Internet Draft
Expires: June 2008

K. Hedayat
Brix Networks
P. Jones
Cisco Systems, Inc.
A. Roychowdhury
Hughes
C. SivaChelvan
Cisco Systems, Inc.
N. Stratton

November 2007

An Extension to the Session Description Protocol (SDP) for Media Loopback
draft-ietf-mmusic-media-loopback-07

Status of this Memo

Copyright Notice

Abstract

The wide deployment of Voice over IP (VoIP), Real-time Text and Video over IP services has introduced new challenges in managing and maintaining voice/real-time Text/video quality, reliability, and overall performance. In particular, media delivery is an area that needs attention. One method of meeting these challenges is monitoring the media delivery performance by looping media back to the transmitter. This is typically referred to as "active monitoring" of services. Media loopback is especially popular in ensuring the quality of transport to the edge of a given VoIP, Real-time Text or Video over IP service. Today in networks that deliver real-time media, short of running 'ping' and 'traceroute' to the edge, service providers are left without the necessary tools to actively monitor, manage, and diagnose quality issues with their service. The extension defined herein adds new SDP media attributes which enables establishment of media sessions where the media is looped back to the transmitter. Such media sessions will serve as monitoring and troubleshooting tools by providing the means for measurement of more advanced VoIP, Real-time Text and Video Over IP performance metrics.

Table of Contents

1. Introduction

The overall quality, reliability, and performance of VoIP, Real-time Text and Video over IP services rely on the performance and quality of the media path. In order to assure the quality of the delivered media there is a need to monitor the performance of the media transport. One method of monitoring and managing the overall quality of VoIP, Real-time Text and Video over IP Services is through monitoring the quality of the media in an active session. This type of "active monitoring" of services is a method of pro-actively managing the performance and quality of VoIP based services.

The goal of active monitoring is to measure the media quality of a VoIP, Real-time Text or Video over IP session. A way to achieve this goal is to request an endpoint to loop media back to the other endpoint and to provide media statistics (e.g., RTCP and RTCP XR information). Another method involves deployment of special endpoints that always loop incoming media back for sessions. Although the latter method has been used and is functional, it does not scale to support large networks and introduces new network management challenges. Further, it does not offer the granularity of testing a specific endpoint that may be exhibiting problems.

The extension defined in this memo introduces new SDP media attributes that enable establishment of media sessions where the media is looped back to the transmitter. The offer/answer model [RFC3264] is used to establish a loopback connection. Furthermore, this extension provides guidelines on handling RTP [RFC3550], as well as usage of RTCP [RFC3550] and RTCP XR [RFC3611] for reporting media related measurements.

2. Terminology

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

3. Offering Entity Behavior

An offering entity compliant to this memo and attempting to establish a media session with media loopback MUST include "loopback" media attributes for each individual media description in the offer message. The offering entity MUST look for the "loopback" media attributes in the media description(s) of the response from the answering entity for confirmation that the request is accepted.

4. Answering Entity Behavior

An answering entity compliant to this specification and receiving an offer containing media descriptions with the "loopback" media attributes, MUST acknowledge the request by including the received "loopback" media attributes for each media description in its response. The server MAY reject the "loopback" request for specific media types as defined in section 5.4.1 of this specification.

An answering entity that is not compliant to this specification and which receives an offer with the "loopback" media attributes MAY ignore the attribute and treat the incoming offer as a normal request.

5. SDP Constructs Syntax

Two new media attributes are defined: one indicates the type of loopback and one indicates the mode of the loopback.

