H. Asaeda
Keio University
T. Jinmei
ISC
W. Fenner
Arastra, Inc.
S. Casner
Packet Design, Inc.
March 9, 2009

Mtrace Version 2: Traceroute Facility for IP Multicast
draft-ietf-mboned-mtrace-v2-03

Status of this Memo

Copyright Notice

Abstract

This document describes the IP multicast traceroute facility. Unlike unicast traceroute, multicast traceroute requires special implementations on the part of routers. This specification describes the required functionality in multicast routers, as well as how management applications can use the router functionality.

Table of Contents

1. Introduction

The unicast "traceroute" program allows the tracing of a path from one machine to another. The key mechanism for unicast traceroute is the ICMP TTL exceeded message, which is specifically precluded as a response to multicast packets. On the other hand, the multicast traceroute facility allows the tracing of an IP multicast routing paths. In this document, we specify the multicast "traceroute" facility to be implemented in multicast routers and accessed by diagnostic programs. The multicast traceroute described in this document named as mtrace version 2 or mtrace2 provides additional information about packet rates and losses that the unicast traceroute cannot, and generally requires fewer packets to be sent.

- o. To be able to trace the path that a packet would take from some source to some destination.

- o. To be able to isolate packet loss problems (e.g., congestion).

- o. To be able to isolate configuration problems (e.g., TTL threshold).

- o. To minimize packets sent (e.g. no flooding, no implosion).

This document supports both IPv4 and IPv6 multicast traceroute facility. The protocol design, concept, and program behavior are same between IPv4 and IPv6 mtrace2. While the original IPv4 multicast traceroute, mtrace, the query and response messages are implemented as IGMP messages [4], all mtrace2 messages are carried on UDP. The packet formats of IPv4 and IPv6 mtrace2 are different because of the different address families, but the syntax is similar.

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 [1].

Since multicast traceroutes flow in the opposite direction to the data flow, we refer to "upstream" and "downstream" with respect to data, unless explicitly specified.

Incoming interface:
The interface on which traffic is expected from the specified source and group.

Outgoing interface:
The interface on which traffic is forwarded from the specified source and group toward the destination. It is the interface on which the multicast traceroute Request was received.

Previous-hop router:
The router that is on the link attached to the Incoming Interface and is responsible for forwarding traffic for the specified source and group.

Group state:
It is the state in which a shared-tree protocol (e.g., PIM-SM [9]) running on a router chooses the previous-hop router toward the core router or Rendezvous Point (RP) as its parent router. In this state, source-specific state is not available for the corresponding multicast address on the router.

Source-specific state:
It is the state in which a routing protocol running on a router chooses the path that would be followed for a source-specific join.

ALL-[protocol]-ROUTERS.MCAST.NET:
It is a dedicated multicast address for a multicast router to communicate with other routers that are working with the same routing protocol. For instance,the address of ALL-PIM-ROUTERS.MCAST.NET is '224.0.0.13' for IPv4 and 'ff02::d' for IPv6.

3. Overview

Given a multicast distribution tree, tracing from a source to a multicast destination is hard, since you don't know down which branch of the multicast tree the destination lies. This means that you have to flood the whole tree to find the path from one source to one destination. However, walking up the tree from destination to source is easy, as most existing multicast routing protocols know the previous hop for each source. Tracing from destination to source can involve only routers on the direct path.

The party requesting the traceroute sends a traceroute Query packet to the last-hop multicast router for the given destination. The last-hop router turns the Query into a Request packet by adding a response data block containing its interface addresses and packet statistics, and then forwards the Request packet via unicast to the router that it believes is the proper previous hop for the given source and group. Each hop adds its response data to the end of the Request packet, then unicast forwards it to the previous hop. The first hop router (the router that believes that packets from the source originate on one of its directly connected networks) changes the packet type to indicate a Response packet and sends the completed response to the response destination address. The response may be returned before reaching the first hop router if a fatal error condition such as "no route" is encountered along the path.

Multicast traceroute uses any information available to it in the router to attempt to determine a previous hop to forward the trace towards. Multicast routing protocols vary in the type and amount of state they keep; multicast traceroute endeavors to work with all of them by using whatever is available. For example, if a PIM-SM router is on the (*,G) tree, it chooses the parent towards the RP as the previous hop. In these cases, no source/group-specific state is available, but the path may still be traced.

4. Packet Formats

Mtrace2 message is encoded in TLV format. If an implementation receives a TLV whose length exceeds the TLV length specified in the Length field, the TLV SHOULD be accepted but any additional data SHOULD be ignored.

