFC 5117 [14] describes a number of scenarios using mixers and translators in single-party (point-to-point), and multi-party (point-to-multipoint) scenarios. RFC 3550 [1] (Section 2.3 and 7.x) discuss in detail the impact of mixers and translators on RTP and RTCP packets. MPRTP assumes that if a mixer or translator exists in the network, then either all of the multiple paths or none of the multiple paths go via this component.

4. MPRTP Architecture

In a typical scenario, an RTP session uses a single path. In an

MPRTP scenario, an RTP session uses multiple subflows that each use a different path. Figure 1 shows the difference.

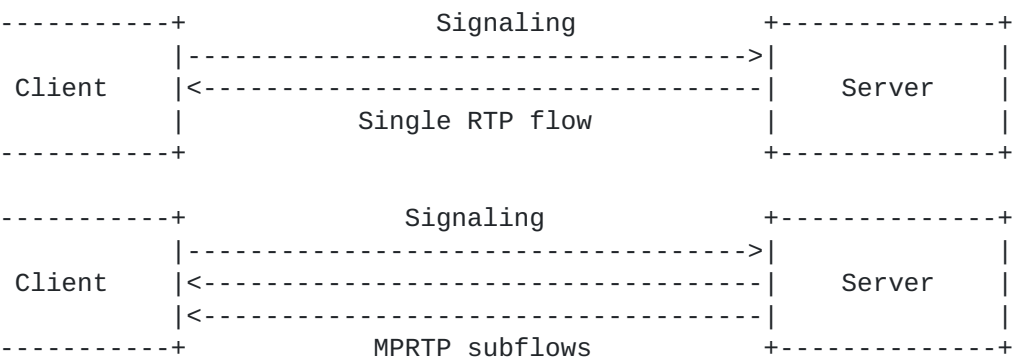


Figure 1: Comparison between traditional RTP streaming and MPRTP

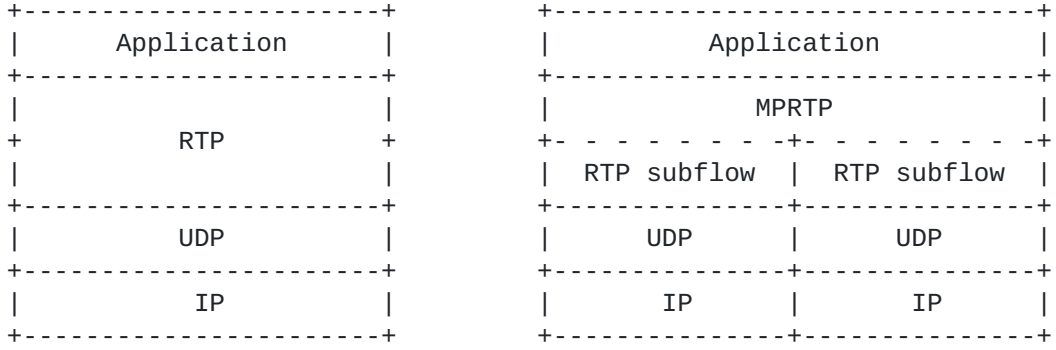


Figure 2: MPRTP Architecture

Figure 2 illustrates the differences between the standard RTP stack and the MPRTP stack. MPRTP receives a normal RTP session from the application and splits it into multiple RTP subflows. Each subflow is then sent along a different path to the receiver. To the network, each subflow appears as an independent, well-formed RTP flow. At the receiver, the subflows are combined to recreate the original RTP session. The MPRTP layer performs the following functions:

- Path Management: The layer is aware of alternate paths to the other host, which may, for example, be the peer's multiple interfaces. This enables the endpoint to transmit differently marked packets along separate paths. MPRTP also selects interfaces to send and receive data. Furthermore, it manages the port and IP address pair bindings for each subflow.

- Packet Scheduling: the layer splits a single RTP flow into multiple subflows and sends them across multiple interfaces (paths). The splitting MAY BE done using different path characteristics.

- Subflow recombination: the layer creates the original stream by recombining the independent subflows. Therefore, the multipath subflows appear as a single RTP stream to applications.

5. Example Media Flow Diagrams

There may be many complex technical scenarios for MPRTP, however, this memo only considers the following two scenarios: 1) a unidirectional media flow that represents the streaming use-case, and 2) a bidirectional media flow that represents a conversational use-case.

5.1. Streaming use-case

In the unidirectional scenario, the receiver (client) initiates a multimedia session with the sender (server). The receiver or the sender may have multiple interfaces and both endpoints are MPRTP-capable. Figure 3 shows this scenario. In this case, host A has multiple interfaces. Host B performs connectivity checks on host A's multiple interfaces. If the interfaces are reachable, then host B streams multimedia data along multiple paths to host A. Moreover, host B also sends RTCP Sender Reports (SR) for each subflow and host A responds with a normal RTCP Receiver Report (RR) for the overall session as well as the receiver statistics for each subflow. Host B distributes the packets across the subflows based on the individually measured path characteristics.

Alternatively, to reduce media startup time, host B may start streaming multimedia data to host A's initiating interface and then perform connectivity checks for the other interfaces. This method of updating a single path session to a multipath session is called "multipath session upgrade".

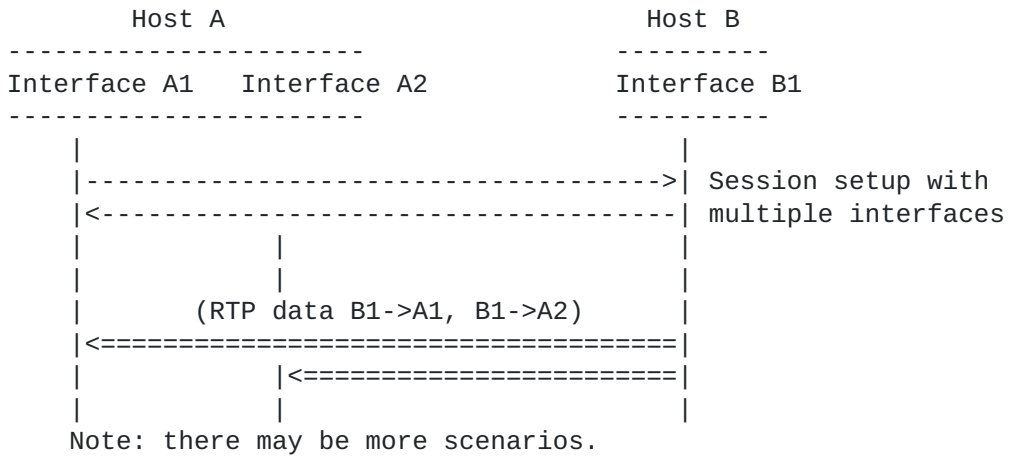


Figure 3: Unidirectional media flow

5.2. Conversational use-case

In the bidirectional scenario, multimedia data flows in both directions. The two hosts exchange their lists of interfaces with each other and perform connectivity checks. Communication begins after each host finds suitable address, port pairs. Interfaces that receive data send back RTCP receiver statistics for that path (based on the 5-tuple). The hosts balance their multimedia stream across multiple paths based on the per path reception statistics and its own volume of traffic. Figure 4 describes an example of a bidirectional flow.

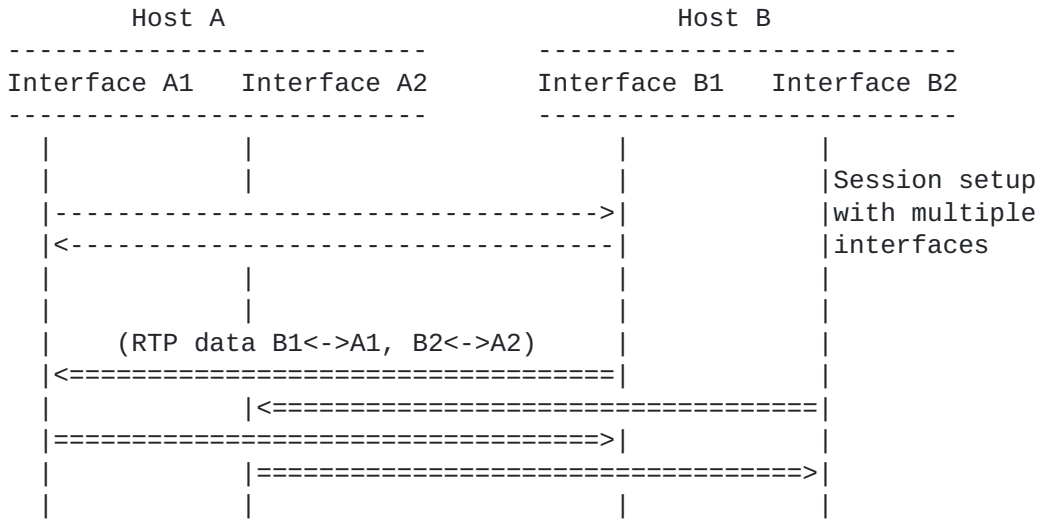


Figure 4: Bidirectional flow

6. MPRTP Functional Blocks

This section describes some of the functional blocks needed for MPRTP. We then investigate each block and consider available mechanisms in the next section.

1. Session Setup: Interfaces may appear or disappear at anytime during the session. To preserve backward compatibility with legacy applications, a multipath session MUST look like a bundle of individual RTP sessions. A multipath session may be upgraded from a typical single path session, as and when new interfaces appear or disappear. However, it is also possible to setup a multipath session from the beginning, if the interfaces are available at the start of the multimedia session.

2. Expanding RTP: For a multipath session, each subflow MUST look like an independent RTP flow, so that individual RTCP messages can be generated per subflow. Furthermore, MPRTP splits the single multimedia stream into multiple subflows based on path characteristics (e.g. RTT, loss-rate, receiver rate, bandwidth-delay product etc.) and dynamically adjusts the load on each link.