5.1 Loopback Type Attribute

The loopback type is a property media attribute with the following syntax:

a=loopback:<loopback-type>

Following is the Augmented BNF [RFC2234] for loopback-type:

```
loopback-type = loopback-type-choices | loopback-type-choice-3
loopback-choices = loopback-type-choice-1 | loopback-type-choice-2
| loopback-type-choice-1 1*space loopback-type-choice-2 |
loopback-type-choice-2 1*space loopback-type-choice-1
loopback-type-choice-1 = "rtp-pkt-loopback"
loopback-type-choice-2 = "rtp-media-loopback"
loopback-type-choice-3 = "rtp-start-loopback"
```

The loopback type is used to indicate the type of loopback. The loopback-type values are rtp-pkt-loopback, rtp-media-loopback, and rtp-start-loopback.

rtp-pkt-loopback: In this mode, the RTP packets are looped back to the sender at a point before the encoder/decoder function in the receive direction to a point after the encoder/decoder function in the send direction. This effectively re-encapsulates the RTP payload with the RTP/UDP/IP overheads appropriate for sending it in the reverse direction. Any type of encoding related functions, such as packet loss concealment, MUST NOT be part of this type of loopback path. In this mode the RTP packets are looped back with a new payload type and format. Section 7 describes the payload formats that MUST be used for this type of loopback.

rtp-media-loopback: This loopback is activated as close as possible to the analog interface and after the decoder so that the RTP packets are subsequently re-encoded prior to transmission back to the sender.

rtp-start-loopback: In certain scenarios it is possible that the media transmitted by the loopback-source is blocked by a network element until the loopback-mirror starts transmitting packets. Loopback-source and loopback-mirror are loopback modes defined in section 5.2. One example of this scenario is the presence of an RTP relay in the path of the media. RTP relays exist in VoIP networks for purpose of NAT and Firewall traversal. If an RTP relay is present, the loopback-source's packets are dropped by the RTP relay until the loopback-mirror has started transmitting media and the media state within the RTP relay is established. This loopback attribute is used to specify the media type for transmitting media packets by the loopback-mirror prior to the loopback process for the purpose of setting media state within the network. In the presence of this loopback attribute the loopback-mirror will transmit media, according to the description that contains this attribute, until it receives media from the loopback-

source. The loopback-mirror MAY include this attribute in the answer if it is not present in the offer. This may be necessary if the loopback-mirror is aware of NAT's, firewalls, or RTP relays on the path of the call. In this case the loopback-source MUST accept media according to rtp-start-loopback attribute. After the first media packet is received from the loopback-source, the loopback-mirror MUST terminate the transmission of rtp-start-loopback media and MUST start looping back media as defined by the other loopback attributes present in the offer. If an offer includes the rtp-start-loopback attribute it MUST also include at least one other attribute as defined in this section. The loopback-source is able to filter rtp-start-loopback packets from other types of loopback with the payload type of the packet. The media port number for rtp-start-loopback MUST be the same as the corresponding loopback attribute that will take over after the reception of first media packet from the offering entity.

It is recommended that an offering entity specifying media with either rtp-pkt-loopback or rtp-media-loopback attribute also specify the rtp-start-loopback attribute unless the offering entity is certain that its media will not be blocked by a network entity as explained above.

5.2 Loopback Mode Attribute

The loopback mode is a value media attribute that is used to indicate the mode of the loopback. These attributes are additional mode attributes like sendonly, recvonly, etc. The syntax of the loopback mode media attribute is:

```
a=<loopback-mode>:<fmt>...
```

The loopback-mode values are loopback-source and loopback-mirror.

loopback-source: This attribute specifies that the sender is the media source and expects the receiver to act as a loopback-mirror.

loopback-mirror: This attribute specifies that the receiver will mirror (echo) all received media back to the sender of the RTP stream. No media is generated locally by the receiver for transmission in the mirrored stream unless rtp-start-loopback is requested.

<fmt> is a media format description. The format descrption has the semantics as defined in section 5.14 of RFC 4566 [RFC2234]. When loopback-mode is specified as loopback-source, the media format corresponds to the RTP payload types the source is willing to send.

When loopback-mode is specified as loopback-mirror, the media format corresponds to the RTP payload types the mirror is willing to receive. The payload types specified in m= line for a loopback-source specify the payloads the source is willing to receive. Similarly, for the loopback-mirror these payload types specify the payloads it is willing to send.

