TRAM
Internet-Draft
Intended status: Standards Track
Expires: December 3, 2015

P. Martinsen
D. Wing
Cisco

June 1, 2015

STUN Traceroute
draft-martinsen-tram-stuntrace-01

Abstract

After a UDP protocol such as RTP determines a network path is experiencing problems, a traceroute is often useful to determine which router or which link is contributing to the problem. However, operating system traceroute commands follow a different path than the actual UDP flow which complicates troubleshooting. A superior method is shown which is absolutely path-congruent with the UDP protocol itself, works on IPv4 and IPv6, and does not require administrative privileges on most operating systems.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 3, 2015.

Table of Contents

1. Introduction

Traceroute [RFC1393] is a simple tool available on most operating systems and is popular to debug the network by simply getting round-trip time along each hop to a remote IP address. More advanced tools, such as MTR, provide more metrics such as packet loss and round trip time to each hop over several seconds or minutes.

To simplify network debugging when dealing with bi-directional real time media it is often useful to get as much information as possible regarding the network path. In this specification probe packets are sent using the same 5-tuple where (S)RTP media is flowing. This will provide the most accurate results, as probe packets sent on a different 5-tuple may take another path due to Equal-Cost Multipath (ECMP, [RFC2992]), policy-based routing, and similar techniques.

To avoid those problems, the probe packets need to be sent from the same socket and with the same DiffServ code point the normal (S)RTP

media packets. As shown in Appendix A, most operating systems can pass the ICMP "Time to Live Exceeded" error to the application, so the application can perform the diagnostics over that network path.

This specifications uses STUN [RFC5389] packets as probes. STUN packets are designed to be multiplexed together with RTP [RFC3550] (and SRTP [RFC3711]) and are unlikely to cause any "problems" for the (S)RTP receiver. To differentiate each hop count, classic traceroute uses different UDP port numbers (e.g., TTL=1 uses UDP port 55001, TTL=2 uses UDP port 55002, etc.). The mechanism described here uses the same UDP port number (so that the trace is path-congruent with the (S)RTP packets), and uses different length UDP packets to differentiate each hop count (e.g., TTL=1 uses length 501, TTL=2 uses length 502, etc.).

Using a technique based on ICMP replies avoids a forklift upgrade of the network to provide host applications with useful information. ICMP is already supported in most network and application stacks.

Additional network characteristics like MTU and bandwidth availability can be discovered by using [I-D.petithuguenin-behave-stun-pmtud] and [I-D.martinsen-tram-turnbandwidthprobe].

2. Notational Conventions

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. Overview of Operation

An application using (S)RTP to send and receive media like audio and video following the guidelines in [RFC4961] uses symmetric send and receive ports. The application opens one socket that it uses to both send and receive media on.

It is important to note that the functionality described here can be done on most OSes without any administrative privileges.

Figure 1 depicts the various components needed for this to work. The application opens up its media socket as it would in normal cases where media is to be sent and received. It also opens up a ICMP socket or installs an error listener on the media socket.

