Mtrace Version 2: Traceroute Facility for IP Multicast
draft-ietf-mboned-mtrace-v2-17

Abstract

   This document describes the IP multicast traceroute facility, named
   Mtrace version 2 (Mtrace2). Unlike unicast traceroute, Mtrace2
   requires special implementations on the part of routers. This
   specification describes the required functionality in multicast
   routers, as well as how an Mtrace2 client invokes a query and
   receives a reply.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

   Given a multicast distribution tree, tracing from a multicast source
   to a receiver is difficult, since we do not know which branch of the
   multicast tree the receiver lies. This means that we have to flood
   the whole tree to find the path from a source to a receiver. On the
   other hand, walking up the tree from a receiver to a source is easy,
   as most existing multicast routing protocols know the upstream router
   for each source. Tracing from a receiver to a source can involve
   only the routers on the direct path.

   This document specifies the multicast traceroute facility named
   Mtrace version 2 or Mtrace2 which allows the tracing of an IP
   multicast routing path. Mtrace2 is usually initiated from an Mtrace2
   client by sending an Mtrace2 Query to a Last Hop Router (LHR) or to a
   Rendezvous Point (RP). The RP is a special router where sources and

receivers meet in PIM-SM [5]. From the LHR/RP receiving the query,
the tracing is directed towards a specified source if a source
address is specified and source specific state exists on the
receiving router. If no source address is specified or if no source
specific state exists on a receiving LHR, the tracing is directed
toward the RP for the specified group address. Moreover, Mtrace2
provides additional information such as the packet rates and losses,
as well as other diagnostic information. Mtrace2 is primarily
intended for the following purposes:

o To trace the path that a packet would take from a source to a
   receiver.

o To isolate packet loss problems (e.g., congestion).

o To isolate configuration problems (e.g., TTL threshold).

Figure 1 shows a typical case on how Mtrace2 is used. FHR represents
the first-hop router, LHR represents the last-hop router, and the
arrow lines represent the Mtrace2 messages that are sent from one
node to another. The numbers before the Mtrace2 messages represent
the sequence of the messages that would happen. Source, Receiver and
Mtrace2 client are typically hosts.

```
                2. Request                 2. Request
                  +----+                     +----+
                  |    |                     |    |
                  v    |                     v    |
 +--------+    +-----+                    +-----+    +----------+
 | Source |----| FHR |----- The Internet -----| LHR |----| Receiver |
 +--------+    +-----+          |             +-----+    +----------+
                  \             |               ^
                   \            |              /
                    \           |             /
                     \          |            /
             3. Reply \         |           / 1. Query
                       \        |          /
                        \       |         /
                         \  +---------+ /
                        v | Mtrace2 |/
                          | client  |
                          +---------+
```
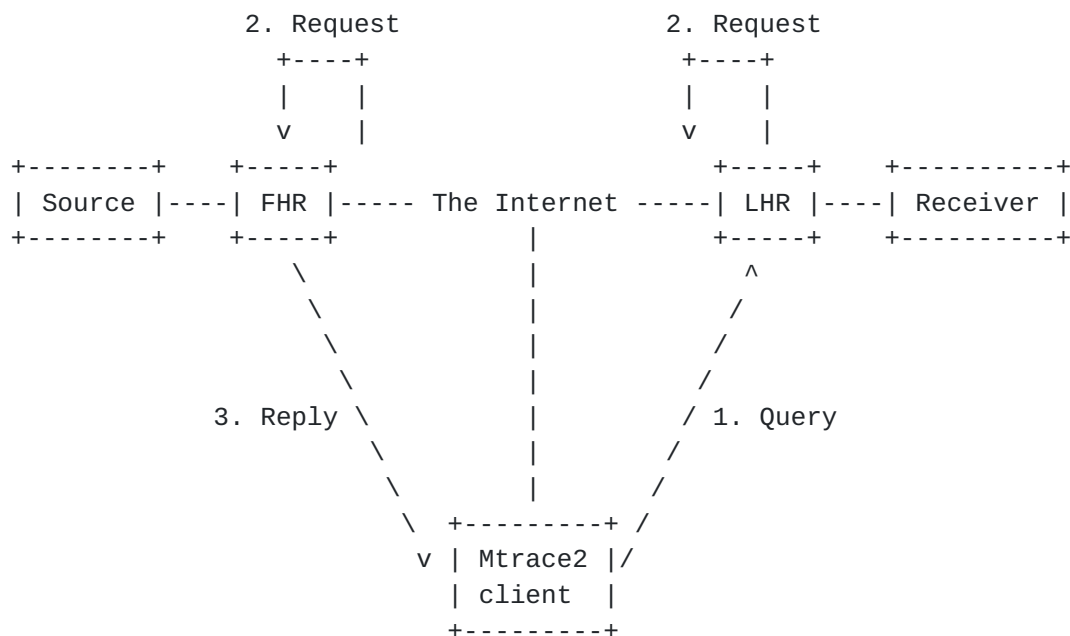
                             Figure 1

When an Mtrace2 client initiates a multicast trace, it sends an
Mtrace2 Query packet to the LHR or RP for a multicast group and,
optionally, a source address. The LHR/RP turns the Query packet into

a Request. The Request message type enables each of the upstream
routers processing the message to apply different packet and message
validation rules than those required for handling of a Query message.
The LHR/RP then appends a standard response block containing its
interface addresses and packet statistics to the Request packet, then
forwards the packet towards the source/RP. The Request packet is
either unicasted to its upstream router towards the source/RP, or
multicasted to the group if the upstream router's IP address is not
known. In a similar fashion, each router along the path to the
source/RP appends a standard response block to the end of the Request
packet before forwarding it to its upstream router. When the FHR
receives the Request packet, it appends its own standard response
block, turns the Request packet into a Reply, and unicasts the Reply
back to the Mtrace2 client.

The Mtrace2 Reply may be returned before reaching the FHR under some
circumstances. This can happen if a Request packet is received at an
RP or gateway, or when any of several types of error or exception
conditions occur which prevent sending of a request to the next
upstream router.

The Mtrace2 client waits for the Mtrace2 Reply message and displays
the results. When not receiving an Mtrace2 Reply message due to
network congestion, a broken router (see Section 5.6), or a non-
responding router (see Section 5.7), the Mtrace2 client may resend
another Mtrace2 Query with a lower hop count (see Section 3.2.1), and
repeat the process until it receives an Mtrace2 Reply message. The
details are Mtrace2 client specific, and it is outside the scope of
this document.

