Status of This Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts. Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt
The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Abstract

The multiple Destination option on IPv6(MDO6) is an elastic multicast delivery system. Elastic multicast uses a list of final unicast addresses of each destination node, that is embedded into an optional routing header, as the destination of a datagram. Routers on the multicast path could forward datagrams by looking up the next hops router for individual final unicast addresses with the existing unicast routing table. Elastic Multicast has several advantages as follows.

- Scalability concerns with a number of multicast groups, because no special multicast routing information is needed.
- Multicast application programs can freely join and prune by adding/deleting destination addresses to/from a list of destinations without any request for intermediate routers on a multicast delivery tree.
- For efficiency of forwarding, MDO6 has a mechanism for a "tractable list". MDO6 routers can forward datagrams with "tractable list" saving route look up of same direction group of destination nodes.
- For Gradual deployment of Elastic Multicast, datagrams can reach a final destination even if there are routers which can not recognize MDO6 on the multicast delivery path.

1. Introduction

Current multicast uses the Host Group Model, the destination of a multicast datagram is identified by a Group Multicast Address. Routers
that relay datagrams have to maintain routing information for each multicast spanning tree, which causes several scalability problems. [Sala, CM]

Multicast is useful not only for broadcast applications but also many-to-many applications like a video conference. Because many-to-many applications need multicast groups much more, the scalability problem described above becomes pretty critical.

Multiple Destination option on IPv6 is a yet another multicast delivery mechanism that depends only on the existing unicast routing environment. The destination of a multicast datagram is specified by a list of unicast addresses instead of a group multicast address. The list is stored in a routing option header with a bitmap that represents the destinations to send on the list. The router looks up the next hop of each unicast address, using their unicast routing table, if their bitmap is on. Then routers copy the datagram and diverge it for the next hops routers sharing the datagrams if any have destinations have a common next hop entry.

The routing header has to be evaluated on hop-by-hop basis. The hop-by-hop option header indicates on MDO6 routing header follows. The IPv6 destination field of MDO6 datagrams are filled by one of the unicast addresses of the destinations and the type of the MDO6 Hop-by-hop option has a bit prefix ‘00’ so that routers that cannot recognize MDO6 can treat the MDO6 datagram as an ordinal IPv6 datagram and forward to one of the destinations. Datagram reachability is preserved because if it passed a preferable router to divide, it can go back on the next MDO6 router the datagram reached.

"Tractable list" is a technology for routing efficiency. A sender is able to sort the destination list so that the on-bits of destination bitmap must appear continuously on whole stems of the spanning tree. In the sparse multicast situation, multicast packets are passed on to almost routers without diverging. In that case, intermediate routers distinguish the packet that they do not need to diverge only by looking up two addresses located at both ends of the bitmap.

In section 2, the header format of MDO6 is specified.

2. Header Format

MDO6 datagram has 3 extra option headers described below.

IPv6 header
Hop-by-hop options(Elastic multicast)
Routing(Type Elastic multicast address list)
Destination options(Elastic multicast)
other header
upper protocols header
payload
2.1 IPv6 Header

An IPv6 header of MDO6 datagrams have the same format as ordinal IPv6 datagrams. The source address of an IPv6 header is a unicast address of the transmitter. The destination address is a unicast address whose node appears first in the destination bitmap, specified in Section 2.3 and 2.4. Next Header should name "Hop-by-hop option" (00), because MDO6 datagrams must include the MDO6 Hop-by-hop option.

```
+----------------------------------------------+
|Version| Class  | Flow Label| Payload Length| NextHeader(00)| Hop Limit| |
+----------------------------------------------+
| Source Address | (transmitter address) |
+--------------------------------------------------+
| Destination Address | (head of address list) |
+----------------------------------------------+
```

2.2 hop-by-hop option header

```
+----------------------------------------------+
|NextHdr=Routing|HdrExtLen=2|Type=MDO6|Opt Data Len=20|
+----------------------------------------------+
|MDO6 options|PadN Option=1|Opt Data len=0|
+----------------------------------------------+
|Destination Address | (tail of address list) |
+----------------------------------------------+
```

MDO6 datagrams must have the Hop-by-hop option because they should be checked on every intermediate router on their spanning trees.
The Next Header field and Header Extension Length field must be filled by the type of the next header and length of this hop-by-hop option specified by [RFC1883].

