Reverse Traceroute
draft-heiwin-intarea-reverse-traceroute-00

Abstract

   Only very few troubleshooting tools exist, that universally work on
   the public internet. Ping and traceroute are the ones that are most
   frequently used, when issues arise that are outside the user's
   administrative reach. Both perform quite basic checks. Ping can
   only confirm basic reachability of an interface. Traceroute can
   enumerate routers in the forward direction of a path but remains
   blind to the reverse path. In this memo, ICMP extensions are
   defined, that allow to build a reverse traceroute tool for the public
   internet.

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 https://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 9 March 2023.

Table of Contents

1. Introduction

   Network operators use traceroute to troubleshoot problems that occur
   within their own but also outside their own network. While
   traceroute provides valuable information about the path to the target
   node, it can not gather information about the reverse path. As the
   internet proves to be widely asymmetric, meaning that forward and
   reverse path differ, traceroute's output cannot easily be used to
   find problems arising on the reverse path. This memo addresses this
   limitation by defining a simple protocol and associated extensions
   that allow network operators to use traceroute's functionality from a
   target host back towards a client.

In the past, a few attempts have been made within the IETF to provide similar functionality. [RFC1393] e.g. defines an IPv4 Traceroute option. A packet carrying that option is signaling to routers, that they should send an ICMP Traceroute message, which is also defined in [RFC1393], to the originator of that packet. When the packet finally arrives at the target host, it also uses the IP Traceroute option in its answer to enable reverse traceroute. It does so by copying the IPv4 address of the originator into the option. Routers use that address to again send ICMP Traceroute message to the originator while forwarding the answer towards it. This method differs from regular traceroute in particular in its use of IP options and the need to have routers support a new feature. The IP option defined in [RFC1393] was subsequently obsoleted in [RFC6814] and the ICMP traceroute message was deprecated in [RFC6918]. The mechanism defined in this memo does not require router changes and does neither rely on IPv4 options nor on IPv6 extension headers. In fact, as much as possible, the traceroute operation as in use today is utilized to implement reverse traceroute itself.

Another attempt at remotely triggering traceroute is documented in [RFC4560]. Performing traceroute on a remote host and collecting the results is essentially reverse traceroute if the performed traceroute is towards the client that actually triggered it. The document specifies an elaborate SNMP MIP module to perform this function. It does however not restrict the host to which the traceroute can be sent, and access to SNMP functionality is typically restricted and not a good choice for a facility that is supposed to work across the public internet. The mechanism defined in this document does rely on ICMP and restricts the reverse traceroute operation to the client that issues the request.

Most basic OAM tasks on the public internet involve ICMP and the fact that ICMP is still being actively developed and enhanced shows its continued relevance and utility. [RFC4884] e.g. has made ICMP messages more extensible by defining multipart messages. ICMP has also been extended to probe interfaces similar to ping, but without the need to have bidirectional connectivity between the probing and probed interface in [RFC8335]. ICMP has also been extended to aid the operation of traceroute to, amongst others, indicate the interface where the expiring packet has been received, as that might differ from the interface that was used to send the ICMP Time Exceeded message (see [RFC5837]).

Mechanisms defining new ICMP types and codes are supposed to fall into one of two categories as per [RFC7279]. They either should serve to inform about forwarding plane anomalies or they should facilitate the discovery of dynamic information about the network. The mechanism described in this memo falls into the latter category.

1.1. Terminology

Client: The machine that sends reverse traceroute requests, i.e. the
   machine towards which a traceroute should be performed.

Server: The machine that responds to reverse traceroute requests,
   i.e. the machine that actually performs the traceroute operation.

Probe, probe packet, traceroute probe: A packet, that is sent as
   part of a traceroute operation, intended to expire at a certain
   router, causing it to respond with a ICMP Time Exceeded message.
   Typically these are UDP packets, but could also be ICMP or TCP
   packets.

1.2. Requirements Language

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 [RFC2119] .

2. Overview

Before describing protocol extensions and the detailed operation of
the protocol, the following outlines a rough sketch of how reverse
traceroute operates. The general aim is to define a protocol that
allows a host (the client) to identify the routers on a path from
another host (the server) towards itself. In addition, just as with
the classic traceroute, RTT measurements should also be part of the
information provided to the client.