3. Adding Interfaces: Interfaces on the host need to be regularly discovered and advertised. This can be done at session setup and/or during the session. Discovering interfaces is outside the scope of this document.

4. Expanding RTCP: MPRTP MUST provide per path RTCP reports for monitoring the quality of the path, for load balancing, or for congestion control, etc. To maintain backward compatibility with legacy applications, the receiver MUST also send aggregate RTCP reports along with the per-path reports.

5. Maintenance and Failure Handling: In a multi-homed endpoint interfaces may appear and disappear. If this occurs at the sender, it has to re-adjust the load on the available links. On the other hand, if this occurs at the receiver, then the multimedia data transmitted by the sender to those interfaces is lost. This data may be re-transmitted along a different path i.e., to a different interface on the receiver. Furthermore, the endpoint has to either explicitly signal the disappearance of an interface, or the other endpoint has to detect it (by lack of media packets, RTCP feedback, or keep-alive packets).

6. Teardown: The MPRTP layer releases the occupied ports on the interfaces.

7. Available Mechanisms within the Functional Blocks

This section discusses some of the possible alternatives for each functional block mentioned in the previous section.

7.1. Session Setup

MPRTP session can be set up in many possible ways e.g., during handshake, or upgraded mid-session. The capability exchange may be done using out-of-band signaling (e.g., Session Description Protocol (SDP) [15] in Session Initiation Protocol (SIP) [16], Real-Time Streaming Protocol (RTSP) [17]) or in-band signaling (e.g., RTP/RTCP header extension, Session Traversal Utilities for NAT (STUN) messages).

7.1.1. Connectivity Checks

The endpoint SHOULD be capable of performing connectivity checks (e.g., using ICE [3]). If the IP addresses of the endpoints are reachable (e.g., globally addressable, same network etc) then connectivity checks are not needed.

7.1.2. Adding New or Updating Interfaces

Interfaces can appear and disappear during a session, the endpoint MUST be capable of advertising the changes in its set of usable interfaces. Additionally, the application or OS may define a policy on when and/or what interfaces are usable. However, MPRTP requires a method to advertise or notify the other endpoint about the updated set of usable interfaces.

7.1.3. In-band vs. Out-of-band Signaling

MTRTP nodes will generally use a signaling protocol to establish their MPRTP session. With the existence of such a signaling relationship, two alternatives become available to exchange information about the available interfaces on each side for extending RTP sessions to MPRTP and for modifying MPRTP sessions: in-band and out-of-band signaling.

In-band signaling refers to using mechanisms of RTP/RTCP itself to communicate interface addresses, e.g., a dedicated RTCP extensions along the lines of the one defined to communicate information about the feedback target for RTP over SSM [4]. In-band signaling does not

rely on the availability of a separate signaling connection and the information flows along the same path as the media streams, thus minimizing dependencies. Moreover, if the media channel is secured (e.g., by means of SRTP), the signaling is implicitly protected as well if SRTCP encryption and authentication are chosen. In-band signaling is also expected to take a direct path to the peer, avoiding any signaling overlay-induced indirections and associated processing overheads in signaling elements -- avoiding such may be especially worthwhile if frequent updates may occur as in the case of mobile users. Finally, RTCP is usually sent sufficiently frequently (in point-to-point settings) to provide enough opportunities for transmission and (in case of loss) retransmission of the corresponding RTCP packets.

Examples for in-band signaling include RTCP extensions as noted above or suitable extensions to STUN.

Out-of-band signaling refers to using a separate signaling connection (via SIP, RTSP, or HTTP) to exchange interface information, e.g., expressed in SDP. Clear benefits are that signaling occurs at setup time anyway and that experience and SDP syntax (and procedures) are available that can be re-used or easily adapted to provide the necessary capabilities. In contrast to RTCP, SDP offers a reliable communication channel so that no separate retransmissions logic is necessary. In SDP, especially when combined with ICE, connectivity check mechanisms including sophisticated rules are readily available. While SDP is not inherently protected, suitable security may need to be applied anyway to the basic session setup.

Examples for out-of-band signaling are dedicated extensions to SDP; those may be combined with ICE.

Both types of mechanisms have their pros and cons for middleboxes. With in-band signaling, control packets take the same path as the media packets and they can be directly inspected by middleboxes so that the elements operating on the signaling channel do not need to understand new SDP. With out-of-band signaling, only the middleboxes processing the signaling need to be modified and those on the data forwarding path can remain untouched.

Overall, it may appear sensible to provide a combination of both mechanisms: out-of-band signaling for session setup and initial interface negotiation and in-band signaling to deal with frequent changes in interface state (and for the potential case, albeit rather theoretical case of MPRTP communication without a signaling channel).

In its present version, this document explores both options to provide a broad understanding of how the corresponding mechanisms

would look like.

[[Comment.1: Some have suggested STUN may be suitable for doing in-band interface advertisement. This is still under consideration, but depends on implementation challenges as many legacy systems don't implement STUN and many RTP systems ignore STUN messages. --Editor]]

7.2. Expanding RTP

RTCP [1] is generated per media session. However, with MPRTP, the media sender spreads the RTP load across several interfaces. The media sender SHOULD make the path selection, load balancing and fault tolerance decisions based on the characteristics of each path. Therefore, apart from normal RTP sequence numbers defined in [1], the MPRTP sender MUST add subflow-specific sequence numbers to help calculate fractional losses, jitter, RTT, playout time, etc., for each path, and a subflow identifier to associate the characteristics with a path. The RTP header extension for MPRTP is shown in Section 9.1.

7.3. Expanding RTCP

To provide accurate per path information an MPRTP endpoint MUST send (SR/RR) report for each unique subflow along with the overall session RTCP report. Therefore, the additional subflow reporting affects the RTCP bandwidth and the RTCP reporting interval. RTCP report scheduling for each subflow may cause a problem for RTCP recombination and reconstruction in cases when 1) RTCP for a subflow is lost, and 2) RTCP for a subflow arrives later than the other subflows. (There may be other cases as well.)

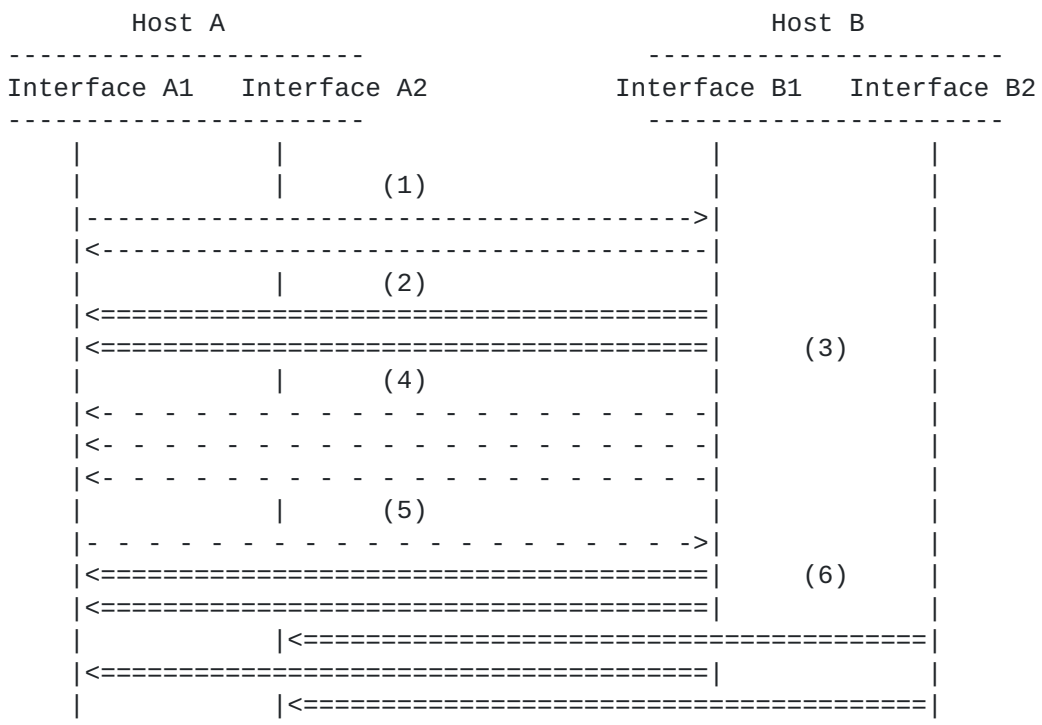
The sender distributes the media across different paths using the per path RTCP reports. However, this document doesn't cover algorithms for congestion control or load balancing.

7.4. Failure Handling and Teardown

An MPRTP endpoint MUST keep alive subflows that have been negotiated but no media is sent on them. Moreover, using the information in the subflow reports, a sender can monitor an active subflow for failure (errors, unreachability, congestion) and decide not to use (make the active subflow passive), or teardown the subflow.

If an interface disappears, the endpoint MUST send an updated interface advertisement without the interface and release the the associated subflows.

8. MPRTP Protocol



Key:
| Interface
---> Signaling Protocol
<=== RTP Packets
- -> RTCP Packet

Figure 5: MPRTP New Interface

8.1. Overview

The bullet points explain the different steps shown in Figure 5 for upgrading a single path multimedia session to multipath session.

(1) The first two interactions between the hosts represents the establishment of a normal RTP session. This may performed e.g. using SIP or RTSP.

(2) When the RTP session has been established, host B streams media using its interface B1 to host A at interface A1.

(3) Host B supports sending media using MPRTP and becomes aware of an additional interface B2.

(4) Host B advertises the multiple interface addresses.

(5) Host A supports receiving media using MPRTP and becomes aware of an additional interface A2.

Side note, even if an MPRTP-capable host has only one interface, it MUST respond to the advertisement with its single interface.

(6) Each host receives information about the additional interfaces and the appropriate endpoints starts to stream the multimedia content using the additional paths.

If needed, each endpoint will need to independently perform connectivity checks (not shown in figure) and ascertain reachability before using the paths.

