AVT
Internet-Draft
Intended status: Standards Track
Expires: July 9, 2011

A. Begen
D. Wing
Cisco
T. VanCaenegem
Alcatel-Lucent

January 5, 2011

Port Mapping Between Unicast and Multicast RTP Sessions
draft-ietf-avt-ports-for-ucast-mcast-rtp-11

Abstract

This document presents a port mapping solution that allows RTP receivers to choose their own ports for an auxiliary unicast session in RTP applications using both unicast and multicast services. The solution provides protection against denial-of-service or packet amplification attacks that could be used to cause one or more RTP packets to be sent to a victim client.

Status of this Memo

Copyright Notice

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 needs to choose its ports for RTP and RTCP since these ports are local resources and only the receiving end can determine which ports are available to use. In addition, Network Address Port Translators (NAPT - hereafter simply called NAT) devices are commonly deployed in networks, thus, static port assignments cannot be used. The receiving end 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 desirable in delay-sensitive real-time applications. Furthermore, the Offer/Answer Model may be burdensome for the endpoints that are concurrently running a large number of unicast sessions with other endpoints.

In this specification, we consider an RTP application that uses one or more unicast and multicast RTP sessions together. 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 NACK feedback to a local repair server (also functioning as a unicast RTCP feedback target) [RFC5760] asking for a retransmission of the packets it is missing, and the local repair server sends the retransmission packets 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 burst packets 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 a solution that we refer to as Port Mapping. This solution allows receivers to choose their desired UDP ports for RTP and RTCP in every unicast

session when they are running RTP applications using both unicast and multicast services, and offer/answer exchange is not available. The solution includes a Token-based protection mechanism against denial-of-service (DoS) or packet amplification attacks that could be used to cause one or more RTP packets to be sent to a victim client. This solution is not applicable in cases where TCP is used as the transport protocol in the unicast sessions. For such scenarios, refer to [RFC4145].

```
        -----------
       |  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. This terminology also reflects the fact that when port mapping is used, the RTP packets can only flow in one direction (from the server to the client) in the unicast sessions.

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. Token-Based Port Mapping

Token-based Port Mapping consists of two steps: (i) Token request and retrieval, and (ii) unicast session establishment.

Once a Token is retrieved from a particular server, it can be used for all the unicast sessions the client will be running with this particular server till the Token expires. By default, Tokens are server specific. However, the client can use the same Token to communicate with different servers if these servers are provided with the same secret key and algorithm used to generate the Token and are at least loosely clock-synchronized.

The Token is essentially an opaque encapsulation that is based on client's IP address (as seen by the server). When a Token request is received, the server creates a Token for this particular client, and sends it back to the client. The Token becomes invalid if client's IP address (as seen by the server) changes (note that the client cannot necessarily detect this in a timely manner) or when the server expires the Token. In these cases, the client has to request a new Token.

As the second step, when the client wants to establish a unicast session, the client includes the Token with its RTCP feedback message. The server validates the Token, making sure that the IP address information matches. This is effective against DoS attacks, e.g., an attacker cannot simply spoof another client's IP address and start a unicast transmission towards random clients. If the validation passes, the unicast session gets established. Otherwise, the server notifies the client that the validation has failed, and in this case, the unicast session will not be established.

Upon successful validation and once the unicast session is established, all the RTP and RTCP rules specified in [RFC3550] and other relevant specifications also apply in this session until it is terminated. During the lifetime of a unicast session, a client might need to send RTCP messages that require authorization. Since such messages require a valid Token for authorization, the client needs to include the Token along with such RTCP messages as explained in detail in later sections of this document.

Below, we first present a motivating scenario for port mapping and then describe the normative behavior and requirements.

3.1. Motivating Scenario

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 [RFC5760]. The retransmission servers also join the multicast session to receive the multicast packets and cache them for a certain time period. When a client detects missing packets in the multicast session, it requests a retransmission from one of the retransmission servers by using an RTCP NACK message [RFC4585]. The retransmission server pulls the requested packet(s) out of the cache and retransmits them to the requesting client [RFC4588].

The RTP and RTCP flows pertaining to the scenario described above are sketched in Figure 2. Between the client and server, we assume there exists at least one NAT device [RFC4787] (If there is no NAT devices between the server and client, the method still works the same fashion). The multicast and unicast sessions are clearly identified with their individual RTP and RTCP flows and port numbers.

