AVT
Internet-Draft
Intended status: Standards Track
Expires: June 5, 2011

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

December 2, 2010

Port Mapping Between Unicast and Multicast RTP Sessions
draft-ietf-avt-ports-for-ucast-mcast-rtp-05

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 attacks that could be used to cause one or more RTP packets to be sent to a victim client.

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 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. 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].

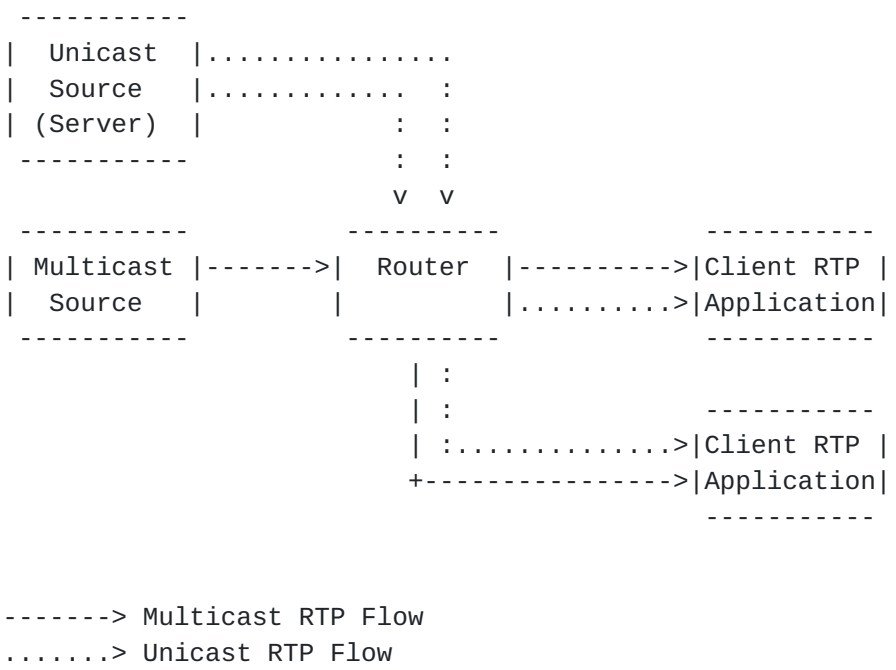

Figure 1: RTP applications simultaneously using both unicast and multicast services

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. Token-Based Port Mapping

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

3.1. Token Request and Retrieval

This first step is required to be completed only once. 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. 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 becomes invalid if client's public IP address changes or when the server expires the Token. In these cases, the client has to request a new Token.

The Token is essentially an opaque encapsulation that is based on client's IP address (as seen by the server). When a request is received, the server creates a Token for this particular client, and sends it back to the client. Later, when the client wants to establish a unicast session, the Token will be validated by the server, 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.

3.2. Unicast Session Establishment

The second step is the unicast session establishment. We illustrate this step with an example. First, we describe the motivation scenario and then define the normative behavior and requirements.

3.2.1. 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, there can be one or more NAT devices [RFC4787].