Note that when a router's control plane and forwarding plane are out
of sync, the Mtrace2 Requests might be forwarded based on the control
states instead. In which case, the traced path might not represent
the real path the data packets would follow.

Mtrace2 supports both IPv4 and IPv6. Unlike the previous version of
Mtrace, which implements its query and response as IGMP messages [8],
all Mtrace2 messages are UDP-based. Although the packet formats of
IPv4 and IPv6 Mtrace2 are different because of the address families,
the syntax between them is similar.

2. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED",
"SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY",
and "OPTIONAL" are to be interpreted as described in RFC 2119 [1],
and indicate requirement levels for compliant Mtrace2
implementations.

2.1. **Definitions**

   Since Mtrace2 Queries and Requests 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 data is expected to arrive from the
      specified source and group.

   Outgoing interface
      This is one of the interfaces to which data from the source or RP
      is expected to be transmitted for the specified source and group.
      It is also the interface on which the Mtrace2 Request was
      received.

   Upstream router
      The router, connecting to the Incoming interface of the current
      router, which is responsible for forwarding data for the specified
      source and group to the current router.

   First-hop router (FHR)
      The router that is directly connected to the source the Mtrace2
      Query specifies.

   Last-hop router (LHR)
      A router that is directly connected to a receiver. It is also the
      router that receives the Mtrace2 Query from an Mtrace2 client.

   Group state
      It is the state a shared-tree protocol, such as PIM-SM [5], uses
      to choose the upstream router towards the RP for the specified
      group. In this state, source-specific state is not available for
      the corresponding group address on the router.

   Source-specific state
      It is the state that is used to choose the path towards the source
      for the specified source and group.

   ALL-[protocol]-ROUTERS.MCAST.NET
      It is a link-local multicast address for multicast routers to
      communicate with their adjacent routers that are running the same
      routing protocol. For instance, the address of ALL-PIM-
      ROUTERS.MCAST.NET [5] is '224.0.0.13' for IPv4 and 'ff02::d' for
      IPv6.

3. Packet Formats

   This section describes the details of the packet formats for Mtrace2
   messages.

   All Mtrace2 messages are encoded in TLV format (see Section 3.1).
   The first TLV of a message is a message header TLV specifying the
   type of message and additional context information required for
   processing of the message and for parsing of subsequent TLVs in the
   message. Subsequent TLVs in a message, referred to as Blocks, are
   appended after the header TLV to provide additional information
   associated with the message. If an implementation receives an
   unknown TLV type for the first TLV in a message, it SHOULD ignore and
   silently discard the TLV and any subsequent TLVs in the packet
   containing the TLV. If an implementation receives an unknown TLV
   type for a subsequent TLV within a message, it SHOULD ignore and
   silently discard the TLV. If the length of a TLV exceeds the
   available space in the containing packet, the implementation MUST
   ignore and silently discard the TLV and any remaining portion of the
   containing packet. Any data in the packet after the specified TLV
   length is considered to be outside the boundary of the TLV and MUST
   be ignored during processing of the TLV.

   All Mtrace2 messages are UDP packets. For IPv4, Mtrace2 Query and
   Request messages MUST NOT be fragmented. For IPv6, the packet size
   for the Mtrace2 messages MUST NOT exceed 1280 bytes, which is the
   smallest MTU for an IPv6 interface [2]. The source port is uniquely
   selected by the local host operating system. The destination port is
   the IANA reserved Mtrace2 port number (see Section 8). All Mtrace2
   messages MUST have a valid UDP checksum.

   Additionally, Mtrace2 supports both IPv4 and IPv6, but not mixed.
   For example, if an Mtrace2 Query or Request message arrives in as an
   IPv4 packet, all addresses specified in the Mtrace2 messages MUST be
   IPv4 as well. Same rule applies to IPv6 Mtrace2 messages.

3.1. 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

      Describes the format of the Value field. For all the available
      types, please see Section 3.2

Length: 16 bits

> Length of Type, Length, and Value fields in octets. Minimum
> length required is 3 octets. The maximum TLV length is not
> defined; however the entire Mtrace2 packet length SHOULD NOT
> exceed the available MTU.

Value: variable length

> The format is based on the Type value. The length of the value
> field is Length field minus 3. All reserved fields in the Value
> field MUST be transmitted as zeros and ignored on receipt.

3.2. Defined TLVs

The following TLV Types are defined:

```
Code          Type
====          ===============================
0x01          Mtrace2 Query
0x02          Mtrace2 Request
0x03          Mtrace2 Reply
0x04          Mtrace2 Standard Response Block
0x05          Mtrace2 Augmented Response Block
0x06          Mtrace2 Extended Query Block
```

Each Mtrace2 message MUST begin with either a Query, Request or Reply
TLV. The first TLV determines the type of each Mtrace2 message.
Following a Query TLV, there can be a sequence of optional Extended
Query Blocks. In the case of a Request or a Reply TLV, it is then
followed by a sequence of Standard Response Blocks, each from a
multicast router on the path towards the source or the RP. In the
case more information is needed, a Standard Response Block can be
followed by one or multiple Augmented Response Blocks.

We will describe each message type in detail in the next few
sections.

3.2.1. Mtrace2 Query

An Mtrace2 Query is usually originated by an Mtrace2 client which
sends an Mtrace2 Query message to the LHR. When tracing towards the
source or the RP, the intermediate routers MUST NOT modify the Query
message except the Type field.

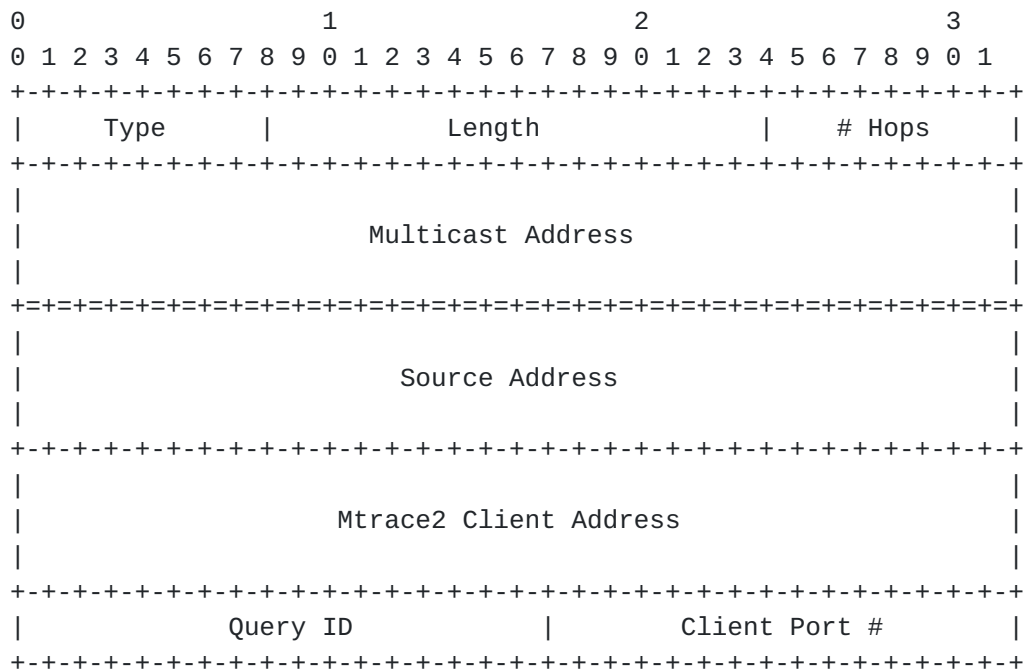An Mtrace2 Query message is shown as follows:

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Type      |            Length             |    # Hops     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   |                      Multicast Address                        |
   |                                                               |
   +=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
   |                                                               |
   |                       Source Address                          |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   |                   Mtrace2 Client Address                      |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           Query ID           |        Client Port #          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

                                Figure 2

   # Hops: 8 bits
      This field specifies the maximum number of hops that the Mtrace2
      client wants to trace. If there are some error conditions in the
      middle of the path that prevent an Mtrace2 Reply from being
      received by the client, the client MAY issue another Mtrace2 Query
      with a lower number of hops until it receives a Reply.

   Multicast Address: 32 bits or 128 bits
      This field specifies an IPv4 or IPv6 address, which can be either:

      m-1: a multicast group address to be traced; or,

      m-2: all 1's in case of IPv4 or the unspecified address (::) in
            case of IPv6 if no group-specific information is desired.

   Source Address: 32 bits or 128 bits
      This field specifies an IPv4 or IPv6 address, which can be either:

      s-1: an unicast address of the source to be traced; or,

      s-2: all 1's in case of IPv4 or the unspecified address (::) in
            case of IPv6 if no source-specific information is desired.
            For example, the client is tracing a (*,g) group state.

Note that it is invalid to have a source-group combination of
(s-2, m-2). If a router receives such combination in an Mtrace2
Query, it MUST silently discard the Query.

Mtrace2 Client Address: 32 bits or 128 bits
    This field specifies the Mtrace2 client's IPv4 address or IPv6
    global address. This address MUST be a valid unicast address, and
    therefore, MUST NOT be all 1's or an unspecified address. The
    Mtrace2 Reply will be sent to this address.

Query ID: 16 bits
    This field is used as a unique identifier for this Mtrace2 Query
    so that duplicate or delayed Reply messages may be detected.

Client Port #: 16 bits
    This field specifies the destination UDP port number for receiving
    the Mtrace2 Reply packet.

3.2.2. Mtrace2 Request

The format of an Mtrace2 Request message is similar to an Mtrace2
Query except the Type field is 0x02.

When a LHR receives an Mtrace2 Query message, it would turn the Query
into a Request by changing the Type field of the Query from 0x01 to
0x02. The LHR would then append an Mtrace2 Standard Response Block
(see Section 3.2.4) of its own to the Request message before sending
it upstream. The upstream routers would do the same without changing
the Type field until one of them is ready to send a Reply.

3.2.3. Mtrace2 Reply

The format of an Mtrace2 Reply message is similar to an Mtrace2 Query
except the Type field is 0x03.

When a FHR or a RP receives an Mtrace2 Request message which is
destined to itself, it would append an Mtrace2 Standard Response
Block (see Section 3.2.4) of its own to the Request message. Next,
it would turn the Request message into a Reply by changing the Type
field of the Request from 0x02 to 0x03. The Reply message would then
be unicasted to the Mtrace2 client specified in the Mtrace2 Client
Address field.

There are a number of cases an intermediate router might return a
Reply before a Request reaches the FHR or the RP. See Section 4.1.1,
Section 4.2.2, Section 4.3.3, and Section 4.5 for more details.

3.2.4. IPv4 Mtrace2 Standard Response Block

This section describes the message format of an IPv4 Mtrace2 Standard
Response Block. The Type field is 0x04.
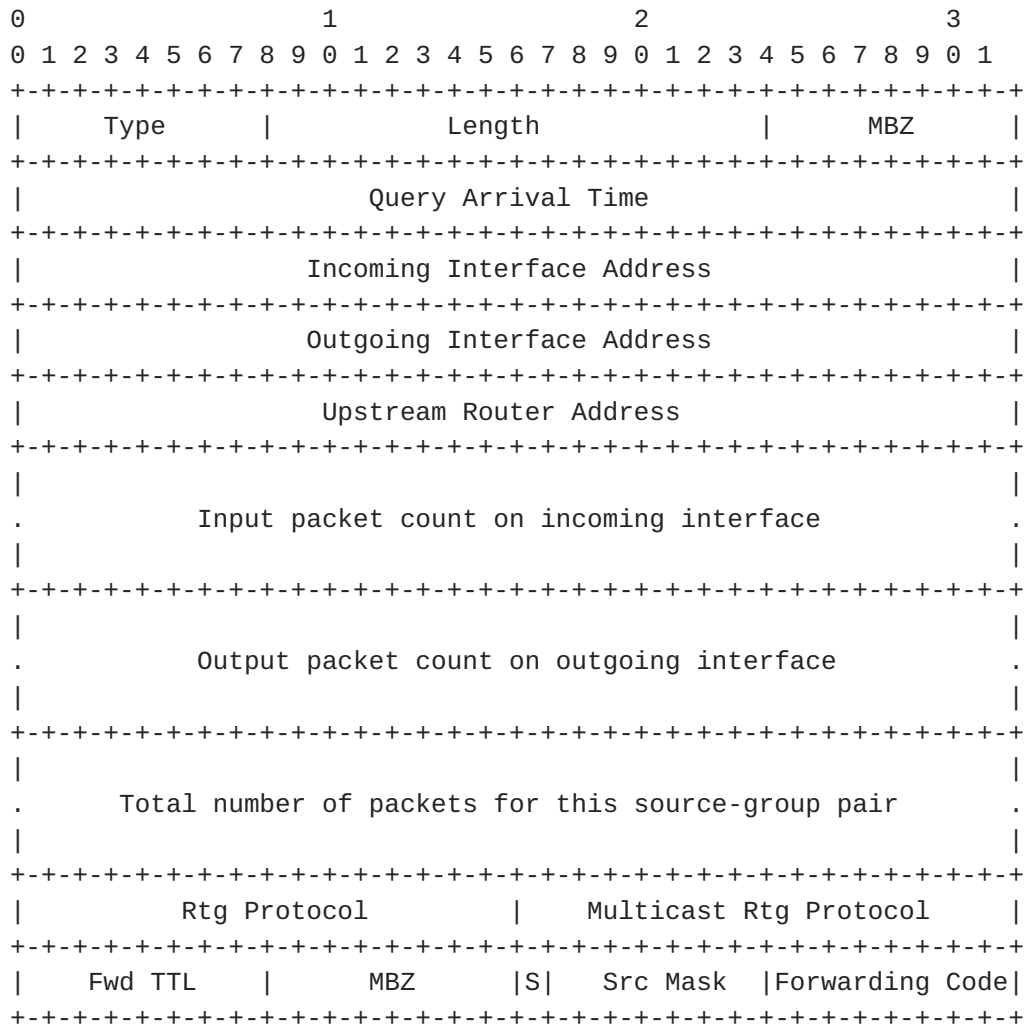
```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |            Length             |      MBZ      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Query Arrival Time                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Incoming Interface Address                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Outgoing Interface Address                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Upstream Router Address                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.           Input packet count on incoming interface           .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.           Output packet count on outgoing interface          .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.        Total number of packets for this source-group pair    .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Rtg Protocol          |     Multicast Rtg Protocol    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Fwd TTL     |     MBZ      |S|   Src Mask   |Forwarding Code|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

MBZ: 8 bits
   This field MUST be zeroed on transmission and ignored on
   reception.

Query Arrival Time: 32 bits
   The Query Arrival Time is a 32-bit NTP timestamp specifying the
   arrival time of the Mtrace2 Query or 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_nsec << 7) / 1953125)
```