The option type number of the MDO6 hop-by-hop option, temporarily in this draft, is XX. (It must be assigned by ICANN in future.) The first 2 bits of XX must be 00 so that an intermediate router that cannot recognize MDO6 skips evaluation of the option and the datagrams are able to pass the router for the IPv6 destination that is one of the destinations on the MDO6 destination list. The third bit of the Hop-by-hop option must be 1 so that the source and destination nodes exclude this options header from targets of the calculation of Authentication Header. Because the destination address field is changed in the multicast delivery path.

MDO6 Options field describes how this datagram should behave. Its value is obtained by applying logical AND to the values below. The meaning of these flags will be explained in section 5.

0x01: tractable
0x02: explore branch
0x04: explore branch exhaustively

The destination address field has a unicast address whose node appears last in the destinations bitmap, that makes a pair with the destination field of the IPv6 header.

Options other than the MDO6 option can be packed in the Hop-by-hop option header. Appearance order of the options is not specified, but it is recommended to settle MDO6 options first. It helps in handling hardware evaluation of tractable list handling that is explained in section 2.1.

2.3 Routing Header

Complete list of unicast addresses of the destinations is enclosed as a routing optional header. An MDO6 routing header is defined as a variation of an IPv6 routing header specified by RFC1883.

Following accordingly RFC1833 the Next Header and Header Extension Length fields are filled with the type of next header and length of the routing header. The type value in the routing header is YY temporarily.

RFC1833 stipulates that when the 4th octet of the routing header is not 0 and it’s type is not recognized by the router, the router must discard the datagram and reply with an ICMP error to the source of the datagram. This enables DoS spoofing attacks, if combined with multicast delivery. So MDO6 strongly recommends that the 4th octet of the routing header is always filled by 0. That guarantees that routers that cannot recognize the MDO6 option only discard datagrams without replying with an ICMP error.
The number of destination unicast addresses must be stored in the 5th octet of the routing header. The maximum number of destinations is 126. This restriction relates to the length limit (8 * 255 octet) of the routing header itself. The 6th to 8 octet are filled by the padding option.

In the 9th to 24th octet, the destination bitmap is stored, that indicates which unicast destinations in the address list that follows are to be sent to. 1 represents "send-to" and 0 is done. If the number of destinations is less than 126, the following bitmap field must be filled by 0.
2.4 Destination Option Header

In the IPv6 specification, it has been carefully determined that the ICMP error reply for multicast datagrams must not occur. But MDO6 datagrams are treated as simple unicast datagrams by routers that cannot recognize MDO6. So error reply may occur anyway.

To prevent this behavior, MDO6 datagram must enclose the destination option header. The prefix to 2 bits of the option type is 010 that means discard the datagram and do not reply with an ICMP error if the node cannot recognize the option.

The identifier is the last Identifier of the ICMP branch explore when a tractable list is established. If a datagram has no tractable list, the Identifier field must be zero-filled.

3. Packet delivery

In this section, method of datagram delivery is explained as scenario based behaviors.

3.1 Delivery Scenario with Non-tractable Destination List.

This is a basic scenario. The sender of MDO6 an datagram stores the destination unicast address list in random order and transmits the packet for the node or next hop router.

The router receives the datagram and checks the hop-by-hop option. Because the MDO6 option type represents that it is a non-tractable list, the router starts route evaluation for all unicast addresses on the list of destinations which the destination bitmap is setted. The routers choose all unicast addresses that have the same next hop result, set these in destination bitmap to 1, and forward the datagrams to the next hop routers.

When the final receiver captures a datagram, the receiver searches for its unicast address in the destination list, then passes it to the upper protocol layer.
An example network was built with 8 hosts connected by 4
ethernets (W,X,Y,Z), 1 leased line and 2 routers. Each host has
addresses a-h. The routers have an interface for leased line(r1,r2)
and 2 interfaces for ethernet(w,x,y,z).

The routing table for each host and router is set as below.