4.1. Mtrace2 TLV format

```
  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      |             Length            |  Value ....   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type (8 bits)

Length (16 bits)

Value (variable length)

4.2. Defined TLVs

The following TLV Types are defined:

Code	Type
1	Mtrace2 Query
2	Mtrace2 Response
3	Mtrace2 Standard Response Block
4	Mtrace2 Augmented Response Block

An mtrace2 message MUST contain one Mtrace2 Query or Response. An mtrace2 message MAY contain one or multiple Mtrace2 Standard and Augmented Responses. A multicast router that sends mtrace2 request MUST NOT contain multiple Mtrace2 Standard blocks but MAY contain multiple Augmented Response blocks.

The type field is defined to be "0x1" for mtrace2 queries and requests. The type field is changed to "0x2" when the packet is completed and sent as a response from the first hop router to the querier. Two codes are required so that multicast routers will not attempt to process a completed response in those cases where the initial query was issued from a router.

5. Mtrace2 Query Header

The mtrace2 message is carried as a UDP packet. The UDP source port is uniquely selected by the local host operating system. The UDP destination port is the IANA reserved mtrace2 port number (see Section 13). The UDP checksum MUST be valid in mtrace2 messages.

The mtrace2 message includes the common mtrace2 Query header as follows. The header is only filled in by the originator of the mtrace2 Query; intermediate routers MUST NOT modify any of the fields.

```
    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
                                                   +-+-+-+-+-+-+-+-+
                                                   |    # hops     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   |                      Multicast Address                        |
   |                                                               |
   +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
   |                                                               |
   |                        Source Address                         |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   |                      Destination Address                      |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |             Query ID              |         Client Port #     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1

5.1. # hops: 8 bits

This field specifies the maximum number of hops that the requester wants to trace. If there is some error condition in the middle of the path that keeps the mtrace2 request from reaching the first-hop router, this field can be used to perform an expanding-ring search to trace the path to just before the problem.

5.2. Multicast Address

This field specifies the 32 bits length IPv4 or 128 bits length IPv6 multicast address to be traced, or is filled with "all 1" in case of IPv4 or with the unspecified address (::) in case of IPv6 if no group-specific information is desired. Note that non-group-specific

mtrace2 MUST specify source address.

5.3. Source Address

This field specifies the 32 bits length IPv4 or 128 bits length IPv6 address of the multicast source for the path being traced, or is filled with "all 1" in case of IPv4 or with the unspecified address (::) in case of IPv6 if no source-specific information is desired. Note that non-source-specific traceroutes may not be possible with certain multicast routing protocols.

5.4. Destination Address

This field specifies the 32 bits length IPv4 or 128 bits length IPv6 address of the multicast receiver for the path being traced. The trace starts at this destination and proceeds toward the traffic source.

5.5. Query ID: 16 bits

This field is used as a unique identifier for this traceroute request so that duplicate or delayed responses may be detected and to minimize collisions when a multicast response address is used.

5.6. Client Port

Mtrace2 response is sent back to the address specified in a Response Address field. This field specifies the UDP port number the router will send Mtrace2 Response. This client port number MUST NOT be changed by any router.

6. IPv4 Mtrace2 Standard Response Block

Each intermediate IPv4 router in a trace path appends "response data block" to the forwarded trace packet. The standard response data block looks as follows.

```
  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
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                      Query Arrival Time                       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                  Incoming Interface Address                   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                  Outgoing Interface Address                   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                  Previous-Hop Router Address                  |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 .           Input packet count on incoming interface            .
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 .           Output packet count on outgoing interface           .
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                                                               |
 .      Total number of packets for this source-group pair       .
 |                                                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |               |               |M| |           |               |
 | Rtg Protocol  |    Fwd TTL    |B|S| Src Mask  |Forwarding Code|
 |               |               |Z| |           |               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

6.1. Query Arrival Time: 32 bits

The Query Arrival Time is a 32-bit NTP timestamp specifying the arrival time of the traceroute request packet at this router. The 32-bit form of an NTP timestamp consists of the middle 32 bits of the full 64-bit form; that is, the low 16 bits of the integer part and the high 16 bits of the fractional part.

The following formula converts from a UNIX timeval to a 32-bit NTP timestamp:

```
query_arrival_time
= (tv.tv_sec + 32384) << 16 + ((tv.tv_usec << 10) / 15625)
```

The constant 32384 is the number of seconds from Jan 1, 1900 to Jan 1, 1970 truncated to 16 bits. ((tv.tv_usec << 10) / 15625) is a reduction of ((tv.tv_usec / 100000000) << 16).

6.2. Incoming Interface Address: 32 bits

This field specifies the address of the interface on which packets from this source and group are expected to arrive, or 0 if unknown.

6.3. Outgoing Interface Address: 32 bits

This field specifies the address of the interface on which packets from this source and group flow to the specified destination, or 0 if unknown.

6.4. Previous-Hop Router Address: 32 bits

This field specifies the router from which this router expects packets from this source. This may be a multicast group (e.g. ALL-[protocol]-ROUTERS.MCAST.NET) if the previous hop is not known because of the workings of the multicast routing protocol. However, it should be 0 if the incoming interface address is unknown.

6.5. Input packet count on incoming interface: 64 bits

This field contains the number of multicast packets received for all groups and sources on the incoming interface, or "all 1" if no count can be reported. This counter may have the same value as ifHCInMulticastPkts from the IF-MIB [14] for this interface.

6.6. Output packet count on incoming interface: 64 bits

This field contains the number of multicast packets that have been transmitted or queued for transmission for all groups and sources on the outgoing interface, or "all 1" if no count can be reported. This counter may have the same value as ifHCOutMulticastPkts from the IF-MIB for this interface.

6.7. Total number of packets for this source-group pair: 64 bits

This field counts the number of packets from the specified source forwarded by this router to the specified group, or "all 1" if no count can be reported. If the S bit is set, the count is for the source network, as specified by the Src Mask field. If the S bit is set and the Src Mask field is 63, indicating no source-specific state, the count is for all sources sending to this group. This counter should have the same value as ipMcastRoutePkts from the IPMROUTE-STD-MIB [15] for this forwarding entry.

6.8. Rtg Protocol: 8 bits

This field describes the routing protocol in use between this router and the previous-hop router. Specified values include:

0	Unknown
1	PIM
2	PIM using special routing table
3	PIM using a static route
4	PIM using MBGP route
5	PIM using state created by Assert processing
6	Bi-directional PIM
7	IGMP/MLD proxy
8	AMT relay
9	AMT gateway
10	AMT gateway with IGMP/MLD proxy

To obtain these values, multicast routers access to ipMcastRouteProtocol, ipMcastRouteRtProtocol, and ipMcastRouteRtType in IpMcastRouteEntry specified in IPMROUTE-STD-MIB [15], and combine these MIB values to recognize above routing protocol values.

6.9. Fwd TTL: 8 bits

This field contains the TTL that a packet is required to have before it will be forwarded over the outgoing interface.

6.10. MBZ: 1 bit

Must be zeroed on transmission and ignored on reception.

6.11. S: 1 bit

This S bit indicates that the packet count for the source-group pair is for the source network, as determined by masking the source address with the Src Mask field.

6.12. Src Mask: 6 bits

This field contains the number of 1's in the netmask this router has for the source (i.e. a value of 24 means the netmask is 0xffffff00). If the router is forwarding solely on group state, this field is set to 63 (0x3f).

6.13. Forwarding Code: 8 bits

This field contains a forwarding information/error code. Section 9.2 explains how and when the forwarding code is filled. Defined values

are as follows;

Value	Name	Description
0x00	NO_ERROR	No error
0x01	WRONG_IF	Mtrace2 request arrived on an interface to which this router would not forward for this source, group, destination.
0x02	PRUNE_SENT	This router has sent a prune upstream which applies to the source and group in the traceroute request.
0x03	PRUNE_RCVD	This router has stopped forwarding for this source and group in response to a request from the next hop router.
0x04	SCOPED	The group is subject to administrative scoping at this hop.
0x05	NO_ROUTE	This router has no route for the source or group and no way to determine a potential route.
0x06	WRONG_LAST_HOP	This router is not the proper last-hop router.
0x07	NOT_FORWARDING	This router is not forwarding this source, group out the outgoing interface for an unspecified reason.
0x08	REACHED_RP	Reached Rendezvous Point or Core
0x09	RPF_IF	Mtrace2 request arrived on the expected RPF interface for this source and group.
0x0A	NO_MULTICAST	Mtrace2 request arrived on an interface which is not enabled for multicast.
0x0B	INFO_HIDDEN	One or more hops have been hidden from this trace.
0x81	NO_SPACE	There was not enough room to insert another response data block in the packet.

0x82	OLD_ROUTER	The previous-hop router does not understand mtrace2 requests.
0x83	ADMIN_PROHIB	Mtrace2 is administratively prohibited.

Note that if a router discovers there is not enough room in a packet to insert its response, it puts the NO_SPACE error code in the previous router's Forwarding Code field, overwriting any error the previous router placed there. After the router sends the response to the Destination Address in the header, the router continues the mtrace2 query by sending an mtrace2 request containing the same mtrace2 query header. Section 9.3 and Section 10.8 include the details.

The 0x80 bit of the Forwarding Code is used to indicate a fatal error. A fatal error is one where the router may know the previous hop but cannot forward the message to it.

7. IPv6 Mtrace2 Standard Response Block

Each intermediate IPv6 router in a trace path appends "response data block" to the forwarded trace packet. The standard response data block looks as follows.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Query Arrival Time                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Incoming Interface ID                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Outgoing Interface ID                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
*                         Local Address                         *
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
*                         Remote Address                        *
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.           Input packet count on incoming interface            .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.           Output packet count on outgoing interface           .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.      Total number of packets for this source-group pair       .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Rtg Protocol  |     MBZ     |S|Src Prefix Len |Forwarding Code|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

7.1. Query Arrival Time: 32 bits

Same definition described in Section 6.1

7.2. Incoming Interface ID: 32 bits

This field specifies the interface ID on which packets from this source and group are expected to arrive, or 0 if unknown. This ID should be the value taken from InterfaceIndex of the IF-MIB [14] for this interface. This field is carried in network byte order.

7.3. Outgoing Interface ID: 32 bits

This field specifies the interface ID on which packets from this source and group flow to the specified destination, or 0 if unknown. This ID should be the value taken from InterfaceIndex of the IF-MIB for this interface. This field is carried in network byte order.

7.4. Local Address

This field specifies a global IPv6 address that uniquely identifies the router. A unique local unicast address [13] SHOULD NOT be used unless the router is only assigned link-local and unique local addresses. If the router is only assigned link-local addresses, its link-local address can be specified in this field.

7.5. Remote Address

This field specifies the address of the previous-hop router, which, in most cases, is a link-local unicast address for the queried source and destination addresses.

Although a link-local address does not have enough information to identify a node, it is possible to detect the previous-hop router with the assistance of Incoming Interface ID and the current router address (i.e., Local Address).

This may be a multicast group (e.g., ALL-[protocol]-ROUTERS.MCAST.NET) if the previous hop is not known because of the workings of the multicast routing protocol. However, it should be the unspecified address (::) if the incoming interface address is unknown.

7.6. Input packet count on incoming interface

Same definition described in Section 6.5

7.7. Output packet count on incoming interface

Same definition described in Section 6.6

7.8. Total number of packets for this source-group pair

This field counts the number of packets from the specified source forwarded by this router to the specified group, or "all 1" if no count can be reported. If the S bit is set, the count is for the source network, as specified by the Src Prefix Len field. If the S bit is set and the Src Prefix Len field is 255, indicating no source-specific state, the count is for all sources sending to this group.

This counter should have the same value as ipMcastRoutePkts from the IPMROUTE-STD-MIB for this forwarding entry.

7.9. Rtg Protocol: 8 bits

Same definition described in Section 6.8

7.10. MBZ: 7 bits

Must be zeroed on transmission and ignored on reception.

7.11. S: 1 bit

This S bit indicates that the packet count for the source-group pair is for the source network, as determined by masking the source address with the Src Prefix Len field.

7.12. Src Prefix Len: 8 bits

This field contains the prefix length this router has for the source. If the router is forwarding solely on group state, this field is set to 255 (0xff)

7.13. Forwarding Code: 8 bits

Same definition described in Section 6.13

8. Mtrace2 Augmented Response Block

In addition to the standard response block, a multicast router on the path will be able to add "augumented response block" when it sends the request to its upstream router or sends the response to the Response Address. This augmented response block is flexible to add various information.