Below is an illustrative example, showing why such a function could
help troubleshooting problems that are difficult to identify using
classic traceroute today.

```
                     +------+    +------+
                     |router|    |router|
                +---->|      +--->|       +-----+
                |     | B    |    | C  |     |
                |     +------+    +------+    v
  +------+    +---+--+                      +------+    +------+
  |      |    |router|                      |router|    |      |
  |client|<-->|      |                      |       |<-->|server|
  |      |    | A  |                        | F  |    |      |
  +------+    +------+                      +--+---+    +------+
                ^      +------+    +------+    |
                |      |router|    |router|    |
                +-----+        +----+       |<----+
                       | D  |    | E  |
                       +------+    +------+
```
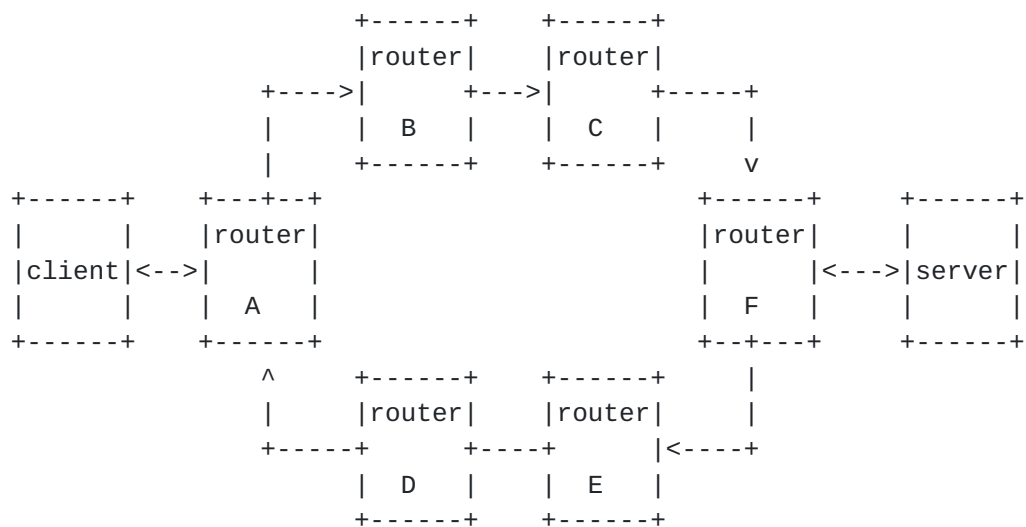
                    Figure 1: Exemplary scenario

   In the figure above we see a client and a server connected through a
   network of routers. A packet from the client towards the server is
   sent along the path consisting of routers A, B, C, and F. A packet
   from the server towards the client follows a different set of
   routers, namely, F, E, D and A. Assuming that all routers will send
   ICMP Time Exceeded messages when the TTL/hop count of an IP packet
   becomes zero and currently are not hitting any rate limiting for ICMP
   message generation, using traceroute today, the client would only see
   routers A, B, C and D as part of traceroute's output.

   To make the example more interesting, let us assume that there is a
   problem between routers D and E causing unusually high delay. Using
   traceroute today, the output might look similar to this:

   1.   A 1ms 2ms 1ms
   2.   B 3ms 3ms 2ms
   3.   C 5ms 6ms 6ms
   4.   F 340ms 320ms 350ms
   5.   server 345ms 310ms 360ms

   This could be misinterpreted as a problem between F and C because the
   starting at F, the reverse path is different and the additional delay
   starts to appear in traceroute's output.

   Reverse traceroute would make the path between the server towards the
   client visible. Therefore it's corresponding output should look
   similar to this, indicating the problem between D and E:

   1.   F 1ms 2ms 1ms
   2.   E 3ms 2ms 3ms

   3. D 300ms 320ms 310ms
   4. A 330ms 315ms 332ms
   5. client 345ms 360ms 360ms

   A client requesting a reverse traceroute does so by using new ICMP
   messages defined in this document. The same is true for reporting
   the result of a reverse traceroute operation. Everything else relies
   as much as possible on existing traceroute operations.

