AVT A. Begen
Internet-Draft B. VerSteeg
Intended status: Standards Track Cisco
Expires: August 30, 2010 February 26, 2010

Port Mapping Between Unicast and Multicast RTP Sessions
draft-ietf-avt-ports-for-ucast-mcast-rtp-00

Abstract

This document presents port mapping solutions that allow RTP receivers to choose their own RTP and RTCP receive ports for the unicast session(s) in RTP applications using both unicast and multicast services.

Table of Contents

1. Introduction

In (any-source or source-specific) multicast RTP applications, destination ports, i.e., the ports on which the multicast receivers receive the RTP and RTCP packets, are defined declaratively. In other words, the receivers cannot choose their receive ports and the sender(s) use the pre-defined ports.

In unicast RTP applications, the receiving end often needs to choose its receive ports for RTP and RTCP. It may convey its request to the sending end through different ways, one of which is the Offer/Answer Model [RFC3264] for the Session Description Protocol (SDP) [RFC4566]. However, the Offer/Answer Model requires offer/answer exchange(s) between the endpoints, and the resulting delay may not be acceptable in delay-sensitive real-time applications.

RTP sessions are defined based on the destination addresses [RFC3550]. While the declaration and selection of the ports are well defined and work well for multicast and unicast RTP applications, respectively, the usage of the ports introduces complications when a receiving end mixes unicast and multicast RTP sessions within the same RTP application.

An example scenario is where the RTP packets are distributed through source-specific multicast (SSM) and a receiver sends unicast RTCP feedback to a local repair server (also functioning as a feedback target) [I-D.ietf-avt-rtcpssm] asking for a retransmission of the packets it is missing, and the local repair server sends the retransmissions over a unicast RTP session [RFC4588].

Another scenario is where a receiver wants to rapidly acquire a new primary multicast RTP session and receives one or more RTP retransmissions over a unicast session before joining the SSM session [I-D.ietf-avt-rapid-acquisition-for-rtp]. Similar scenarios exist in applications where some part of the content is distributed through multicast while the receivers get additional and/or auxiliary content through one or more unicast connections, as sketched in Figure 1.

In this document, we discuss this problem and introduce alternative solutions that we refer to as Port Mapping. These solutions allow receivers to choose their desired RTP and RTCP receive ports for every unicast session when they are running RTP applications using both unicast and multicast services.

```
          -----------
         |  Unicast  |................
         |  Source   |............  :
         | (Server)  |           :  :
          -----------            :  :
                                 v  v
          -----------          ----------             -----------
         | Multicast |------->|  Router  |---------->|Client RTP |
         |  Source   |        |          |.........>|Application|
          -----------          ----------             -----------
                                  | :
                                  | :                 -----------
                                  | :..............>|Client RTP |
                                  +---------------->|Application|
                                                      -----------


         -------> Multicast RTP Flow
         .......> Unicast RTP Flow
```

Figure 1: RTP applications simultaneously using both unicast and multicast services

In the remainder of this document, we refer to the RTP endpoints that serve other RTP endpoints over a unicast session as the Servers. The receiving RTP endpoints are referred to as Clients.

2. Requirements Notation

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. Design Guidelines

We have the following design guidelines in developing a port mapping solution:

- o Design a scalable and distributable system. This drives the design towards a system in which all of the actions associated with a given set of flows at a given instant in time are distinct from actions on other flows. This allows the system to be dynamically segmented as dictated by dynamic conditions in the network.

- Use atomic, client-driven transactions in order to limit the amount of state information maintained by the server.

- Use idempotent transactions in order to limit the impact to the overall system when messages are lost. The state of the system thus only depends on the last successfully received message.

- Do not create dependency among messages carried in different packets if possible. In other words, if an information is logically coupled to other information, send all of the data in a single transaction to the extent that this is practical.

- Do not introduce new vectors for attacks.