The constant 32384 is the number of seconds from Jan 1, 1900 to
Jan 1, 1970 truncated to 16 bits. ((tv.tv_nsec << 7) / 1953125)
is a reduction of ((tv.tv_nsec / 1000000000) << 16).

Note that Mtrace2 does not require all the routers on the path to
have synchronized clocks in order to measure one-way latency.

Additionally, Query Arrival Time is useful for measuring the
packet rate. For example, suppose that a client issues two
queries, and the corresponding requests R1 and R2 arrive at router
X at time T1 and T2, then the client would be able to compute the
packet rate on router X by using the packet count information
stored in the R1 and R2, and the time T1 and T2.

Incoming Interface Address: 32 bits
   This field specifies the address of the interface on which packets
   from the source or the RP are expected to arrive, or 0 if unknown
   or unnumbered.

Outgoing Interface Address: 32 bits
   This field specifies the address of the interface on which packets
   from the source or the RP are expected to transmit towards the
   receiver, or 0 if unknown or unnumbered. This is also the address
   of the interface on which the Mtrace2 Query or Request arrives.

Upstream Router Address: 32 bits
   This field specifies the address of the upstream 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
   upstream router is not known because of the workings of the
   multicast routing protocol. However, it should be 0 if the
   incoming interface address is unknown or unnumbered.

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's if no
   count can be reported. This counter may have the same value as
   ifHCInMulticastPkts from the IF-MIB [10] for this interface.

Output packet count on outgoing interface: 64 bit
   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's if no count can be reported.
   This counter may have the same value as ifHCOutMulticastPkts from
   the IF-MIB [10] for this interface.

Total number of packets for this source-group pair: 64 bits
    This field counts the number of packets from the specified source
    forwarded by the router to the specified group, or all 1's if no
    count can be reported. If the S bit is set (see below), the count
    is for the source network, as specified by the Src Mask field (see
    below). If the S bit is set and the Src Mask field is 127,
    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 [11] for this
    forwarding entry.

Rtg Protocol: 16 bits
    This field describes the unicast routing protocol running between
    this router and the upstream router, and it is used to determine
    the RPF interface for the specified source or RP. This value
    should have the same value as ipMcastRouteRtProtocol from the
    IPMROUTE-STD-MIB [11] for this entry. If the router is not able
    to obtain this value, all 0's must be specified.

Multicast Rtg Protocol: 16 bits
    This field describes the multicast routing protocol in use between
    the router and the upstream router. This value should have the
    same value as ipMcastRouteProtocol from the IPMROUTE-STD-MIB [11]
    for this entry. If the router cannot obtain this value, all 0's
    must be specified.

Fwd TTL: 8 bits
    This field contains the configured multicast TTL threshold, if
    any, of the outgoing interface.

S: 1 bit
    If this bit is set, it 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.

Src Mask: 7 bits
    This field contains the number of 1's in the netmask the 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 127 (0x7f).

Forwarding Code: 8 bits
    This field contains a forwarding information/error code. Values
    with the high order bit set (0x80-0xff) are intended for use as
    error or exception codes. Section 4.1 and Section 4.2 explain 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
                           the specified group towards the source or RP.
   0x02    PRUNE_SENT      This router has sent a prune upstream which
                           applies to the source and group in the
                           Mtrace2 Request.
   0x03    PRUNE_RCVD      This router has stopped forwarding for this
                           source and group in response to a request
                           from the downstream router.
   0x04    SCOPED          The group is subject to administrative
                           scoping at this router.
   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 LHR.
   0x07    NOT_FORWARDING  This router is not forwarding this source and
                           group out the outgoing interface for an
                           unspecified reason.
   0x08    REACHED_RP      Reached the Rendezvous Point.
   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.
   0x0C    REACHED_GW      Mtrace2 Request arrived on a gateway (e.g.,
                           a NAT or firewall) that hides the
                           information between this router and the
                           Mtrace2 client.
   0x0D    UNKNOWN_QUERY   A non-transitive Extended Query Type was
                           received by a router which does not support
                           the type.
   0x80    FATAL_ERROR     A fatal error is one where the router may
                           know the upstream router but cannot forward
                           the message to it.
   0x81    NO_SPACE        There was not enough room to insert another
                           Standard Response Block in the packet.
   0x83    ADMIN_PROHIB    Mtrace2 is administratively prohibited.
```

3.2.5. IPv6 Mtrace2 Standard Response Block

   This section describes the message format of an IPv6 Mtrace2 Standard
   Response Block. The Type field is also 0x04.

```
    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            |      MBZ      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      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        |     Multicast Rtg Protocol       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            MBZ 2           |S|Src Prefix Len |Forwarding Code|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

MBZ: 8 bits
    This field MUST be zeroed on transmission and ignored on
    reception.