   Clearly, the ability to trigger a traceroute on a remote host offers
   plenty of potential concerns, in particular in terms of amplification
   attacks, when a single reverse traceroute request could trigger a
   larger traceroute operation involving tens of packets. Therefore,
   the general design of the protocol is to require the client to send a
   single request for each and every packet that the server is supposed
   to send. In other words, a single traceroute request from the client
   only triggers the server to send a single traceroute packet.

   A detailed description of the operation follows in Section 5. A
   further discussion about security considerations follows in
   Section 10.

3. Protocol design

   Reverse traceroute messages are defined for both ICMPv4 and ICMPv6.

3.1. Traceroute Request

   A traceroute request is sent by the client to the server to make the
   server send a single traceroute packet towards the client.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Code=1     |            Checksum           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Identifier          |           (Unused)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     TTL       |     Proto     |             Flow              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

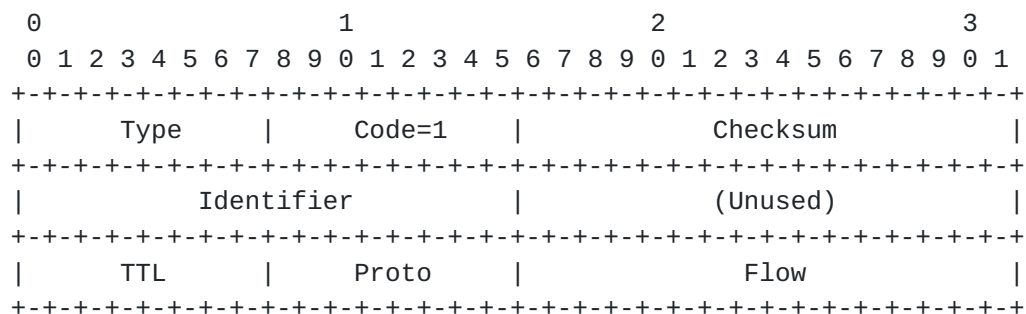   Figure 2

   Header fields:

   NOTE: see Section 9 for why those protocol field values were chose.

   * Type: The ICMP Echo type of the message.

The value for ICMPv4 is 8. The value for ICMPv6 is 128.

* Code: Reverse traceroute messages are identified by code 1.

* Identifier: The ICMP query identifier of the message.

* Unused: MUST be set to 0.

* TTL: The Time-To-Live the traceroute probe MUST use.

* Protocol: The protocol the traceroute probe MUST use.

  Supported SHOULD be ICMP, TCP and UDP. The code points are
  defined by IANA in the list of protocol numbers.

* Flow: The flow identifier the traceroute probe MUST use.

The identifier is employed by a client application to match
traceroute requests with traceroute responses. The TTL specifies the
Time-To-Live of the resulting traceroute probe.

The protocol field specifies the protocol to be employed in the
traceroute probes. A value of 0 indicates to the target that it MUST
choose a suitable protocol on its own.

The flow field contains a number that impacts the next-hop forwarding
decision in load-balancing routers. Adjusting the Flow field can
result in the probes taking different paths. In other words, pinning
the Flow field to a fixed value will make sure, that all probes with
the same Flow value will follow the same path. A value of 0
indicates to the target that it MUST choose a suitable flow
identifier on its own. See Section 4 for details.

3.2. Traceroute Response

A traceroute response is sent by a server to the client containing
information about the answer to a traceroute probe such as an ICMP
Time Exceeded message.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Code=1     |            Checksum           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Identifier           |            (Unused)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Status     |    Length     |          (Reserved)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3

Header fields:

* Type: The ICMP Echo type of the message.

   The value for ICMPv4 is 0. The value for ICMPv6 is 129.

* Code: Reverse traceroute messages are identified by code 1.

* Identifier: The ICMP query identifier of the message.

* Unused: MUST be set to 0.

* Status: Supplies information about the status of a request.

   Following values are defined:

   0x00: Success

   0x01: Invalid TTL

   0x02: Invalid Protocol

   0x03: Invalid Flow

* Length: The length of the optional error message following the
   header.

* Reserved: MUST be set to 0.

The identifier of the response MUST match the the identifier of the
corresponding request. If a malformed request is encountered, it
MUST be silently dropped. If an invalid probe configuration is
supplied by the request, the server MUST respond with a suitable
error code. Additional information about an error condition MAY be
supplied to the client by specifying an error message, whose length
MUST be reflected in the length field.