8.1.1. Gather or Discovering Candidates

The endpoint periodically polls the operating system or is notified when an additional interface appears. If the endpoint wants to use the additional interface for MPRTP it MUST advertise it to the other peers. The endpoint may also use ICE [3] to gather additional candidates.

8.1.2. NAT Traversal

After gathering their interface candidates, the endpoints decide internally if they wish to perform connectivity checks.

[[Comment.2: Legacy applications do not require ICE for session establishment, therefore, MPRTP should not require it as well. --Editor]]

If the endpoint chooses to perform connectivity checks then it MUST first advertise the gathered candidates as ICE candidates in SDP during session setup and let ICE perform the connectivity checks. As soon as a sufficient number of connectivity checks succeed, the endpoint can use the successful candidates to advertise its MPRTP interface candidates.

8.1.3. Choosing between In-band (in RTCP) and Out-of-band (in SDP) Interface Advertisement

If there is no media flowing at the moment and the application wants to use the interfaces from the start of the session, it should advertise them in SDP (See Section 11.3). Alternatively, the endpoint can setup the session as a single path media session and upgrade the session to multipath by advertising the session in-band

(See Section 8.1.4). Moreover, if an interface appears and disappears, the endpoint SHOULD prefer to advertise it in-band because the endpoint would not have to wait for a response from the other endpoint before starting to use the interface. However, if there is a conflict between an in-band and out-of-band advertisement, i.e., the endpoint receives an in-band advertisement while it has a pending out-of-band advertisement, or vice versa then the session is setup using out-of-band mechanisms.

8.1.4. In-band Interface Advertisement

To advertise the multiple interfaces in RTCP, an MPRTP-capable endpoint MUST add the MPRTP Interface Advertisement defined in Figure 13 with the RTCP Sender Report (SR). Each unique address is encapsulated in an Interface Advertisement block and contains the IP address, RTP and RTCP port addresses. The Interface Advertisement blocks are ordered based on a decreasing priority level. On receiving the MPRTP Interface Advertisement, an MPRTP-capable receiver MUST respond with the set of interfaces (subset or all available) it wants to use.

If the sender and receiver have only one interface, then the endpoints MUST indicate the negotiated single path IP, RTP port and RTCP port addresses.

8.1.5. Subflow ID Assignment

After interface advertisements have been exchanged, the endpoint MUST associate a Subflow ID to each unique subflow. Each combination of sender and receiver IP addresses and port pairs (5-tuple) is a unique subflow. If the connectivity checks have been performed then the endpoint MUST only use the subflows for which the connectivity checks have succeeded.

8.1.6. Active and Passive Subflows

To send and receive data an endpoint MAY use any number of subflows from the set of available subflows. The subflows that carry media data are called active subflows, while those subflows that don't send any media packets (fallback paths) are called passive subflows.

An endpoint MUST multiplex the subflow specific RTP and RTCP packets on the same port to keep the NAT bindings alive. This is inline with the recommendation made in RFC6263[18]. Moreover, if an endpoint uses ICE, multiplexing RTP and RTCP reduces the number of components from 2 to 1 per media stream. If no MPRTP or MPRTCP packets are received on a particular subflow at a receiver, the receiver SHOULD remove the subflow from active and passive lists and not send any

MPRTCP reports for that subflow. It may keep the bindings in a dead-pool, so that if the 5-tuple or subflow reappears, it can quickly reallocate it based on past history.

8.2. RTP Transmission

If both endpoints are MPRTP-capable and if they want to use their multiple interfaces for sending the media stream then they MUST use the MPRTP header extensions. They MAY use normal RTP with legacy endpoints (see Appendix A).

An MPRTP endpoint sends RTP packets with an MPRTP extension that maps the media packet to a specific subflow (see Figure 8). The MPRTP layer SHOULD associate an RTP packet with a subflow based on a scheduling strategy. The scheduling strategy may either choose to augment the paths to create higher throughput or use the alternate paths for enhancing resilience or error-repair. Due to the changes in path characteristics, the endpoint should be able change its scheduling strategy during an ongoing session. The MPRTP sender MUST also populate the subflow specific fields described in the MPRTP extension header (see Section 9.1.1).

8.3. Playout Considerations at the Receiver

A media receiver, irrespective of MPRTP support or not, should be able to playback the media stream because the received RTP packets are compliant to [1], i.e., a non-MPRTP receiver will ignore the MPRTP header and still be able to playback the RTP packets. However, the variation of jitter and loss per path may affect proper playout. The receiver can compensate for the jitter by modifying the playout delay (i.e., by calculating skew across all paths) of the received RTP packets.

8.4. Subflow-specific RTCP Statistics and RTCP Aggregation

Aggregate RTCP provides the overall media statistics and follows the normal RTCP defined in RFC3550 [1]. However, subflow specific RTCP provides the per path media statistics because the aggregate RTCP report may not provide sufficient per path information to an MPRTP scheduler. Specifically, the scheduler should be aware of each path's RTT and loss-rate, which an aggregate RTCP cannot provide. The sender/receiver MUST use non-compound RTCP reports defined in RFC5506 [5] to transmit the aggregate and subflow-specific RTCP reports. Also, each subflow and the aggregate RTCP report MUST follow the timing rules defined in [6].

The RTCP reporting interval is locally implemented and the scheduling of the RTCP reports may depend on the the behavior of each path. For

instance, the RTCP interval may be different for a passive path than an active path to keep port bindings alive. Additionally, an endpoint may decide to share the RTCP reporting bit rate equally across all its paths or schedule based on the receiver rate on each path.

8.5. RTCP Transmission

The sender sends an RTCP SR on each active path. For each SR the receiver gets, it echoes one back to the same IP address-port pair that sent the SR. The receiver tries to choose the symmetric path and if the routing is symmetric then the per-path RTT calculations will work out correctly. However, even if the paths are not symmetric, the sender would at maximum, under-estimate the RTT of the path by a factor of half of the actual path RTT.

9. Packet Formats

In this section we define the protocol structures described in the previous sections.

9.1. RTP Header Extension for MPRTP

The MPRTP header extension is used to distribute a single RTP stream over multiple subflows.

The header conforms to the one-byte RTP header extension defined in [7]. The header extension contains a 16-bit length field that counts the number of 32-bit words in the extension, excluding the four-octet extension header (therefore zero is a valid length, see Section 5.3.1 of [1] for details).

The RTP header for each subflow is defined below:

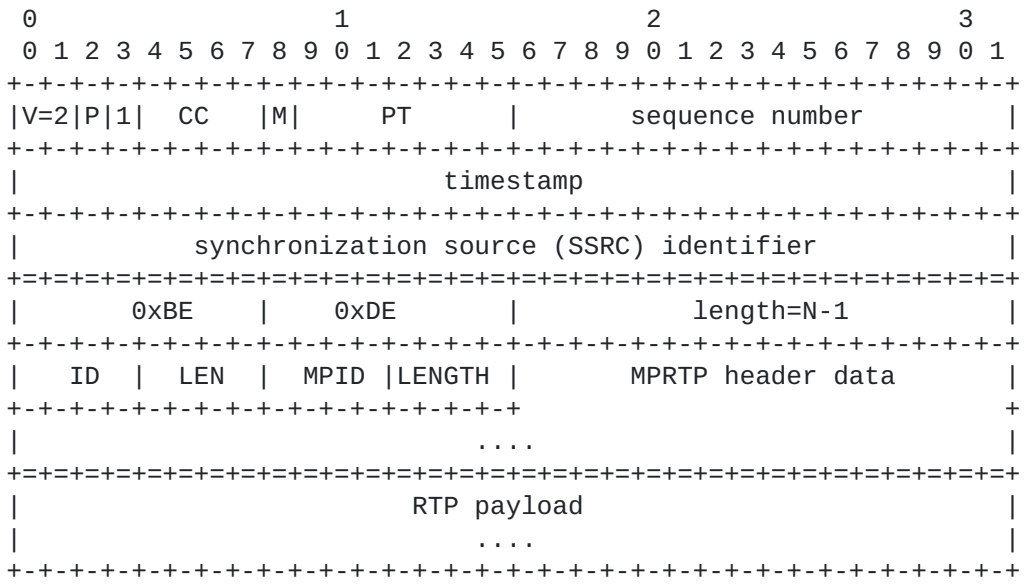


Figure 6: Generic MPRTP header extension

MPID:

The MPID field corresponds to the type of MPRTP packet. Namely:

MPID ID Value	Use
0x0	Subflow RTP Header. For this case the Length is set to 6

Figure 7: RTP header extension values for MPRTP (H-Ext ID)

length

The 4-bit length field is the length of extension data in bytes not including the H-Ext ID and length fields. The value zero indicates there is no data following.

MPRTP header data

Carries the MPID specific data as described in the following sub-sections.

9.1.1. MPRTP RTP Extension for a Subflow

The RTP header for each subflow is defined below:

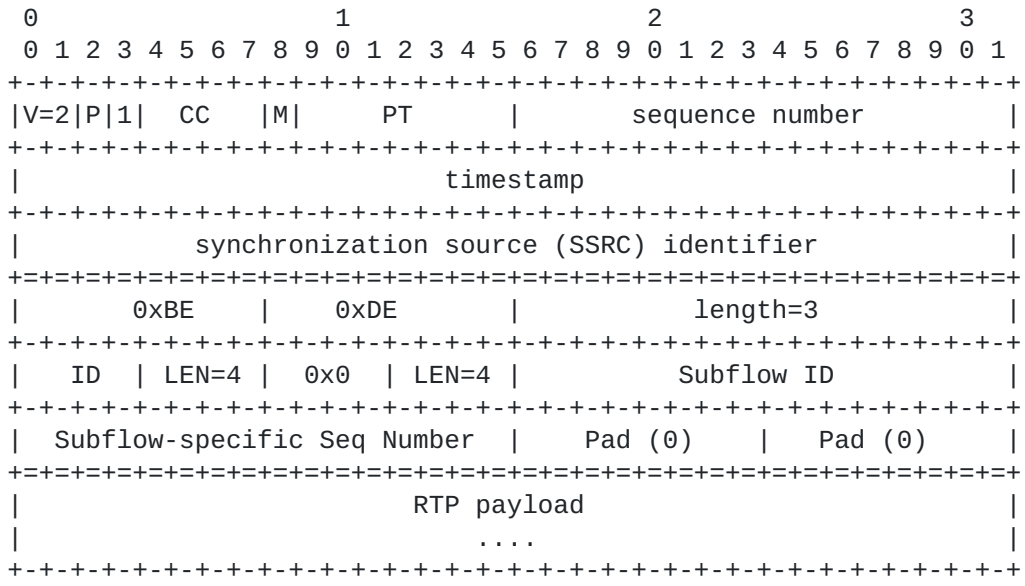


Figure 8: MPRTP header for subflow

MP ID = 0x0

Indicates that the MPRTP header extension carries subflow specific information.

length = 4

Subflow ID: Identifier of the subflow. Every RTP packet belonging to the same subflow carries the same unique subflow identifier.