Query Arrival Time: 32 bits
    Same definition as in IPv4.

Incoming Interface ID: 32 bits
    This field specifies the interface ID on which packets from the
    source or RP are expected to arrive, or 0 if unknown. This ID
    should be the value taken from InterfaceIndex of the IF-MIB [10]
    for this interface.

Outgoing Interface ID: 32 bits
   This field specifies the interface ID to which packets from the
   source or RP are expected to transmit, or 0 if unknown. This ID
   should be the value taken from InterfaceIndex of the IF-MIB [10]
   for this interface

Local Address: 128 bits
   This field specifies a global IPv6 address that uniquely
   identifies the router. An unique local unicast address [9] 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.

Remote Address: 128 bits
   This field specifies the address of the upstream router, which, in
   most cases, is a link-local unicast address for the upstream
   router.

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

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

Input packet count on incoming interface: 64 bits
   Same definition as in IPv4.

Output packet count on outgoing interface: 64 bits
   Same definition as in IPv4.

Total number of packets for this source-group pair: 64 bits
   Same definition as in IPv4, except if the S bit is set (see
   below), 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 [11]
   for this forwarding entry.

Rtg Protocol: 16 bits
   Same definition as in IPv4.

Multicast Rtg Protocol: 16 bits

      Same definition as in IPv4.

   MBZ 2: 15 bits
      This field MUST be zeroed on transmission and ignored on
      reception.

   S: 1 bit
      Same definition as in IPv4, except the Src Prefix Len field is
      used to mask the source address.

   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).

   Forwarding Code: 8 bits
      Same definition as in IPv4.

3.2.6. Mtrace2 Augmented Response Block

   In addition to the Standard Response Block, a multicast router on the
   traced path can optionally add one or multiple Augmented Response
   Blocks before sending the Request to its upstream router.

   The Augmented Response Block is flexible for various purposes such as
   providing diagnosis information (see Section 7) and protocol
   verification. Its Type field is 0x05, and its format is 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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    Type       |            Length             |      MBZ      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    Augmented Response Type    |        Value ....             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   MBZ: 8 bits
      This field MUST be zeroed on transmission and ignored on
      reception.

   Augmented Response Type: 16 bits
      This field specifies the type of various responses from a
      multicast router that might need to communicate back to the
      Mtrace2 client as well as the multicast routers on the traced
      path.

      The Augmented Response Type is defined as follows:

```
          Code    Type
          ====    ============================================
          0x01    # of the returned Standard Response Blocks
```

   When the NO_SPACE error occurs on a router, the router should send
   the original Mtrace2 Request received from the downstream router
   as a Reply back to the Mtrace2 client, and continue with a new
   Mtrace2 Request. In the new Request, the router would add a
   Standard Response Block followed by an Augmented Response Block
   with 0x01 as the Augmented Response Type, and the number of the
   returned Mtrace2 Standard Response Blocks as the Value.

   Each upstream router would recognize the total number of hops the
   Request has been traced so far by adding this number and the
   number of the Standard Response Block in the current Request
   message.

   This document only defines one Augmented Response Type in the
   Augmented Response Block. The description on how to provide
   diagnosis information using the Augmented Response Block is out of
   the scope of this document, and will be addressed in separate
   documents.

Value: variable length
   The format is based on the Augmented Response Type value. The
   length of the value field is Length field minus 6.

3.2.7. Mtrace2 Extended Query Block

   There may be a sequence of optional Extended Query Blocks that follow
   an Mtrace2 Query to further specify any information needed for the
   Query. For example, an Mtrace2 client might be interested in tracing
   the path the specified source and group would take based on a certain
   topology. In which case, the client can pass in the multi-topology
   ID as the Value for an Extended Query Type (see below). The Extended
   Query Type is extensible and the behavior of the new types will be
   addressed by separate documents.

   The Mtrace2 Extended Query Block's Type field is 0x06, and is
   formatted 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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      Type     |             Length            |    MBZ    |T|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       Extended Query Type     |          Value ....         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

MBZ: 7 bits
   This field MUST be zeroed on transmission and ignored on
   reception.

T-bit (Transitive Attribute): 1 bit
   If the TLV type is unrecognized by the receiving router, then this
   TLV is either discarded or forwarded along with the Query,
   depending on the value of this bit. If this bit is set, then the
   router MUST forward this TLV. If this bit is clear, the router
   MUST send an Mtrace2 Reply with an UNKNOWN_QUERY error.

Extended Query Type: 16 bits
   This field specifies the type of the Extended Query Block.

Value: 16 bits
   This field specifies the value of this Extended Query.

4. Router Behavior

This section describes the router behavior in the context of Mtrace2
in detail.

4.1. Receiving Mtrace2 Query

An Mtrace2 Query message is an Mtrace2 message with no response
blocks filled in, and uses TLV type of 0x01.

4.1.1. Query Packet Verification

Upon receiving an Mtrace2 Query message, a router MUST examine
whether the Multicast Address and the Source Address are a valid
combination as specified in Section 3.2.1, and whether the Mtrace2
Client Address is a valid IP unicast address. If either one is
invalid, the Query MUST be silently ignored.

Mtrace2 supports a non-local client to the LHR/RP. A router SHOULD,
however, support a mechanism to filter out queries from clients
beyond a specified administrative boundary. The potential approaches
are described in Section 9.2.

In the case where a local LHR client is required, the router must
then examine the Query to see if it is the proper LHR/RP for the
destination address in the packet. It is the proper local LHR if it
has a multicast-capable interface on the same subnet as the Mtrace2
Client Address and is the router that would forward traffic from the
given (S,G) or (*,G) onto that subnet. It is the proper RP if the
multicast group address specified in the query is 0 and if the IP
header destination address is a valid RP address on this router.

If the router determines that it is not the proper LHR/RP, or it
cannot make that determination, it does one of two things depending
on whether the Query was received via multicast or unicast. If the
Query was received via multicast, then it MUST be silently discarded.
If it was received via unicast, the router turns the Query into a
Reply message by changing the TLV type to 0x03 and appending a
Standard Response Block with a Forwarding Code of WRONG_LAST_HOP.
The rest of the fields in the Standard Response Block MUST be zeroed.
The router then sends the Reply message to the Mtrace2 Client Address
on the Client Port # as specified in the Mtrace2 Query.

Duplicate Query messages as identified by the tuple (Mtrace2 Client
Address, Query ID) SHOULD be ignored. This MAY be implemented using
a cache of previously processed queries keyed by the Mtrace2 Client
Address and Query ID pair. The duration of the cached entries is
implementation specific. Duplicate Request messages MUST NOT be
ignored in this manner.