The server MUST respond with an Invalid TTL status if the requested
TTL equals zero. The server MUST choose suitable default values if
the requested protocol or flow values are zero.

If the response status indicates success, the length field MUST be
set to 0 and the message MUST be followed by a single traceroute
payload structure. The traceroute payload structure MUST NOT be
appended to the response message if the status does not indicate
success.

3.3. Traceroute Payload Structure

The traceroute payload structure contains information about a single
successful traceroute probe.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                          IPv6 Address                         +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                            Timespan                           +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

                                Figure 4
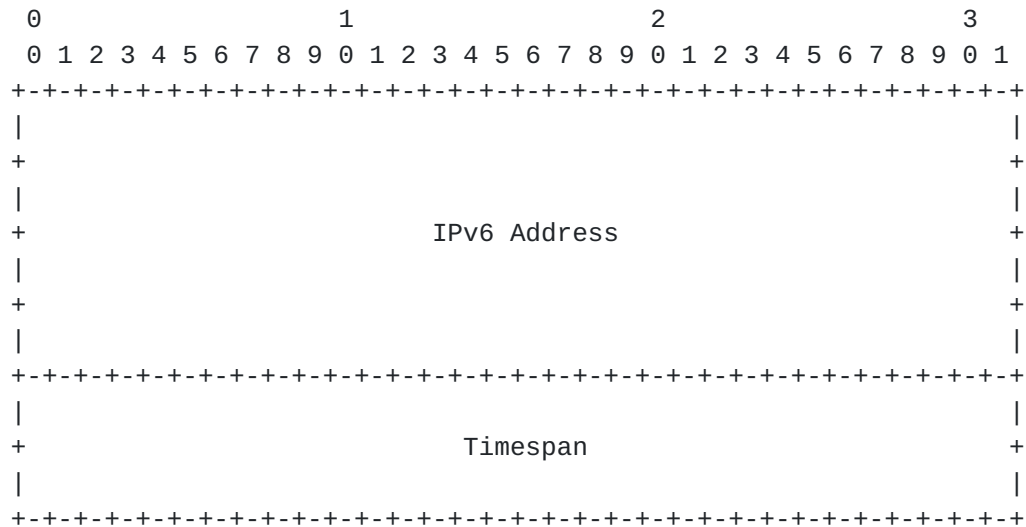
*   IPv6 Address: Address of the node that responded to the traceroute
    probe.

    The address format is defined in [RFC4291].

    IPv4 addresses are encoded as IPv4-mapped IPv6 addresses.

*   Timespan: The timespan elapsed between sending a traceroute probe
    and receiving an answer.

    The timespan is defined in nanoseconds.

4. Probes

The probes sent by a reverse traceroute server application share a
set of common information, that must be encoded inside them. The set
of information consists of:

Flow identifier: An identifier that affects the next-hop forwarding
    decision in load-balancing routers.

Probe identifier: A constant identifier that identifies the packet
    as a reverse traceroute probe.

Request identifier: An identifier that is used by the client to

match requests and responses.

[RFC0792] guarantees that Time Exceeded and Destination Unreachable
responses contain at least the first eight bytes of the original
datagram. As a consequence the needed information MUST be encoded in
those eight bytes. This section gives suggestions as to how the
state SHOULD be encoded inside different probes. In all probes
supporting port numbers, e.g. UDP and TCP, the probe identifier is
encoded into the source port. As the probe identifier is a constant
value, we suggest to assign an unused user port number (see Section 8
) for reverse traceroute server implementations. [paris-traceroute]
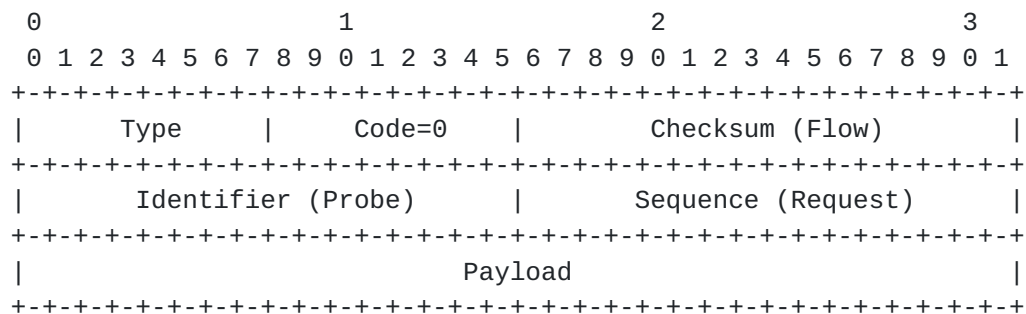provides information on load balancing and the flow identifier.