Flow-Specific Sequence Number (FSSN): Sequence of the packet in the subflow. Each subflow has its own strictly monotonically increasing sequence number space.

9.2. RTCP Extension for MPRTP (MPRTCP)

The MPRTP RTCP header extension is used to 1) provide RTCP feedback per subflow to determine the characteristics of each path, and 2) advertise each interface.

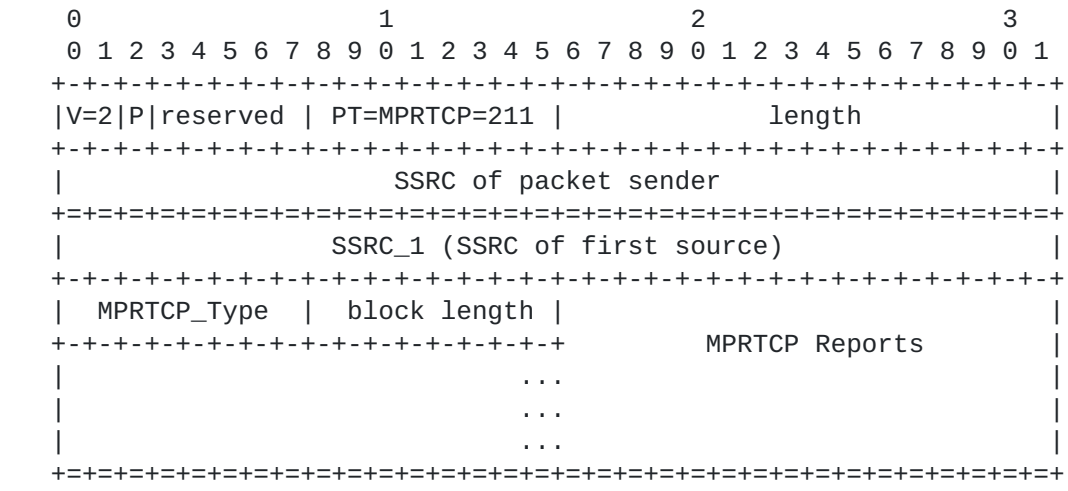


Figure 9: Generic RTCP Extension for MPRTP (MPRTCP) [appended to normal SR/RR]

MPRTCP: 8 bits

Contains the constant 211 to identify this as an Multipath RTCP packet.

length: 16 bits

As described for the RTCP packet (see Section 6.4.1 of the RTP specification [1]), the length of this is in 32-bit words minus one, including the header and any padding.

MPRTCP_Type: 8-bits

The MPRTCP_Type field corresponds to the type of MPRTP RTCP packet. Namely:

MPRTCP_Type Value	Use
0	Subflow Specific Report
1	Interface Advertisement (IPv4 Address)
2	Interface Advertisement (IPv4 Address)
3	Interface Advertisement (DNS Address)

Figure 10: RTP header extension values for MPRTP (MPRTCP_Type)

block length: 8-bits

The 8-bit length field is the length of the encapsulated MPRTCP reports in 32-bit word length not including the MPRTCP_Type and length fields. The value zero indicates there is no data following.

MPRTCP Reports: variable size

Defined later in 9.2.1 and 9.3.

9.2.1. MPRTCP Extension for Subflow Reporting

When sending a report for a specific subflow the sender or receiver MUST add only the reports associated with that 5-tuple. Each subflow is reported independently using the following MPRTCP Feedback header.

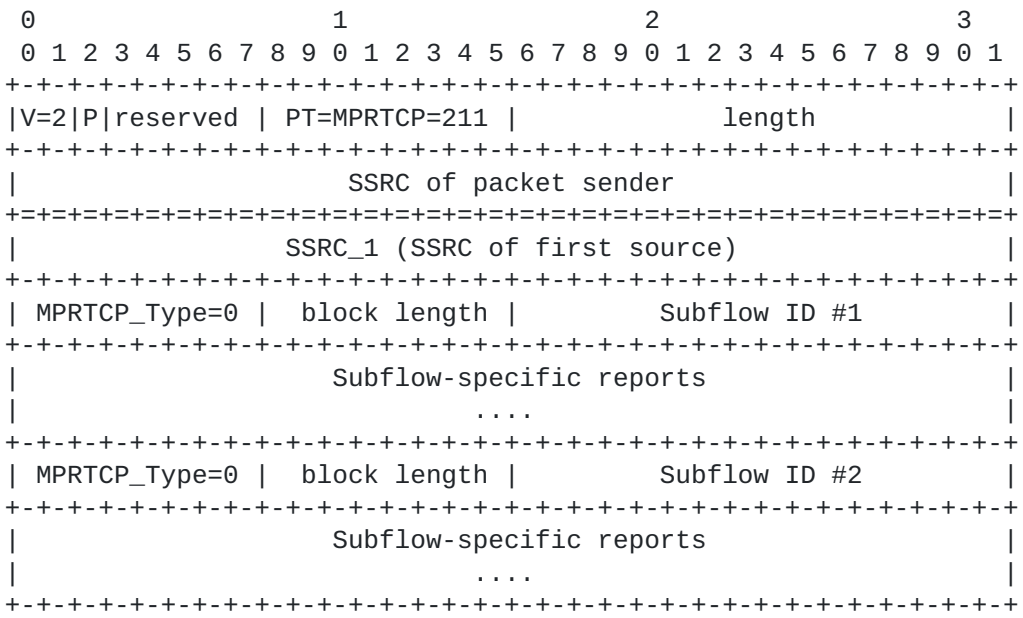


Figure 11: MPRTCP Subflow Reporting Header

MPRTCP_Type: 0

The value indicates that the encapsulated block is a subflow report.

block length: 8-bits

The 8-bit length field is the length of the encapsulated subflow-specific reports in 32-bit word length not including the

MPRTCP_Type and length fields.

Subflow ID: 16 bits

Subflow identifier is the value associated with the subflow the endpoint is reporting about. If it is a sender it MUST use the Subflow ID associated with the 5-tuple. If it is a receiver it MUST use the Subflow ID received in the Subflow-specific Sender Report.

Subflow-specific reports: variable

Subflow-specific report contains all the reports associated with the Subflow ID. For a sender, it MUST include the Subflow-specific Sender Report (SSR). For a receiver, it MUST include Subflow-specific Receiver Report (SRR). Additionally, if the receiver supports subflow-specific extension reports then it MUST append them to the SRR.

9.2.1.1. MPRTCP for Subflow-specific SR, RR and XR

[[Comment.3: inside the context of subflow specific reports can we reuse the payload type code for Sender Report (PT=200), Receiver Report (PT=201), Extension Report (PT=207).Transport and Payload specific RTCP messages are session specific and SHOULD be used as before. --Editor]]

Example:

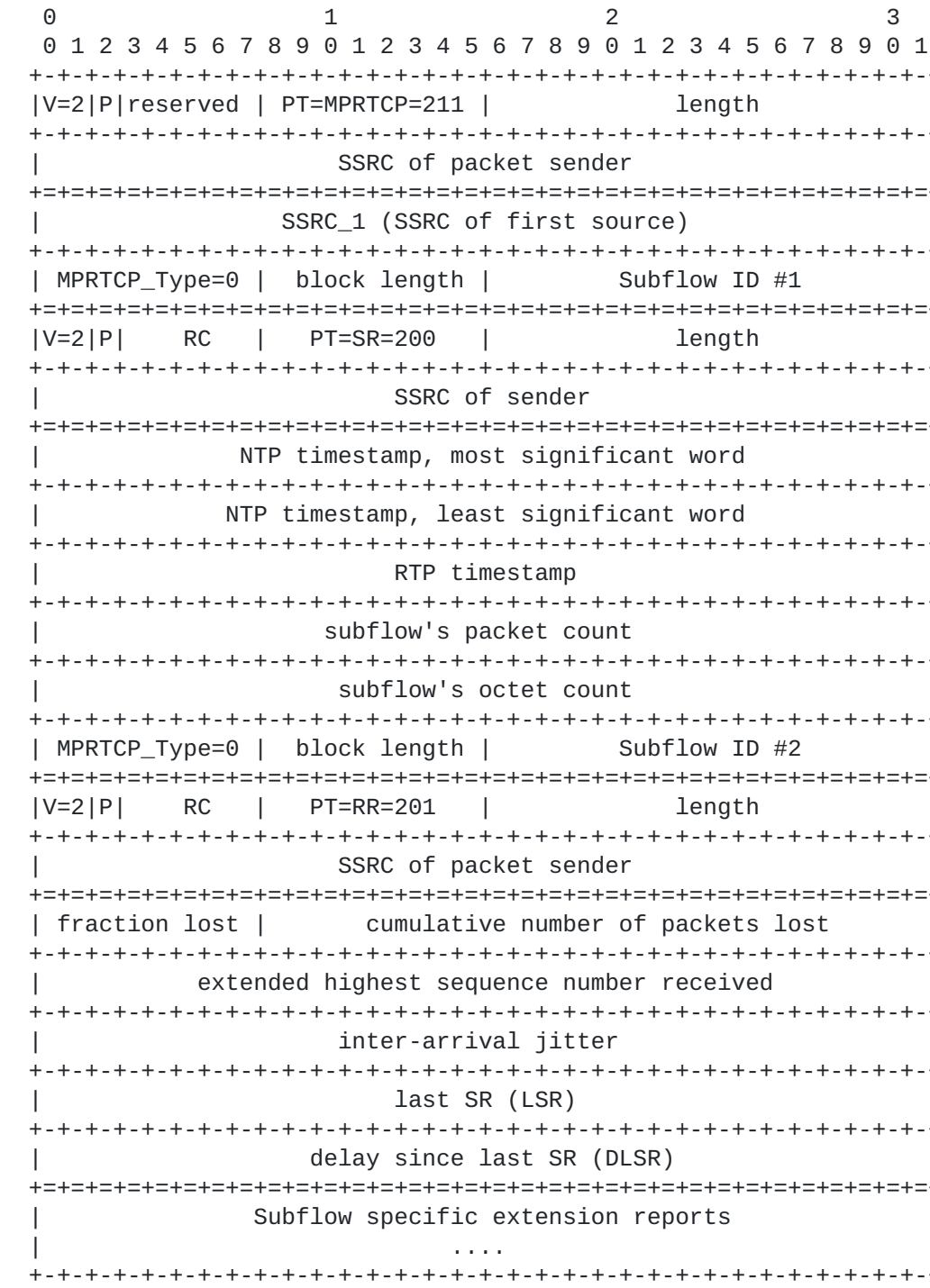


Figure 12: Example of reusing RTCP SR and RR inside an MPRTCP header (Bi-directional use-case, in case of uni-directional flow the subflow will only send an SR or RR).

9.3. MPRTCP Extension for Interface advertisement

This sub-section defines the RTCP header extension for in-band interface advertisement by the receiver. The interface advertisement block describes a method to represent IPv4, IPv6 and generic DNS-type addresses in a block format. It is based on the sub-reporting block in [4]. The interface advertisement SHOULD immediately follow the Receiver Report. If the Receiver Report is not present, then it MUST be appended to the Sender Report.

The endpoint MUST advertise the interfaces it wants to use whenever an interface appears or disappears and also when it receives an Interface Advertisement.

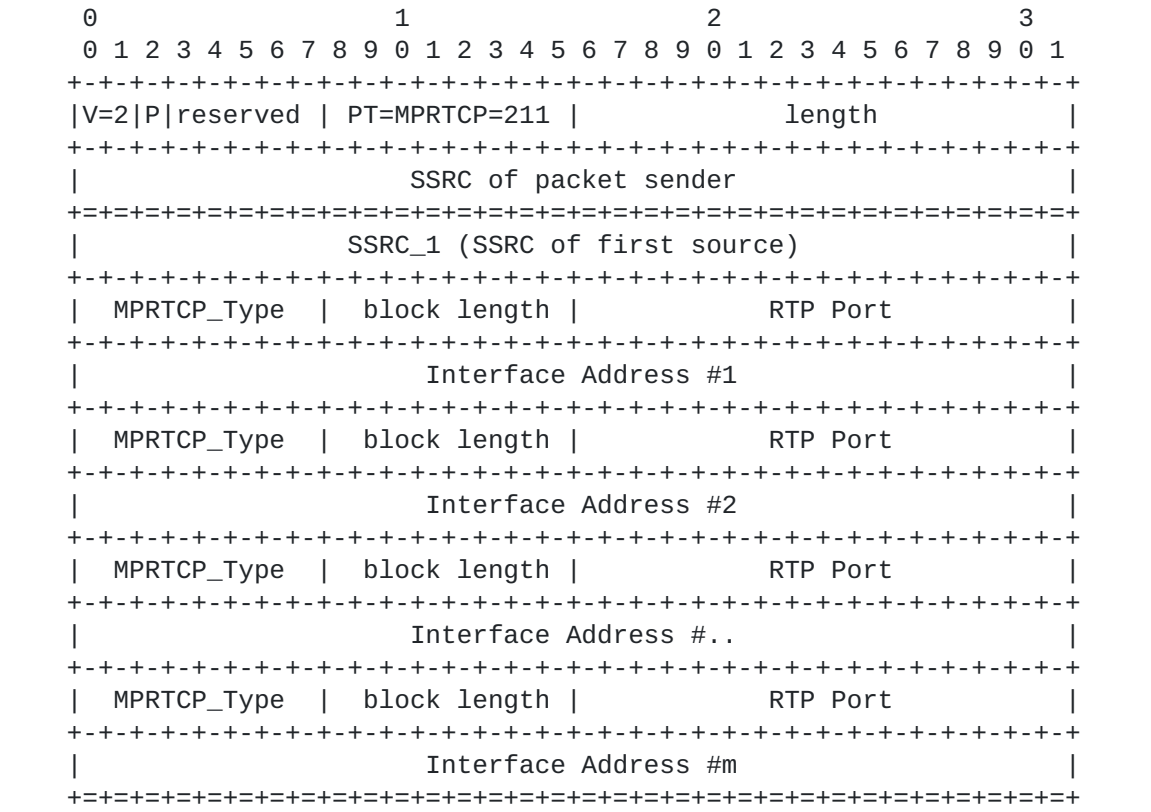


Figure 13: MPRTP Interface Advertisement. (appended to SR/RR)

MPRTCP_Type: 8 bits

The MPRTCP_Type corresponds to the type of interface address. Namely:

1: IPv4 address

2: IPv6 address

3: DNS name

block length: 8 bits

The length of the Interface Advertisement block in bytes.

For an IPv4 address, this should be 9 (i.e., 5 octets for the header and 4 octets for IPv4 address).

For an IPv6 address, this should be 21.

For a DNS name, the length field indicates the number of octets making up the string plus the 5 byte header.

RTP Port: 2 octets

The port number to which the sender sends RTP data. A port number of 0 is invalid and MUST NOT be used.

Interface Address: 4 octets (IPv4), 16 octets (IPv6), or n octets (DNS name)

The address to which receivers send feedback reports. For IPv4 and IPv6, fixed-length address fields are used. A DNS name is an arbitrary-length string. The string MAY contain Internationalizing Domain Names in Applications (IDNA) domain names and MUST be UTF-8 [8] encoded.

10. RTCP Timing reconsiderations for MPRTCP

MPRTP endpoints MUST conform to the timing rule imposed in [6], i.e., the total RTCP rate between the participants MUST NOT exceed 5% of the media rate. For each endpoint, a subflow MUST send the aggregate and subflow-specific report. The endpoint SHOULD schedule the RTCP reports for the active subflows based on the share of the transmitted or received bit rate to the average media bit rate, this method ensures fair sharing of the RTCP bandwidth. Alternatively, the endpoint MAY schedule the reports in round-robin.

11. SDP Considerations

11.1. Signaling MPRTP Header Extension in SDP

To indicate the use of the MPRTP header extensions (see Section 9) in SDP, the sender MUST use the following URI in extmap: urn:ietf:params:rtp-hdrext:mprtp. This is a media level parameter. Legacy RTP (non-MPRTP) clients will ignore this header extension, but can continue to parse and decode the packet (see Appendix A).

Example:

```
v=0
o=alice 2890844526 2890844527 IN IP4 192.0.2.1
s=
c=IN IP4 192.0.2.1
t=0 0
m=video 49170 RTP/AVP 98
a=rtpmap:98 H264/90000
a=fmtp:98 profile-level-id=42A01E;
a=extmap:1 urn:ietf:params:rtp-hdrext:mprtp
```

11.2. Signaling MPRTP capability in SDP

A participant of a media session MUST use SDP to indicate that it supports MPRTP. Not providing this information will make the other endpoint ignore the RTCP extensions.

```
    mprtp-attrib = "a=" "mprtp" [
            SP mprtp-optional-parameter]
            CRLF   ; flag to enable MPRTP
```

The endpoint MUST use 'a=mprtp', if it is able to send and receive MPRTP packets. Generally, senders and receivers MUST indicate this capability if they support MPRTP and would like to use it in the specific media session being signaled. To exchange the additional interfaces, the endpoint SHOULD use the in-band signaling (Section 9.3). Alternatively, advertise in SDP (Section 11.3).

MPRTP endpoint multiplexes RTP and RTCP on a single port, sender MUST indicate support by adding "a=rtcp-mux" in SDP. If an endpoint receives an SDP without "a=rtcp-mux" but contains "a=mprtp", then the endpoint MUST infer support for multiplexing.

[[Comment.4: If a=mprtp is indicated, does the endpoint need to indicate a=rtcp-mux? because MPRTP mandates RTP RTCP multiplexing. --Editor]]

11.3. MPRTP Interface Advertisement in SDP (out-of-band signaling)

If the endpoint is aware of its multiple interfaces and wants to use them for MPRTP then it MAY use SDP to advertise these interfaces. Alternatively, it MAY use in-band signaling to advertise its interfaces, as defined in Section 9.3. The receiving endpoint MUST use the same mechanism to respond to an interface advertisement. In particular, if an endpoint receives an SDP offer, then it MUST respond to the offer in SDP.

11.3.1. "interface" attribute

The interface attribute is an optional media-level attribute and is used to advertise an endpoint's interface address.