<table>
<thead>
<tr>
<th>Network</th>
<th>Gateway</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>loopback</td>
<td>-</td>
<td>loopback</td>
</tr>
<tr>
<td>W</td>
<td>-</td>
<td>ethernet</td>
</tr>
<tr>
<td>default</td>
<td>w</td>
<td>ethernet</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network</th>
<th>Gateway</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>loopback</td>
<td>-</td>
<td>loopback</td>
</tr>
<tr>
<td>Y</td>
<td>-</td>
<td>ethernet</td>
</tr>
<tr>
<td>default</td>
<td>y</td>
<td>ethernet</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[Host a,b]</th>
<th>[Host c,d]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network</td>
<td>Gateway</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>loopback</td>
<td>-</td>
</tr>
<tr>
<td>W</td>
<td>-</td>
</tr>
<tr>
<td>default</td>
<td>w</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[Host e,f]</th>
<th>[Host g,h]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network</td>
<td>Gateway</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>loopback</td>
<td>-</td>
</tr>
<tr>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td>default</td>
<td>y</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>[R1]</th>
<th>[R2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network</td>
<td>Gateway</td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>loopback</td>
<td>-</td>
</tr>
<tr>
<td>W</td>
<td>r1</td>
</tr>
<tr>
<td>X</td>
<td>r2</td>
</tr>
<tr>
<td>Y</td>
<td>r2</td>
</tr>
<tr>
<td>Z</td>
<td>r2</td>
</tr>
</tbody>
</table>

A process on host a sends a multicast datagram for destinations b,
c, d, e, f, g and h using sendmsg( socket, msg, flags ). A parameter
msg includes a list of unicast destination addresses in struct iovec
style. The protocol stack makes a datagram referencing this iovec as
below.
Here is the detailed datagram 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  Class        |  Flow Label                           |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
| Payload Length               |NextHdr=Option |  Hop Limit    |  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
|                                                               | IPv6
+                                                               +  Hdr
|                                                               |
+                                                               +  |
|                                                               |  Source Address ( = a )
+                                                               +  |
|                                                               +  |
|                                                               +  |
|                                                               +  |
|                                                               |  |
|                                                               +  |
|                                                               +  |
|                                                               |  Destination Address ( = b )
|                                                               +  |
|                                                               +  |
|                                                               +  |
|                                                               +  |
|                                                               +  |
|                                                               +  |
|                                                               |
| NextHdr=Routing| HdrExtLen=2   | RouteType=MDO6|Opt Data Len=20|  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
| # of dest = 7 | PadN Option=1 | Opt Datalen=1 | 0  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  |
| (continued for next page) |
```
Host a looks up the next hop for each address then gets the result below.

b: directly deliverable via ethernet W.
c,d,e,f,g,h: relay via router w.

For host b, host a rewrites the header as below, then transmit it to b.

IPv6 dest = b
bitmap = [1,0,0,0,0,0,0]
dest addr = [b,c,d,e,f,g,h]

Host b captures this datagram and checks the MDO0 hop-by-hop option. A destinations bitmap in a routing header means only one destination is to be sent to, and it find the destination host is itself. Then it passes the datagram to the upper level protocol stack.

For router w, host a rewrites the bitmap as below.
IPv6 dest = c  
bitmap = [0,1,1,1,1,1]
dest addr = [b,c,d,e,f,g,h]

Router R1 captures this datagram and checks the MDO0 hop-by-hop option. It finds the datagram is not for this router and looks up the next hop using the routing table as below.

c,d,g,h: to be relay via router r2.
e,f: directly deliverable via ethernet Y.

Now, for c,d,g,h, r1 sends the datagram rewriting the bitmap as below.
IPv6 dest = c  
bitmap = [0,1,1,0,1,1]
dest addr = [b,c,d,e,f,g,h]

In the same way,
IPv6 dest = e  
bitmap = [0,0,0,1,0,0]
dest addr = [b,c,d,e,f,g,h]

for e and
IPv6 dest = f  
bitmap = [0,0,0,0,1,0]
dest addr = [b,c,d,e,f,g,h]

and for f.

R2 relays the datagrams as well as R1. Then datagrams can reach every destination.

4. Impact for Upper Layer Protocol

MDO6 datagrams have option headers that are related to other options and upper layer protocols. In this section, we describe the impact for upper layer protocols.

The fields below for MDO6 options is rewritten as it travels along the delivery path.

- Destination address of IPv6 header.
- Hop-by-hop option header.
- Destinations bitmap in the routing header.

This has an impact on i) checksum calculation in UDP and ICMP headers and ii) IPsec Authentication Header.

i) Checksum calculation in UDP and ICMP

In both UDP and ICMP, the target of the checksum calculation includes the pseudo header, transport header and payload. Optional headers are not a target. In the target, only the destination address of the pseudo header may be rewritten.
UDP and ICMP datagrams on MDO6 must use 0:0:0(address all zeros) as the destination address of the pseudo header for calculating a checksum.

ii) IPsec Authentication header

Unlike checksum calculation, optional headers are the target of the calculation of hashed value of the IPsec Authentication header. It must be controlled as to which option should be the target of hash value calculation.

- Hop-by-hop option header and routing header must be rewritten by intermediate routers. Third bit of the MDO6 option type must be 1 so that the hop-by-hop option header is excluded from the calculation of the hash value.

- Destination option header of MDO6 is not rewritten by intermediate routers. So it should be included in target. The third bit of the MDO6 option type must be 0 so that the hop-by-hop option header is included in the calculation of the hash value.

5. Tractable Order List

As described in section 3, intermediate routers must look up the next hop for all destinations, if addresses are put in random order. It is too expensive, it will not be compatible with the hardware routing facility because the destination list would have variable length.

A tractable ordered list is a list of destination addresses that has been ordered so that the intermediate router can skip looking up the routing entry. An MDO6 application delivers datagrams for a small number of receivers sparsely distributed over the Internet. Most intermediate routers only relay from an interface to an interface without diverging. With a tractable ordered list, such routers can found that they need not to diverge by looking up the routing entry only 2 times.

To make the tractable ordered list, the sender searches the multicast spanning tree and lines up the leaf destinations in depth first search order. With the example described in section 3, the multicast delivery spanning tree is as below.