```
  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              |           Value ....          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The augmented response block is always appended to mtrace2 TLV header (0x04). The 16 bits Type filed of the augmented response block is defined for various purposees, such as diagnosis (as in Section 12) and protocol verification. The packet length of the augmented response block is specified in the augmented response block TLV header as see in Section 4.1.

This document does not define any augmented response block type. Specifing how to deal with diagnosis information will be also described in separate documents.

9. Router Behavior

All of these actions are performed in addition to (NOT instead of) forwarding the packet, if applicable. E.g. a multicast packet that has TTL or the hop limit remaining MUST be forwarded normally, as MUST a unicast packet that has TTL or the hop limit remaining and is not addressed to this router.

9.1. Traceroute Query

An mtrace2 Query message is a traceroute message with no response blocks filled in, and uses TLV type 0x1 for IPv4 and IPv6 mtrace2.

9.1.1. Packet Verification

Upon receiving an mtrace2 Query message, a router must examine the Query to see if it is the proper last-hop router for the destination address in the packet. It is the proper last-hop router if it has a multicast-capable interface on the same subnet as the Destination Address and is the router that would forward traffic from the given (S,G) onto that subnet.

If the router determines that it is not the proper last-hop router, or it cannot make that determination, it does one of two things depending if the Query was received via multicast or unicast. If the Query was received via multicast, then it MUST be silently dropped. If it was received via unicast, a forwarding code of WRONG_LAST_HOP is noted and processing continues as in Section 9.2

Duplicate Query messages as identified by the tuple (IP Source, Query ID) SHOULD be ignored. This MAY be implemented using a simple 1-back cache (i.e. remembering the IP source and Query ID of the previous Query message that was processed, and ignoring future messages with the same IP Source and Query ID). Duplicate Request messages MUST NOT be ignored in this manner.

9.1.2. Normal Processing

When a router receives an mtrace2 Query and it determines that it is the proper last-hop router, it treats it like an mtrace2 Request and performs the steps listed in Section 9.2

9.2. Mtrace2 Request

An mtrace2 Request is a traceroute message with some number of response blocks filled in, and uses TLV type 0x1 for IPv4 and IPv6 mtrace2. Routers can tell the difference between Queries and Requests by checking the length of the packet.

