Internet Engineering Task Force   Inter-Domain Multicast Routing Working Group
INTERNET-DRAFT   W. Fenner
draft-ietf-idmr-traceroute-ipm-07.txt   AT&T Research
                                                                  S. Casner
                                                              Cisco Systems
                                                              July 14, 2000
                                                       Expires January 2001

A "traceroute" facility for IP Multicast.

Status of this Memo

Abstract

This draft describes the IGMP multicast traceroute facility. Unlike unicast traceroute, multicast traceroute requires a special packet type and implementation on the part of routers. This specification describes the required functionality in multicast routers, as well as how management applications can use the new router functionality.

This document is a product of the Inter-Domain Multicast Routing working group within the Internet Engineering Task Force. Comments are solicited and should be addressed to the working group's mailing list at idmr@cs.ucl.ac.uk and/or the author(s).

Key Words

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 [Brad97].

1. Introduction

The unicast "traceroute" program allows the tracing of a path from one machine to another, using a mechanism that already existed in IP. Unfortunately, no such existing mechanism can be applied to IP multicast paths. The key mechanism for unicast traceroute is the ICMP TTL exceeded message, which is specifically precluded as a response to multicast packets. Thus, we specify the multicast "traceroute" facility to be implemented in multicast routers and accessed by diagnostic programs. While it is a disadvantage that a new mechanism is required, the multicast traceroute facility can provide additional information about packet rates and losses that the unicast traceroute cannot, and generally requires fewer packets to be sent.

Goals:

- 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).

2. 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 (which need be neither the source nor the destination) 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 DVMRP router has no active state for a particular source but does have a DVMRP route, it chooses the parent of the DVMRP route as the previous hop. 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.

3. Multicast Traceroute header

The header for all multicast traceroute packets is as follows. The header is only filled in by the originator of the traceroute Query; intermediate hops 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    IGMP Type  |     # hops    |           checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Multicast Group Address                     |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                      Source Address                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Response Address                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    resp ttl   |               Query ID                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.1. IGMP Type: 8 bits

The IGMP type field is defined to be 0x1F for traceroute queries and requests. The IGMP type field is changed to 0x1E 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 won't attempt to process a completed response in those cases where the initial query was issued from a router or the response is sent via multicast.

3.2. # 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 traceroute request from reaching the first-hop router, this field can be used to perform an expanding-length search to trace the path to just before the problem.

3.3. Checksum: 16 bits

The checksum is the 16-bit one's complement of the one's complement sum of the whole IGMP message (the entire IP payload)[Brad88]. When computing the checksum, the checksum field is set to zero. When transmitting packets, the checksum MUST be computed and inserted into this field. When receiving packets, the checksum MUST be verified before processing a packet.

3.4. Group address

This field specifies the group address to be traced, or zero if no group-specific information is desired. Note that non-group-specific traceroutes may not be possible with certain multicast routing protocols.

3.5. Source address

This field specifies the IP address of the multicast source for the path being traced, or 0xFFFFFFFF if no source-specific information is desired. Note that non-source-specific traceroutes may not be possible with certain multicast routing protocols.

3.6. Destination address

This field specifies the IP address of the multicast receiver for the path being traced. The trace starts at this destination and proceeds toward the traffic source.

3.7. Response Address

This field specifies where the completed traceroute response packet gets sent. It can be a unicast address or a multicast address, as explained in section 6.2.

3.8. resp ttl: 8 bits

This field specifies the TTL at which to multicast the response, if the response address is a multicast address.

3.9. Query ID: 24 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.

4. Definitions

Since multicast traceroutes flow in the opposite direction to the data flow, we always 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 towards the destination. Also called the "Reception Interface", since it is the interface on which the multicast traceroute Request was received.

Previous-Hop Router
The router, on the Incoming Interface, which is responsible for forwarding traffic for the specified source and group.