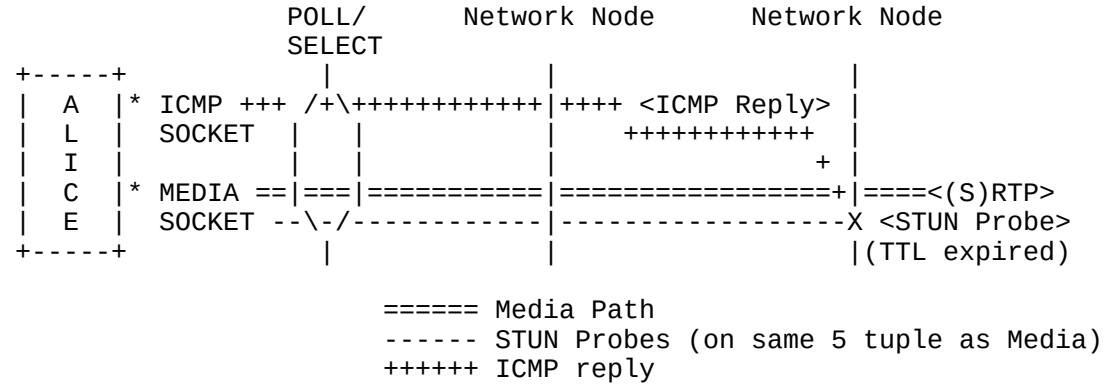

Figure 1

The application also need to listen on the sockets for any incoming ICMP packets or socket error messages. This is usually done with the socket calls select() or poll(). How to actually receive the ICMP messages will vary from OS to OS. See Appendix A for implementation details on various OSes.

Once the application have media running and is listening for ICMP replies it can start sending probes to detect networks nodes in the media path. This is done by sending STUN messages and setting the TTL/MAX_HOP limit in the IPv4/IPv6 header. Appendix A.1 explains how to set this on various platforms.

The STUN packet is sent on the same socket as the media packet are sent and received on. Mixing (S)RTP and STUN is well known behavior and should not cause any problems.

Along the path, every layer 3 network node (a.k.a. router) decreases the IPv4 TTL or IPv6 HOP_LIMIT field. If the field becomes 0 the network node responds with a ICMP error "Time to Live Exceeded" (TTL Exceeded) or "Hop Limit Exceeded in Transit" (Time Exceeded Message).

The application will receive a ICMP error in response to the offending probe packet. The source IP address of the ICMP packet will be the sending network node. This enables the application to trace the path towards the destination. The ICMP reply contains at least 8 bytes of the offending packet. The IP fragment of the offending packet in the ICMP reply can be used to determining if this ICMP reply actually was a reply to an offending packet the application did send out.

4. New STUN Attributes

This STUN extension defines the following new attribute:

0xXXX0: PATH-NODE-PROBE

4.1. PATH-NODE-PROBE

This attribute have a length of 8. Padding is needed to hit the required STUN 32 bit STUN attribute boundary.

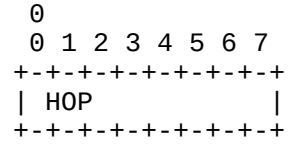

Figure 2: PATH-NODE-PROBE Attribute

The HOP field indicates what hop in the network path (relative to the application) the application is trying to learn the IP address of. This field should be set to the same value as the TTL/HOP_LIMIT field in the IPv4/IPv6 header of the probe packet leaving the application. Note that the TTL/HOP_LIMIT field in the IPv4/IPv6 header will decrease as the packet traverses the path. The HOP field in the attribute will remain unchanged.

This attribute is useful for clients when receiving the whole offending IP packet in the ICMP reply. The attribute will be reflected back in a STUN response if the remote application supports is. This makes it easier to correlate sent probe packets and ICMP responses.

5. Base Protocol Procedures

The procedures are simple; send a probe packet that may or may not trigger a reply from one of the nodes in the network path and then listen and parse any incoming replies. The reply might be an ICMP Time To Live Exceeded (from an intermediate hop), a STUN response (from the (S)RTP peer), or any other ICMP error message.

5.1. Forming STUN Packet Probes

To reduce chances of a STUN traceroute probe being stopped by various middle-boxes it is RECOMMENDED to use a STUN binding request as described in ICE [RFC5245].

Since the STUN packet can traverse the whole media-path and reach the remote peer it is RECOMMENDED the agent follows the guidelines for sending connectivity checks defined in ICE [RFC5245]. Adding a USERNAME attribute and integrity protecting the STUN message enables the remote peer to authenticate the STUN message and create an appropriate response. If the remote peer is unable to authenticate the STUN request it will not send any response. Getting a response from the remote peer is useful as it is an indication the probe have traveled the whole network path.