The loopback mode attribute does not apply to rtp-start-loopback attribute and MUST be ignored if received by the answering entity.

5.3 Generating the Offer for Loopback Session

If an offerer wishes to make a loopback request, it MUST include both the loopback-type and loopback-mode attribute in a valid SDP offer:

```
Example:   m=audio 41352 RTP/AVP 0 8
           a=loopback:rtp-media-loopback
           a=loopback-source:0 8
```

Note: A loopback offer in a given media description MUST NOT contain the standard mode attributes sendonly, recvonly, sendrecv or inactive. The loopback-mode attributes (loopback-source and loopback-mirror) replace the standard attributes.

The offerer may offer more than one loopback-type in the SDP offer. In this case the answer MUST include only one of the loopback types that are accepted by the answerer. The answerer SHOULD give preference to the first loopback-type in the SDP offer.

For loopback-source media (e.g. audio) streams, the port number and the address in the offer (m= line) indicate where the offerer would like to receive the media stream. The payload type numbers indicate the value of the payload the offerer expects to receive. The RTP payload types indicated in the a=loopback-source line are the payload types for the codecs the offerer is willing to send. However, the answer might indicate a different payload type number for the same codec. In that case, the offerer MUST send the payload type received in the answer.

If loopback-type is rtp-pkt-loopback, the loopback-mirror MUST send and the loopback-source MUST receive the looped back packets encoded in one of the two payload formats (encapsulated RTP or payload loopback) as defined in section 7.

```
Example:   m=audio 41352 RTP/AVP 112
           a=loopback:rtp-pkt-loopback
```

```
              a=loopback-source:0 8
              a=rtpmap:112 encaprtp/8000

   Example:   m=audio 41352 RTP/AVP 112
              a=loopback:rtp-pkt-loopback
              a=loopback-source:0 8
              a=rtpmap:112 rtploopback/8000
```

Note: NAT devices may change the actual port number that is used for transmission and the expected receive port.

5.4 Generating the Answer for Loopback Session

If an answerer wishes to accept the loopback request it MUST include both the loopback mode and loopback type attribute in the answer. If a stream is offered with loopback-source or loopback-mirror attributes, the corresponding stream MUST be loopback-mirror or loopback-source respectively, provided that answerer is capable of supporting the requested loopback-type.

For example, if the offer contains:

```
m=audio 41352 RTP/AVP 0 8
a=loopback:rtp-media-loopback
a=loopback-source:0 8
```

The answer that is capable of supporting the offer MUST contain:

```
m=audio 41352 RTP/AVP 0 8
a=loopback:rtp-media-loopback
a=loopback-mirror:0 8
```

As previously stated if a stream is offered with multiple loopback type attributes, the corresponding stream MUST contain only one loopback type attribute selected by the answerer.

For example, if the offer contains:

```
m=audio 41352 RTP/AVP 0 8 112
a=loopback:rtp-media-loopback rtp-pkt-loopback
a=loopback-source:0 8
```

The answer that is capable of supporting the offer and chooses to loopback the media using the rtp-media-loopback type MUST contain:

```
m=audio 41352 RTP/AVP 0 8
```

```
a=loopback:rtp-media-loopback
a=loopback-mirror:0 8
```

As specified in section 7, if the loopback-type is rtp-pkt-loopback, either the encapsulated RTP payload format or direct loopback RTP payload format MUST be used for looped back packets.

For example, if the offer contains:

```
m=audio 41352 RTP/AVP 112 113
a=loopback:rtp-pkt-loopback
a=loopback-source:0 8
a=rtpmap:112 encaprtp/8000
a=rtpmap:113 rtploopback/8000
```

The answer that is capable of supporting the offer MUST contain one of the following:

```
m=audio 41352 RTP/AVP 112
a=loopback:rtp-pkt-loopback
a=loopback-mirror:0 8
a=rtpmap:112 encaprtp/8000

m=audio 41352 RTP/AVP 113
a=loopback:rtp-pkt-loopback
a=loopback-mirror:0 8
a=rtpmap:113 rtploopback/8000
```

5.4.1 Rejecting the Loopback Offer

An offered stream with loopback-source MAY be rejected if the loopback-type is not specified, the specified loopback-type is not supported, or the endpoint cannot honor the offer for any other reason. The Loopback request may be rejected by setting the media port number to zero in the answer as per RFC 3264 [RFC3264].