- Do not have any IPv4/IPv6 dependencies. To the extent that addressing information is required to persist across transactions, handle the addresses in a manner that allows the server to give opaque address information (called Cookie) to the client. The client then presents the opaque addressing information back to the server in subsequent transactions. This allows the system to maintain connectivity information without unduly burdening the server(s) with state information.

  The cookie is generated by the server (Section 4.2) or the client (Section 4.3), and is only understood by the server or the client, respectively. To other systems, the cookie is opaque data. Thus, the endpoint generating the cookie may use any method of its choice to make the cookie data opaque.

- Be NAT-tolerant [RFC5389] [RFC4787]. Considerations for IPv6/IPv4 translation are out of scope of this specification.

4. Port Mapping

We present the details of the proposed solutions in the context of an example application.

Consider an SSM distribution network where a distribution source multicasts RTP packets to a large number of clients, and one or more retransmission servers function as feedback targets to collect unicast RTCP feedback from these clients [I-D.ietf-avt-rtcpssm]. When a client detects a missing packet in the primary multicast session, it requests a retransmission from one of the retransmission servers. The client may or may not be behind a NAT device. We first consider the simpler scenario where there are no NAT devices between the server and client. We then discuss the implications of NAT devices.

The pertaining RTP and RTCP flows are sketched in Figure 2.

```
     --------------                                 ---     ----------
    |              |-------------------------------|   |-->|P1        |
    |              |-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-|   |.->|P2        |
    |              |                               |   |   |          |
    | Distribution |      ----------------         |   |   |          |
    |    Source    |     |                |        |   |   |          |
    |              |---->|P1              |        |   |   |          |
    |              |.-.->|P2              |        |   |   |          |
    |              |     |                |        |   |   |          |
     --------------      |              P2|<.=.=.=.|   |=.=|*c1       |
                         |              P2|<~~~~~~~|   |~~~|*c1       |
                         |                |        | N |   |          |
                         | Retransmission |        | A |   |  Client  |
                         |     Server     |        | T |   |          |
                         |                |        |   |   |          |
                         |              P3|........|   |..>|*c2       |
                         |              P4|<.=.=.=.|   |=.>|*c3       |
                         |                |        |   |   |          |
                          ----------------          ---     ----------

    -------> Multicast RTP Flow
    .-.-.-.> Multicast RTCP Flow
    .=.=.=.> Unicast RTCP Reports
    ~~~~~~~> Unicast RTCP Feedback Messages
    .......> Unicast RTP Flow
```

Figure 2: Example scenario showing an SSM distribution with support for retransmissions from a local server

4.1. SDP Description

The SDP describing the scenario given in Figure 2 can be written as:

```
v=0
o=ali 1122334455 1122334466 IN IP4 nack.example.com
s=Local Retransmissions
t=0 0
a=group:FID 1 2
a=rtcp-unicast:rsi
m=video 41000 RTP/AVPF 98
i=Primary Multicast Stream
c=IN IP4 233.252.0.2/255
a=source-filter: incl IN IP4 233.252.0.2 198.51.100.1
a=rtpmap:98 MP2T/90000
a=rtcp:41001 IN IP4 192.0.2.1
a=rtcp-fb:98 nack
a=mid:1
m=video 41002 RTP/AVPF 99
i=Unicast Retransmission Stream
c=IN IP4 192.0.2.1
a=rtpmap:99 rtx/90000
a=rtcp:41003
a=fmtp:99 apt=98; rtx-time=5000
a=mid:2
```

Figure 3: SDP describing an SSM distribution with support for retransmissions from a local server

In this SDP, the source stream is multicast from a distribution source (with a source IP address of 198.51.100.1) to the multicast destination address of 233.252.0.2 and port 41000. A retransmission server including feedback target functionality (with an address of 192.0.2.1 and port of 41001) is specified with the 'rtcp' attribute. The RTCP port for the unicast session (41003) is also specified with the 'rtcp' attribute.