4.1. ICMP

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Code=0     |        Checksum (Flow)        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Identifier (Probe)      |       Sequence (Request)      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Payload                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5

The payload MUST be adjusted so that the flow identifier encoded
inside the checksum is valid.

4.2. UDP

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Source Port (Probe)      |     Destination Port (Flow)   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Length             |       Checksum (Request)      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Payload                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
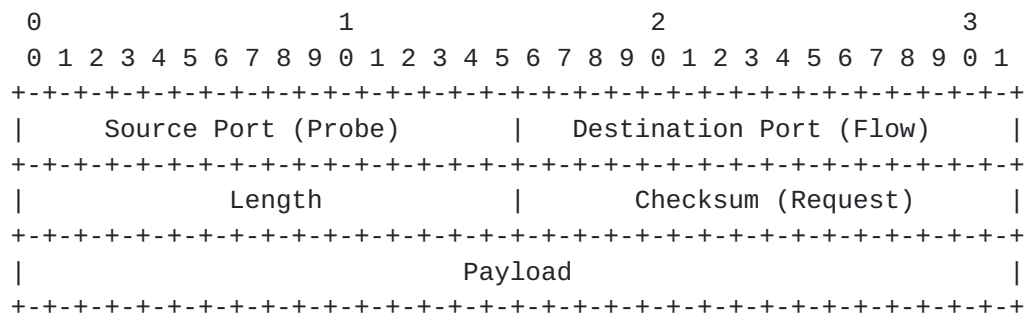
Figure 6

The payload MUST be adjusted so that the request identifier encoded
inside the checksum is valid.

4.3. TCP

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Source Port (Probe)      |    Destination Port (Flow)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Sequence Number (Request)                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Acknowledgment Number                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Offset|  Res. |     Flags       |             Window           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum              |         Urgent Pointer       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |     Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
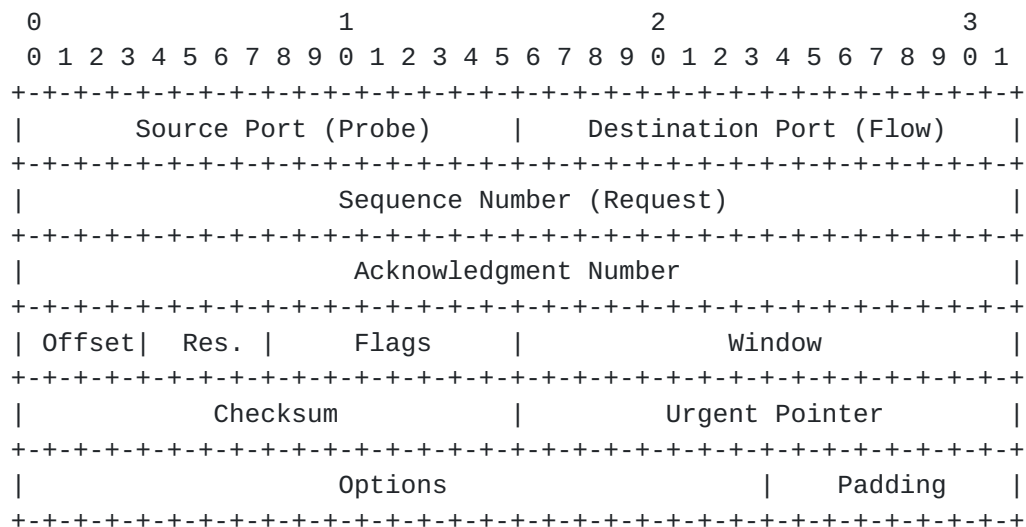
Figure 7

5. Operation

5.1. Client

Before sending traceroute requests, a client could first discovers whether a target server runs reverse traceroute. It does so by setting the TTL inside the traceroute request message to 0, thus creating an error condition. If a response with a non-zero status, signaling an error, is received, the server runs reverse traceroute. Otherwise, the originator of the response does not provide a reverse traceroute instance (or other circumstances prevent the instance to respond to or receive requests such as firewalls).