```
     a
    /|
    / |
   b R1
  //|
  // |
 ef R2
  ||\\ gh cd
```

Then [b,e,f,g,h,d,c] is a result of depth first search, and it is tractable ordered.
A tractable ordered tree has the characteristic that on every stems of the delivery tree, the series of 1 in the destination bitmap are always continuous. This means that if an intermediate router determines the first and last destinations to be sent have the same next hop, they also can deduce that destinations between the two have the same next hop without looking up the routing entry. In order for intermediate routers to determine easily whether or not they must diverge the MDO6 datagram or not, the first destination address to be sent is stored in the destination address field of the IPv6 header and last destination address is stored in the MDO6 Hop-by-hop option header.

To make the tractable ordered list, three additional types of ICMP datagram (explore branch, branch history and request for re-explore) are introduced. Tractable ordered lists are combined using the procedure outlined below.

i) The sender of MDO6 transmits ICMP_EXPLORE datagrams encapsulated by MDO6 datagrams.

ii) Intermediate routers route the ICMP datagrams recording branch histories at diverging points.

iii) The receiver sends back the ICMP_EXPLORE datagrams to the transmitter with the branch history.

iv) The sender the calculates spanning tree from branch histories.

After the sender has probed the branch environment, the unicast routing may be changed by the alteration of the network topology. That may destroy the tractability of the destination list. Receivers are able to determine some kinds of routing environment changes by observing the TTLs of received datagrams. Receivers notify senders by ICMP_REQ_EXPLORE when they detect the variation of TTLs to recombine tractable ordered list.

5.1 Detailed Behavior of MDO6 Datagram with Tractable Ordered List

The example below details how an MDO6 datagram with a tractable ordered list is relayed by intermediate routers. Host a transmits the datagram to destinations c,d,g,h with the MDO6 option header as below.

| IPv6 src | a |
| IPv6 dst | c |
| IPv6 Opt | MDO6 followed |
| MDO6 option | tractable |
| MDO6 dst | h |
| RouteType | MDO6 |
| # of dest | 4 |
| bitmap | [1,1,1,1] |
| dest addr | [c,d,g,h] |

As explained in section 3, in case the destination list is not tractable, intermediate router R1 must check all destination addresses...
and determines that all four destinations have same next hop r2.

At this time, MDO6 specifies that the list is tractable ordered. R1 looks up the routing table with IPv6 destination address "c" and with MDO6 dest address "h". These have the same next hop and it reasons that destination between c and h also has the same next hop without looking it up. Then R1 forwards datagram toward next hop r2.

R2 captures the datagram then looks up with c and h. At this time, it is found that the next hop for c is X and for h is Z. R2 also needs to look up the next hop for d and h. It found X and Z are next hops.

Toward X, R2 modifies the bitmap of the routing header, destination addresses of IPv6 header and MDO6 hop-by-hop option header and hop limit counter as below, then forwards it.