When forming the STUN packet probe the agent SHOULD add the PATH-NODE-PROBE attribute and MAY add a PADDING attribute as described in [RFC5780] Section 7.6. The PATH-NODE-PROBE attribute is useful for STUN servers receiving the STUN probe and it can be used to correlate any ICMP replies if the reply contains the complete offending packet. Adding the PADDING attribute is useful for clients that needs to have several outstanding probe packets on the same 5-tuple. The length of the offending packet reported back in any ICMP reply will make it possible to correlate this to the correct probe.

The agent sending the STUN packet probe MUST store the length of the UDP packet (as reported in the IP header) containing the STUN probe.

Before sending the probe on the wire it is important to set the appropriate TTL or HOP_LIMIT field in the IPv4 or IPv6 header before the packet is sent. How to do this on various OSes are described in Appendix A.1.

The probe MUST also be sent with the same DSCP value as the (S)RTP packets. This is normally not a problem as the STUN probes and (S)RTP packets are sent on the same socket.

5.2. Receiving a STUN Packet Probe

An agent that listens for STUN requests (a.k.a STUN server) that receives a STUN request with a PATH-NODE-PROBE attribute, MUST include a PATH-NODE-PROBE attribute with the same value in the generated response.

Any PADDING attributes as defined in [RFC5780] SHOULD be ignored by the STUN server.

5.3. Receiving ICMP Messages

After an agent sends a STUN probe it must be ready to receive a ICMP reply or a STUN reply. Details on how to do this on various OSes are described in Appendix A.2.

To prevent ICMP spoofing attacks [RFC5927] , the received ICMP packet MUST be validated by port number and length in the IP fragment of the offending packet contained in the ICMP payload. Port number validation checks that the port number in the offending IP fragment of the probe packet contained in the ICMP payload corresponds to the (S)RTP media (and STUN probe) 5-tuple. The length validation checks IP packet length field in the IP fragment of the offending packet received in the ICMP reply. This value MUST correspond to any length stored when the agent sent the STUN probe. If the agent uses the PADDING (Defined in [RFC5780]) attribute to generate different length on the STUN probes it is possible to have several outstanding probes, thus speeding up the trace.

6. IPv4 and IPv6 Differences

Core functionality is the same. In IPv6 the IPv4 TTL field is renamed to HOP_LIMIT to better reflect what it actually represent.

7. IANA Considerations

The code-point for the new STUN attribute defined in this specification is described in Section 4.

8. Security Considerations

ICMP messages does leak network topology, which is a well-known threat to networks and mitigations have long existed in routers and firewalls so that networks can be configured to not leak this topology information beyond their borders.

ICMP spoofing and DOS attack prevention exist in routers deployed on the Internet today.

No new threats have been added in this specification.

9. Acknowledgements

Trond Andersen for actually implementing this and Wilson Chen for helping out with different OS behavior testing.

10. References

10.1. Normative References

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

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

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

[RFC4961] Wing, D., "Symmetric RTP / RTP Control Protocol (RTCP)", BCP 131, RFC 4961, July 2007.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

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

[RFC5780] MacDonald, D. and B. Lowekamp, "NAT Behavior Discovery Using Session Traversal Utilities for NAT (STUN)", RFC 5780, May 2010.

10.2. Informative References

[I-D.martinsen-tram-turnbandwidthprobe] Martinsen, P., Andersen, T., Salgueiro, G., and M. Petit-Huguenin, "Traversal Using Relays around NAT (TURN) Bandwidth Probe", draft-martinsen-tram-turnbandwidthprobe-00 (work in progress), May 2015.

[I-D.petithuguenin-behave-stun-pmtud] Petit-Huguenin, M., "Path MTU Discovery Using Session Traversal Utilities for NAT (STUN)", draft-petithuguenin-behave-stun-pmtud-03 (work in progress), March 2009.

[ICMPTest] "ICMP test github repo", <https://github.com/palerikm/ICMPTest/>.

[RFC1393] Malkin, G., "Traceroute Using an IP Option", RFC 1393, January 1993.