```
     --------------                                 ---     ----------
    |              |-------------------------------|   |-->|P1        |
    |              |-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-|   |.->|P2        |
    |              |                               |   |   |          |
    | Distribution |      ----------------         |   |   |          |
    |    Source    |     |                |        |   |   |          |
    |              |---->|P1              |        |   |   |          |
    |              |.-.->|P2              |        |   |   |          |
    |              |     |                |        |   |   |          |
     --------------      |              P3|<.=.=.=.|   |=.=|*c0       |
                         |              P3|<~~~~~~~|   |~~~|*c1       |
    MULTICAST RTP        |                |        |   |   |          |
    SESSION with         |                |        | N |   |          |
    UNICAST FEEDBACK     |                |        | A |   |          |
                         | Retransmission |        | T |   |  Client  |
    - - - - - - - - - - -| - - - - - - - -| - - - -| - |- -| - - - - -|-
                         |     Server     |        |   |   |          |
                         |                |        |   |   |          |
    PORT MAPPING         |              PT|<~~~~~~~|   |~~>|*cT       |
                         |                |        |   |   |          |
    - - - - - - - - - - -| - - - - - - - -| - - - -| - |- -| - - - - -|-
                         |                |        |   |   |          |
    AUXILIARY UNICAST    |                |        |   |   |          |
    RTP SESSION          |                |        |   |   |          |
                         |              P3|........|   |..>|*c1       |
                         |              P3|=.=.=.=.|   |=.>|*c1       |
                         |              P4|<.=.=.=.|   |=.=|*c2       |
                         |                |        |   |   |          |
                          ----------------          ---     ----------


    -------> 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 server

In Figure 2, we have the following multicast and unicast ports:

o Ports P1 and P2 denote the destination RTP and RTCP ports in the multicast session, respectively. The clients listen to these ports to receive the multicast RTP and RTCP packets. Ports P1 and P2 are defined declaratively.

- o Port P3 denotes the RTCP port on the feedback target running on the retransmission server to collect any RTCP packet sent by the clients including feedback messages, and RTCP receiver and extended reports. This is also the port that the retransmission server uses to send the RTP packets and RTCP sender reports in the unicast session. Port P3 is defined declaratively.

- o Port P4 denotes the RTCP port on the retransmission server used to collect the RTCP receiver and extended reports for the unicast session. Port P4 is defined declaratively.

- o Ports *c0, *c1 and *c2 are chosen by the client. *c0 denotes the port on the client used to send the RTCP reports for the multicast session. *c1 denotes the port on the client used to send the unicast RTCP feedback messages in the multicast session and to receive the RTP packets and RTCP sender reports in the unicast session. *c2 denotes the port on the client used to send the RTCP receiver and extended reports in the unicast session. Ports c0, c1 and c2 could be the same port or different ports. There are two advantages of using the same port for both c0 and c1:

  1. Some NATs only keep bindings active when a packet goes from the inside to the outside of the NAT (See REQ-6 of Section 4.3 of [RFC4787]). When the gap between the packets sent from the client to the server is long, this can exceed that timeout. If c0=c1, the occasional (periodic) RTCP receiver reports sent from port c0 (for the multicast session's RTCP port P3) will ensure the NAT does not time out the public port associated with the incoming unicast traffic to port c1.

  2. Having c0=c1 conserves NAT port bindings.

- o Ports PT and *cT denote the ports through which the Token request and retrieval occur at the server and client sides, respectively. Port PT is declared on a per unicast session basis, although the same port could be used for two or more unicast sessions sourced by the server. A Token once requested and retrieved by a client from port PT remains valid until its expiration time.

We assume that the information declaratively defined is available as part of the session description information and is provided to the clients. The Session Description Protocol (SDP) [RFC4566] and other session description methods can be used for this purpose.

3.2. Normative Behavior and Requirements

In this section, we describe the normative behavior and requirements. To simplify the presentation, we refer to the port numbers described

in the example presented in Figure 2. However, the behavior and requirements described here are not specific to that particular example, and can be applied to any scenario where analogous ports can be identified.

First of all, a client compliant with this specification MUST be able to include a Token with any type of RTCP message (as described below) when it is needed. That is, clients MUST NOT implement this specification with only a specific use case in mind.

Second, the solution provided in this specification is not applicable in cases where there is RTP traffic flowing from the client to the server in the unicast session. In other words, the direction of RTP traffic MUST be only from the server to the client in the unicast session. If the client wants to send RTP traffic back to the server, the regular session establishment methods such as [RFC3264] need to be used.

The following steps summarize the Token-based solution:

1. The client ascertains server address and port numbers (P3, P4 and PT) from the session description. Port P4 MUST be different from port P3. Port PT MAY be equal to port P3.

2. The client selects its local port numbers (*c0, *c1, *c2 and *cT). It is strongly RECOMMENDED that the client uses the same port for c0 and c1. Port cT MAY be equal to ports c0 and c1.

3. If the client does not have a Token (or the existing Token has expired):

   A. The client first sends a Port Mapping Request message (Section 4.1) to port PT. This message is sent from port *cT on the client side. The server learns client's IP address from the received message. The client can send this message anytime it wants (e.g., during initialization), and does not normally ever need to re-send this message (See Section 6).

   B. The server generates an opaque encapsulation (i.e., the Token) based on certain information including the client's IP address.

   C. The server sends the Token back to the client using a Port Mapping Response message (Section 4.2). This message MUST be sent from port PT to port cT.