The syntax of the interface attribute is defined using the following Augmented BNF, as defined in [9]. The definitions of unicast-address, port, token, SP, and CRLF are according to RFC4566 [19].

```
mprtp-optional-parameter = mprtp-interface
                ; other optional parameters may be added later

mprtp-interface = "interface" ":" counter SP unicast-address
                    ":" rtp_port
                     *(SP interface-description-extension)

counter = 1*DIGIT
rtp_port = port    ;port from RFC4566
```

<mprtp-interface>: specifies one unicast IP address, the RTP and RTCP port number of the endpoint. The unicast address with lowest counter value MUST match the connection address ('c=' line). Similarly, the RTP and RTCP ports MUST match the RTP and RTCP ports in the associated 'm=' line. The counter should start at 1 and increment with each additional interface. Multiple interface lines MUST be ordered in a decreasing priority level as is the case with the Interface Advertisement blocks in in-band signaling (See Figure 13).

<unicast-address>: is taken from RFC4566 [19]. It is one of the IP addresses of the endpoint and allows the use of IPv4 addresses, IPv6 addresses and Fully Qualified Domain Names (FQDN). An endpoint MUST only include the IP address for which the connectivity checks have succeeded.

<port>: is from RFC4566 [19]. It is the RTP port associated with the unicast address and note that the RTP and RTCP ports are multiplexed for MPRTP subflows.

<counter>: is a monotonically increasing positive integer starting at 1. The counter MUST reset for each media line. The counter value for an 'mprtp-interface' should remain the same for the session.

The 'mprtp-interface' can be extended using the 'interface-description-extension' parameter. An endpoint MUST ignore any extensions it does not understand.

11.3.2. Example

The ABNF grammar is illustrated by means of an example:

```
v=0
o=alice 2890844526 2890844527 IN IP4 192.0.2.1
s=
c=IN IP4 192.0.2.1
t=0 0
m=video 49170 RTP/AVP 98
a=rtpmap:98 H264/90000
a=fmtp:98 profile-level-id=42A01E;
a=extmap:1 urn:ietf:params:rtp-hdrext:mprtp
a=rtcp-mux
a=mprtp interface:1 192.0.2.1:49170    ;primary interface
a=mprtp interface:2 198.51.100.1:51372    ;additional interface
```

11.4. MPRTP with ICE

If the endpoints intend to use ICE [3] for discovering interfaces and running connectivity checks then the following two step procedure MUST be followed:

1. Advertise ICE candidates: in the initial OFFER the endpoints exchange candidates, as defined in ICE [3]. Thereafter the endpoints run connectivity checks.

2. Advertise MPRTP interfaces: When a sufficient number of connectivity checks succeed the endpoint MUST send an updated offer containing the interfaces that they want to use for MPRTP.

When an endpoint uses ICE's regular nomination [3] procedure, it chooses the best ICE candidate as the default path. In the case of an MPRTP endpoint, if more than one ICE candidate succeeded the connectivity checks then an MPRTP endpoint MAY advertise (some of) these as "mprtp-interfaces" in an updated offer.

When an endpoint uses ICE's aggressive nomination [3] procedure, the selected candidate may change as more ICE checks complete. Instead of sending updated offers as additional ICE candidates appear

(transience), the endpoint it MAY use in-band signaling to advertise its interfaces, as defined in Section 9.3. Additionally, it MAY send an updated offer when the transience stabilizes.

If the default interface disappears and the paths used for MPRTP are different from the one in the c= and m= lines then the mprtp-intefaces with the lowest counter value should be promoted to the c= and m= lines in the updated offer.

When a new interface appears then the application/endpoint should internally decide if it wishes to use it and sends an updated offer with ICE candidates of the new interface. The receiving endpoint responds to the offer with all its ICE candidates in the answer and starts connectivity checks between all its candidates and the offerer's new ICE candidate. Similarly, the initiating endpoint starts connectivity checks between the new interface and all the received ICE candidates in the answer. If the connectivity checks succeed, the initiating endpoint MAY send an updated offer with the new interface as an additional mprtp-interface.

11.5. Increased Throughput

The MPRTP layer MAY choose to augment paths to increase throughput. If the desired media rate exceeds the current media rate, the endpoints MUST renegotiate the application specific ("b=AS:xxx") [19] bandwidth.

11.6. Offer/Answer

When the SDP [19] is used to negotiate MPRTP sessions following the offer/answer model [15], the "a=mprtp" attribute (see Section 11.2) indicates the desire to use multiple interfaces to send or receive media data. The initial SDP offer MUST include this attribute at the media level. If the answerer wishes to also use MPRTP, it MUST include a media-level "a=mprtp" attribute in the answer. If the answer does not contain an "a=mprtp" attribute, the offerer MUST NOT stream media over multiple paths and the offerer MUST NOT advertise additional MPRTP interfaces in-band or out-of-band.

When SDP is used in a declarative manner, the presence of an "a=mprtp" attribute signals that the sender can send or receive media data over multiple interfaces. The receiver SHOULD be capable to stream media to the multiple interfaces and be prepared to receive media from multiple interfaces.

The following sections shows examples of SDP offer and answer for in-band and out-of-band signaling.

11.6.1. In-band Signaling Example

The following offer/answer shows that both the endpoints are MPRTP capable and SHOULD use in-band signaling for interfaces advertisements.

```
Offer:
  v=0
  o=alice 2890844526 2890844527 IN IP4 192.0.2.1
  s=
  c=IN IP4 192.0.2.1
  t=0 0
  m=video 49170 RTP/AVP 98
  a=rtpmap:98 H264/90000
  a=fmtp:98 profile-level-id=42A01E;
  a=rtcp-mux
  a=mprtp


Answer:
  v=0
  o=bob 2890844528 2890844529 IN IP4 192.0.2.2
  s=
  c=IN IP4 192.0.2.2
  t=0 0
  m=video 4000 RTP/AVP 98
  a=rtpmap:98 H264/90000
  a=fmtp:98 profile-level-id=42A01E;
  a=rtcp-mux
  a=mprtp
```

The endpoint MAY now use in-band RTCP signaling to advertise its multiple interfaces. Alternatively, it MAY make another offer with the interfaces in SDP (out-of-band signaling).

11.6.2. Out-of-band Signaling Example

If the multiple interfaces are included in an SDP offer then the receiver MUST respond to the request with an SDP answer.

11.6.2.1. Without ICE

In this example, the offerer advertises two interfaces and the answerer responds with a single interface description. The endpoint MAY use one or both paths depending on the end-to-end characteristics of each path.

```
Offer:
  v=0
  o=alice 2890844526 2890844527 IN IP4 192.0.2.1
  s=
  c=IN IP4 192.0.2.1
  t=0 0
  m=video 49170 RTP/AVP 98
  a=rtpmap:98 H264/90000
  a=fmtp:98 profile-level-id=42A01E;
  a=rtcp-mux
  a=mprtp interface:1 192.0.2.1:49170
  a=mprtp interface:2 198.51.100.1:51372


Answer:
  v=0
  o=bob 2890844528 2890844529 IN IP4 192.0.2.2
  s=
  c=IN IP4 192.0.2.2
  t=0 0
  m=video 4000 RTP/AVP 98
  a=rtpmap:98 H264/90000
  a=fmtp:98 profile-level-id=42A01E;
  a=rtcp-mux
  a=mprtp interface:1 192.0.2.2:4000
```

11.6.2.2. With ICE

In this example, the endpoint first sends its ICE candidates in the initial offer and the other endpoint answers with its ICE candidates.

Initial offer (with ICE candidates):

```
Offer:
  v=0
  o=alice 2890844526 2890844527 IN IP4 192.0.2.1
  s=
  c=IN IP4 192.0.2.1
  t=0 0
  a=ice-pwd:asd88fgpdd777uzjYhagZg
  a=ice-ufrag:8hhY
  a=mprtp
  m=video 49170 RTP/AVP 98
  a=rtpmap:98 H264/90000
  a=fmtp:98 profile-level-id=42A01E;
  a=rtcp-mux
  a=candidate:1 1 UDP 2130706431 192.0.2.1 49170 typ host
  a=candidate:2 1 UDP 1694498815 198.51.100.1 51372 typ host
```

```
   Answer:
     v=0
     o=bob 2890844528 2890844529 IN IP4 192.0.2.2
     s=
     c=IN IP4 192.0.2.2
     t=0 0
     a=ice-pwd:YH75Fviy6338Vbrhrlp8Yh
     a=ice-ufrag:9uB6
     a=mprtp
     m=video 4000 RTP/AVP 98
     a=rtpmap:98 H264/90000
     a=fmtp:98 profile-level-id=42A01E;
     a=rtcp-mux
     a=candidate:1 1 UDP 2130706431 192.0.2.2 4000 typ host
```

Thereafter, each endpoint conducts ICE connectivity checks and when sufficient number of connectivity checks succeed, the endpoint sends an updated offer. In the updated offer, the endpoint advertises its multiple interfaces for MPRTP.

Updated offer (with MPRTP interfaces):

```
   Offer:
     v=0
     o=alice 2890844526 2890844527 IN IP4 192.0.2.1
     s=
     c=IN IP4 192.0.2.1
     t=0 0
     m=video 49170 RTP/AVP 98
     a=rtpmap:98 H264/90000
     a=fmtp:98 profile-level-id=42A01E;
     a=rtcp-mux
     a=mprtp interface:1 192.0.2.1:49170
     a=mprtp interface:2 198.51.100.1:51372


   Answer:
     v=0
     o=bob 2890844528 2890844529 IN IP4 192.0.2.2
     s=
     c=IN IP4 192.0.2.2
     t=0 0
     m=video 4000 RTP/AVP 98
     a=rtpmap:98 H264/90000
     a=fmtp:98 profile-level-id=42A01E;
     a=rtcp-mux
     a=mprtp interface:1 192.0.2.2:4000
```

12. MPRTP in RTSP

Endpoints MUST use RTSP 2.0 [17] for session setup. Endpoints MUST multiplex RTP and RTCP on a single port [10] and follow the recommendations made in Appendix C of [17].

12.1. Solution Overview without ICE

1. The RTSP Server should include all of its multiple interfaces via the SDP attribute ("a=mprtp interface") in the RTSP DESCRIBE message.