9.2.1. Packet Verification

If the mtrace2 Request does not come from an adjacent host or router, it MUST be silently ignored. If the mtrace2 Request is not addressed to this router, or if the Request is addressed to a multicast group which is not a link-scoped group (i.e. 224/24 for IPv4, FFx2::/16 [3] for IPv6), it MUST be silently ignored.

9.2.2. Normal Processing

When a router receives an mtrace2 Request, it performs the following steps. Note that it is possible to have multiple situations covered by the Forwarding Codes. The first one encountered is the one that is reported, i.e. all "note forwarding code N" should be interpreted as "if forwarding code is not already set, set forwarding code to N".

1. If there is room in the current buffer (or the router can efficiently allocate more space to use), insert a new response block into the packet and fill in the Query Arrival Time, Outgoing Interface Address (for IPv4 mtrace2) or Outgoing Interface ID (for IPv6 mtrace2), Output Packet Count, and Fwd TTL (for IPv4 mtrace2). If there was no room, fill in the response code "NO_SPACE" in the *previous* hop's response block, and forward the packet to the address specified in the Destination Address field and continue the trace as described in Section 9.3.

2. Attempt to determine the forwarding information for the source and group specified, using the same mechanisms as would be used when a packet is received from the source destined for the group. State need not be instantiated, it can be "phantom" state created only for the purpose of the trace, such as "dry-run".

   If using a shared-tree protocol and there is no source-specific state, or if the source is specified as "all 1", group state should be used. If there is no group state or the group is specified as 0, potential source state (i.e. the path that would be followed for a source-specific Join) should be used. If this router is the Core or RP and no source-specific information is available, note an error code of REACHED_RP.

3. If no forwarding information can be determined, the router notes an error code of NO_ROUTE, sets the remaining fields that have not yet been filled in to zero, and then forwards the packet to the requester as described in Section 9.3.

4. Fill in the Incoming Interface Address, Previous-Hop Router Address, Input Packet Count, Total Number of Packets, Routing Protocol, S, and Src Mask from the forwarding information that was determined.

5. If mtrace2 is administratively prohibited or the previous hop router does not understand mtrace2 requests, note the appropriate forwarding code (ADMIN_PROHIB or OLD_ROUTER). If mtrace2 is administratively prohibited and any of the fields as filled in step 4 are considered private information, zero out the applicable fields. Then the packet is forwarded to the requester as described in Section 9.3.

6. If the reception interface is not enabled for multicast, note forwarding code NO_MULTICAST. If the reception interface is the interface from which the router would expect data to arrive from the source, note forwarding code RPF_IF. Otherwise, if the reception interface is not one to which the router would forward data from the source to the group, a forwarding code of WRONG_IF is noted.

7. If the group is subject to administrative scoping on either the Outgoing or Incoming interfaces, a forwarding code of SCOPED is noted.

8. If this router is the Rendezvous Point or Core for the group, a forwarding code of REACHED_RP is noted.

9. If this router has sent a prune upstream which applies to the source and group in the mtrace2 Request, it notes forwarding code PRUNE_SENT. If the router has stopped forwarding downstream in response to a prune sent by the next hop router, it notes forwarding code PRUNE_RCVD. If the router should normally forward traffic for this source and group downstream but is not, it notes forwarding code NOT_FORWARDING.

10. The packet is then sent on to the previous hop or the Destination Address as described in Section 9.3.

9.3. Forwarding Mtrace2 Requests

If the Previous-hop router is known for this request and the number of response blocks is less than the number requested (i.e., the "# hops" field in mtrace2 header), the packet is sent to that router. If the Incoming Interface is known but the Previous-hop router is not known, the packet is sent to an appropriate multicast address on the Incoming Interface. The appropriate multicast address may depend on the routing protocol in use, MUST be a link-scoped group (i.e. 224/24

for IPv4, FF02::/16 for IPv6), MUST NOT be ALL-SYSTEMS.MCAST.NET (224.0.0.1) for IPv4 and All Nodes Address (FF02::1) for IPv6, and MAY be ALL-ROUTERS.MCAST.NET (224.0.0.2) for IPv4 or All Routers Address (FF02::2) for IPv6 if the routing protocol in use does not define a more appropriate group. Otherwise, it is sent to the Destination Address in the header.

When the NO_SPACE error occurs, the multicast routers sends back the mtrace2 response with contained data and the NO_SPACE error code to the address specified in the Destination Address field in the mtrace2 query header, and continues the mtrace2 query by sending an mtrace2 request containing the same mtrace2 query header except the # hops field and its response block. The # hops field must be decreased according to the number of standard response blocks in the mtrace2 request received by the router.

9.4. Sending Mtrace2 Responses

9.4.1. Destination Address

An mtrace2 Response must be sent to the address specified in the Destination Address field in the mtrace2 query header.