4. The client needs to provide the Token to the server using a Token Verification Request message (Section 4.3), whenever the client sends an RTCP feedback message for triggering or controlling a unicast session (See Section 4.3.1). If the Token is invalid or missing, the server sends a Token Verification Failure message (Section 4.4) to the client.

   Note that the unicast session is only established after the server has received a feedback message (along with a valid Token) from the client for which it needs to react by sending unicast data. Until a unicast session is established, neither the server nor the client needs to send RTCP reports for the unicast session.

5. Normal flows ensue as shown in Figure 2. Note that in the unicast session, traffic from the server to the client (i.e., both the RTP and RTCP packets sent from port P3 to port c1) MUST be multiplexed on the (same) port c1. If the client uses the same port for both c0 and c1, the RTCP reports sent for the multicast session keep the P3->c1(=c0) binding alive. If the client uses different ports for c0 and c1, the client needs to periodically send an explicit keep-alive message [I-D.ietf-avt-app-rtp-keepalive] to keep the P3->c1 binding alive during the lifetime of the unicast session if the time between unicast feedback messages (from c1 to P3) is likely to exceed the NAT's timeout value (For the NAT timeout value requirements, see [RFC4787]).

A unicast session on a particular receive port c1 can last as long as the associated multicast session lasts. However, a client cannot keep using the same receive port c1 for different subsequent unicast sessions since there could be packet leakage when switching from one unicast session to another unless each received unicast stream has its own distinct Synchronization Source (SSRC) identifier to allow the client to filter out the undesired packets. Unless this is guaranteed (which is not often easy), a client SHOULD use separate receive ports for subsequent unicast sessions. After a sufficient time, a previously used receive port could be used again.

4. Message Formats

This section defines the formats of the RTCP messages that are exchanged between a server and a client for the purpose of port mapping. A new RTCP control packet type is introduced and four port mapping messages using this control packet are defined:

1. Port Mapping Request

2. Port Mapping Response

3. Token Verification Request

4. Token Verification Failure

Each message has a fixed-length field for version (V), padding (P), sub-message type (SMT), packet type (PT), length and SSRC of packet sender. Messages have other fields as defined below. In all messages defined in this section, the PT field is set to TOKEN. Individual messages are identified by the SMT field. The length field indicates the message size in 32-bit words minus one, including the header and any padding. This definition is in line with the definition of the length field used in RTCP sender and receiver reports. In all messages, any Reserved field SHALL be set to zero and ignored.

Following the rules specified in [RFC3550], all integer fields in the messages defined below are carried in network-byte order, that is, most significant byte (octet) first, also known as big-endian. Unless otherwise stated, numeric constants are in decimal (base 10).

Note that RTCP is not a timely or reliable protocol. The RTCP packets might get lost or re-ordered in the network. When sending a new Port Mapping Request message, the scheduling rules that apply to sending initial RTCP messages [RFC4585] apply. When a client sends a Port Mapping Request or Token Verification Request message but it does not receive a response back from the server (either a Port Mapping Response or Token Verification Failure message), it MAY resend its request by following the timer rules defined for RTCP feedback messages in Section 3.5 of [RFC4585] as a good practice. When sending an RTCP (feedback) message bundled with a Token Verification Request message, the timer rules of [RFC4585] apply as usual.

4.1. Port Mapping Request

The Port Mapping Request message is identified by SMT=1. This message is transmitted by the client to a dedicated server port (and

possibly a dedicated address) to request a Token. In the Port Mapping Request message, the packet sender SSRC is set to the client's SSRC, which is chosen randomly by the client. The packet format has the structure depicted in Figure 3.

```
    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|  SMT=1  |    PT=TOKEN   |         Length=3              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      SSRC of Packet Sender                    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                             Random                            |
   |                             Nonce                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Packet format for the Port Mapping Request message

o Random Nonce (64 bits): Field that contains a random value generated by the client following the procedures of [RFC4086]. This nonce is taken into account by the server when generating a Token for the client to enable better security for clients that share the same IP address (Such clients need to produce a random value guaranteed to be temporally and globally unique). If the same Port Mapping Request message is transmitted multiple times for redundancy reasons, the random nonce value MUST remain the same in these duplicated messages. However, the client MUST generate a new random nonce for every new Port Mapping Request message.

4.2. Port Mapping Response

The Port Mapping Response message is identified by SMT=2. This message is sent by the server and delivers the Token to the client as a response to the Port Mapping Request message. In the Port Mapping Response message, the packet sender SSRC is set to the server's SSRC. The packet format has the structure depicted in Figure 4.