2. The RTSP Client should include all its multiple interface in the RTSP SETUP message using the new attribute ("dest_mprtp_addr=") in the Transport header. [[Comment.5: Do we need a new lower layer transport MPRTP?. --Editor]]

3. The RTSP Server responds to the RTSP SETUP message with a 200 OK containing its MPRTP interfaces (using the "src_mprtp_header=") in the Transport header. After this the RTSP Client can issue a PLAY request.

4. If a new interface appears or an old one disappear at the RTSP Client during playback, it should send a new RTSP SETUP message containing the updated interfaces ("dest_mprtp_addr") in the Transport header. [[Comment.6: Sending a re-SETUP to update the interfaces during PLAY state would require a change in behavior of the server. Similar to Section 5.12 of [draft-ietf-mmusic-rtsp-nat]. --Editor]]

5. If a new interface appear or an old one disappear at the RTSP Server during playback, the RTSP Server should send a PLAY_NOTIFY message with a new Notify-Reason: "src-mprtp-interface-update". The request must contain the updated interfaces ("dest_mprtp_addr") in the "MPRTP-Interfaces" header.

6. Alternatively, the RTSP Server or Client may use the RTCP (in-band) mechanism to advertise their interfaces.

[[Comment.7: Does it make sense to advertise out-of-band (in RTSP SETUP) when advertising in-band in RTCP is less complex? --Editor]]

The overview is illustrated by means of an example:

```
C->S: DESCRIBE rtsp://server.example.com/fizzle/foo RTSP/2.0
      CSeq: 111
      User-Agent: PhonyClient 1.3
      Accept: application/sdp, application/example
      Supported: setup.mprtp, setup.rtp.rtcp.mux

S->C: RTSP/2.0 200 OK
      CSeq: 111
      Date: 23 Jan 2011 15:35:06 GMT
      Server: PhonyServer 1.3
      Content-Type: application/sdp
      Content-Length: 367
      Supported: setup.mprtp, setup.rtp.rtcp.mux

       v=0
       o=mprtp-rtsp-server 2890844526 2890844527 IN IP4 192.0.2.1
       s=
       c=IN IP4 192.0.2.1
       t=0 0
       m=video 49170 RTP/AVP 98
       a=rtpmap:98 H264/90000
       a=fmtp:98 profile-level-id=42A01E;
       a=extmap:1 urn:ietf:params:rtp-hdrext:mprtp
       a=rtcp-mux
       a=mprtp interface:1 192.0.2.1:49170
       a=mprtp interface:2 198.51.100.1:51372
```

On receiving the response to the RTSP DESCRIBE message, the RTSP Client sends a RTSP SETUP message containing its MPRTP interfaces in the Transport header using the "dest_mprtp_addr=" attribute. The RTSP Server responds with a 200 OK containing both the RTSP client's and the RTSP Server's MPRTP interfaces.

```
C->S: SETUP rtsp://server.example.com/fizzle/foo/audio RTSP/2.0
      CSeq: 112
      Transport: RTP/AVPF/UDP; unicast; dest_mprtp_addr="
      1 192.0.2.2 4000"; RTCP-mux,
      RTP/AVP/UDP; unicast; dest_addr=":6970"/":6971",
      RTP/AVP/TCP;unicast;interleaved=0-1
      Accept-Ranges: NPT, UTC
      User-Agent: PhonyClient 1.3
      Supported: setup.mprtp, setup.rtp.rtcp.mux

S->C: RTSP/2.0 200 OK
      CSeq: 112
      Session: 12345678
      Transport: RTP/AVPF/UDP; unicast; dest_mprtp_addr="
      1 192.0.2.2 4000";
      src_mprtp_addr="1 192.0.2.1 49170;
      2 198.51.100.1 51372"; RTCP-mux
      Accept-Ranges: NPT
      Date: 23 Jan 2012 15:35:06 GMT
      Server: PhonyServer 1.3
      Supported: setup.mprtp, setup.rtp.rtcp.mux
```

The RTSP Client can issue a PLAY request on receiving the 200 OK and media can start to stream once the RTSP server receives the PLAY request.

12.2. Solution Overview with ICE

This overview uses the ICE mechanisms [20] defined for RTSP 2.0 [17].

1. The RTSP Server should include the "a=rtsp-ice-d-m" attribute and also indicate that it supports MPRTP by including the "a=mprtp" attribute in the SDP of the RTSP DESCRIBE message.

2. The client sends a RTSP SETUP message containing the D-ICE in lower level transport and ICE candidates in the transport header. The RTSP Server and client then follow the procedures (Steps 2 to 8) described in [20].

3. When the connectivity checks conclude the RTSP Client can send an updated RTSP SETUP message with its MPRTP interfaces (ICE candidates that were successful) in the Transport header ("dest_mprtp_addr="). The RTSP Server responds to the RTSP SETUP message with a 200 OK containing its MPRTP interfaces (ICE candidates that were successful) in the Transport header ("src_mprtp_header="). After receiving the 200 OK, the RTSP client can issue the PLAY request.

4. Alternatively after the connectivity checks conclude, the RTSP Client can issue the PLAY request (Step 9 and 10 of [20]) and the endpoints can use the RTCP (in-band) mechanism to advertise their interfaces.

5. If a new interface appears or an old one disappears, the RTSP Client should issue an updated SETUP message with the new candidates (See Section 5.12 of [20]) or the RTSP Server should send a PLAY_NOTIFY message (See Section 5.13 of [20]). After connectivity checks succeed for the new interfaces, the RTSP Client can proceed with the instructions in Step 3 or 4.

The overview is illustrated by means of an example:

```
C->S: DESCRIBE rtsp://server.example.com/foo RTSP/2.0
      CSeq: 312
      User-Agent: PhonyClient 1.3
      Accept: application/sdp, application/example
      Supported: setup.mprtp, setup.ice-d-m, setup.rtp.rtcp.mux

S->C: RTSP/2.0 200 OK
      CSeq: 312
      Date: 23 Jan 2012 15:35:06 GMT
      Server: PhonyServer 1.3
      Content-Type: application/sdp
      Content-Length: 367
      Supported: setup.mprtp, setup.ice-d-m, setup.rtp.rtcp.mux

      v=0
      o=mprtp-rtsp-server 2890844526 2890842807 IN IP4 192.0.2.1
      s=SDP Seminar
      i=A Seminar on the session description protocol
      u=http://www.example.com/lectures/sdp.ps
      e=seminar@example.com (Seminar Management)
      t=2873397496 2873404696
      a=recvonly
      a=rtsp-ice-d-m
      a=control: *
      m=video 49170 RTP/AVP 98
      a=rtpmap:98 H264/90000
      a=fmtp:98 profile-level-id=42A01E;
      a=rtcp-mux
      a=mprtp
      a=control: /video
```

```
C->S: SETUP rtsp://server.example.com/foo/video RTSP/2.0
      CSeq: 302
      Transport: RTP/AVP/D-ICE; unicast; ICE-ufrag=9uB6;
            ICE-Password=YH75Fviy6338Vbrhrlp8Yh;
            candidates="1 1 UDP 2130706431 192.0.2.2
            4000 typ host"; RTCP-mux,
            RTP/AVP/UDP; unicast; dest_addr=":6970"/":6971",
            RTP/AVP/TCP;unicast;interleaved=0-1
      Accept-Ranges: NPT, UTC
      User-Agent: PhonyClient 1.3
      Supported: setup.mprtp, setup.ice-d-m, setup.rtp.rtcp.mux

S->C: RTSP/2.0 200 OK
      CSeq: 302
      Session: 12345678
      Transport: RTP/AVP/D-ICE; unicast; RTCP-mux;
            ICE-ufrag=8hhY; ICE-Password=
            asd88fgpdd777uzjYhagZg; candidates="
            1 1 UDP 2130706431 192.0.2.1 49170 typ host;
            2 1 UDP 1694498815 198.51.100.1 51372 typ host"
      Accept-Ranges: NPT
      Date: 23 Jan 2012 15:35:06 GMT
      Server: PhonyServer 1.3
      Supported: setup.mprtp, setup.ice-d-m, setup.rtp.rtcp.mux
```

After the connectivity checks complete, the RTSP Client can send an updated RTSP SETUP message containing the MPRTP interfaces for which the connectivity checks were successful. These steps are the same as the ones in the previous example.

12.3. RTSP Extensions

12.3.1. MPRTP Interface Transport Header Parameter

This section defines a new RTSP transport parameter for carrying MPRTP interfaces. The transport parameters may only occur once in each transport specification. The parameter can contain one or more MPRTP interfaces. In the SETUP response if the RTSP server support MPRTP it MUST be include one or more MPRTP interfaces.

```
trns-parameter = <Defined in Section 20.2.3 of
                      [I-D.ietf-mmusic-rfc2326bis]>
trns-parameter =/ SEMI dest-mprtp-interface-par
trns-parameter =/ SEMI src-mprtp-interface-par
dest-mprtp-interface-par = "dest_mprtp_addr" EQUAL DQ SWS
                        interface *(SEMI interface) SWS DQ
src-mprtp-interface-par = "src_mprtp_addr" EQUAL DQ SWS
                        interface *(SEMI interface) SWS DQ

interface  = counter SP
             unicast-address SP
             rtp_port SP
             *(SP interface-description-extension)

 counter         = See section 11.3.1
 unicast-address = See section 11.3.1
 rtp_port        = See section 11.3.1
 interface-description-extension = See section 11.3.1
```

12.3.2. MPRTP Feature Tag

A feature tag is defined for indicating MPRTP support in the RTSP capabilities mechanism: "setup.mprtp". This feature tag indicates that the endpoint supports all the mandatory extensions defined in this specification and is applicable to all types of RTSP agents; clients, servers and proxies.

The MPRTP compliant RTSP Client MUST send the feature tag "setup.mprtp" in the "Supported" header of all DESCRIBE and SETUP requests.