5.5 Offerer Processing of the Answer

The answer to a loopback-source MUST be loopback-mirror. The answer to a loopback-mirror MUST be loopback-source. The loopback-mode line MUST contain at least one codec the answerer is willing to send or receive depending on whether it is the loopback-source or the loopback-mirror. In addition, the "m=" line MUST contain at least one codec that the answerer is willing to send or receive

depending on whether it is the loopback-mirror or the loopback-source.
If the answer does not contain a=loopback-mirror or a=loopback-source or contains any other standard mode attributes, it is assumed that the loopback extensions are not supported by the target UA.

5.6 Modifying the Session

At any point during the loopback session, either participant may issue a new offer to modify the characteristics of the previous session. In case of SIP this is defined in section 8 of RFC 3264 [RFC3264]. This also includes transitioning from a normal media processing mode to loopback mode, and vice a versa.

6. RTP Requirements

An answering entity that is compliant to this specification and accepting a media with rtp-pkt-loopback loopback-type MUST loopback the incoming RTP packets using either the encapsulated RTP payload format or the direct loopback RTP payload format as defined in section 7 of this specification.

An answering entity that is compliant to this specification and accepting a media with rtp-media-loopback loopback-type MUST transmit all received media back to the sender. The incoming media MUST be treated as if it were to be played (e.g. the media stream MAY receive treatment from PLC algorithms). The answering entity MUST re-generate all the RTP header fields as it would when transmitting media. The answering entity MAY choose to encode the loopback media according to any of the media descriptions supported by the offering entity. Furthermore, in cases where the same media type is looped back, the answering entity MAY choose to preserve number of frames/packet and bitrate of the encoded media according to the received media.

7. Payload formats for Packet loopback

The payload formats described in this section MUST be used by a loopback-mirror when rtp-pkt-loopback is the specified loopback-type. Two different formats are specified here - an encapsulated RTP payload format and a direct loopback RTP payload format. The encapsulated RTP payload format should be used when

the incoming RTP header information needs to be preserved during the loopback operation. This is useful in cases where loopback source needs to measure performance metrics in both directions. However, this comes at the expense of increased packet size as described in section 7.1. The direct loopback RTP payload format should be used when bandwidth requirement prevent the use of encapsulated RTP payload format.

7.1 Encapsulated Payload format

A received RTP packet is encapsulated in the payload section of the RTP packet generated by a loopback-mirror. Each received packet MUST be encapsulated in a different packet, the encapsulated packet MAY be fragmented only if required (for example: due to MTU limitations).

7.1.1 Usage of RTP Header fields

Payload Type (PT): The assignment of an RTP payload type for this packet format is outside the scope of this document; it is either specified by the RTP profile under which this payload format is used or more likely signaled dynamically out-of-band (e.g., using SDP; section 7.3 defines the name binding).

Marker (M) bit: If the received RTP packet is looped back in multiple RTP packets, the M bit is set to 1 in the last packet, otherwise it is set to 0.

Extension (X) bit: Defined by the RTP Profile used.

Sequence Number: The RTP sequence number SHOULD be generated by the loopback-mirror in the usual manner with a constant random offset.