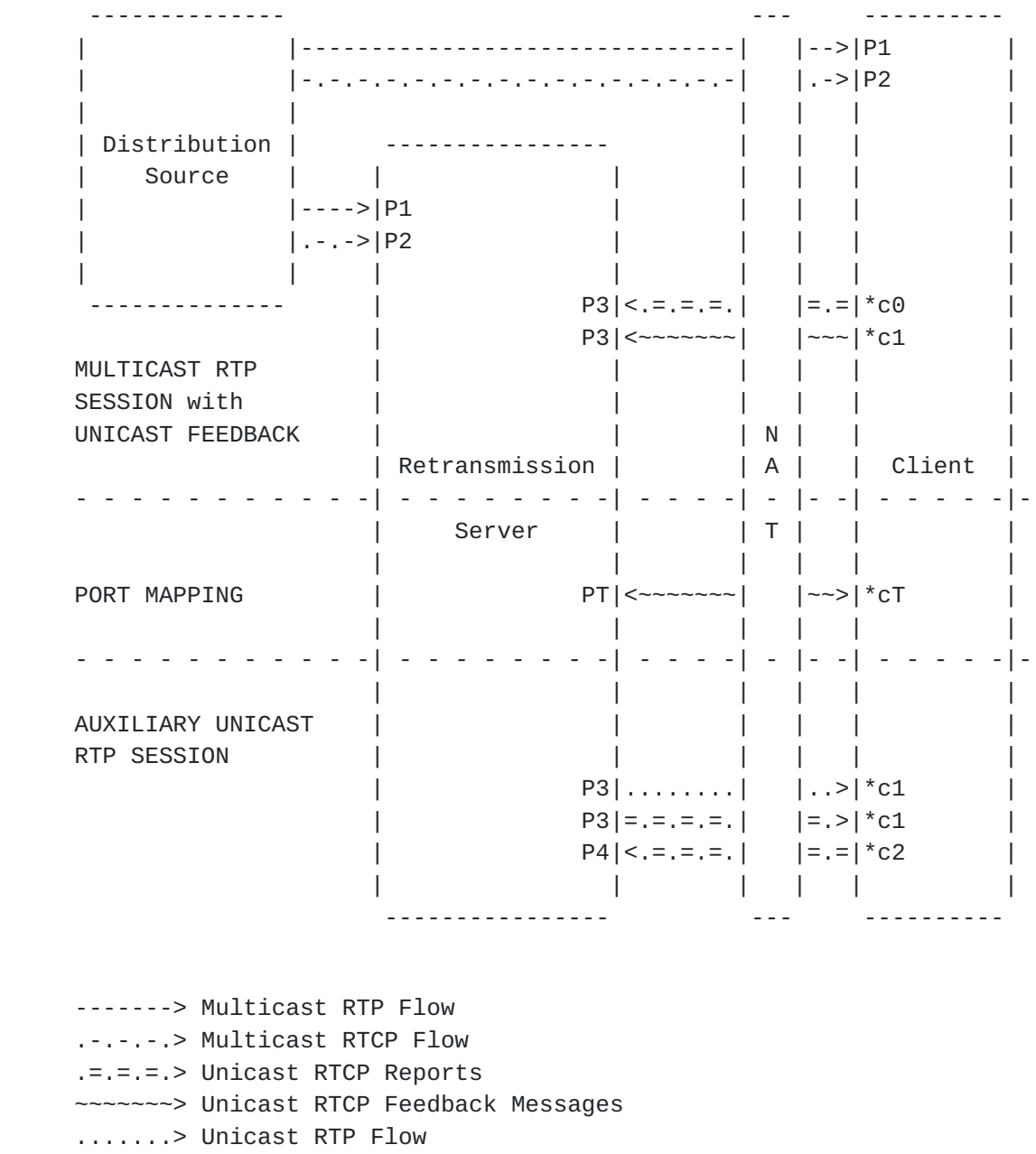

Figure 2: Example scenario showing an SSM distribution with support for retransmissions from a server

3.2.2. Normative Behavior and Requirements

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

- 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.

- 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.

- 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 and MUST be different from port P3.

- 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 MAY be the same port or different ports. However, 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 retransmission requests (or other traffic 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.

  Thus, it is strongly RECOMMENDED that the client uses the same port for c0 and c1.

- 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 its value MAY be the same 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. Port PT MAY be equal to port P3. Port cT MAY also be equal to ports c0 and c1.

In addition to the ports, we use the following notation:

- DS: IP address of the distribution source
- G: Destination multicast address
- S: IP address of the retransmission server
- C: IP address of the client
- C': Public IP address of the client (as seen by the server)

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.

The following steps summarize the Token-based solution:

1. The client ascertains server address (S) and port numbers (P3 and P4) from the session description.
2. The client selects its local port numbers (*c0, *c1 and *c2).
3. If the client does not have a Token (or the existing Token has expired):
    A. The client first sends a message to the server via a new RTCP message, called Port Mapping Request to port PT. This message is sent from port cT on the client side. The server learns client's public IP address (C') 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 client's IP address.
    C. The server sends the Token back to the client using a new RTCP message, called Port Mapping Response. This message MUST be sent from port PT to port cT.

4. The client needs to provide the Token to the server using a new RTCP message, called Token Verification Request, whenever the client sends an RTCP feedback message for triggering or controlling a unicast session (See Section 4.3). 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 unicast session's lifetime is likely to exceed the NAT's timeout value.

4. Message Formats

This section defines the formats of the RTCP transport-layer feedback messages that are exchanged between a server and a client for the purpose of Token-based port mapping. Four RTCP messages are defined:

1. Port Mapping Request

2. Port Mapping Response

3. Token Verification Request

4. Token Verification Failure

These are all payload-independent RTCP feedback messages with a common format defined in Section 6.1 of [RFC4585], also sketched in Figure 3.