```
      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|  SMT=2  |    PT=TOKEN   |           Length              |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                      SSRC of Packet Sender                    |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                    SSRC of Requesting Client                  |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                           Associated                          |
     |                             Nonce                             |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     :                          Token Element                        :
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                            Absolute                           |
     |                         Expiration Time                       |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                     Relative Expiration Time                  |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     :                       Packet Types Element                    :
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: Packet format for the Port Mapping Response message

o SSRC of Requesting Client (32 bits): Field that contains the SSRC of the client who sent the request.

o Associated Nonce (64 bits): Field that contains the nonce received in the Port Mapping Request message and used in Token construction.

o Token Element (Variable size): Element that is used to carry the Token generated by the server. This element is a 32-bit aligned Length-Value element. The Length field, which is 8 bits, indicates the length (in octets) of the Value field that follows the Length field. This length does not include any padding that is required for alignment. The Value field carries the Token (or more accurately, the output of the encoding process on the server). If the Token element does not fall on a 32-bit boundary, the last word MUST be padded to the boundary using further bits set to zero.

o Absolute Expiration Time (64 bits): Field that contains the absolute expiration time of the Token. The absolute expiration time is expressed as a Network Time Protocol (NTP) timestamp value in seconds since year 1900 [RFC5905]. The client does not need to use this element directly, thus, does not need to synchronize its clock with the server. However, the client needs to send this

element back to the server along with the associated nonce in the Token Verification Request message, thus, needs to keep it associated with the Token.

- Relative Expiration Time (32 bits): Field that contains the relative expiration time of the Token. The relative expiration time is expressed in seconds from the time the Token was generated. Whenever a server decides to not grant a Token to a requesting client, the relative expiration time will be set to zero (and hence, the accompanying Token will be invalid).

  The server conveys the relative expiration time in the clear to the client to allow the client to request a new Token well before the expiration time.

- Packet Types Element (Variable size): Element that is used to signal which RTCP packet types require Token-based authentication. This element is a 32-bit aligned Length-Value element. The Length field, which is 8 bits, indicates the length (in octets) of the Value field that follows the Length field. This length does not include any padding that is required for alignment. The Value field carries zero or more 8-bit sub-fields each carrying an RTCP packet type. If the Packet Types element does not fall on a 32-bit boundary, the last word MUST be padded to the boundary using further bits set to zero. An example Packet Types element is shown in Figure 5.

  A server MAY change its policy on which RTCP packet types would require Token-based authentication based on observations, configuration or other policies. However, upon such a change the server SHALL NOT send a new Port Mapping Response message to the clients who requested a Token earlier. A client learns about this change when and if it gets a Token Verification Failure message.

```
     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
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |    Length=4   |      205      |      206      |      203      |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |      204      |                  Padding                      |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: Example Packet Types element

4.3. Token Verification Request

The Token Verification Request message is identified by SMT=3. This message contains the Token and accompanies any RTCP message that would trigger a new or control an existing unicast session. For a list of such messages, see Section 4.3.1.

In the Token Verification Request message, the packet sender SSRC is set to the client's SSRC. The client MUST NOT send a Token Verification Request message with a Token that has expired. The packet format has the structure depicted in Figure 6.

```
    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|  SMT=3  |    PT=TOKEN   |           Length              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     SSRC of Packet Sender                     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          Associated                           |
   |                            Nonce                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   :                        Token Element                          :
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      Associated Absolute                      |
   |                        Expiration Time                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: Packet format for the Token Verification Request message

- o Associated Nonce (64 bits): Field that contains the nonce associated with the Token above.

- o Token Element (Variable size): Token element that was previously received in the Port Mapping Response message.

- o Associated Absolute Expiration Time (64 bits): Field that contains the absolute expiration time associated with the Token above.

4.3.1. Where to Include Token

This section provides guidelines about which RTCP packet types would need to be accompanied by a Token Verification Request message. However, since a server might determine in real time that other RTCP messages also need to be authenticated by a Token, a client MUST act according to the up-to-date list provided to the client in the Port Mapping Response message (in the Packet Types element). Clients need

to support using Token-based authentication with any necessary RTCP message (See Section 3.2).

As a general rule, when the Token capability is declared in the session description, the RTCP messages that trigger transmission of RTP packets in a port-mapped unicast session are REQUIRED to be authenticated by using a Token. Such messages include but are not limited to:

o NACK messages [RFC4585]

o RAMS-R messages [I-D.ietf-avt-rapid-acquisition-for-rtp]

Additionally, some RTCP messages might directly or indirectly control an existing unicast session associated with a multicast session. Unless another authentication method as described in their respective specifications is used, such RTCP messages might also need to be authenticated by using a Token. Examples are:

o BYE messages [RFC3550]

o RAMS-T messages [I-D.ietf-avt-rapid-acquisition-for-rtp]

o CCM messages [RFC5104]

Note that even if a packet type is listed to require Token-based authentication, it does not need to be authenticated when it does not control the unicast session. For example, if BYE (203) is listed in the Port Mapping Response message as one of the packet types that requires authentication, the client does not need to bundle the RTCP BYE message with a Token when it is sending it for the multicast session.

The Token Verification Request message might also be bundled with packets carrying RTCP receiver and/or extended reports. While such packets do not have a strong security impact, a specific application might desire to have a more controlled reporting scheme from the clients. In this case, the server lists the packet types for the receiver (201) and/or extended reports (207) in the Port Mapping Response message.