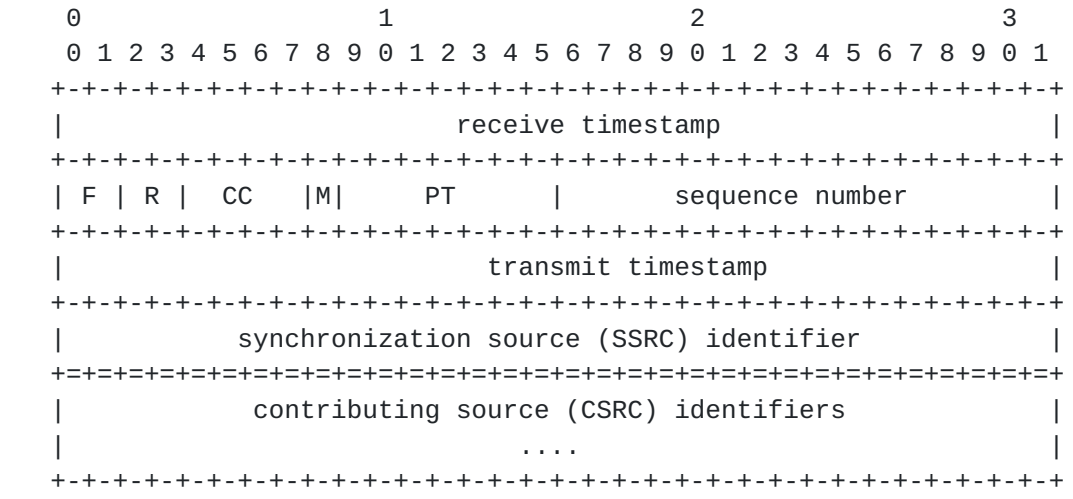
Timestamp: The RTP timestamp denotes the sampling instant for when the loopback-mirror is transmitting this packet to the loopback-source. The RTP timestamp MUST based on the same clock used by the loopback-source. The initial value of the timestamp SHOULD be random for security reasons (see Section 5.1 of RFC 3550 [RFC3550]).

SSRC: set as described in RFC 3550 [RFC3550].

CC and CSRC fields are used as described in RFC 3550 [RFC3550].

7.1.2 RTP Payload Structure

The RTP header in the encapsulated packet MUST be followed by the payload header defined in this section. If the received RTP packet has to be looped back in multiple packets due to fragmentation, the RTP header in each packet MUST be followed by the payload header defined in this section. The header is devised so that the loopback-source can usefully decode looped back packets in the presence of moderate packet loss [RFC3550].



The 12 octets after the receive timestamp are identical to the RTP header in the received packet except for the first 4 bits of the first octet.

Receive Timestamp: 32 bits

The Receieve timestamp denotes the sampling instant for when the last octet of the media packet that is being encapsulated by the loopback-mirror is received from the loopback-source. The Receive timestamp MUST be based on the same clock used by the loopback-source. The initial value of the timestamp SHOULD be random for security reasons (see Section 5.1 of RFC 3550 [RFC3550]).

Fragmentation (F): 2 bits

First Fragment (00) /Last Fragment (01) /No Fragmentation(10)/ Intermediate Fragment (11). This field identifies how much of the received packet is encapsulated in this packet by the loopback-mirror. If the received packet is not fragmented, this field is set to 10; otherwise the packet that contains the first fragments sets this field to 00, the packet that contains the last fragment sets this field to 01, all other packets set this field to 11.

Reserved: 2 bits

This field is reserved for future definition. In the absence of such a definition, the bits in this field MUST be set to zero and MUST be ignored by the receiver.

Any padding octets in the original packet MUST not be included in the loopback packet generated by a loopback-mirror. The loopback-mirror MAY add padding octets if required.

7.1.3 Usage of SDP

The payload type number for the encapsulated stream can be negotiated using a mechanism like SDP. There is no static payload type assignment for the encapsulated stream, so dynamic payload type numbers MUST be used. The binding to the name is indicated by an rtpmap attribute. The name used in this binding is "encaprtp".