Based on this SDP, we define the following parameters:

- DS=198.51.100.1 - Address of the distribution source
- G=233.252.0.2 - Destination address where the primary multicast stream is sent to
- P1=41000 - Destination RTP port where the primary multicast stream is sent to
- P2=41001 - Destination RTCP port on the retransmission server and clients for the primary multicast session
- S=192.0.2.1 - Address of the retransmission server

o P3=41002 - Source RTP port on the retransmission server for the unicast session

o P4=41003 - RTCP port on the retransmission server for the unicast session

We denote the client address by C. *c1 denotes the port on the client used to send the unicast feedback in the primary multicast session. *c2 and *c3 denote the RTP and RTCP ports on the client used in the unicast session, respectively. The '*' before the port numbers means that these port numbers are chosen by the client, and not assigned/ imposed by another entity. Note that if the client implements RTP/ RTCP port muxing [I-D.ietf-avt-rtp-and-rtcp-mux] in the unicast session, c2 will equal c3.

During the lifetime of a unicast session, the server needs to remember the public IP address and public RTP and RTCP ports of the client as a part of the session state information.

4.2. Server-Generated Cookie Approach

4.2.1. Steps

This approach follows the steps outlined below:

1. The client ascertains server address and port number(s) from the SDP description (S, P3 and P4).

2. The client determines its port numbers (*c2 and *c3).

3. The client sends a message to the server via a new RTCP message, called PortMappingRequest. Separate messages are sourced from ports c2 and c3 on the client. Note that normally the message sent from port c2 should be addressed to port P3 on the server, and the message sent from port c3 should be addressed to port P4 on the server. However, since the former RTCP message is sent to an RTP port (P3), the server is required to implement RTP/RTCP port muxing on this port [I-D.ietf-avt-rtp-and-rtcp-mux]. Thus, the server MUST support RTP/RTCP port muxing, and both PortMappingRequest messages sourced from ports c2 and c3 MUST be sent to port P3 on the server.

4. The server derives client address (C) and its RTP and RTCP ports (c2 and c3) from the received messages.

5. For each PortMappingRequest message, the server generates an opaque encapsulation (called Cookie) that conveys client's addressing information (IP address and port) using a reversible

transform only known to the server.

6. The server sends each cookie back to the client using a new RTCP message, called PortMappingResponse. Assuming that the client does not support port muxing, two separate PortMappingResponse messages MUST be sent to port c3 on the client and the server MUST indicate in each PortMappingResponse message whether it is for an RTP or for an RTCP port using an appropriate field.

   For the server to be able to send the PortMappingResponse for port c2 to port c3, the client needs to include the cookie for port c3 when requesting the cookie for port c2. This introduces delay and dependency, which may be a drawback in certain applications (See Figure 4).

7. If the client supports port muxing, then there is no need to select a port c3 and the client needs one cookie only.

8. The client includes the cookie(s) when necessary in the subsequent messages sent to the server.

9. Normal flows ensue, with the server using the addressing encapsulated in the opaque cookie(s).

4.2.2. Implications of NATs

If there are no NAT devices between the server and client, the client MUST acquire a cookie for each distinct 2-tuple of (S, c2) and (S, c3). In other words, as long as the client uses the same local ports and the same server, it can use the same cookies when communicating with any feedback target running on this server. The advantage here is that the client can acquire the necessary cookies at the very beginning for every port pair (if it is not port-muxing) it is planning to use, and thus, can avoid the delays incurred to acquire the cookies later when it wants to use a new unicast service.

If there is a NAT device between the server and client, the client may still acquire the cookies at the beginning, provided that it is behind a NAT that assigns the same public IP address and port for the messages sent from the same internal IP address and port even when the client is talking to different destinations ("endpoint-independent mapping" [RFC4787]). However, if the NAT has endpoint-dependent mapping [RFC4787], the client MUST fall back to acquiring a cookie for each distinct 3-tuple of (S, P3, c2) and (S, P3, c3). In practice, however, it is a difficult task to determine the type of a NAT device [I-D.ietf-behave-nat-behavior-discovery].