5. Response data

Each router adds a "response data" segment to the traceroute packet before it forwards it on. The response data looks like this:

```
 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  |    FwdTTL     |B|S| Src Mask  |Forwarding Code|
|               |               |Z| |           |               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

5.1. Query Arrival Time

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).

5.2. Incoming Interface Address

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

5.3. Outgoing Interface Address

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.

5.4. Previous-Hop Router Address

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.

5.5. Packet counts

Note that these packet counts SHOULD be as up to date as possible. If packet counts are not being maintained on the processor that handles the traceroute request in a multi-processor router architecture, the packet SHOULD be delayed while the counters are gathered from the remote processor(s). If this occurs, the Query Arrival Time should be updated to reflect the time at which the packet counts were learned.

5.6. Input packet count on incoming interface

This field contains the number of multicast packets received for all groups and sources on the incoming interface, or 0xffffffff if no count can be reported. This counter should have the same value as ifInMulticastPkts from the IF-MIB for this interface.

5.7. Output packet count on outgoing interface

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 0xffffffff if no count can be reported. This counter should have the same value as ifOutMulticastPkts from the IF-MIB for this interface.

5.8. 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 0xffffffff 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 ipMRoutePkts from the IPMROUTE-STD-MIB for this forwarding entry.

5.9. Rtg Protocol: 8 bits

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

1	DVMRP
2	MOSPF
3	PIM
4	CBT
5	PIM using special routing table
6	PIM using a static route
7	DVMRP using a static route
8	PIM using MBGP (aka BGP4+) route
9	CBT using special routing table
10	CBT using a static route
11	PIM using state created by Assert processing

5.10. FwdTTL: 8 bits

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

5.11. MBZ: 1 bit

Must be zeroed on transmission and ignored on reception.

5.12. 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.

5.13. 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).

5.14. Forwarding Code: 8 bits

This field contains a forwarding information/error code. Defined values include:

Value	Name	Description
0x00	NO_ERROR	No error
0x01	WRONG_IF	Traceroute 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 tracer- oute 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 Rendez-vous Point or Core
0x09	RPF_IF	Traceroute request arrived on the expected RPF interface for this source,group.
0x0A	NO_MULTICAST	Traceroute 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 traceroute requests.
0x83	ADMIN_PROHIB	Traceroute is administratively prohibited.

Note that if a router discovers there is not enough room in a packet to insert its response, it puts the 0x81 error code in the previous router's Forwarding Code field, overwriting any error the previous router placed there. A multicast traceroute client, upon receiving this error, MAY restart the trace at the last hop listed in the packet.

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.

6. 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 remaining MUST be forwarded normally, as MUST a unicast packet that has TTL remaining and is not addressed to this router.

6.1. Traceroute Query

A traceroute Query message is a traceroute message with no response blocks filled in, and uses IGMP type 0x1F.

6.1.1. Packet Verification

Upon receiving a traceroute 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 source 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 6.2.

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.

6.1.2. Normal Processing

When a router receives a traceroute Query and it determines that it is the proper last-hop router, it treats it like a traceroute Request and performs the steps listed in section 6.2.

6.2. Traceroute Request

A traceroute Request is a traceroute message with some number of response blocks filled in, and also uses IGMP type 0x1F. Routers can tell the difference between Queries and Requests by checking the length of the packet.

6.2.1. Packet Verification

If the traceroute Request is not addressed to this router, or if the Request is addressed to a multicast group which is not a link-scoped group (e.g. 224.0.0.x), it MUST be silently ignored.

6.2.2. Normal Processing

When a router receives a traceroute 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, Output Packet Count, and FwdTTL. 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 requester as described in "Forwarding Traceroute Requests".

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.

   If using a shared-tree protocol and there is no source-specific state, or if the source is specified as 0xFFFFFFFF, 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 the forwards the packet to the requester as described in "Forwarding Traceroute

Requests".

