TRAM
Internet-Draft
Intended status: Standards Track
Expires: February 24, 2017
P. Martinsen
T. Reddy
D. Wing
Cisco
V. Singh
callstats.io
August 23, 2016

# Measurement of Round Trip Time and Fractional Loss Using STUN
## draft-ietf-tram-stun-path-data-05

Abstract

A host with multiple interfaces needs to choose the best interface for communication. Oftentimes, this decision is based on a static configuration and does not consider the path characteristics, which may affect the user experience.

This document describes a mechanism for an endpoint to measure the path characteristics fractional loss and RTT using Session Traversal Utilities for NAT (STUN) messages.

Table of Contents



## 1. Introduction

This document extends STUN [RFC5389] to make it possible to correlate STUN responses to specific request when re-transmits occur. This assists the client in determining path characteristics like round-trip time (RTT) and fractional packet loss.

The TRANSACTION_TRANSMIT_COUNTER attribute introduced in section Section 3.1 can be used in ICE [RFC5245] connectivity checks (STUN Binding request and response). It can also be used with TURN [RFC5766] by adding this attribute to Allocate requests and responses to measure loss and RTT between the client and respective TURN server.

ICE is a mechanism commonly used in VoIP applications to traverse NATs, and it uses a static prioritization formula to order the candidate pairs and perform connectivity checks, in which the most preferred address pairs are tested first and when a sufficiently good pair is discovered, that pair is used for communications and further connectivity tests are stopped.

When multiple paths are available for communication, the endpoint sends ICE connectivity checks across each path (candidate pair). Choosing the path with the lowest round trip time is a reasonable

approach, but re-transmits can cause an otherwise good path to appear flawed. However, STUN's retransmission algorithm [RFC5389] cannot determine the round-trip time (RTT) if a STUN request packet is re-transmitted, because each request and retransmission packet is identical. Further, several STUN requests may be sent before the connectivity between candidate pairs are ascertained (see Section 16 of [RFC5245]). To resolve the issue of identical request and response packets in a STUN transaction, this document changes the retransmission behavior for idempotent packets. In addition to determining RTT, it is also possible to get a hint regarding which path direction caused packet loss. This is achieved by defining a new STUN attribute and requires compliant STUN (TURN, ICE) endpoints to count request packets.

The mechanisms described in this document can be used by the controlling agent to influence the ICE candidate pair selection. How ICE actually will use this information to improve the active candidate pair selection is outside the scope of this document.

## 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].

This specification uses terminology defined in ICE [RFC5245] and STUN [RFC5389].

## 3. Measuring RTT and Fractional Loss

This document defines a new comprehension-optional STUN attribute TRANSACTION_TRANSMIT_COUNTER with a STUN Type TBD-CA. This type is in the comprehension-optional range, which means that STUN agents can safely ignore the attribute. If ICE is in use it will fallback to normal procedures.

If a client wishes to measure RTT, it inserts the TRANSACTION_TRANSMIT_COUNTER attribute in a STUN request. In this attribute the client sends the number of times the STUN request is transmitted with the same Transaction ID. The server would echo back the transmission count in the response so that client can distinguish between STUN responses coming from re-transmitted requests. Hence, the endpoint can use the STUN requests and responses to determine the round-trip time (RTT). The server may also convey the number of responses it has sent for the STUN request to the client. Further, this information enables the client to get a hint regarding what direction the packet loss occurred. In some cases, it is impossible to distinguish between packet reordering and packet loss. However if

this information is collected as network metrics from several clients over a longer time period, it will be easier to detect a pattern that can provide useful information.

### 3.1. TRANSACTION_TRANSMIT_COUNTER attribute

The TRANSACTION_TRANSMIT_COUNTER attribute in a STUN request takes a 32-bit value. This document updates one of the STUN message structuring rules explained in Section 6 of [RFC5389] wherein retransmit of the same request reuse the same transaction ID and are bit-wise identical to the previous request. For idempotent packets, the Req and Resp fields in the TRANSACTION_TRANSMIT_COUNTER attribute will be incremented by 1 by the client or server for every transmission with the same transaction id. Any re-transmitted STUN request MUST be bit-wise identical to the previous request except for the values in the TRANSACTION_TRANSMIT_COUNTER attribute.