The following is an example SDP fragment for encapsulated RTP.

```
m=audio 41352 RTP/AVP 112
a=rtpmap:112 encaprtp/8000
```

7.2 Direct loopback RTP payload format

The direct loopback RTP payload format can be used in scenarios where the 16 byte overhead of the encapsulated payload format is significant. This payload format MUST not be used in cases where the MTU on the loopback path is less than the MTU on the transmit path. When using this payload format, the receiver MUST loop back each received packet in a separate RTP packet.

7.2.1 Usage of RTP Header fields

Payload Type (PT): The assignment of an RTP payload type for this packet format is outside the scope of this document; it is either specified by the RTP profile under which this payload format is used or more likely signaled dynamically out-of-band (e.g., using SDP; section 7.3 defines the name binding).

Marker (M) bit: Set to the value in the received packet.

Extension (X) bit: Defined by the RTP Profile used.

Sequence Number: The RTP sequence number SHOULD be generated by the loopback-mirror in the usual manner with a constant random offset.

Timestamp: The RTP timestamp denotes the sampling instant for when the loopback-mirror is transmitting this packet to the loopback-source. The RTP timestamp MUST based on the same clock used by the loopback-source. The initial value of the timestamp SHOULD be random for security reasons (see Section 5.1 of RFC 3550 [RFC3550]).

SSRC: set as described in RFC 3550 [RFC3550].

CC and CSRC fields are used as described in RFC 3550 [RFC3550].

7.2.2 RTP Payload Structure

This payload format does not define any payload specific headers. The loopback-mirror simply copies the payload data from the payload portion of the packet received from the loopback-source.

7.2.3 Usage of SDP

The payload type number for the payload loopback stream can be negotiated using a mechanism like SDP. There is no static payload type assignment for the stream, so dynamic payload type numbers MUST be used. The binding to the name is indicated by an rtpmap attribute. The name used in this binding is "rtploopback".

The following is an example SDP fragment for encapsulated RTP.

```
m=audio 41352 RTP/AVP 112
a=rtpmap:112 rtploopback/8000
```

8. RTCP Requirements

The use of the loopback attribute is intended for monitoring of media quality of the session. Consequently the media performance information should be exchanged between the offering and the answering entities. An offering or answering entity that is compliant to this specification SHOULD support RTCP per [RFC3550] and RTCP-XR per RFC 3611 [RFC3611]. Furthermore, if the client or the server choose to support RTCP-XR, they SHOULD support RTCP-XR

Loss RLE report block, Duplicate RLE report block, Statistics Summary report block, and VoIP Metric Reports Block per sections 4.1, 4.2, 4.6, and 4.7 of RFC 3611 [RFC3611]. The client and the server MAY support other RTCP-XR reporting blocks as defined by RFC 3611 [RFC3611].

9. Congestion Control

All the participants in a loopback session SHOULD implement congestion control mechanisms as defined by the RTP profile under which the loopback mechanism is implemented. For audio video profiles, implementations SHOULD conform to the mechanism defined in Section 2 of RFC 3551.

10. Examples

This section provides examples for media descriptions using SDP for different scenarios. The examples are given for SIP-based transactions and are abbreviated and do not show the complete signaling for convenience.

10.1 Offer for specific media loopback type

A client sends an INVITE request with SDP which looks like:

```
v=0
o=user1 2890844526 2890842807 IN IP4 192.0.2.11
s=Example
i=An example session
e=user@example.com
c=IN IP4 192.0.2.12/127
t=0 0
m=audio 49170 RTP/AVP 0
a=loopback:rtp-media-loopback
a=loopback-source:0
```

The client is offering to source the media and expects the server to mirror the RTP stream per rtp-media-loopback loopback type.

A server sends a response with SDP which looks like:

```
v=0
o=user1 2890844526 2890842807 IN IP4 192.0.2.11
s=Example
```

```
i=An example session
e=user@example.com
c=IN IP4 192.0.2.12/127
t=0 0
m=audio 49170 RTP/AVP 0
a=loopback:rtp-media-loopback
a=loopback-mirror:0
```

The server is accepting to mirror the media from the client at the media level.