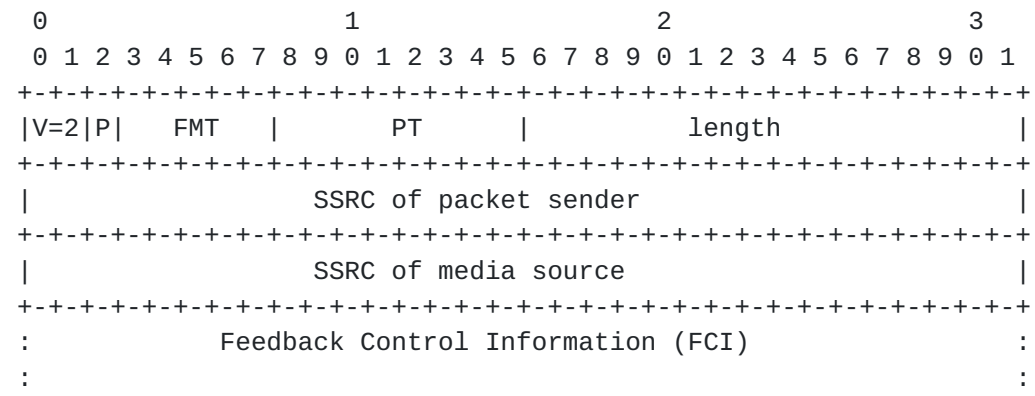

Figure 3: The common packet format for the RTCP feedback messages

Each feedback message has a fixed-length field for version, padding, feedback message type (FMT), packet type (PT), length, SSRC of packet sender, SSRC of media source as well as a variable-length field for feedback control information (FCI).

In the new messages defined in this section, the PT field is set to RTPFB (205) and the FMT field is set to Port Mapping (7). Individual Port Mapping messages are identified by a sub-field called Sub Feedback Message Type (SFMT). 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 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 when it is eligible to do so based on the timer rules defined in [RFC4585].

4.1. Port Mapping Request

The Port Mapping Request message is identified by SFMT=1. This message is a unicast feedback message transmitted by the client to a dedicated server port to request a Token. In the Port Mapping Request message, the client MUST set both the packet sender SSRC and media source SSRC fields to its own SSRC since the Port Mapping Request message is not necessarily linked to any specific media source. The FCI field has the structure depicted in Figure 4.

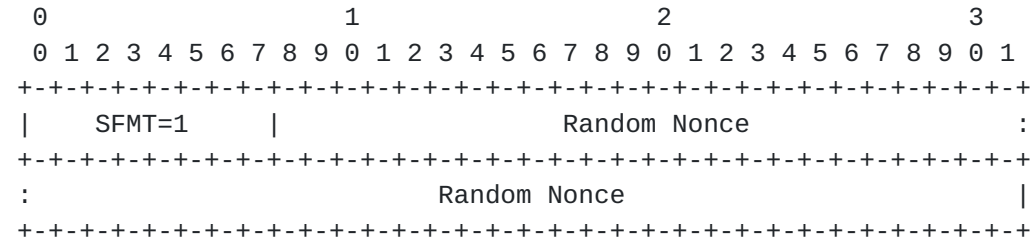

Figure 4: The FCI field of Port Mapping Request message

- Random Nonce (56 bits): Mandatory field that contains a random nonce 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. If the 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 SFMT=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 and media sender SSRC fields are both set to the client's SSRC since the Port Mapping Response message is not necessarily linked to any specific media source. The FCI field has the structure depicted in Figure 5.

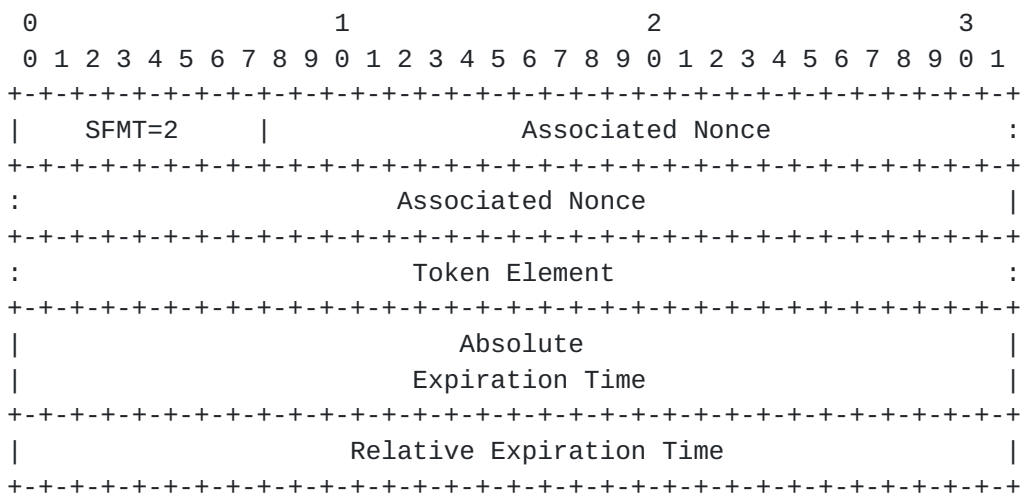

Figure 5: FCI field syntax for the Port Mapping Response message

- Associated Nonce (56 bits): Mandatory field that contains the nonce received in the Port Mapping Request message and used in Token construction.