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 traceroute is administratively prohibited or the previous hop router does not understand traceroute requests, note the appropriate forwarding code (ADMIN_PROHIB or OLD_ROUTER). If traceroute 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 "Forwarding Traceroute Requests".

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 Rendez-vous 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 traceroute Request, it notes forwarding code PRUNE_SENT. If the router has stopped forwarding down-stream in response to a prune sent by the next hop router, it notes forwarding code PRUNE_RCVD. If the router should nor-mally 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 requester as described in "Forwarding Traceroute Requests".

6.3. Traceroute response

A router must forward all traceroute response packets normally, with no special processing. If a router has initiated a traceroute with a Query or Request message, it may listen for Responses to that traceroute but MUST still forward them as well.

6.4. Forwarding Traceroute Requests

If the Previous-hop router is known for this request and the number of response blocks is less than the number requested, 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. 224.0.0.x), MUST NOT be ALL-SYSTEMS.MCAST.NET (224.0.0.1) and MAY be ALL-ROUTERS.MCAST.NET (224.0.0.2) if the routing protocol in use does not define a more appropriate group. Otherwise, it is sent to the Response Address in the header, as described in "Sending Traceroute Responses". Note that it is not an error for the number of response blocks to be greater than the number requested; such a packet should simply be forwarded to the requester as described in "Sending Traceroute Responses".

6.5. Sending Traceroute Responses

6.5.1. Destination Address

A traceroute response must be sent to the Response Address in the traceroute header.

6.5.2. TTL

If the Response Address is unicast, the router inserts its normal unicast TTL in the IP header. If the Response Address is multicast, the router copies the Response TTL from the traceroute header into the IP header.

6.5.3. Source Address

If the Response Address is unicast, the router may use any of its interface addresses as the source address. Since some multicast routing protocols forward based on source address, if the Response Address is multicast, the router MUST use an address that is known in the multicast routing table if it can make that determination.

6.5.4. Sourcing Multicast Responses

When a router sources a multicast response, the response packet MUST be sent on a single interface, then forwarded as if it were received on that interface. It MUST NOT source the response packet individually on each interface, in order to avoid duplicate packets.

6.6. 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 (0xffffffff).

7. Using multicast traceroute

7.1. Sample Client

This section describes the behavior of an example multicast traceroute client.

7.1.1. Sending Initial Query

When the destination of the trace is the machine running the client, the traceroute Query packet can be sent to the ALL-ROUTERS multicast group (224.0.0.2). 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 trace 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 trace 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, to make sure that routers that are not members of the multicast group notice the packet. See also section 7.2 on determining the last-hop router.

7.1.2. 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 7.3 and 7.4 on receiving the results of a trace.

7.1.3. Collecting Statistics

After a client has determined that it has traced the whole path or as much as it can expect to (see section 7.5), 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 8.3 and 8.4.

Details of performing a multicast traceroute:

7.2. Last hop router

The traceroute 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 traceroute request should be multicasted to the group being traced (since the last hop router listens to that group). All routers except the correct last hop router should ignore any multicast traceroute request received via multicast. Traceroute requests which are multicasted to the group being traced must include the Router Alert IP option [Katz97].

Another alternative is to unicast to the trace destination. Traceroute requests which are unicasted to the trace destination must include the Router Alert IP option [Katz97], in order that the last-hop router is aware of the packet.

If the traceroute querier is attached to the same router as the destination of the request, the traceroute request may be multicasted to 224.0.0.2 (ALL-ROUTERS.MCAST.NET) if the last-hop router is not known.

7.3. First hop router

The traceroute querier may not be unicast reachable from the first hop router. In this case, the querier should set the traceroute response address to a multicast address, and should set the response TTL to a value sufficient for the response from the first hop router to reach the querier. It may be appropriate to start with a small TTL and increase in subsequent attempts until a sufficient TTL is reached, up to an appropriate maximum (such as 192).