```
IPv6 src   = a  
IPv6 dst   = c  
IPv6 Opt   = MDO6 followed
MDO6 option = tractable
MDO6 dst    = d  
RouteType  = MDO6
# of dest  = 4
bitmap     = [1,1,0,0] 
dest addr  = [c,d,g,h]
```

The same as in the above, R2 modifies headers as below and forwards toward Z.

```
IPv6 src   = a  
IPv6 dst   = g  
IPv6 Opt   = MDO6 followed
MDO6 option = tractable
MDO6 dst    = h  
RouteType  = MDO6
# of dest  = 4
bitmap     = [0,0,1,1] 
dest addr  = [c,d,g,h]
```

5.2 Explore Branches

MDO6 system sorts the list of destination addresses into tractable order exploring the topology of the multicast delivery tree. The transmitter sends ICMP datagram assembled as a MDO6 datagram itself. Intermediate routers relays the MDO6 datagram recording the histories of divergence. The exploring datagrams reaches the destination then the receiver sends them back to the transmitter in a unicast ICMP datagram with the branch history. The transmitter collects the histories, analyzes the topology of the multicast spanning tree and sorts the destination list into tractable order.
Two types of ICMP datagrams were needed to explore.

- ICMP MD06 explore branch
- ICMP MD06 branch history

Explore branch has 2 modes: effective exploring and exhaustive exploring. Effective exploring means omit exploring sub-trees if the tractable ordered list of a sub-tree is able to be combined trivially.

Exploring is needed in the case below.

- When a new receiver is subscribed to the list.
- When the routing environment is changed.

It is allowable but not recommended to explore periodically because of the cost of processing on intermediate routers. Instead of periodical exploring, it is recommended for receivers to observe the TTLs of received datagrams. The receiver notifies a transmitter when TTLs change, then the transmitter starts re-exploring.

ICMP explore branches datagrams have the format below.

```
+---------+-----------------+----------------+---------+------------------
| IPv6    | Hop-by-hop      | Routing header | IPsec   | ICMP MDO6
| Header | option header   | type = MDO6    | Auth    |   explore branch
|         | type = MDO6     | non-tractable | addr=[a,,z] | Header | length = n
|         | explore branch  | map=[1,,1]     |         | history=
[1,,1],[1,,0],
```

---

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------------------+
| Version| Class        | Flow Label    |
| ^      | Payload Length| NextHdr=Option|
| Hop Limit |              | Hop Limit |
| IPv6    |              | +       |
| + Hdr   |              | +      |
| +      |              | +     |
| Source Address | +    |
| +      |              | +     |
| +      | Destination Address | + |
| (most upper address) | + |
| +      | v |

---
As well as ordinal MDO6 datagrams, the first address of the destination addresses must be set in the destination address of the IPv6 header of ICMP explore branches. The MDO6 hop-by-hop option must be set no-tractable & explore branches, and tail destination address must be set.

To prevent a source address spoofing attack, the ICMP explore branch must be protected by an IPsec Authentification Header. Because branch history is appended by intermediate routers, the target area or AH hash calculation are from IPv6 header to the identifier field. The transmitter and receivers must exchange Security Association information before exploring branches.

The type field of an ICMP header must be MDO6 explore branch(XXX) and the code field must be explore(YY). Checksum must be a calculated result the same as in ordinal ICMP headers. The number of recorded history must be stored in the # of the bitmap field. The identifier is a unique ID that identifies each explore datagram. It is also embedded in the ICMP branch history datagram so that transmitter can collect result datagrams for the explore request. Branch history bitmaps are a
When intermediate MDO6 routers capture the ICMP explore branch datagrams, they check whether they need to diverge them or not. If they need to diverge them, they append a new destination bitmap to the tail of branch history and increment # of bitmaps. If the exploring mode is exhaustive, they forward the datagram for all next hop routers. If the mode is effective they choose a direction to forward in the way described later.

5.2.2 ICMP branch history

ICMP MDO6 branch history is a type of datagram that contains the result of exploring branches sent from receivers back to the transmitter.
This datagram is delivered by an ordinal unicast IP datagram.

IPv6 Header:

Destination address:
unicast address of transmitter.

Source address:
unicast address that send back this history

ICMP header:
Type: ICMP branch explore (= XX)
Code: history (= ZZ)

# of destination addresses:
Destination Addresses:
As explained in the above section, the transmitter can get branch history records from receivers. Remember the example described in 3.1. A transmitter wants to sort the destination list \([b,c,d,e,f,g,h]\) then send ICMP MDO6 explore branch datagram. Receivers send back the datagrams below as a result of exploring.