10.2 Offer for choice of media loopback type

A client sends an INVITE request with SDP which looks like:

```
v=0
o=user1 2890844526 2890842807 IN IP4 192.0.2.11
s=Example
i=An example session
e=user@example.com
c=IN IP4 192.0.2.12/127
t=0 0
m=audio 49170 RTP/AVP 0 112 113
a=loopback:rtp-media-loopback rtp-pkt-loopback
a=loopback-source:0
a=rtpmap:112 encaprtp/8000
a=rtpmap:113 rtploopback/8000
```

The client is offering to source the media and expects the server to mirror the RTP stream at either the media or rtp level.

A server sends a response with SDP which looks like:

```
v=0
o=user1 2890844526 2890842807 IN IP4 192.0.2.11
s=Example
i=An example session
e=user@example.com
c=IN IP4 192.0.2.12/127
t=0 0
m=audio 49170 RTP/AVP 112
a=loopback:rtp-pkt-loopback
a=loopback-mirror:0
a=rtpmap:112 encaprtp/8000
```

The server is accepting to mirror the media from the client at the packet level using the encapsulated RTP payload format.

10.3 Offer for choice of media loopback type with rtp-start-loopback

A client sends an INVITE request with SDP which looks like:

```
v=0
o=user1 2890844526 2890842807 IN IP4 192.0.2.11
s=Example
i=An example session
e=user@example.com
c=IN IP4 192.0.2.12/127
t=0 0
m=audio 49170 RTP/AVP 0 112 113
a=loopback:rtp-media-loopback rtp-pkt-loopback
a=loopback-source:0
a=rtpmap:112 encaprtp/8000
a=rtpmap:113 rtploopback/8000
m=audio 49170 RTP/AVP 100
a=loopback:rtp-start-loopback
```

The client is offering to source the media and expects the server to mirror the RTP stream at either the media or rtp level. The client also expects the server to source media until it receives packets from the server per media described with the rtp-start-loopback attribute.

A server sends a response with SDP which looks like:

```
v=0
o=user1 2890844526 2890842807 IN IP4 192.0.2.11
s=Example
i=An example session
e=user@example.com
c=IN IP4 192.0.2.12/127
t=0 0
m=audio 49170 RTP/AVP 113
a=loopback:rtp-pkt-loopback
a=loopback-mirror:0
a=rtpmap:113 rtploopback/8000
m=audio 49170 RTP/AVP 100
a=rtpmap:100 pcmu/8000
a=loopback:rtp-start-loopback
```

The server is accepting to mirror the media from the client at the packet level using the direct loopback RTP payload format. The

server is also accepting to source media until it receives media packets from the client.

10.4 Response to INVITE request rejecting loopback media

A client sends an INVITE request with SDP which looks like:

```
v=0
o=user1 2890844526 2890842807 IN IP4 192.0.2.11
s=Example
i=An example session
e=user@example.com
c=IN IP4 192.0.2.12/127
t=0 0
m=audio 49170 RTP/AVP 0
a=loopback:rtp-media-loopback
a=loopback-source:0
```

The client is offering to source the media and expects the server to mirror the RTP stream at the media level.

A server sends a response with SDP which looks like:

```
v=0
o=user1 2890844526 2890842807 IN IP4 192.0.2.11
s=Example
i=An example session
e=user@example.com
c=IN IP4 192.0.2.12/127
t=0 0
m=audio 0 RTP/AVP 0
a=loopback:rtp-media-loopback
a=loopback-mirror:0
```

NOTE: Loopback request may be rejected by either not including the loopback mode attribute (for backward compatibility) or setting the media port number to zero, or both, in the response.