The IANA has assigned 224.0.1.32, MTRACE.MCAST.NET, as the default multicast group for multicast traceroute responses. Other groups may be used if needed, e.g. when using mtrace to diagnose problems with the IANA-assigned group.

7.4. Broken intermediate router

A broken intermediate router might simply not understand traceroute packets, and drop them. The querier would then get no response at all from its traceroute 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 time-out).

7.5. Trace termination

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

7.5.1. 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.

7.5.2. Fatal Error

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

7.5.3. No Previous Hop

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

7.5.4. Trace 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.

7.6. Continuing after an error

When the NO_SPACE error occurs, the client might try to continue the trace by starting it at the last hop in the trace. It can do this by unicasting to this router's outgoing interface address, keeping all fields the same. If this results in a single hop and a "WRONG_IF" error, the client may try setting the trace destination to the same outgoing interface address.

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 reply packet received. A client may be able to determine (via mrinfo[Pusa99] or SNMP[Thal99,Thal00]) 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.

7.7. Multicast Traceroute and shared-tree routing protocols

When using shared-tree routing protocols like PIM-SM and CBT, a more advanced client may use multicast traceroute to determine paths or potential paths.

7.7.1. 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.

7.7.2. CBT

When a multicast traceroute reaches a CBT Core, it must simply stop since CBT does not have source-specific state. However, a second trace can be performed, setting the trace destination to the traffic source, the trace group to the group being traced, and the trace source to the Core (or to 0, since CBT does not have source-specific state). This trace Query may be unicasted to the Core. There are two possibilities when combining the two traces:

7.7.2.1. No overlap

If there is no overlap between the two traces, the second trace can be reversed and appended to the first trace. This composite trace shows the full path from the source to the destination.

7.7.2.2. Overlapping paths

If there is a portion of the path that is common to the ends of the two traces, that portion is removed from both traces. Then, as in the no overlap case, the second trace is reversed and appended to

the first trace, and the composite trace again contains the full path.

This algorithm works whether the source has joined the CBT tree or not.

7.8. Protocol-specific considerations

7.8.1. DVMRP

DVMRP's dominant router election and route exchange guarantees that DVMRP routers know whether or not they are the last-hop forwarder for the link and who the previous hop is.

7.8.2. PIM Dense Mode

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.

8. Problem Diagnosis

8.1. Forwarding Inconsistencies

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

8.2. TTL problems

By taking the maximum of (hops from source + forwarding TTL threshold) over all hops, you can discover the TTL required for the source to reach the destination.

8.3. Packet Loss

By taking two traces, you can find packet loss information by comparing the difference in input packet counts to the difference in

output packet counts 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 trace query is propagating, there may be small errors (off by 1 or 2) 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.

8.4. 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.

8.5. 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.

9. Implementation-specific Caveats

Some routers with distributed forwarding architectures may not update the main processor's packet counts often enough for the packet counters to be meaningful on a small time scale. This can be recognized during a periodic trace by seeing positive loss in one trace and negative loss in the next, with no (or small) net loss over a longer interval. The suggested solution to this problem is to simply collect statistics over a longer interval.

In the multicast extensions for SunOS 4.1.x from Xerox PARC, which are the basis for many UNIX-based multicast routers, both the output packet count and the packet forwarding count for the source-group pair are incremented before priority dropping for rate limiting occurs and before the packets are put onto the interface output queue which may overflow. These drops will appear as (positive) loss on the link even though they occur within the router.

In release 3.3/3.4 of the UNIX multicast extensions, a multicast packet generated on a router will be counted as having come in an interface even though it did not. This can create the appearance of negative loss even on a point-to-point link.

In releases up through 3.5/3.6, packets were not counted as input on an interface if the reverse-path forwarding check decided that the packets should be dropped. That causes the packets to appear as lost on the link if they were output by the upstream hop. This situation can arise when two routers on the path for the group being traced are connected by a shared link, and the path for some other group does not flow between those two routers because the downstream router receives packets for the other group on another interface, but the upstream router is the elected forwarder to other routers or hosts on the shared link.