```
[b,c,d,e,f,g,h] [b,c,d,e,f,g,h] [b,c,d,e,f,g,h]
=============== =============== ===============
b: [1,0,0,0,0,0,0] c: [0,1,1,1,1,1,1] d: [0,1,1,0,0,1,1]
    [0,1,1,0,0,1,1]    [0,1,1,0,0,0,0]    [0,1,1,0,0,0,0]
    [0,1,0,0,0,0,0]    [0,0,1,0,0,0,0]    [0,0,1,0,0,0,0]
    [0,1,0,0,0,1,1]

[b,c,d,e,f,g,h] [b,c,d,e,f,g,h] [b,c,d,e,f,g,h]
=============== =============== ===============
e: [0,1,1,1,1,1,1] f: [0,1,1,1,1,1,1] g: [0,1,1,1,1,1,1]
    [0,0,0,1,1,0,0]    [0,0,0,1,1,0,0]    [0,1,1,0,0,1,1]
    [0,0,0,1,0,0,0]    [0,0,0,0,1,0,0]    [0,0,0,0,0,1,1]
    [0,0,0,0,1,0]

[b,c,d,e,f,g,h]
===============
h: [0,1,1,1,1,1,1]
    [0,1,1,0,0,1,1]
    [0,0,0,0,0,1,1]
    [0,0,0,0,0,0,1]
```

The transmitter sorts the list by the procedure "make_tractable_list()" explained below.

```
/* branch_tree is temporal data that holds such spanning tree structures.
   
   o   o
   /|   /|
  b o(e,f) b o
   |   /|
 o(g,h) o   o
   |   /| /
 o   f e o o
   \| \ \ \ 
 c d c d gh
```

o: divergent point

(e,f) attached by divergent point means that the destination that
is diverged at this point but the topology is not yet known below the point.

Right example is final structure of analyzing. Left one is intermediate one.

```c
make_tree() { /* procedure to combine spanning tree structure. */

make root node and attach all nodes by the root.
/* o(b,c,d,e,f,g,h) */

for ( i = 0; i < # of node ; i++ ) { /* for all destination nodes */
  check the history result from node_i
  depth = length of history node_i /* depth(b) = 1 , depth(h) = 4 */
  hang node_i at a divergent point it is attached \ \ 
  with a stem that has new divergent points \ \ 
  (depth - depth(divergent point)).

  foreach( j = i; i < # of nodes; i++ ) { /* for nodes remains */
    if ( node_j is located on the path from root to the node_i ) {
      attach node_j at the divergent point that fork with node_i;
    }
  }
}

make_regular_list(tree) {
  retrieve tree in depth first order and pick nodes up.
}

make_tractable_list () {
  make_tree(); /* compose spanning tree */
  make_list(); /* make tractable ordered list */
}
```

With this procedure, spanning tree and tractable list are composed as followed.

initialized hang b