4.4. Token Verification Failure

The Token Verification Failure message is identified by SMT=4. This message is sent by the server and notifies the client that the Token was invalid or that the client did not include a Token Verification Request message in the RTCP packet although it was supposed to. In the Token Verification Failure message, the packet sender SSRC is set

to the server's SSRC. The packet format has the structure depicted in Figure 7.

```
    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|  SMT=4  |    PT=TOKEN   |         Length=5              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      SSRC of Packet Sender                    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    SSRC of Requesting Client                  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   Failed PT   |   FMT   |             Reserved                |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           Associated                          |
   |                             Nonce                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: Packet format for the Token Verification Failure message

o SSRC of Requesting Client (32 bits): Field that contains the SSRC of the client who sent the verification request.

o Failed PT (8 bits): Field that indicates the type of the RTCP packet that caused this failure message.

o FMT (5 bits): Field that indicates the feedback message type (FMT) value of the RTCP packet that caused this failure. Together with the field above, the client can infer which RTCP message it has previously sent caused this failure message to be sent by the server. For example, if the client did not include a valid Token with an RTCP NACK message, the Failed PT field will indicate 205 (RTPFB) and the FMT field will indicate 1 (Generic NACK). If the RTCP message did not have an associated FMT value (such as an RTCP BYE message), the FMT field SHALL be set to zero.

o Associated Nonce (64 bits): Field that contains the nonce received in the Token Verification Request message. If there was no Token Verification Request message included by the client, this field is set to 0.

5. Procedures for Token Construction

The Token encoding is known to the server but opaque to the client. Implementations MUST encode the following information into the Token as a minimum, in order to provide adequate security:

- o Client's IP address as seen by the server (32/128 bits for IPv4/ IPv6 addresses)

- o The nonce generated and inserted in the Port Mapping Request message by the client (64 bits)

- o The absolute expiration time chosen by the server indicated as an NTP timestamp value in seconds since year 1900 [RFC5905] (64 bits, to protect against replay attacks)

An example way for constructing Tokens is to perform HMAC-SHA1 [RFC2104] on the concatenated values of the information listed above. The HMAC key needs to be at least 160 bits long, and generated using a cryptographically secure random source [RFC4086]. While HMAC-SHA1 is the RECOMMENDED procedure, implementations might adopt different approaches.

In addition to the information listed above, implementations are encouraged to encode whatever additional information is deemed necessary or useful. For example, key rollover is simplified by encoding a key-id into the Token. As another example, a cluster of anycast servers could find advantage by encoding a server identifier into the Token. As another example, while HMAC-SHA1 provides a level of security that is widely regarded as being more than sufficient for providing message authentication and it is secure against all known cryptanalytic attacks that use computational resources that are currently economically feasible, if HMAC-SHA1 has been compromised, a replacement HMAC algorithm could be used instead (e.g., HMAC-SHA256).

To protect from offline attacks, the server SHOULD occasionally choose a new HMAC key. To ease implementation, a key-id can be assigned to each HMAC key. This can be encoded as simply as one bit (where the new key is X (e.g., 1) and the old key is the inverted value of X (e.g., 0)), or if several keys are supported at once could be encoded into several bits. As the encoding of the Token is entirely private to the server and opaque to the clients, any encoding can be used. By encoding the key-id into the Token element, the server can reject an old key without bothering to do HMAC validation (saving CPU cycles). The key-id can be encoded into the Value field of the Token element by simply concatenating the (plaintext) key-id with the hashed information (i.e., the Token itself).

For example, the Value field in the Token element can be computed as:

```
key-id || hash-alg (client-ip | nonce | abs-expiration)
```

During Token construction, the expiration time has to be chosen carefully based on the intended service duration. Tokens that are valid for an unnecessarily long period of time (e.g., several hours) might impose security risks. Depending on the application and use cases, a reasonable value needs to be chosen by the server. Note that using shorter lifetimes requires the clients to acquire Tokens more frequently. However, since a client can acquire a new Token well before it will need to use it, the client will not necessarily be penalized for the acquisition delay.

Finally, be aware that NTP timestamps will wrap around in year 2036 and implementations might need to handle this eventually. Refer to Section 6 of [RFC5905] for further details.

6. Validating Tokens

Upon receipt of an RTCP feedback message along with the Token Verification Request message that contains a Token, nonce and absolute expiration time, the server MUST validate the Token.

The server first applies its own procedure for constructing the Tokens by using client's IP address from the received Token Verification Request message, and the nonce and absolute expiration time values reported in the received Token Verification Request message. The server then compares the resulting output with the Token sent by the client in the Token Verification Request message. If they match and the absolute expiration time has not passed yet, the server declares that the Token is valid.

Note that if the client's IP address changes, the Token will not validate. Similarly, if the client inserts an incorrect nonce or absolute expiration time value in the Token Verification Request message, validation will fail. It is also possible that the server wants to expire the Token prematurely. In these cases, the server MUST reply back to the client with a Token Verification Failure message (that goes from port P3 on the server to port c1 on the client).

In addition to the Token Verification Failure message, it is RECOMMENDED that applications define an application-specific error response to be sent by the server when the server detects that the Token is invalid. For applications using [I-D.ietf-avt-rapid-acquisition-for-rtp], this document defines a new 4xx-level response code in the RAMS Response Code Space Registry. A client that received a Token Verification Failure message can request a new Token from the server.

7. SDP Signaling

7.1. The portmapping and portmapping-req Attributes

The 'portmapping' attribute is used declaratively without any parameters in a multicast block in an SDP description. It provides a hint information to the SDP recipient that one or more unicast sessions associated with this multicast session require the use of Tokens for certain RTCP messages. The 'portmapping-req' attribute is also used declaratively but in a unicast block. It indicates the port and optionally the address for obtaining a Token.

In an SDP description, there could be one or more multicast sessions described, each associated with zero or more unicast sessions. The recipient of the SDP description infers the association(s) between the multicast and unicast sessions through the "a=group" line(s).

The presence of the 'portmapping' and 'portmapping-req' attribute pair indicates that (i) a Token MUST be included in certain feedback messages sent to the server triggering or controlling a unicast session (See Section 4.3.1), and (ii) the client MUST receive the unicast session's RTP and RTCP packets from the server on the port from which it sent the RTCP message triggering or controlling the unicast session.

> Note: This does not imply that Token Verification Request messages need to be sent in the unicast session. Token Verification Request messages accompany RTCP messages that trigger or control this unicast session, and are sent either in the multicast session or the unicast session, depending on the RTCP message (See Section 4.3.1).

7.1.1. ABNF Definition of portmapping

The formal description of the 'portmapping' attribute is defined by the following ABNF [RFC5234] syntax:

```
                  portmapping-attr = "a=portmapping"
```

This attribute SHALL be used as a media-level attribute in a multicast block in SDP descriptions.

7.1.2. ABNF Definition of portmapping-req

The formal description of the 'portmapping-req' attribute is defined by the following ABNF [RFC5234] syntax:

```
 portmapping-req-attr = "a=portmapping-req" [":" port [SP nettype SP
```

```
                              addrtype SP connection-address]] CRLF
```

Here, 'port', 'nettype', 'addrtype' and 'connection-address' are defined as specified in Section 9 of [RFC4566].

The 'portmapping-req' attribute SHALL be used as a media-level attribute in a unicast block in SDP descriptions.

In the optional address value, only unicast addresses SHOULD be used unless one wants to use a multicast address after evaluating the additional security risks such as non-legit servers generating fake Tokens. If the address is not specified, the (source) address in the "c" line corresponding to the unicast media stream is implied.

7.1.3. Offer/Answer Model Considerations

When using the attributes defined above in SDP offer/answer exchanges [RFC3264], the following considerations apply. When an offerer sends an answerer an offer of an SDP description making use of the Token approach described in this specification, the 'portmapping' and 'portmapping-req' attributes are included declaratively. There will not be offer/answer exchanges between the answerer and the actual server providing the unicast service(s).

When the answerer supports the Token approach, it MUST echo in its answer back to the offerer the 'portmapping' and 'portmapping-req' attributes from the offer including the same port number and address (if any) for the 'portmapping-req' attribute. If the answerer does not implement this specification, it follows normal SDP parsing of unknown attributes (they are ignored and are not sent in the answer). This means that the answerer can still join the multicast session, but will not be able to use the unicast service(s) that require the use of Tokens.

7.2. Requirements

The use of SDP for the port mapping solution normatively requires the support for:

- o The SDP grouping framework and flow identification (FID) semantics [RFC5888]

- o The RTP/AVPF profile [RFC4585]

- o Multiplexing RTP and RTCP on a single port on both endpoints in the unicast session [RFC5761]

7.3. Example and Discussion

The declarative SDP describing the scenario given in Figure 2 is 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=Multicast Stream
     c=IN IP4 233.252.0.2/255
     a=source-filter:incl IN IP4 233.252.0.2 198.51.100.1   ; Note 1
     a=rtpmap:98 MP2T/90000
     a=multicast-rtcp:41500                                 ; Note 1
     a=rtcp:42000 IN IP4 192.0.2.1                          ; Note 2
     a=rtcp-fb:98 nack                                      ; Note 2
     a=portmapping                                          ; Note 3
     a=mid:1
     m=video 42000 RTP/AVPF 99                              ; Note 4
     i=Unicast Retransmission Stream
     c=IN IP4 192.0.2.1
     a=sendonly
     a=rtpmap:99 rtx/90000
     a=rtcp-mux                                             ; Note 5
     a=rtcp:42500                                           ; Note 6
     a=fmtp:99 apt=98; rtx-time=5000
     a=portmapping-req:30000                                ; Note 7
     a=mid:2
```

Figure 8: SDP describing an SSM distribution with support for retransmissions from a local server

In this description, we highlight the following notes:

Note 1: 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 (P1). The associated RTCP packets are multicast in the same group to port 41500 (P2).