The packet counts for source/group pairs are generally kept in router forwarding caches. These cache entries may be occasionally garbage-collected on routers, so a multicast traceroute client should be prepared to see packet counts decrease. If a long-running traceroute is keeping a "base" to compare against, it should use the post-reset trace as the new "base", as previous values returned by this hop are no longer valid. In addition, it may choose to discard the data for all other hops to cover the same amount of time for all hops.

Some routers (notably the obsolete mrouted 3.3 and 3.4) can constantly reset these packet counts. A client might want to detect routers that are constantly resetting and simply fail to collect statistics for that hop (instead of allowing it to cause all other data to be discarded).

Some routers send byte-swapped counter values. If the difference between a pair of measurements is extremely large, a traceroute client may want to see if the difference is more reasonable when byte-swapped. Note that this heuristic may start misfiring when packet rates get high, so implementations may want to only attempt this heuristic when the packet rate is much different on one router than on surrounding routers.

Some implementations (e.g. UNIX mrouted 3.8 and before) return incorrect time values; the difference between the time values for the same hop in two traces may have no relationship with the amount of time that passed between making the traces. Implementations should check that time values look valid before using them.

10. Acknowledgments

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. A multicast traceroute client, mtrace, 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.

11. IANA Considerations

11.1. Routing Protocols

The IANA is responsible for allocating new Routing Protocol codes. The Routing Protocol code is somewhat problematic, since in the case of protocols like CBT and PIM it must encode both a unicast routing algorithm and a multicast tree-building protocol. The space was not divided into two fields because it was already small and some combinations (e.g. DVMRP) would be wasted.

Routing Protocol codes should be allocated for any combination of protocols that are in common use in the Internet.

11.2. Forwarding Codes

New Forwarding codes must only be created by an RFC that modifies this document's section 7, 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.

12. Security Considerations

12.1. Topology discovery

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

12.2. Traffic rates

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

12.3. Unicast replies

The "Response address" field may be used to send a single packet (the traceroute Reply packet) to an arbitrary unicast address. It is possible to use this facility as a packet amplifier, as a small multicast traceroute Query may turn into a large Reply packet.

13. References

Brad88	Braden, B., D. Borman, C. Partridge, "Computing the Internet Checksum", RFC 1071, ISI, September 1988.
Brad97	Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119/BCP 14, Harvard University, March 1997.
Katz97	Katz, D., "IP Router Alert Option," RFC 2113, Cisco Systems, February 1997.
Pusa99	Pusateri, T., "DVMRP Version 3", work in progress, September 1999.
Thal00	Thaler, D., "PIM MIB", work in progress, July 2000.
Thal99	Thaler, D., "DVMRP MIB", work in progress, October 1999.

14. Authors' Addresses

William C. Fenner
AT&T Labs -- Research
75 Willow Rd.
Menlo Park, CA 94025
United States
Email: fenner@research.att.com

Stephen L. Casner
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134
United States
Email: casner@cisco.com

15. Change History

(To be removed before publication as RFC)

15.1. Changes from draft-ietf-idmr-traceroute-ipm-06.txt:

- Added implementation-specific notes as suggested by Dave Thaler:
  - Forwarding cache entries going away while traffic is flowing, causing reset counters.
  - mrouted 3.3 and 3.4 constant resets
  - byte-swapped counters
  - bogus time due to missed ntohl() parenthesis in mrouted <= 3.8
- Add example of ALL-[protocol]-ROUTERS.MCAST.NET for the multicast-on-prev-hop. (Maybe this isn't important any more; PIM used to be allowed to not know the proper prev hop but that's not true any more)

15.2. Changes from draft-ietf-idmr-traceroute-ipm-05.txt:

- Changes section added.
- Updated abstract

- Added mention of up-to-date packet counts, in particular allowing the delay of an mtrace packet while the counts are fetched in a distributed architecture.

- Added mention of ifInMulticastPkts, ifOutMulticastPkts, and ipM-RoutePkts for clarification of what counts should be used.

- Note that the dropping of duplicate Queries MAY be a 1-back cache and that duplicate Requests MUST NOT be dropped

- Add no-space processing rule

- Note that it's not an error for there to be more blocks than requested, just send it back after adding yours.

- Clean up some of section 8 - move implementation-specific stuff to a separate section, rename "Congestion" to "Packet Loss", note that time delay isn't actually that useful.