12.3.3. Status Codes

TBD

12.3.4. New Reason for PLAY_NOTIFY

A new value used in the PLAY_NOTIFY methods Notify-Reason header is defined: "src-mprtp-interface-update". This reason indicates that the RTSP Server has updated set of MPRTP interfaces.

```
Notify-Reas-val =/ "src-mprtp-interface-update"
```

PLAY_NOTIFY requests with Notify-Reason header set to src-mprtp-interface-update MUST include a mprtp-interfaces header.

```
mprtp-interfaces        =  "mprtp-interfaces" HCOLON interface
                          *(COMMA interface)
interface  = counter SP
             unicast-address SP
             rtp_port SP
             *(SP interface-description-extension)

 counter         = See section 11.3.1
 unicast-address = See section 11.3.1
 rtp_port        = See section 11.3.1
 interface-description-extension = See section 11.3.1
```

[[Comment.8: Do we need to add a new header attribute?. Alternatively, the RTSP Server could just send the PLAY_NOTIFY and let the RTSP client initiate a new RTSP SETUP message with its current interfaces and the RTSP Server can then respond with its updated set of interfaces. This will make it a 3-way exchange as opposed to a 1-way notification. Alternatively, using SET_PARAMETER reduces it to a 2-way exchange and can be initiated by both the RTSP Server and the RTSP Client. However, SET_PARAMETER can only be used when the endpoints are in SETUP state. --Editor]]

Example:

```
S->C: PLAY_NOTIFY rtsp://server.example.com/foo RTSP/2.0
      CSeq: 305
      Notify-Reason: src-mprtp-interface-update
      Session: 12345678
      mprtp-interfaces: 2 192.0.2.10 48211, 3 198.51.100.11 38703
      Server: PhonyServer 1.3

C->S: RTSP/2.0 200 OK
      CSeq: 305
      User-Agent: PhonyClient 1.3
```

12.3.5. re-SETUP

The server SHALL support SETUP requests in PLAYING state if it is only updating the transport parameter (dest_mprtp_addr). If the session is established using ICE then the RTSP Server and Client MUST also follow the procedures described for re-SETUP in [20].

13. IANA Considerations

The following contact information shall be used for all registrations in this document:

Contact: Varun Singh
mailto:varun.singh@iki.fi
tel:+358-9-470-24785

Note to the RFC-Editor: When publishing this document as an RFC, please replace "RFC XXXX" with the actual RFC number of this document and delete this sentence.

13.1. MPRTP Header Extension

This document defines a new extension URI to the RTP Compact Header Extensions sub-registry of the Real-Time Transport Protocol (RTP) Parameters registry, according to the following data:

Extension URI: urn:ietf:params:rtp-hdrext:mprtp
Description: Multipath RTP
Reference: RFC XXXX

13.2. MPRTCP Packet Type

A new RTCP packet format has been registered with the RTCP Control Packet type (PT) Registry:

Name:	MPRTCP
Long name:	Multipath RTCP
Value:	211
Reference:	RFC XXXX.

This document defines a substructure for MPRTCP packets. A new sub-registry has been set up for the sub-report block type (MPRTCP_Type) values for the MPRTCP packet, with the following registrations created initially:

```
Name:           Subflow Specific Report
Long name:      Multipath RTP Subflow Specific Report
Value:          0
Reference:      RFC XXXX.

Name:           IPv4 Address
Long name:      IPv4 Interface Address
Value:          1
Reference:      RFC XXXX.

Name:           IPv6 Address
Long name:      IPv6 Interface Address
Value:          2
Reference:      RFC XXXX.

Name:           DNS Name
Long name:      DNS Name indicating Interface Address
Value:          3
Reference:      RFC XXXX.
```

Further values may be registered on a first come, first served basis. For each new registration, it is mandatory that a permanent, stable, and publicly accessible document exists that specifies the semantics of the registered parameter as well as the syntax and semantics of the associated sub-report block. The general registration procedures of [19] apply.

13.3. SDP Attributes

This document defines a new SDP attribute, "mprtp", within the existing IANA registry of SDP Parameters.

13.3.1. "mprtp" attribute

- Attribute Name: MPRTP
- Long Form: Multipath RTP
- Type of Attribute: media-level
- Charset Considerations: The attribute is not subject to the charset attribute.
- Purpose: This attribute is used to indicate support for Multipath RTP. It can also provide one of many possible interfaces for communication. These interface addresses may have been validated using ICE procedures.

- Appropriate Values: See Section 11.2 and Section 11.3.1 of RFC XXXX.

13.4. RTSP

This document requests registration in a number of registries for RTSP.

13.4.1. RTSP Feature Tag

This document request that one RTSP 2.0 feature tag be registered in the "RTSP 2.0 feature tag" registry:

setup.mprtp See Section 12.3.2.

13.4.2. RTSP Transport Parameters

This document requests that 2 transport parameters be registered in RTSP 2.0's "Transport Parameters":

"dest_mprtp_addr": See Section 12.3.1.

"src_mprtp_addr": See Section 12.3.1.

13.4.3. Notify-Reason value

This document requests that one assignment be done in the RTSP 2.0 Notify-Reason header value registry. The defined value is:

"src-mprtp-interface-update": See Section 12.3.4.

14. Security Considerations

TBD

All drafts are required to have a security considerations section. See RFC 3552 [21] for a guide.

15. Acknowledgements

Varun Singh, Saba Ahsan, and Teemu Karkkainen are supported by Trilogy (http://www.trilogy-project.org), a research project (ICT-216372) partially funded by the European Community under its Seventh Framework Program. The views expressed here are those of the author(s) only. The European Commission is not liable for any use that may be made of the information in this document.

The authors would also like acknowledge the contribution of Ralf Globisch and Thomas Schierl for providing the input and text for the MPRTP interface advertisement in SDP.

Thanks to Miguel A. Garcia, Ralf Globisch, Christer Holmberg, and Roni Even for providing valuable feedback on earlier versions of this draft

16. References

16.1. Normative References

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

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

[3] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[4] Ott, J., Chesterfield, J., and E. Schooler, "RTP Control Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback", RFC 5760, February 2010.

[5] Johansson, I. and M. Westerlund, "Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences", RFC 5506, April 2009.

[6] 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.

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

[8] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[9] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

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

16.2. Informative References

[11] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

[12] Ford, A., Raiciu, C., Handley, M., Barre, S., and J. Iyengar, "Architectural Guidelines for Multipath TCP Development", RFC 6182, March 2011.

[13] Nordmark, E. and M. Bagnulo, "Shim6: Level 3 Multihoming Shim Protocol for IPv6", RFC 5533, June 2009.

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

[15] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[16] 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.

[17] Schulzrinne, H., Rao, A., Lanphier, R., Westerlund, M., and M. Stiemerling, "Real Time Streaming Protocol 2.0 (RTSP)", draft-ietf-mmusic-rfc2326bis-28 (work in progress), October 2011.

[18] Marjou, X. and A. Sollaud, "Application Mechanism for Keeping Alive the NAT Mappings Associated with RTP / RTP Control Protocol (RTCP) Flows", RFC 6263, June 2011.

[19] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[20] Goldberg, J., Westerlund, M., and T. Zeng, "A Network Address Translator (NAT) Traversal mechanism for media controlled by Real-Time Streaming Protocol (RTSP)", draft-ietf-mmusic-rtsp-nat-11 (work in progress), October 2011.

[21] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.

Appendix A. Interoperating with Legacy Applications

An MPRTP sender can use its multiple interfaces to send media to a legacy RTP client. The legacy receiver will ignore the subflow RTP header and the receiver's de-jitter buffer will try to compensate for

the mismatch in per-path delay. However, the receiver can only send the overall or aggregate RTCP report which may be insufficient for an MPRTP sender to adequately schedule packets or detect if a path disappeared.

An MPRTP receiver can only use one of its interface when communicating with a legacy sender.

Appendix B. Change Log

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

B.1. changes in draft-singh-avtcore-mprtp-04

- Fixed missing 0xBEDE header in MPRTP header format.
- Removed connectivity checks and keep-alives from in-band signaling.
- MPRTP and MPRTCP are multiplexed on a single port.
- MPRTCP packet headers optimized.
- Made ICE optional
- Updated Sections: 7.1.2, 8.1.x, 11.2, 11.4, 11.6.
- Added how to use MPRTP in RTSP (Section 12).
- Updated IANA Considerations section.

B.2. changes in draft-singh-avtcore-mprtp-03

- Added this change log.
- Updated section 6, 7 and 8 based on comments from MMUSIC.
- Updated section 11 (SDP) based on comments of MMUSIC.
- Updated SDP examples with ICE and non-ICE in out-of-band signaling scenario.
- Added Appendix A on interop with legacy.
- Updated IANA Considerations section.

B.3. changes in draft-singh-avtcore-mprtp-02

- MPRTCP protocol extensions use only one PT=210, instead of 210 and 211.
- RTP header uses 1-byte extension instead of 2-byte.
- Added section on RTCP Interval Calculations.
- Added "mprtp-interface" attribute in SDP considerations.

B.4. changes in draft-singh-avtcore-mprtp-01

- Added MPRTP and MPRTCP protocol extensions and examples.
- WG changed from -avt to -avtcore.

Authors' Addresses

Varun Singh
Aalto University
School of Science and Technology
Otakaari 5 A
Espoo, FIN 02150
Finland

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

Teemu Karkkainen
Aalto University
School of Science and Technology
Otakaari 5 A
Espoo, FIN 02150
Finland

Email: teemuk@comnet.tkk.fi

Joerg Ott
Aalto University
School of Science and Technology
Otakaari 5 A
Espoo, FIN 02150
Finland

Email: jo@comnet.tkk.fi

Saba Ahsan
Aalto University
School of Science and Technology
Otakaari 5 A
Espoo, FIN 02150
Finland

Email: sahsan@cc.hut.fi

Lars Eggert
NetApp
Sonnenallee 1
Kirchheim 85551
Germany

Phone: +49 151 12055791
Email: lars@netapp.com
URI: http://eggert.org/