After such a potential discovery phase, the client sends regular traceroute request messages with an increasing TTL to the server. Each of these request messages SHOULD employ a unique identifier, as the server associates each traceroute probe with a single identifier. The client SHOULD pace those requests appropriately as to avoid packet loss due to e.g. rate limiting, which, for security reasons is recommended (see Section 10). If the specified probe configuration is considered invalid by the server, an error message is immediately dispatched to the client. The client examines the status in order to find the cause of the error and prints the optional error message inside the response. If the error occured due to an invalid protocol or flow, the client can set the respective value to zero in order to ask the server to choose a suitable default value on its own.

If the status indicates success, the payload data structure
immediately following the header is examined to retrieve the round-
trip-time encoded in the timespan field and the address of the node
that responded to the traceroute probe. If no response to a
traceroute request message is received, then the server e.g. did not
receive a response to the probes or timed out before a response was
received or some error occured. In either case the client is not
notified and SHOULD assume that the traceroute probe was not
answered.

[5.2](). **Server**

If a request with an invalid configuration was received by the
server, it MUST respond with a traceroute response messages
indicating a suitable error status and MAY supplement it with an
error message immediately following the header, whose length is
encoded in the corresponding field.

If a request with a valid configuration was received by the server,
the server creates a new session entry consisting of the source
address and query identifier of the request. If the resulting
session entry is already present, the associated traceroute request
is discarded without further action. Otherwise the server creates a
session state consisting of at least the current timestamp as well.
The server stores the session entry and associated session state for
later retrieval. A probe with the configuration supplied in the
request is formed and sent back towards the client. Then a timer
associated with the current session is configured and started.

If a response to a sent traceroute probe is received, a session entry
is formed in one of the two following ways depending on the response:

1. ICMP error messages from routers:

    The original IP header and the first eight bytes of the
    traceroute probe are contained in the payload of these messages.

    The address for the session entry is the destination address of
    the original IP header.

    The identifier for the session entry is identifier probe specific
    and encoded in the first eight bytes of the probe.

2. Response from client:

    The address for the session entry is the source address of the
    response.

The identifier for the session entry is probe specific.

If such a session entry exists, the session state associated with it is retrieved. The round-trip-time is computed by subtracting the previous timestamp inside the session state from the current timestamp. The node address is the source address of the response packet. A traceroute payload structure is appended to the response message and filled with both timestamp and node address and sent back towards the client with a status indicating success.

If no probe response is received before a timeout occurs, the server MUST silently discard the affected session entry and associated session state.

6. Updates to RFC792

This document extends the ICMP Echo Request and ICMP Echo Response messages with code 1. It redefines the Sequence field of ICMP Echo Request and ICMP Echo Response messages for code 1.

7. Updates to RFC4443

This document extends the ICMPv6 Echo Request and ICMPv6 Echo Response messages with code 1. It redefines the Sequence field of ICMPv6 Echo Request and ICMPv6 Echo Response messages for code 1.

8. IANA Considerations

If this document is to be published as an RFC, IANA is asked to:

* assign a unique user port number to be encoded as the probe identifier in the source port of reverse traceroute probes (Section 4)

* extend type 8 with code 1 as reverse traceroute request in the list of ICMP Parameters

* extend type 0 with code 1 as reverse traceroute response in the list of ICMP Parameters

* extend type 128 with code 1 as reverse traceroute request in the list of ICMPv6 Parameters

* extend type 129 with code 1 as reverse traceroute response in the list of ICMPv6 Parameters

9. Discussion

The ICMP messages defined in this memo use the ICMP codes of Echo
request and Echo reply but use different codes. The main reason for
this is that by using these particular codes, reverse traceroute has
a higher chance of being immediately deployable on the public
internet, as middleboxes are familiar with that code and especially a
large number of NATs could readily translate these packets.