When the client is behind a NAT, it needs to send periodic packets to

keep the NAT bindings alive [RFC4787]. If the NAT device fails for some reason and then restarts, the public IP address and ports assigned to a client may change. This will invalidate the previously acquired cookies. Upon detecting the failure, the client must acquire new cookies.

4.2.3. Message Flows

Figure 4 shows the message flows, where each message is appended with the (Source Address, Source Port, Destination Address, Destination Port) information. In this section, we assume that the client does not mux the RTP and RTCP ports.

```
        -------------    ---------------                     --------
       | Distribution | | Retransmission |                   |        |
       |    Source    | |     Server     |                   | Client |
       |     (DS)     | |       (S)      |                   |   (C)  |
        -------------    ---------------                     --------
           |                    |                                  |
           |                    |                                  |
           |- (DS, *, G, P1) ->|--------- RTP Multicast --------->|
           |- (DS, *, G, P2) .>|.-.-.-.-. RTCP Multicast -.-.-.-.>|
           |                    |                                  |
           |                    |                                  |
           |    (C, c1, S, P2) |<.=.=. RTCP Receiver Reports =.=.=|
           |                    |    (for the multicast session)   |
           :                    :                                  :
           |    (C, c3, S, P3) |<~~~~ PortMappingRequest(c3) ~~~~~|
           |                    |                                  |
           |    (S, P3, C, c3) |~~~~~~~ PortMappingResponse ~~~~~>|
           |                    |            Cookie(c3)            |
           |                    |                                  |
           |    (C, c2, S, P3) |<~~~~ PortMappingRequest(c2) ~~~~~|
           |                    |          with Cookie(c3)         |
           |                    |                                  |
           |    (S, P3, C, c3) |~~~~~~~ PortMappingResponse ~~~~~>|
           |                    |            Cookie(c2)            |
           |                    |                                  |
           |    (C, c1, S, P2) |<~ RTCP NACK with Cookie(c2,c3) ~~|
           |                    |                                  |
           |                    |**********************************|
           |                    |*   UNICAST SESSION ESTABLISHED  *|
           |                    |**********************************|
           |                    |                                  |
           |    (S, P3, C, c2) |....... RTP Retransmissions .....>|
           |                    |                                  |
           |                    |                                  |
           |    (C, c3, S, P3) |<.=.=. RTCP Receiver Reports =.=.=|
           |                    |     (for the unicast session)    |
           |                    |                                  |
           |    (S, P3, C, c3) |.=.=.=. RTCP Sender Reports =.=.=>|
           |                    |     (for the unicast session)    |
           |                    |                                  |


       -------> Multicast RTP Flow
       .-.-.-.> Multicast RTCP Flow
       .=.=.=.> Unicast RTCP Reports
       ~~~~~~~> Unicast RTCP Feedback Messages
       .......> Unicast RTP Flow
```

Figure 4: Message flows for server-side cookie approach

In the example above, the compound RTCP packet carrying the NACK message also carries the Cookie(c2) and Cookie(c3) since the server must know which ports the client is expecting to receive the RTP retransmission packet(s) and RTCP sender reports on. If an RTCP message from the client will not trigger any transmission from the server (e.g., RTCP receiver and extended reports), it does not have to include any cookies.

4.3. Client-Generated Cookie Approach

4.3.1. Steps

This approach follows the steps outlined below:

1. The client ascertains server address and port number from the SDP description (S and P3).

2. The client determines its port numbers (*c2 and *c3).

3. The client generates a random cookie.

4. The client sends separate RTCP packets from its ports c2 and c3 to the server port P3 to setup the NAT. Each RTCP packet indicates through a bit/field whether it source port will be used for RTP or RTCP traffic by the client. The client repeats this step as deemed necessary to keep the NAT bindings alive [RFC4787].