4.1.2. Query Normal Processing

When a router receives an Mtrace2 Query and it determines that it is
the proper LHR/RP, it turns the Query to a Request by changing the
TLV type from 0x01 to 0x02, and performs the steps listed in
Section 4.2.

4.2. Receiving Mtrace2 Request

An Mtrace2 Request is an Mtrace2 message that uses TLV type of 0x02.
With the exception of the LHR, whose Request was just converted from
a Query, each Request received by a router should have at least one
Standard Response Block filled in.

4.2.1. Request Packet Verification

If the Mtrace2 Request does not come from an adjacent router, or if
the 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.
224.0.0.0/24 for IPv4, FFx2::/16 [3] for IPv6), it MUST be silently
ignored. GTSM [12] SHOULD be used by the router to determine whether
the router is adjacent or not.

If the sum of the number of the Standard Response Blocks in the
received Mtrace2 Request and the value of the Augmented Response Type
of 0x01, if any, is equal or more than the # Hops in the Mtrace2
Request, it MUST be silently ignored.

4.2.2. Request Normal Processing

   When a router receives an Mtrace2 Request message, 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". Note that in the steps described
   below the "Outgoing Interface" is the one on which the Mtrace2
   Request message arrives.

   1.   Prepare a Standard Response Block to be appended to the packet,
        setting all fields to an initial default value of zero.

   2.   If Mtrace2 is administratively prohibited, note the Forwarding
        Code of ADMIN_PROHIB and skip to step 4.

   3.   In the Standard Response Block, fill in the Query Arrival Time,
        Outgoing Interface Address (for IPv4) or Outgoing Interface ID
        (for IPv6), Output Packet Count, and Fwd TTL (for IPv4).

   4.   Attempt to determine the forwarding information for the
        specified source and group, using the same mechanisms as would
        be used when a packet is received from the source destined for
        the group. A state need not be instantiated, it can be a
        "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 no source-specific information is desired (i.e.,
        all 1's for IPv4 or unspecified address (::) for IPv6), group
        state should be used. If there is no group state or no group-
        specific information is desired, potential source state (i.e.,
        the path that would be followed for a source-specific Join)
        should be used.

   5.   If no forwarding information can be determined, the router notes
        a Forwarding Code of NO_ROUTE, sets the remaining fields that
        have not yet been filled in to zero, and then sends an Mtrace2
        Reply back to the Mtrace2 client.

   6.   If a Forwarding Code of ADMIN_PROHIB has been set, skip to step
        7. Otherwise, fill in the Incoming Interface Address (or
        Incoming Interface ID and Local Address for IPv6), Upstream
        Router Address (or Remote Address for IPv6), Input Packet Count,
        Total Number of Packets, Routing Protocol, S, and Src Mask (or
        Src Prefix Len for IPv6) using the forwarding information
        determined in step 4.

7.   If the Outgoing interface is not enabled for multicast, note
     Forwarding Code of NO_MULTICAST. If the Outgoing interface is
     the interface from which the router would expect data to arrive
     from the source, note forwarding code RPF_IF. If the Outgoing
     interface is not one to which the router would forward data from
     the source or RP to the group, a Forwarding code of WRONG_IF is
     noted. In the above three cases, the router will return an
     Mtrace2 Reply and terminate the trace.

8.   If the group is subject to administrative scoping on either the
     Outgoing or Incoming interfaces, a Forwarding Code of SCOPED is
     noted.

9.   If this router is the RP for the group for a non-source-specific
     query, note a Forwarding Code of REACHED_RP. The router will
     send an Mtrace2 Reply and terminate the trace.

10. If this router is directly connected to the specified source or
     source network on the Incoming interface, it sets the Upstream
     Router Address (for IPv4) or the Remote Address (for IPv6) of
     the response block to zero. The router will send an Mtrace2
     Reply and terminate the trace.

11. If this router has sent a prune upstream which applies to the
     source and group in the Mtrace2 Request, it notes a Forwarding
     Code of PRUNE_SENT. If the router has stopped forwarding
     downstream in response to a prune sent by the downstream router,
     it notes a Forwarding Code of PRUNE_RCVD. If the router should
     normally forward traffic downstream for this source and group
     but is not, it notes a Forwarding Code of NOT_FORWARDING.

12. If this router is a gateway (e.g., a NAT or firewall) that hides
     the information between this router and the Mtrace2 client, it
     notes a Forwarding Code of REACHED_GW. The router continues the
     processing as described in Section 4.5.

13. If the total number of the Standard Response Blocks, including
     the newly prepared one, and the value of the Augmented Response
     Type of 0x01, if any, is less than the # Hops in the Request,
     the packet is then forwarded to the upstream router as described
     in Section 4.3; otherwise, the packet is sent as an Mtrace2
     Reply to the Mtrace2 client as described in Section 4.4.

4.3. Forwarding Mtrace2 Request

   This section describes how an Mtrace2 Request should be forwarded.

4.3.1. Destination Address

   If the upstream router for the Mtrace2 Request is known for this
   request, the Mtrace2 Request is sent to that router. If the Incoming
   interface is known but the upstream router is not, the Mtrace2
   Request is sent to an appropriate multicast address on the Incoming
   interface. The multicast address SHOULD depend on the multicast
   routing protocol in use, such as ALL-[protocol]-ROUTERS.MCAST.NET.
   It MUST be a link-scoped group (i.e. 224.0.0.0/24 for IPv4, FF02::/16
   for IPv6), and MUST NOT be ALL-SYSTEMS.MCAST.NET (224.0.0.1) for IPv4
   and All Nodes Address (FF02::1) for IPv6. It MAY also 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 multicast address.

4.3.2. Source Address

   An Mtrace2 Request should be sent with the address of the Incoming
   interface. However, if the Incoming interface is unnumbered, the
   router can use one of its numbered interface addresses as the source
   address.

4.3.3. Appending Standard Response Block

   An Mtrace2 Request MUST be sent upstream towards the source or the RP
   after appending a Standard Response Block to the end of the received
   Mtrace2 Request. The Standard Response Block includes the multicast
   states and statistics information of the router described in
   Section 3.2.4.

   If appending the Standard Response Block would make the Mtrace2
   Request packet longer than the MTU of the Incoming Interface, or, in
   the case of IPv6, longer than 1280 bytes, the router MUST change the
   Forwarding Code in the last Standard Response Block of the received
   Mtrace2 Request into NO_SPACE. The router then turns the Request
   into a Reply, and sends the Reply as described in Section 4.4.