9.4.2. Source Address

An mtrace2 Response must be sent with the address of the router's reception interface.

9.5. Hiding Information

Information about a domain's topology and connectivity may be hidden from multicast traceroute requests. The exact mechanism is not specified here; however, the INFO_HIDDEN forwarding code may be used to note that, for example, the incoming interface address and packet count are for the entrance to the domain and the outgoing interface address and packet count are the exit from the domain. The source-group packet count may be from either router or not specified (all 1).

10. Client Behavior

10.1. Sending Mtrace2 Query

When the destination of the mtrace2 is the machine running the client, the mtrace2 Query packet can be sent to the ALL-ROUTERS.MCAST.NET (224.0.0.2) for IPv4 or All Routers Address (FF02::2) for IPv6. This will ensure that the packet is received by the last-hop router on the subnet. Otherwise, if the proper last-hop router is known for the mtrace2 destination, the Query could be unicasted to that router. Otherwise, the Query packet should be multicasted to the group being queried; if the destination of the mtrace2 is a member of the group, this will get the Query to the proper last-hop router. In this final case, the packet should contain the Router Alert option [7][8], to make sure that routers that are not members of the multicast group notice the packet.

See also Section 10.4 on determining the last-hop router.

10.2. Determining the Path

The client could send a small number of initial query messages with a large "# hops" field, in order to try to trace the full path. If this attempt fails, one strategy is to perform a linear search (as the traditional unicast traceroute program does); set the "# hops" field to 1 and try to get a response, then 2, and so on. If no response is received at a certain hop, the hop count can continue past the non-responding hop, in the hopes that further hops may respond. These attempts should continue until a user-defined timeout has occurred.

See also Section 10.5 and Section 10.6 on receiving the results of a trace.

10.3. Collecting Statistics

After a client has determined that it has traced the whole path or as much as it can expect to (see Section 10.7), it might collect statistics by waiting a short time and performing a second trace. If the path is the same in the two traces, statistics can be displayed as described in Section 12.3 and Section 12.4.

10.4. Last Hop Router

The mtrace2 querier may not know which is the last hop router, or that router may be behind a firewall that blocks unicast packets but passes multicast packets. In these cases, the mtrace2 request should be multicasted to ALL-ROUTERS.MCAST.NET (224.0.0.2) for IPv4 or All

Routers Address (FF02::2) for IPv6. All routers except the correct last hop router should ignore any mtrace2 request received via multicast. Mtrace2 requests which are multicasted to the group being traced must include the Router Alert option[7][8].

Another alternative is to unicast to the trace destination. Mtrace2 requests which are unicasted to the trace destination must include the Router Alert option, in order that the last-hop router is aware of the packet.

10.5. First Hop Router

The IANA assigned 224.0.1.32, MTRACE.MCAST.NET as the default multicast group for IPv4 mtrace responses, in order to support mtrace queriers that are not unicast reachable from the first hop router. However, mtrace2 does not reserve any IPv4/IPv6 multicast addresses for mtrace2 responses. Every mtrace2 response is sent to the unicast address specified in the Destination Address field of the mtrace2 query header.

The mtrace2 querier may be behind a gateway (e.g., a NAT or firewall) that blocks unicast packets. When the gateway receives mtrace2 query from an adjacent host that is not unicast reachable, it sends back the mtrace2 response with contained data and the NO_SPACE error code to the address specified in the Destination Address field in the mtrace2 query header. And to continue the mtrace2 query, the gateway prepares the mtrace2 query containing the same mtrace2 query header, except the # hops field, the Destination Address, and its response block.

The # hops field must be decreased according to the number of standard response blocks in the mtrace2 request received by the gateway. And the original Destination Address MUST be replaced with the gateway address that is unicast reachable from the first hop router. After that, the gateway restarts the mtrace2 query by sending an mtrace2 request. When the gateway receives the mtrace2 response from the last hop router, it MUST forward the mtrace2 response back to the original mtrace2 querier with the original Destination Address in the mtrace2 query header.

10.6. Broken Intermediate Router

A broken intermediate router might simply not understand mtrace2 packets, and drop them. The querier would then get no response at all from its mtrace2 requests. It should then perform a hop-by-hop search by setting the number of responses field until it gets a response (both linear and binary search are options, but binary is likely to be slower because a failure requires waiting for a

timeout).

10.7. Mtrace2 Termination

When performing an expanding hop-by-hop trace, it is necessary to determine when to stop expanding.

10.7.1. Arriving at source

A trace can be determined to have arrived at the source if the Incoming Interface of the last router in the trace is non-zero, but the Previous Hop router is zero.

10.7.2. Fatal error

A trace has encountered a fatal error if the last Forwarding Error in the trace has the 0x80 bit set.

10.7.3. No previous hop

A trace can not continue if the last Previous Hop in the trace is set to 0.