5. The client sends unicast feedback from its port c1 to server port P2 where the RTCP feedback message also carries the cookie from Step 3.

6. The server correlates these three RTCP packets based on the cookie value, and remembers the public IP address(es) and port(s) of the client when sending packets back to the client.

   If the client supports RTP/RTCP port muxing, the server needs to remember only one public IP address and port. The state information the server has to keep is reduced but not totally eliminated.

If the server is about to send an RTP and/or RTCP packet to the client but does not know the port mappings since it has not received one or both of the RTCP packets sent in Step 4, it cannot start transmission. Eventually, the client times out and resends the RTCP packets carrying the cookie from its ports c2 and c3. Note that if

the client supports port muxing, the failure probability is substantially reduced. Once the server figures out the port mappings, it keeps that state information until the unicast session is ended.

After the server has established a port mapping for the 2-tuple of cookie and public IP address of the client, it discards RTCP packets carrying the same cookie coming from the same public IP address but from a different public port. The reason is that such packets are likely to be sent by an attacker since there is no good reason for a client to change its port during a short-lived session. Thus, if two different clients sharing the same public IP address accidentally generate the same random cookie and send it to the same port on the server, only the first port mapping will be valid. If neither client is port-muxing, the (total 4) RTCP packets can cross each other resulting in a failure. To minimize the chances for a failure in the client-generated cookie approach, the client should support port muxing and the generated cookies should be truely random [RFC4086].

4.3.2. Implications of NATs

If there are no NAT devices between the server and client, there is no risk of a cookie collision. Thus, it is safe to use this approach.

When there is a NAT device between the server and client, there is a risk of a cookie collision, although it is unlikely if the random cookies are generated properly [RFC4086].

5. Message Formats

The PortMappingRequest message has the following format:

```
 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|   FMT   |       PT      |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      SSRC of Packet Sender                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      SSRC of Media Source                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: FCI field syntax for the PortMappingRequest message

The PortMappingResponse message has the following format:

```
      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|   FMT   |       PT      |          Length               |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                     SSRC of Packet Sender                     |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                     SSRC of Media Source                      |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     :                             Cookie                            :
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: FCI field syntax for the PortMappingResponse message

Editor's note: We will finalize the message formats in a later version.

6. Security Considerations

TBC.

7. IANA Considerations

TBC.

8. Contributors and Acknowledgments

TBC.

9. References

9.1. 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.

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

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R.

Hakenberg, "RTP Retransmission Payload Format", RFC 4588, July 2006.

[I-D.ietf-avt-rtcpssm]
Ott, J. and J. Chesterfield, "RTCP Extensions for Single-Source Multicast Sessions with Unicast Feedback", draft-ietf-avt-rtcpssm-19 (work in progress), November 2009.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[I-D.ietf-avt-rtp-and-rtcp-mux]
Perkins, C. and M. Westerlund, "Multiplexing RTP Data and Control Packets on a Single Port", draft-ietf-avt-rtp-and-rtcp-mux-07 (work in progress), August 2007.

9.2. Informative References

[I-D.ietf-avt-rapid-acquisition-for-rtp]
Steeg, B., Begen, A., Caenegem, T., and Z. Vax, "Unicast-Based Rapid Acquisition of Multicast RTP Sessions", draft-ietf-avt-rapid-acquisition-for-rtp-07 (work in progress), February 2010.

[I-D.ietf-behave-nat-behavior-discovery]
MacDonald, D. and B. Lowekamp, "NAT Behavior Discovery Using STUN", draft-ietf-behave-nat-behavior-discovery-08 (work in progress), September 2009.

[RFC4787] Audet, F. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

[RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

Authors' Addresses

Ali Begen
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Email: abegen@cisco.com

Bill VerSteeg
Cisco
5030 Sugarloaf Parkway
Lawrenceville, GA 30044
USA

Email: billvs@cisco.com