[RFC2992] Hopps, C., "Analysis of an Equal-Cost Multi-Path Algorithm", RFC 2992, November 2000.

[RFC5927] Gont, F., "ICMP Attacks against TCP", RFC 5927, July 2010.

Appendix A. Platform Implementation Details

This section provides examples and hint on how probe packets can be sent and ICMP messages received on various OSes. For a complete example please refer to [ICMPTest].

A.1. Setting TTL or HOP_LIMIT on Probes

Setting the appropriate value in the IPv4 or IPv6 header is the same for most platforms. Use

```
setsockopt(sockHandle, IPPROTO_IP, IP_TTL, &sock_ttl,
sizeof(sock_ttl));
```

for IPv4 or

```
setsockopt(sockHandle,
         IPPROTO_IPV6, IPV6_UNICAST_HOPS, &sock_ttl,
         sizeof(sock_ttl));
```

for IPv6.

Sending the probes on the same socket as media is flowing requires the implementations to only set this when sending the probe packet. Remember to set it back to initial value when sending media. Most OSes seems to handle the setsockopt call correctly and not set the value in the IP header of any buffered packets.

A.2. Receiving ICMP Messages

A.2.1. OS-X and iOS

Creating a socket to listen for incoming ICMP messages can be done as:

```
icmpSocket=socket(config.remoteAddr.ss_family, SOCK_DGRAM,
                  IPPROTO_ICMP); <<<
```

This is done in addition to the normal socket used to send media on (RTP) and probes. (Yes, even if the probe are sent on the media socket the ICMP reply will be on the ICMP sockets..)

Code in the while(1) loop of poll would look something like:

```
       for(i=0;i<numSockets;i++){
           if (ufds[i].revents & POLLIN) {
               if(i == rtpSock){
                   //Handle "normal" data here.
               }
               if(i == icmpSock){//This is the ICMP socket
                  //Handle ICMP packets here.
               }
           }
       }
```

A.2.2. Linux and Android

For unprivileged recipient of the ICMP messages an error handler must be installed. This can be done like:

```
   setsockopt (config.sockfd, SOL_IP,
               IP_RECVERR, &val, sizeof(val)) < 0);
```

In the poll() section of the code something like this needs to be there:

```
       struct msghdr msg;

       if (ufds[dataSock].revents & POLLERR) {
           if (recvmsg(sockfd, &msg, MSG_ERRQUEUE ) == -1) {
               //Ignore for now. Will get it later..
               continue;
           }
           //possible ICMP message
           //use cmsg to read the structures in msg
       }
```

Failing to call rcvmsg seems to let the msg fall through to the kernel. Looks like it will close down the socket because of the received error. So be careful!

For application with the right administrative privileges it is possible create a separate ICMP listen socket as described in the previous section. The socket() call would then look like:

```
               icmpSocket=socket(config.remoteAddr.ss_family, SOCK_RAW,
               IPPROTO_ICMP);
```

The poll() loop will be as described for OS-X and iOS. No need for a error handler.

A.2.3. Windows

The following code in select() or poll() will read and detect any incoming ICMP messages on the send socket.

```
            if (FD_ISSET(sendsocket, &read_flags)) {
                cc = recvfrom(sendsocket, receivepacket,
                    sizeof(receivepacket), 0,
                    (struct sockaddr *)&receiveaddr, (int*)&fromlen);
                if (cc < 0 && GETERRORCODE == WSAENETRESET) {
                   //ICMP packet handling here
                   //Do:
                   //inet_ntoa(receiveaddr.sin_addr));
                   //to get the address of the router sending the
                   //ICMP reply
                }
            }
```

Authors' Addresses

Paal-Erik Martinsen
Cisco Systems, Inc.
Philip Pedersens Vei 22
Lysaker, Akershus 1325
Norway

Email: palmarti@cisco.com

Dan Wing
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, California 95134
USA

Email: dwing@cisco.com