Note 2: A retransmission server including feedback target functionality with an IP address of 192.0.2.1 and port of 42000 (P3) is specified with the 'rtcp' attribute. The feedback functionality is enabled for the RTP stream with payload type 98 through the 'rtcp-fb' attribute [RFC4585].

Note 3: The "a=portmapping" line together with the "a=group" line hints that a Token needs to be retrieved first before the unicast session can be established.

Note 4: The port specified in the second "m" line (for the unicast stream) does not mean anything in this scenario as the client does not send any RTP traffic back to the server.

Note 5: The server multiplexes RTP and RTCP packets on the same port (c1 in Figure 2).

Note 6: The server uses port 42500 (P4) for the unicast session.

Note 7: The "a=portmapping-req" line indicates that the Port Mapping Request message needs to be sent to port 30000 (PT). Since there is no address indicated in this line, the client needs to send the Token request to the address specified in the "c" line.

8. Address Pooling NATs

Large-scale NAT devices have a pool of public IPv4 addresses and map internal hosts to one of those public IPv4 addresses. As long as an internal host maintains an active mapping in the NAT, the same IPv4 address is assigned to new connections. However, once all of the host's mappings have been deleted (e.g., because of timeout), it is possible that a new connection from that same host will be assigned a different IPv4 address from the pool. When that occurs, the Token will be considered invalid by the server, causing an additional round trip for the client to acquire a fresh Token.

Any traffic from the host which traverses the NAT will prevent this problem. As the host is sending RTCP receiver reports at least every 5 seconds (Section 6.2 of [RFC3550]) for the multicast session it is receiving, those RTCP messages will be sufficient to prevent this problem.

9. Security Considerations

9.1. Tokens

The Token, which is generated based on a client's IP address and expiration date, provides protection against off-path denial-of-service (DoS) attacks. An attacker using a certain IP address cannot cause one or more RTP packets to be sent to a victim client who has a different IP address. However, if the attacker acquires a valid Token for a victim and can spoof the victim's source address, this approach becomes vulnerable to replay attacks. This is especially easy if the attacker and victim are behind a large-scale NAT and share the same IP address.

Multicast is deployed on managed networks - not the Internet. These managed networks will choose to enable network ingress filtering [RFC2827] or not. If ingress filtering is enabled on a network, an attacker cannot spoof a victim's IP address to use a Token to initiate an attack against a victim. However, if ingress filtering is not enabled on a network, an attacker could obtain a Token and spoof the victim's address, causing traffic to flood the victim. On such a network, the server can reduce the time period for such an attack by expiring a Token in a short period of time. In the extreme case, the server can expire the Token in such a short period of time, such that the client will have to acquire a new Token immediately before using it in a Token Verification Request message. One should, however, note that such a behavior might have an adverse effect on the delay in establishing or controlling a unicast session.

RTCP messages could be subject to on-path or man-in-the-middle attacks. For example, an attacker can modify a value in one or more fields in the Port Mapping Response or the Token Verification Request message that are used in Token construction. This will result in Token validation failure. Consequently, the client ends up asking the server to generate a new Token. The resulting delay and extra processing on the server are undesirable.

Alternatively, the attacker can modify a value in a field that is not used in Token construction. For example, the attacker can reduce the value in the Relative Expiration Time field in the Port Mapping Response message from two hours to two minutes. While the Token will still validate, this attack will result in more frequent requests to the server for a new Token. Oppositely, the attacker can increase the value in the Relative Expiration Time field, and make the client think the Token will be valid for a longer time. This attack can be only detected by monitoring the activity on the server. Note that using the relative expiration time in Token construction does not necessarily make this attack easier to detect since the attacker

might revert the modified value back to its original value in the Token Verification Request message. This allows the Token to still validate on the server. In this case, the attack is still only detectable by monitoring the server activity.

If there is a risk or concern for on-path or man-in-the-middle attacks, RTCP messages SHOULD be protected by Secure RTCP (SRTCP) [RFC3711].

9.2. The portmapping and portmapping-req Attributes

The 'portmapping' attribute does not introduce a significant security risk. It is used for informative purposes as a hint in a multicast block in SDP descriptions and without the use of the 'portmapping-req' attribute in an associated unicast block, its existence does not mean anything and SHALL be ignored. However, if the 'portmapping-req' attribute exists in the unicast block but the 'portmapping' attribute is missing in the associated multicast block (which is inferred through the "a=group" line), the clients MUST still behave as if the 'portmapping' attribute existed in the multicast block.