10.7.4. Traceroute shorter than requested

If the trace that is returned is shorter than requested (i.e. the number of response blocks is smaller than the "# hops" field), the trace encountered an error and could not continue.

10.8. Continuing after an error

When the NO_SPACE error occurs, as described in Section 9.3, the multicast routers sends back the mtrace2 response to the address specified in the Destination Address field in the mtrace2 query header. In this case, the mtrace2 client may receive multiple mtrace2 responses from different routers (along the path). After the client receives multiple mtrace2 response messages, it integrates (i.e. constructs) them as a single mtrace2 response message.

If a trace times out, it is likely to be because a router in the middle of the path does not support multicast traceroute. That router's address will be in the Previous Hop field of the last entry in the last response packet received. A client may be able to determine (via mrinfo or SNMP [13][15]) a list of neighbors of the non-responding router. If desired, each of those neighbors could be probed to determine the remainder of the path. Unfortunately, this heuristic may end up with multiple paths, since there is no way of knowing what the non-responding router's algorithm for choosing a

previous-hop router is. However, if all paths but one flow back towards the non-responding router, it is possible to be sure that this is the correct path.

11. Protocol-Specific Considerations

11.1. PIM-SM

When a multicast traceroute reaches a PIM-SM RP and the RP does not forward the trace on, it means that the RP has not performed a source-specific join so there is no more state to trace. However, the path that traffic would use if the RP did perform a source-specific join can be traced by setting the trace destination to the RP, the trace source to the traffic source, and the trace group to 0. This trace Query may be unicasted to the RP.

11.2. Bi-Directional PIM

Bi-directional PIM [10] is a variant of PIM-SM that builds bi-directional shared trees connecting multicast sources and receivers. Along the bi-directional shared trees, multicast data is natively forwarded from sources to the RPA (Rendezvous Point Address) and from the RPA to receivers without requiring source-specific state. In contrast to PIM-SM, RP always has the state to trace.

A Designated Forwarder (DF) for a given RPA is in charge of forwarding downstream traffic onto its link, and forwarding upstream traffic from its link towards the RPL (Rendezvous Point Link) that the RPA belongs to. Hence mtrace2 reports DF addresses or RPA along the path.

11.3. PIM-DM

Routers running PIM Dense Mode do not know the path packets would take unless traffic is flowing. Without some extra protocol mechanism, this means that in an environment with multiple possible paths with branch points on shared media, multicast traceroute can only trace existing paths, not potential paths. When there are multiple possible paths but the branch points are not on shared media, the previous hop router is known, but the last hop router may not know that it is the appropriate last hop.

When traffic is flowing, PIM Dense Mode routers know whether or not they are the last-hop forwarder for the link (because they won or lost an Assert battle) and know who the previous hop is (because it won an Assert battle). Therefore, multicast traceroute is always able to follow the proper path when traffic is flowing.

11.4. IGMP/MLD Proxy

When a mtrace2 Query packet reaches an incoming interface of IGMP/MLD Proxy [11], it put a WRONG_IF (0x01) value in Forwarding Code of

mtrace2 standard response block (as in Section 6.13) and sends the mtrace2 response back to the Response Address. When a mtrace2 Query packet reaches an outgoing interface of IGMP/MLD proxy, it is forwarded through its incoming interface towards the upstream router.

11.5. AMT

AMT [12] provides the multicast connectivity to the unicast-only inter-network. To do this, multicast packets being sent to or from a site are encapsulated in unicast packets. When a mtrace2 query packet reaches an AMT pseudo-interface of an AMT gateway, the AMT gateway encapsulats it to a particular AMT relay reachable across the unicast-only infrastructure. Then the AMT relay decapsulates the mtrace2 query packet and forwards the mtrace2 request to the appropriate multicast router.

12. Problem Diagnosis

12.1. Forwarding Inconsistencies

The forwarding error code can tell if a group is unexpectedly pruned or administratively scoped.

12.2. TTL or Hop Limit Problems

By taking the maximum of hops (from source + forwarding TTL (or hop limit) threshold) over all hops, it is possible to discover the TTL or hop limit required for the source to reach the destination.

12.3. Packet Loss

By taking two traces, it is possible to find packet loss information by comparing the difference in input packet counts to the difference in output packet counts for the specified source-group address pair at the previous hop. On a point-to-point link, any difference in these numbers implies packet loss. Since the packet counts may be changing as the mtrace2 query is propagating, there may be small errors (off by 1 or 2 or more) in these statistics. However, these errors will not accumulate if multiple traces are taken to expand the measurement period. On a shared link, the count of input packets can be larger than the number of output packets at the previous hop, due to other routers or hosts on the link injecting packets. This appears as "negative loss" which may mask real packet loss.

In addition to the counts of input and output packets for all multicast traffic on the interfaces, the response data includes a count of the packets forwarded by a node for the specified source-group pair. Taking the difference in this count between two traces and then comparing those differences between two hops gives a measure of packet loss just for traffic from the specified source to the specified receiver via the specified group. This measure is not affected by shared links.