```
o(b,c,d,e,f,g,h) => o(c,d,e,f,g,h) => next line
/ b

hang c hang d hang e hang f

_ _ _
o | o | o | o |
/ | /
| | | | |
| o(e,f) | o(e,f) | o | o |
| => | => | => | => |
| o(g,h) | o(g,h) | (f) o | o(g,h) | o o(g,h) |
| | | | | |
| o(d) | o | e | o | f e o |
| \ / \ / \ /
c | c d | c d | c d
```
5.2.4 Effective Exploring

In the example above, the transmitter can compose a complete spanning tree even if the result of exploring from node h is lost because g returns a branch history that shows that h forks at the last divergent point on the way to g.

Effective exploring is an exploring method cutting unnecessary exploring datagrams at divergent points for nodes that are trivially settled.

To cut off the exploring stems, MDO6 routers group the to-be-sent destinations by the next hop. Next, they count the number of groups that have only one destination and two destinations. 6 cases can be considered.

<table>
<thead>
<tr>
<th>2 dest \ 1 dest</th>
<th>no group</th>
<th>1 group</th>
<th>2 or more groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>no group</td>
<td>(i)</td>
<td>(ii)</td>
<td>(iii)</td>
</tr>
<tr>
<td>1 or more group</td>
<td>(iii)</td>
<td>(iii)</td>
<td>(iii)</td>
</tr>
</tbody>
</table>

(i) All groups have three or more destinations. And they need to be explored for a tractable list. Then transfer ICMP explore history datagram for all next hops.

(ii) Only one group has one destination. Routers can cut off this group. The transmitter can determine that this node forks at this divergent point by collecting other history branch results.

(iii) Two or more groups have one destination or one or more groups have two or more destinations. Routers can select and cut off 2 groups that have a destination or 1 group that has 2 destinations, because the transmitter can determine that these 2 nodes fork at this divergent point by collecting other history branch result and it is trivial that 2 variation of sequence of these nodes are also tractable ordered.

The transmitter specifies the modes of exploring by setting MDO6 hop-by-hop options. It is recommended to begin exploring in effective mode. In case exploring fails because of lost datagrams, try to explore exhaustively.

5.3 ICMP Re-explore Branch
Because the tractable list that is ordered by exploring the multicast spanning tree is reflect the unicast routing environment, transition of routing environment may break the tractability of the list.

The transmitter can collaborate with a receiver to reform the tractable list. A receiver records the identifier of tractable lists and its ordinal hop limit count, continuously observes the TTLs of the captured datagrams and notifies the transmitter when it changes. The transmitter can explore the branch again.

An ICMP type value of MDO6 Request re-explore must be XX, and subcode must be YY. In the identifier field the receiver set the last ICMP explore branch receiver captured.

When the transmitter receives the ICMP request re-explore the first time, it starts the exploring procedure. The receiver can notify the transmitter again because ICMP datagrams lost on the way to the transmitter. When the transmitter receives a datagram that has the same identifier again, it must discard it.
6. Discussion

6.1 Security

Many ISP prohibit source routing to prevent packets from passing along the route the ISP cannot control. Using the MDO6 tractable list, a cracker transmits a datagram for the target destination along an illegal relay point while putting target address between relay point address. Routers can discard such illegal datagrams checking the head and tail TLA with other destinations. If head and tail have the same TLA but others are different, the router discards it.

A host that recognizes MDO6 options replies with an ICMP error to the sender. It may cause a spoofing attack. In order to prevent it, MDO6 datagram must include a destination header that causes discarding by a non-MDO6 host. And border routers checks that MDO6 datagrams have a destination header.

6.2 MTU

MDO6 option can have up to 126 destinations. But it shares a 2064 octet length and it is longer than the Ethernet MTU. MDO6 is mainly focused on the usage for small groups of participants. Many multicast applications transmit 1024-length payloads. In this case up to 16 participants can join to the group. (IPv6, UDP, RTP header was under consideration.)

7. References

http://www.isoc.org/inet98/proceedings/6d/6d_3.htm


Authors Addresses

IMAI Yuji
Fujitsu LABORATORIES Ltd.
1-1, Kamikodanaka 4-Chome, Nakahara-ku, Kawasaki 211-8588, Japan
Phone : +81-44-754-2628
Fax : +81-44-754-2793
E-mail: kimai@flab.fujitsu.co.jp

IMAI Yuji Draft - Expires March 2000