The 'portmapping-req' attribute is not believed to introduce any significant security risk to multimedia applications. A malevolent third party could use this attribute to redirect the Port Mapping Request messages by altering the port number or cause the unicast session establishment to fail by removing it from the SDP description. But, this requires intercepting and rewriting the packets carrying the SDP description; and if an interceptor can do that, many more attacks are possible, including a wholesale change of the addresses and port numbers at which the media will be sent.

In order to avoid attacks of this sort, the SDP description needs to be integrity protected and provided with source authentication. This can, for example, be achieved on an end-to-end basis using S/MIME [RFC5652] when SDP is used in a signaling packet using MIME types (application/sdp). Alternatively, HTTPS [RFC2818] or the authentication method in the Session Announcement Protocol (SAP) [RFC2974] could be used as well.

10. IANA Considerations

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

Ali Begen
abegen@cisco.com

Note to the RFC Editor: In the following, please replace "XXXX" with the number of this document prior to publication as an RFC.

10.1. Registration of SDP Attributes

This document registers two new attribute names in SDP.

```
     SDP Attribute ("att-field"):
     Attribute name:     portmapping
     Long form:          Hint for the use of port mapping
     Type of name:       att-field
     Type of attribute:  Media level
     Subject to charset: No
     Purpose:            See this document
     Reference:          [RFCXXXX]
     Values:             See this document


     SDP Attribute ("att-field"):
     Attribute name:     portmapping-req
     Long form:          Port and address for requesting Token
     Type of name:       att-field
     Type of attribute:  Media level
     Subject to charset: No
     Purpose:            See this document
     Reference:          [RFCXXXX]
     Values:             See this document
```

10.2. Registration of RTCP Control Packet Types

In accordance with Section 15 of [RFC3550], this specification adds the following value to the RTCP Control Packet types sub-registry of the Real-Time Transport Protocol (RTP) Parameters registry:

Value	Abbrev.	Name	Reference
TBD	TOKEN	Port Mapping	[RFCXXXX]

Note to the IANA: Please assign the next available value, which is currently 210.

10.3. SMT Values for TOKEN Packet Type Registry

This document creates a new sub-registry for the sub-message type (SMT) values to be used with the TOKEN packet type. The registry is called the SMT Values for TOKEN Packet Type Registry. This registry is to be managed by the IANA according to the IETF Review policy of [RFC5226].

The length of the SMT field is five bits, allowing 32 values. The registry is initialized with the following entries:

Value	Name	Reference
0	Reserved	[RFCXXXX]
1	Port Mapping Request	[RFCXXXX]
2	Port Mapping Response	[RFCXXXX]
3	Token Verification Request	[RFCXXXX]
4	Token Verification Failure	[RFCXXXX]
5-30	Unassigned	IETF Review
31	Reserved	[RFCXXXX]

The SMT values 0 and 31 are reserved for future use.

10.4. RAMS Response Code Space Registry

This document adds the following entry to the RAMS Response Code Space Registry.

Code	Description	Reference
405	Invalid Token	[RFCXXXX]

This response code is used when the Token included by the RTP_Rx in the RAMS-R message is invalid.

11. Acknowledgments

The approach presented in this document came out after discussions with various individuals in the AVT and MMUSIC WGs, and the breakout session held in the Anaheim meeting. We thank each of these individuals, in particular to Magnus Westerlund and Colin Perkins.

12. References

12.1. 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.

[RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, July 2006.

[RFC5760] Ott, J., Chesterfield, J., and E. Schooler, "RTP Control Protocol (RTCP) Extensions for Single-Source Multicast Sessions with Unicast Feedback", RFC 5760, February 2010.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

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

[RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC5888] Camarillo, G. and H. Schulzrinne, "The Session Description Protocol (SDP) Grouping Framework", RFC 5888, June 2010.

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

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

12.2. Informative References

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

[RFC4145] Yon, D. and G. Camarillo, "TCP-Based Media Transport in the Session Description Protocol (SDP)", RFC 4145, September 2005.

[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-17 (work in progress), November 2010.

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

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

[I-D.ietf-avt-app-rtp-keepalive] Marjou, X. and A. Sollaud, "Application Mechanism for keeping alive the Network Address Translator (NAT) mappings associated to RTP flows.", draft-ietf-avt-app-rtp-keepalive-09 (work in progress), September 2010.

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, May 2000.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5104] Wenger, S., Chandra, U., Westerlund, M., and B. Burman, "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)", RFC 5104, February 2008.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, September 2009.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[RFC2974] Handley, M., Perkins, C., and E. Whelan, "Session Announcement Protocol", RFC 2974, October 2000.

Authors' Addresses

Ali Begen
Cisco
181 Bay Street
Toronto, ON M5J 2T3
Canada

Email: abegen@cisco.com

Dan Wing
Cisco Systems, Inc.
170 West Tasman Dr.
San Jose, CA 95134
USA

Email: dwing@cisco.com

Tom VanCaenegem
Alcatel-Lucent
Copernicuslaan 50
Antwerpen,   2018
Belgium

Email: Tom.Van_Caenegem@alcatel-lucent.com