10. Security Considerations

The reverse traceroute server needs to maintain state to store
request identifiers and timestamps for probes, which renders the
implementation vulnerable to state exhaustion attacks. Thus,
implementations SHOULD define an upper limit for the number of
reverse traceroute sessions. Additionally, rate limiting SHOULD be
employed to further decrease the effectiveness of said exploits. By
supplementing such measures with a low enough timeout interval for
session state cleanup, the threat posed by state exhaustion attacks
is significantly diminished.

The details of the probe encoding scheme proposed in Section 4 should
be carefully considered. The flow identifier specified inside
reverse traceroute requests is encoded into the destination ports for
UDP and TCP. If an operator deems the control of a probe's
destination port as a security threat, the reverse traceroute server
SHOULD be configured to allow only a single flow identifier.

As the probe encoding scheme uses a four-bytes payload to balance
changes to the checksum, a malicious client could create reverse
traceroute requests that carry a known payload. Combined with the
control over the destination probes destination port and employment
of IP-Spoofing, the probes could be used to deliver a malicious
payload to a service running on the spoofed host.

11. Acknowledgments

12. References

12.1. 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,
           <https://www.rfc-editor.org/info/rfc2119>.

   [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing
              Architecture", RFC 4291, DOI 10.17487/RFC4291, February
              2006, <https://www.rfc-editor.org/info/rfc4291>.

   [RFC4884] Bonica, R., Gan, D., Tappan, D., and C. Pignataro,
              "Extended ICMP to Support Multi-Part Messages", RFC 4884,
              DOI 10.17487/RFC4884, April 2007,
              <https://www.rfc-editor.org/info/rfc4884>.

12.2. Informative References

   [paris-traceroute]
              Augustin, B., Cuvellier, X., Orgogozo, B., Viger, F.,
              Friedman, T., Latapy, M., Magnien, C., and R. Teixeira,
              "Avoiding traceroute anomalies with Paris traceroute",
              IMC 06, October 2006.

   [RFC0792] Postel, J., "Internet Control Message Protocol", STD 5,
              RFC 792, DOI 10.17487/RFC0792, September 1981,
              <https://www.rfc-editor.org/info/rfc792>.

   [RFC1393] Malkin, G., "Traceroute Using an IP Option", RFC 1393,
              DOI 10.17487/RFC1393, January 1993,
              <https://www.rfc-editor.org/info/rfc1393>.

   [RFC4560] Quittek, J., Ed. and K. White, Ed., "Definitions of
              Managed Objects for Remote Ping, Traceroute, and Lookup
              Operations", RFC 4560, DOI 10.17487/RFC4560, June 2006,
              <https://www.rfc-editor.org/info/rfc4560>.

   [RFC5837] Atlas, A., Ed., Bonica, R., Ed., Pignataro, C., Ed., Shen,
              N., and JR. Rivers, "Extending ICMP for Interface and
              Next-Hop Identification", RFC 5837, DOI 10.17487/RFC5837,
              April 2010, <https://www.rfc-editor.org/info/rfc5837>.

   [RFC6814] Pignataro, C. and F. Gont, "Formally Deprecating Some IPv4
              Options", RFC 6814, DOI 10.17487/RFC6814, November 2012,
              <https://www.rfc-editor.org/info/rfc6814>.

   [RFC6918] Gont, F. and C. Pignataro, "Formally Deprecating Some
              ICMPv4 Message Types", RFC 6918, DOI 10.17487/RFC6918,
              April 2013, <https://www.rfc-editor.org/info/rfc6918>.

   [RFC7279] Shore, M. and C. Pignataro, "An Acceptable Use Policy for
              New ICMP Types and Codes", BCP 189, RFC 7279,
              DOI 10.17487/RFC7279, May 2014,
              <https://www.rfc-editor.org/info/rfc7279>.

   [RFC8335]   Bonica, R., Thomas, R., Linkova, J., Lenart, C., and M.
               Boucadair, "PROBE: A Utility for Probing Interfaces",
               RFC 8335, DOI 10.17487/RFC8335, February 2018,
               <https://www.rfc-editor.org/info/rfc8335>.

Authors' Addresses

   Valentin Heinrich
   University of Applied Sciences Augsburg
   Email: valentin.heinrich@hs-augsburg.de

   Rolf Winter
   University of Applied Sciences Augsburg
   Email: rolf.winter@hs-augsburg.de