The IANA assigned STUN type for the new attribute is TBD-CA.

The format of the value in TRANSACTION_TRANSMIT_COUNTER attribute in the request is:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Reserved (Padding)    |     Req       |     Resp      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: TRANSACTION_TRANSMIT_COUNTER attribute in request

The fields is described below:

Req: Number of times request is transmitted with the same transaction ID to the server.

Resp: Number of times a response with the same transaction ID is sent from the server. MUST be set to zero in requests and ignored by the receiver.

The padding is necessary to hit the 32-bit boundary needed for STUN attributes. The padding bits are ignored, but to allow for future reuse of these bits they MUST be set to 0.

### 3.2. Usage in Requests

When sending a STUN request, the TRANSACTION_TRANSMIT_COUNTER Attribute allows a client to indicate to the server that it wants to measure RTT and get a hint of the direction of any packet loss.

The client MUST populate the Req value in the TRANSACTION_TRANSMIT_COUNTER. This value MUST reflect the number of requests that have been transmitted to the server. Initial value for the first request sent is therefore 1. The first re-transmit will set the value to 2 and so on.

The Resp filed in the attribute MUST be set to zero in the request.

### 3.3. Usage in Responses

When a server receives a STUN request that includes a TRANSACTION_TRANSMIT_COUNTER attribute, it processes the request as per the STUN protocol [RFC5389] plus the specific rules mentioned here. The server checks the following:

o If the TRANSACTION_TRANSMIT_COUNTER attribute is not recognized, ignore the attribute because its type indicates that it is comprehension- optional. This should be the existing behavior as explained in section 3.1 of [RFC5389].

o The server that supports TRANSACTION_TRANSMIT_COUNTER attribute MUST echo back the Req field in the response using a TRANSACTION_TRANSMIT_COUNTER attribute.

o If the server is stateless or does not want to remember the transaction ID then it would populate value 0 for the Resp field in TRANSACTION_TRANSMIT_COUNTER attribute sent in the response. If the server is stateful then it populates the Resp field with the number of responses it has sent for the STUN request.

A client that receives a STUN response with a TRANSACTION_TRANSMIT_COUNTER can check the values in the Req field to accurately calculate the RTT if retransmits are occurring.

If the server sending the STUN response is stateless the value of the Resp field will always be 0. If the server keeps state of the numbers of STUN request with that same transaction id the value will reflect how many packets the server have seen and responded to. This gives the client a hint of which direction loss occurred. See section Section 3.4 for more details.

## 3.4. Example Operation

Example operation, when a server is stateful, is described in Figure 2. In the first case, all the requests and responses are received correctly.

In the upstream loss case, the first request is lost, but the second one is received correctly, the client on receiving the response notes that while 2 requests were sent, only one was received by the server. The server also realizes that the value in the Req field does not match the number of received requests, therefore 1 request was lost. This may also occur at startup in the presence firewalls or NATs that block unsolicited incoming traffic.

In the downstream loss case, the responses get lost, client expecting multiple responses, notes that while the server responded to 3 requests but only 1 response was received.

In the both loss case, requests and responses get lost in tandem, the server notes one request packet was not received, while the client expecting 3 responses received only one, it notes that one request and response packets were lost.

```
|     Normal    |  Upstream loss | Downstream loss |     Both loss |
| Client Server |  Client Server |  Client  Server | Client Server |
|+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-|
| 1        1,1  |  1        x    |  1         1,1  |  1        x   |
|   1,1         |                |    x            |               |
|               |  2        2,1  |  2         2,2  |  2        2,1 |
|               |    2,1         |    x            |    x          |
|               |                |  3         3,3  |  3        3,2 |
|               |                |    3,3          |    3,2        |
```

Figure 2: Retransmit Operation between client and Server

Another example could be the client sends two requests but the second request arrives at the server before the first request because of out of order delivery. In this case, the stateful server populates value 1 for the Resp field in TRANSACTION_TRANSMIT_COUNTER attribute sent in response to the second request and value 2 for the Resp field in TRANSACTION_TRANSMIT_COUNTER attribute sent in response to the first request.