- Token Element (Variable size): Mandatory element that is used to carry the Token generated by the server. This element is a Length-Value element. The Length field, which is 8 bits, indicates the length (in octets) of the Value field that follows the Length field. The Value field carries the Token (or more accurately, the output of the encoding process on the server).

- Absolute Expiration Time (64 bits): Mandatory 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): Mandatory 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. A relative expiration time of zero indicates that the accompanying Token is not valid.

  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.

4.3. Token Verification Request

The Token Verification Request message is identified by SFMT=3. This message contains the Token and accompanies any RTCP message that would trigger a new or control an existing unicast session. Currently, the following RTCP messages are REQUIRED to be accompanied by a Token Verification Request message:

- Messages that trigger a new unicast session:
  - NACK messages [RFC4585]
  - RAMS-R messages [I-D.ietf-avt-rapid-acquisition-for-rtp]
- Messages that control an existing unicast session associated with a multicast session:
  - BYE messages [RFC3550]
  - RAMS-T messages [I-D.ietf-avt-rapid-acquisition-for-rtp]
  - CCM messages [RFC5104]

Other RTCP messages defined in the future, which could be abused to cause packet amplification attacks, SHOULD also be authenticated using the mechanism described in this document. The Token Verification Request message might also be bundled with packets carrying RTCP receiver 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 the Token Verification Request message, the client MUST set both the packet sender SSRC and media source SSRC fields to its own SSRC since the media source SSRC may not be known. The client MUST NOT send a Token Verification Request message with a Token that has expired. The FCI field has the structure depicted in Figure 6.

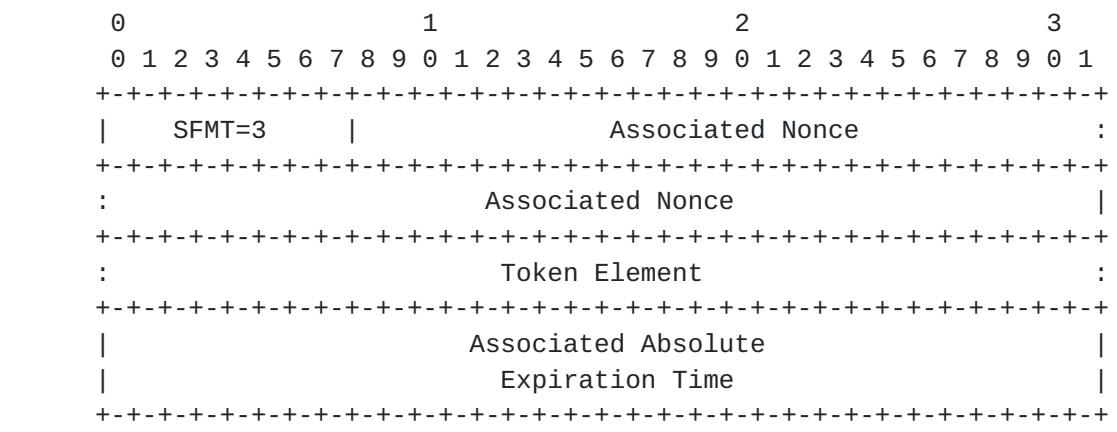

Figure 6: FCI field syntax for the Token Verification message

- Associated Nonce (56 bits): Mandatory field that contains the nonce associated with the Token above.

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

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

4.4. Token Verification Failure

The Token Verification Failure message is identified by SFMT=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 and media sender SSRC fields are both set to the client's SSRC. The FCI field has the structure depicted in Figure 6.