   The router will continue with a new Request by copying from the old
   Request excluding all the response blocks, followed by the previously
   prepared Standard Response Block, and an Augmented Response Block
   with Augmented Response Type of 0x01 and the number of the returned
   Standard Response Blocks as the value. The new Request is then
   forwarded upstream.

4.4. Sending Mtrace2 Reply

   An Mtrace2 Reply MUST be returned to the client by a router if the
   total number of the traced routers is equal to the # Hops in the
   Request. The total number of the traced routers is the sum of the
   Standard Response Blocks in the Request (including the one just
   added) and the number of the returned blocks, if any.

4.4.1. Destination Address

   An Mtrace2 Reply MUST be sent to the address specified in the Mtrace2
   Client Address field in the Mtrace2 Request.

4.4.2. Source Address

   An Mtrace2 Reply SHOULD be sent with the address of the router's
   Outgoing interface. However, if the Outgoing interface address is
   unnumbered, the router can use one of its numbered interface
   addresses as the source address.

4.4.3. Appending Standard Response Block

   An Mtrace2 Reply MUST be sent with the prepared Standard Response
   Block appended at the end of the received Mtrace2 Request except in
   the case of NO_SPACE forwarding code.

4.5. Proxying Mtrace2 Query

   When a gateway (e.g., a NAT or firewall), which needs to block
   unicast packets to the Mtrace2 client, or hide information between
   the gateway and the Mtrace2 client, receives an Mtrace2 Query from an
   adjacent host or Mtrace2 Request from an adjacent router, it appends
   a Standard Response Block with REACHED_GW as the Forwarding Code. It
   turns the Query or Request into a Reply, and sends the Reply back to
   the client.

   At the same time, the gateway originates a new Mtrace2 Query message
   by copying the original Mtrace2 header (the Query or Request without
   any of the response blocks), and makes the changes as follows:

   o sets the RPF interface's address as the Mtrace2 Client Address;

   o uses its own port number as the Client Port #; and,

   o decreases # Hops by ((number of the Standard Response Blocks that
      were just returned in a Reply) - 1). The "-1" in this expression
      accounts for the additional Standard Response Block appended by
      the gateway router.

The new Mtrace2 Query message is then sent to the upstream router or
to an appropriate multicast address on the RPF interface.

When the gateway receives an Mtrace2 Reply whose Query ID matches the
one in the original Mtrace2 header, it MUST relay the Mtrace2 Reply
back to the Mtrace2 client by replacing the Reply's header with the
original Mtrace2 header. If the gateway does not receive the
corresponding Mtrace2 Reply within the [Mtrace Reply Timeout] period
(see Section 5.8.4), then it silently discards the original Mtrace2
Query or Request message, and terminates the trace.

4.6. Hiding Information

Information about a domain's topology and connectivity may be hidden
from the Mtrace2 Requests. The Forwarding Code of INFO_HIDDEN may be
used to note that. For example, the incoming interface address and
packet count on the ingress router of a domain, and the outgoing
interface address and packet count on the egress router of the domain
can be specified as all 1's. Additionally, the source-group packet
count (see Section 3.2.4 and Section 3.2.5) within the domain may be
all 1's if it is hidden.

5. Client Behavior

This section describes the behavior of an Mtrace2 client in detail.

5.1. Sending Mtrace2 Query

An Mtrace2 client initiates an Mtrace2 Query by sending the Query to
the LHR of interest.

5.1.1. Destination Address

If an Mtrace2 client knows the proper LHR, it unicasts an Mtrace2
Query packet to that router; otherwise, it MAY send the Mtrace2 Query
packet 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 LHR on the subnet.

See also Section 5.4 on determining the LHR.

5.1.2. Source Address

An Mtrace2 Query MUST be sent with the client's interface address,
which would be the Mtrace2 Client Address.

5.2. Determining the Path

An Mtrace2 client could send an initial Query messages with a large #
Hops, 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 Reply, then 2, and so on. If no Reply 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 the [Mtrace Reply Timeout] timeout has occurred.

See also Section 5.6 on receiving the results of a trace.

5.3. Collecting Statistics

After a client has determined that it has traced the whole path or as
much as it can expect to (see Section 5.8), 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 7.3 and Section 7.4.

5.4. Last Hop Router (LHR)

The Mtrace2 client 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.

5.5. First Hop Router (FHR)

The IANA assigned 224.0.1.32, MTRACE.MCAST.NET as the default
multicast group for old IPv4 mtrace (v1) responses, in order to
support mtrace clients that are not unicast reachable from the first-
hop router. Mtrace2, however, does not require any IPv4/IPv6
multicast addresses for the Mtrace2 Replies. Every Mtrace2 Reply is
sent to the unicast address specified in the Mtrace2 Client Address
field of the Mtrace2 Reply.

5.6. Broken Intermediate Router

A broken intermediate router might simply not understand Mtrace2
packets, and drop them. The Mtrace2 client will get no Reply at all
as a result. It should then perform a hop-by-hop search by setting
the # Hops field until it gets an Mtrace2 Reply. The client may use
linear or binary search; however, the latter is likely to be slower

because a failure requires waiting for the [Mtrace Reply Timeout] period.

5.7. Non-Supported Router

When a non-supported router receives an Mtrace2 Query or Request message whose destination address is a multicast address, the router will silently discard the message.

When the router receives an Mtrace2 Query which is destined to itself, the router would return an ICMP port unreachable to the Mtrace2 client. On the other hand, when the router receives an Mtrace2 Request which is destined to itself, the router would return an ICMP port unreachable to its adjacent router from which the Request receives. Therefore, the Mtrace2 client needs to terminate the trace when the [Mtrace Reply Timeout] timeout has occurred, and may then issue another Query with a lower number of # Hops.

5.8. Mtrace2 Termination

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

5.8.1. 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 Upstream Router is zero.

5.8.2. Fatal Error

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

5.8.3. No Upstream Router

A trace can not continue if the last Upstream Router in the trace is set to 0.

5.8.4. Reply Timeout

This document defines the [Mtrace Reply Timeout] value, which is used to time out an Mtrace2 Reply as seen in Section 4.5, Section 5.2, and Section 5.7. The default [Mtrace Reply Timeout] value is 10 (seconds), and can be manually changed on the Mtrace2 client and routers.

5.9. Continuing after an Error

When the NO_SPACE error occurs, as described in Section 4.2, a router
will send back an Mtrace2 Reply to the Mtrace2 client, and continue
with a new Request (see Section 4.3.3). In which case, the Mtrace2
client may receive multiple Mtrace2 Replies from different routers
along the path. When this happens, the client MUST treat them as a
single Mtrace2 Reply message.