The intention with this mechanism is not to carry out comprehensive and accurate measurements regarding in what direction loss is occurring. In some cases, it might not be able to distinguish the difference between downstream loss and packet reordering.

## 4. IANA Considerations

[Paragraphs in braces should be removed by the RFC Editor upon publication]

[The TRANSACTION_TRANSMIT_COUNTER attribute requires that IANA allocate a value in the "STUN attributes Registry" from the comprehension-optional range (0x8000-0xBFFF), to be replaced for TBD-CA throughout this document]

This document defines the TRANSACTION_TRANSMIT_COUNTER STUN attribute, described in Section 3. IANA has allocated the comprehension-optional codepoint TBD-CA for this attribute.

## 5. Security Considerations

Security considerations discussed in [RFC5389] are to be taken into account. STUN requires the 96 bits transaction ID to be uniformly and randomly chosen from the interval 0 .. 2**96-1, and be cryptographically strong. This is good enough security against an off-path attacker. An on-path attacker can either inject a fake response or modify the values in TRANSACTION_TRANSMIT_COUNTER attribute to mislead the client and server. This attack can be mitigated using STUN authentication. As TRANSACTION_TRANSMIT_COUNTER is expected to be used between peers using ICE, and ICE uses STUN short-term credential mechanism the risk of on-path attack influencing the messages is minimal. If TRANSACTION_TRANSMIT_COUNTER is used with Allocate request then STUN long-term credential mechanism or STUN Extension for Third-Party Authorization [RFC7635] or (D)TLS connection can be used between the TURN client and the TURN server to prevent attackers from trying to impersonate a TURN server and sending bogus TRANSACTION_TRANSMIT_COUNTER attribute in the Allocate response. However, an attacker could corrupt, remove, or delay an ICE request or response, in order to discourage that path from being used.

The information sent in any STUN packet if not encrypted can potentially be observed passively and used for reconnaissance and later attacks.

## 6. Acknowledgements

Thanks to Brandon Williams, Simon Perreault, Aijun Wang, Martin Thomson, Oleg Moskalenko, Ram Mohan R, Spencer Dawkins, Suresh Krishnan, Ben Campbell, Mirja Kuhlewind, Lionel Morand, Kathleen Moriarty and Alissa Cooper for valuable inputs and comments.

## 7. References

### 7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, DOI 10.17487/RFC5245, April 2010, <http://www.rfc-editor.org/info/rfc5245>.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, DOI 10.17487/RFC5389, October 2008, <http://www.rfc-editor.org/info/rfc5389>.

[RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, DOI 10.17487/RFC5766, April 2010, <http://www.rfc-editor.org/info/rfc5766>.

### 7.2. Informative References

[RFC7635] Reddy, T., Patil, P., Ravindranath, R., and J. Uberti, "Session Traversal Utilities for NAT (STUN) Extension for Third-Party Authorization", RFC 7635, DOI 10.17487/RFC7635, August 2015, <http://www.rfc-editor.org/info/rfc7635>.

Authors' Addresses

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

Email: palmarti@cisco.com

Tirumaleswar Reddy
Cisco Systems, Inc.
Cessna Business Park, Varthur Hobli
Sarjapur Marathalli Outer Ring Road
Bangalore, Karnataka 560103
India

Email: tireddy@cisco.com

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

Email: dwing@cisco.com

Varun Singh
CALLSTATS I/O Oy
Runeberginkatu 4c A 4
Helsinki 00100
Finland

Email: varun@callstats.io
URI: https://www.callstats.io/about