On a point-to-point link that is a multicast tunnel, packet loss is usually due to congestion in unicast routers along the path of that tunnel. On native multicast links, loss is more likely in the output queue of one hop, perhaps due to priority dropping, or in the input queue at the next hop. The counters in the response data do not allow these cases to be distinguished. Differences in packet counts between the incoming and outgoing interfaces on one node cannot generally be used to measure queue overflow in the node.

12.4. Link Utilization

Again, with two traces, you can divide the difference in the input or output packet counts at some hop by the difference in time stamps from the same hop to obtain the packet rate over the link. If the average packet size is known, then the link utilization can also be estimated to see whether packet loss may be due to the rate limit or the physical capacity on a particular link being exceeded.

12.5. Time Delay

If the routers have synchronized clocks, it is possible to estimate propagation and queuing delay from the differences between the timestamps at successive hops. However, this delay includes control processing overhead, so is not necessarily indicative of the delay that data traffic would experience.

13. IANA Considerations

The following new assignments can only be made via a Standards Action as specified in [5].

13.1. Forwarding Codes

New Forwarding codes must only be created by an RFC that modifies this document's Section 10, fully describing the conditions under which the new forwarding code is used. The IANA may act as a central repository so that there is a single place to look up forwarding codes and the document in which they are defined.

13.2. UDP Destination Port and IPv6 Address

The IANA should allocate UDP destination port for multicast traceroute version 2 upon publication of the first RFC.

14. Security Considerations

14.1. Topology Discovery

Mtrace2 can be used to discover any actively-used topology. If your network topology is a secret, mtrace2 may be restricted at the border of your domain, using the ADMIN_PROHIB forwarding code.

14.2. Traffic Rates

Mtrace2 can be used to discover what sources are sending to what groups and at what rates. If this information is a secret, mtrace2 may be restricted at the border of your domain, using the ADMIN_PROHIB forwarding code.

15. Acknowledgements

This specification started largely as a transcription of Van Jacobson's slides from the 30th IETF, and the implementation in mrouted 3.3 by Ajit Thyagarajan. Van's original slides credit Steve Casner, Steve Deering, Dino Farinacci and Deb Agrawal. The original multicast traceroute client, mtrace (version 1), has been implemented by Ajit Thyagarajan, Steve Casner and Bill Fenner.

The idea of unicasting a multicast traceroute Query to the destination of the trace with Router Alert set is due to Tony Ballardie. The idea of the "S" bit to allow statistics for a source subnet is due to Tom Pusateri.

For the mtrace version 2 specification, extensive comments were received from Yiqun Cai, Liu Hui, Bharat Joshi, Shinsuke Suzuki, Achmad Husni Thamrin, and Cao Wei.

16. References

16.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to indicate requirement levels", RFC 2119, March 1997.

[2] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[3] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 2373, July 1998.

[4] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, October 2002.

[5] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 2434, October 1998.

[6] Braden, B., Borman, D., and C. Partridge, "Computing the Internet Checksum", RFC 1071, September 1988.

[7] Katz, D., "IP Router Alert Option", RFC 2113, February 1997.

[8] Partridge, C. and A. Jackson, "IPv6 Router Alert Option", RFC 2711, October 1999.

[9] Fenner, B., Handley, M., Holbrook, H., and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", RFC 4601, August 2006.

[10] Handley, M., Kouvelas, I., Speakman, T., and L. Vicisano, "Bidirectional Protocol Independent Multicast (BIDIR-PIM)", RFC 5015, October 2007.

[11] Fenner, B., He, H., Haberman, B., and H. Sandick, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying")", RFC 4605, August 2006.

[12] Thaler, D., Talwar, M., Aggarwal, A., Vicisano, L., and T. Pusateri, "Automatic IP Multicast Without Explicit Tunnels (AMT)", draft-ietf-mboned-auto-multicast-08.txt (work in progress), October 2007.

16.2. Informative References

[13] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.

[14] McCloghrie, K. and F. Kastenholz, "The Interfaces Group MIB", RFC 2863, June 2000.

[15] McWalter, D., Thaler, D., and A. Kessler, "IP Multicast MIB", RFC 5132, December 2007.

Authors' Addresses

Hitoshi Asaeda
Keio University
Graduate School of Media and Governance
Fujisawa, Kanagawa 252-8520
Japan

Email: asaeda@wide.ad.jp
URI: http://www.sfc.wide.ad.jp/~asaeda/

Tatuya Jinmei
Internet Systems Consortium
Redwood City, CA 94063
US

Email: Jinmei_Tatuya@isc.org

William C. Fenner
Arastra, Inc.
Menlo Park, CA 94025
US

Email: fenner@fenron.com

Stephen L. Casner
Packet Design, Inc.
Palo Alto, CA 94304
US

Email: casner@packetdesign.com