If a trace times out, it is very likely that a router in the middle
of the path does not support Mtrace2. That router's address will be
in the Upstream Router field of the last Standard Response Block in
the last received Reply. A client may be able to determine (via
mrinfo or SNMP [9][11]) 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 an upstream 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.

6. Protocol-Specific Considerations

This section describes the Mtrace2 behavior with the presence of
different multicast protocols.

6.1. PIM-SM

When an Mtrace2 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 Mtrace2 Query
may be unicasted to the RP, and the RP takes the same actions as an
LHR.

6.2. Bi-Directional PIM

Bi-directional PIM [6] 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 the sources to the Rendezvous Point Link (RPL), and
from which, to receivers without requiring source-specific state. In
contrast to PIM-SM, Bi-directional PIM always has the state to trace.

A Designated Forwarder (DF) for a given Rendezvous Point Address
(RPA) is in charge of forwarding downstream traffic onto its link,
and forwarding upstream traffic from its link towards the RPL that
the RPA belongs to. Hence Mtrace2 Reply reports DF addresses or RPA
along the path.

6.3. PIM-DM

Routers running PIM Dense Mode [13] 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, Mtrace2 can only trace
existing paths, not potential paths. When there are multiple
possible paths but the branch points are not on shared media, the
upstream router is known, but the LHR 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 LHR for the link (because they won or lost an Assert
battle) and know who the upstream router is (because it won an Assert
battle). Therefore, Mtrace2 is always able to follow the proper path
when traffic is flowing.

6.4. IGMP/MLD Proxy

When an IGMP/MLD Proxy [7] receives an Mtrace2 Query packet on an
incoming interface, it notes a WRONG_IF in the Forwarding Code of the
last Standard Response Block (see Section 3.2.4), and sends the
Mtrace2 Reply back to the Mtrace2 client. On the other hand, when an
Mtrace2 Query packet reaches an outgoing interface of the IGMP/MLD
proxy, it is forwarded onto its incoming interface towards the
upstream router.

7. Problem Diagnosis

This section describes different scenarios Mtrace2 can be used to
diagnose the multicast problems.

7.1. Forwarding Inconsistencies

The Forwarding Error code can tell if a group is unexpectedly pruned
or administratively scoped.

7.2. TTL or Hop Limit Problems

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

7.3. 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 Request 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 Standard Response Block
   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 Standard Response Block
   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.

7.4. 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.

7.5. 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.

8. IANA Considerations

The following new registries are to be created and maintained under
the "RFC Required" registry policy as specified in [4].

8.1. "Mtrace2 Forwarding Codes" Registry

This is an integer in the range 0-255. Assignment of a Forwarding
Code requires specification of a value and a name for the Forwarding
Code. Initial values for the forwarding codes are given in the table
at the end of Section 3.2.4. Additional values (specific to IPv6)
may also be specified at the end of Section 3.2.5. Any additions to
this registry are required to fully describe the conditions under
which the new Forwarding Code is used.

8.2. "Mtrace2 TLV Types" registry

Assignment of a TLV Type requires specification of an integer value
"Code" in the range 0-255 and a name ("Type"). Initial values for
the TLV Types are given in the table at the beginning of Section 3.2.

8.3. UDP Destination Port

The Mtrace2 UDP destination port is [TBD].

9. Security Considerations

This section addresses some of the security considerations related to
Mtrace2.

9.1. Addresses in Mtrace2 Header

An Mtrace2 header includes three addresses, source address, multicast
address, and Mtrace2 client address. These addresses MUST be
congruent with the definition defined in Section 3.2.1 and forwarding
Mtrace2 messages having invalid addresses MUST be prohibited. For
instance, if Mtrace2 Client Address specified in an Mtrace2 header is
a multicast address, then a router that receives the Mtrace2 message
MUST silently discard it.

9.2. Filtering of Clients

A router SHOULD support a mechanism to filter out queries from
clients beyond a specified administrative boundary. Such a boundary
could, for example, be specified via a list of allowed/disallowed

client addresses or subnets. If a query is received from beyond the
specified administrative boundary, the Query MUST NOT be processed.
The router MAY, however, perform rate limited logging of such events.

[9.3](). 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.

[9.4](). Characteristics of Multicast Channel

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.

[9.5](). Limiting Query/Request Rates

A router may limit Mtrace2 Queries and Requests by ignoring some of
the consecutive messages. The router MAY randomly ignore the
received messages to minimize the processing overhead, i.e., to keep
fairness in processing queries, or prevent traffic amplification.
The rate limit is left to the router's implementation.

[9.6](). Limiting Reply Rates

The proxying and NO_SPACE behaviors may result in one Query returning
multiple Reply messages. In order to prevent abuse, the routers in
the traced path MAY need to rate-limit the Replies. The rate limit
function is left to the router's implementation.

[10](). 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 the
"S" bit to allow statistics for a source subnet is due to Tom
Pusateri.

For the Mtrace version 2 specification, the authors would like to
give special thanks to Tatsuya Jinmei, Bill Fenner, and Steve Casner.
Also, extensive comments were received from David L. Black, Ronald
Bonica, Yiqun Cai, Liu Hui, Bharat Joshi, Robert Kebler, John

Kristoff, Heidi Ou, Pekka Savola, Shinsuke Suzuki, Dave Thaler,
Achmad Husni Thamrin, Stig Venaas, and Cao Wei.

11. References

11.1. 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 4291, February 2006.

[4]   Narten, T. and H. Alvestrand, "Guidelines for Writing an
           IANA Considerations Section in RFCs", RFC 5226, May 2008.

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

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

[7]   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.

11.2. Informative References

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

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

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

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

   [12]   Gill, V., Heasley, J., Meyer, D., Savola, P., and C.
               Pignataro, "The Generalized TTL Security Mechanism
               (GTSM)", RFC 5082, October 2007.

   [13]   Adams, A., Nicholas, J., and W. Siadak, "Protocol
               Independent Multicast - Dense Mode (PIM-DM): Protocol
               Specification (Revised)", RFC 3973, January 2005.

Authors' Addresses

   Hitoshi Asaeda
   National Institute of Information and Communications Technology
   4-2-1 Nukui-Kitamachi
   Koganei, Tokyo 184-8795
   Japan

   Email: asaeda@nict.go.jp

   Kerry Meyer
   Cisco Systems, Inc.
   510 McCarthy Blvd.
   Milpitas, CA 95035
   USA

   Email: kerrymey@cisco.com

   WeeSan Lee (editor)

   Email: weesan@weesan.com