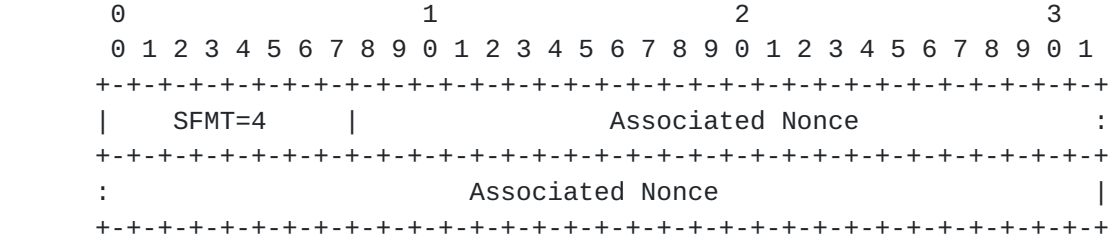

Figure 7: FCI field syntax for the Token Failure message

- Associated Nonce (56 bits): Mandatory 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:

- Client's IP address as seen by the server (32/128 bits for IPv4/ IPv6 addresses)
- The nonce generated and inserted in the Port Mapping Request message by the client (56 bits)
- 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 should be at least 160 bits long, and generated using a cryptographically secure random source [RFC4086]. However, implementations MAY adopt different approaches and 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, 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 the 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-req Attribute

This new SDP attribute is used declaratively to indicate the port for obtaining a Token. Its presence indicates that a Token MUST be included in the feedback messages sent to the server triggering or controlling a unicast session.

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

```
portmapping-req-attribute = "a=portmapping-req:" port CRLF
```

Here, 'port' is defined as specified in Section 9 of [RFC4566]. The 'portmapping-req' attribute is used as a session-level or media-level attribute. If used at a media level, the attribute MUST be used for a unicast media stream.

The Offer/Answer Model behavior [RFC3264] for the 'portmapping-req' attribute is not defined, thus, MUST NOT be used. This attribute MUST only be used in a declarative manner.

7.2. Requirements

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

- The SDP grouping framework and flow identification (FID) semantics [RFC5888]
- The RTP/AVPF profile [RFC4585]
- The RTCP extensions for SSM sessions with unicast feedback [RFC5760]
- The 'multicast-rtcp' attribute [I-D.ietf-avt-rtcp-port-for-ssm]
- 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=mid:1
m=video 42000 RTP/AVPF 99                              ; Note 3
i=Unicast Retransmission Stream
c=IN IP4 192.0.2.1
a=sendonly
a=rtpmap:99 rtx/90000
a=rtcp-mux                                             ; Note 4
a=rtcp:42500                                           ; Note 5
a=fmtp:99 apt=98; rtx-time=5000
a=portmapping-req:30000                                ; Note 6
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 (DS) to the multicast destination address of 233.252.0.2 (G) 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 (S) 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 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 4: The server multiplexes RTP and RTCP packets on the same port

(c1 in Figure 2).

Note 5: The server uses port 42500 (P4) for the unicast sessions.

Note 6: The "a=portmapping-req" line indicates that a Token needs to be retrieved first before a unicast session associated to the multicast session can be established and that the Port Mapping Request message needs to be sent to port 30000 (PT).

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 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 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.

HMAC-SHA1 provides a level of security that is widely regarded as being more than sufficient for providing message authentication. It is believed that the economic cost of breaking that algorithm is significantly higher than the cost of more direct approaches to violating system security, e.g., theft, bribery, wiretapping, and other forms of malfeasance. HMAC-SHA1 is secure against all known cryptanalytic attacks that use computational resources that are currently economically feasible.

9.2. The portmapping-req Attribute

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 a new attribute name in SDP.

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

10.2. Registration of FMT Values

Within the RTPFB range, the following format (FMT) value is registered:

```
Name:       Port Mapping
Long name:  Port Mapping Between Unicast and Multicast RTP Sessions
Value:      7
Reference:  [RFCXXXX]
```

10.3. SFMT Values for Port Mapping Messages Registry

This document creates a new sub-registry for the sub-feedback message type (SFMT) values to be used with the FMT value registered for Port Mapping messages. The registry is called the SFMT Values for Port Mapping Messages Registry. This registry is to be managed by the IANA according to the Specification Required policy of [RFC5226].

The length of the SFMT field in the Port Mapping messages is a single octet, allowing 256 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]
5-254	Assignable - Specification Required	
255	Reserved	[RFCXXXX]

The SFMT values 0 and 255 are reserved for future use.

Any registration for an unassigned SFMT value needs to contain the following information:

- Contact information of the one doing the registration, including at least name, address, and email.

- A detailed description of what the new SFMT represents and how it shall be interpreted.

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.

[I-D.ietf-avt-rtcp-port-for-ssm] Begen, A., "RTP Control Protocol (RTCP) Port for Source-Specific Multicast (SSM) Sessions", draft-ietf-avt-rtcp-port-for-ssm-03 (work in progress), October 2010.

[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.

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