10.5 Response to INVITE request rejecting loopback media with rtp-start-loopback

A client sends an INVITE request with SDP which looks like:

```
v=0
o=user1 2890844526 2890842807 IN IP4 192.0.2.11
```

```
s=Example
i=An example session
e=user@example.com
c=IN IP4 192.0.2.12/127
t=0 0
m=audio 49170 RTP/AVP 0
a=loopback:rtp-media-loopback
a=loopback-source:0
m=audio 49170 RTP/AVP 100
a=loopback:rtp-start-loopback
```

The client is offering to source the media and expects the server to mirror the RTP stream at the media level. The client also expects the server to source media until it receives packets from the server per media described with the rtp-start-loopback attribute.

A server sends a response with SDP which looks like:

```
v=0
o=user1 2890844526 2890842807 IN IP4 192.0.2.11
s=Example
i=An example session
e=user@example.com
c=IN IP4 192.0.2.12/127
t=0 0
m=audio 0 RTP/AVP 0
a=loopback:rtp-media-loopback
a=loopback-mirror:0
m=audio 0 RTP/AVP 0
a=loopback:rtp-start-loopback
```

NOTE: Loopback request may be rejected by either not including the loopback mode attribute (for backward compatibility) or setting the media port number to zero, or both, in the response.

11. Security Considerations

The security considerations of [RFC3261] apply. Furthermore, given that media loopback may be automated without the end user's knowledge, the server of the media loopback should be aware of denial of service attacks. It is recommended that sessions with media loopback are authenticated and the frequency of such sessions is limited by the server.

12. Implementation Considerations

The media loopback approach described in this document is a complete solution that would work under all scenarios. However, it is believed that the solution may not be light-weight enough for the common case. In light of this concern, this section clarifies which features of the loopback proposal MUST be implemented for all implementations and which features MAY be deferred if the complete solution is not desired.

All implementations MUST support the rtp-pkt-loopback option for loopback-type attribute. In addition, for the loopback-mode attribute, all implementations of an offerer MUST at a minimum be able to act as a loopback-source. All implementation MUST also at a minimum support the direct media loopback payload type. Remaining attribute values including rtp-media-loopback and rtp-start-loopback MAY be implemented in complete implementations of this draft.

13. IANA Considerations

There are no IANA considerations associated with this specification.

14. Acknowledgements

The authors wish to thank Nagarjuna Venna, Flemming Andreasen, Jeff Bernstein, Paul Kyzivat, and Dave Oran for their comments and suggestions.

15. Normative References

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

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

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

[RFC3611] Almeroth, K., Caceres, R., Clark, A., Cole, R., Duffield, N., Friedman, T., Hedayat, K., Sarac, K. and M. Westerlund, "RTP Control Protocol Extended Reports (RTCP XR)", RFC 3611, November 2003.

[RFC2234] Crocker, P. Overell, "Augmented ABNF for Syntax Specification: ABNF", RFC 2234, November 1997.

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

[RFC2736] Handley, M., Perkins, C., "Guidelines for Writers of RTP Payload Format Specifications", RFC 2736, BCP 0036, December 1999.

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

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

Authors' Addresses

Kaynam Hedayat
Brix Networks
285 Mill Road
Chelmsford, MA 01824
US

Phone: +1 978 367 5611
EMail: khedayat@brixnet.com
URI: http://www.brixnet.com/

Paul E. Jones
Cisco Systems, Inc.
7025 Kit Creek Rd.
Research Triangle Park, NC 27709

US

Phone: +1 919 392 6948
EMail: paulej@packetizer.com
URI: http://www.cisco.com/

Arjun Roychowdhury
Hughes Systique Corp.
15245 Shady Grove Rd, Ste 330
Rockville MD 20850
US

Phone: +1 301 527 1629
EMail: arjun@hsc.com
URI: http://www. hsc.com/

Chelliah SivaChelvan
Cisco Systems, Inc.
2200 East President George Bush Turnpike
Richardson, TX 75082
US

Phone: +1 972 813 5224
EMail: chelliah@cisco.com
URI: http://www.cisco.com/

Nathan Stratton

663 Salem St.
Lynnfield, MA 01940

Phone: +1 410 908 7587
EMail: nathan@robotics.net
URI: http://www.robotics.net/

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).