Workgroup: Network Working Group
Internet-Draft: draft-chen-pim-be-mrh-01
Published: 16 October 2022
Intended Status: Standards Track
Expires: 19 April 2023

```
Authors: H. Chen     D. Eastlake   M. McBride   Y. Fan
         Futurewei   Futurewei     Futurewei    Casa Systems
         G. Mishra   Y. Liu         A. Wang
         Verizon     China Mobile   China Telecom
         X. Liu            L. Liu
         IBM Corporation   Fujitsu
```

Stateless Best Effort Multicast Using MRH

Abstract

This document describes stateless best effort Multicast along the shortest paths to the egress nodes of a P2MP Path/Tree. The multicast data packet is encapsulated in an IPv6 Multicast Routing Header (MRH). The MRH contains the egress nodes represented by the indexes of the nodes and flexible bit strings for the nodes. The packet is delivered to each of the egress nodes along the shortest path. There is no state stored in the core of the network.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction
  1.1. Acronyms
2. Encoding of P2MP Path/Tree
  2.1. Flexible Bitstring
  2.2. Explicit Nodeindex
  2.3. More Efficient Encoding
3. Node Index Forwarding Table
4. IPv6 Multicast Routing Header (MRH)
5. Procedures at Nodes
  5.1. Procedure at Ingress Node
  5.2. BE Multicast Forwarding Procedure
6. Security Considerations
7. IANA Considerations
8. Acknowledgements
9. References
  9.1. Normative References
  9.2. Informative References
Authors' Addresses

1. Introduction

The potential egress nodes and transit nodes in a network are numbered or indexed from 1 to the number of the nodes. Figure 1 shows an example network having nodes PE1 to PE10 and P1 to P5, where PE1 to PE10 are edge nodes (i.e., potential egress nodes) and P1 to P5 are transit nodes. In this example, these nodes have node indexes 1 to 10 and 11 to 15 respectively. The number labeling a link is the cost of the link. For example, 5 on the link between P5 and PE4 is the cost of the link. The cost of a link without a numeric label is 1.

```
                             [PE2] 2
                             /             PEi(i=1 to 10):Edge Nodes
                            /              Pi(i=1 to 5):Transit Nodes
                           /
                          [P2]--------[PE3] 3
                            /          /
                           /          /4
                          /          /         P2MP path/tree from
                         /        [P5]         PE1 to PE2 - PE6
                        /     ___/ \ \____5
                       /  ___/      \     \___
            1         /  /           \        \
[CE1].....[PE1]------[P1]------[P3]   \      [PE4] 4
            :       / \           \__  \      /
            :      /   \           4\  \  __/
            :     /     \             \ | /
         [PE10] [PE9]   [PE8]         [P4]------[PE5] 5
           10     9       8          /  \
                                    /    \
                                   /      \
                              7  [PE7]    [PE6] 6
```

Figure 1: Network with 10 edges and P2MP tree from PE1 to PE2 - PE6

The P2MP path/tree from ingress PE1 towards egresses PE2 to PE6 (i.e., PE2, PE3, PE4, PE5 and PE6) in Figure 1 is represented by the indexes of the egress nodes of the P2MP path. The indexes of PE2 to PE6 are 2 to 6 (i.e., 2, 3, 4, 5 and 6) respectively. The indexes are represented by a flexible bit string or the indexes directly. A more efficient representation is used. That is that if the latter is more efficient, the indexes are used directly; otherwise, the flexible bit string is used.

A controller such as PCE as a controller can have the information about the node indexes, and send the P2MP path to the ingress of the path.

After receiving a data packet from traffic source CE1, ingress PE1 encapsulates the packet in a MRH with the P2MP path represented by the indexes. The packet is transmitted along the shortest path to each of the egresses.

This document describes the encoding of a P2MP Path/Tree using the indexes of the egress nodes of the tree and specifies the procedure/ behavior of the nodes along the shortest paths to the egresses.

1.1. Acronyms

The following acronyms are used in this document:

CE:
Customer edge/equipment.

MRH: Multicast Routing Header.

P2MP: Point 2 Multi-Point.

PE: Provider Edge.

2. Encoding of P2MP Path/Tree

This section describes two basic encodings of a P2MP tree (i.e., the egress nodes of the tree): flexible bitstring and explicit nodeindex. We encode the tree more efficiently using flexible bitstring and/or explicit nodeindex.

2.1. Flexible Bitstring

A flexible bitstring has four fields:

1). B flag with value 1,

2). start index (StartIndex),

3). size of bitstring (S-BitString) in bytes, and

4). bitstring (BitString), where each bit with value 1 indicates a node index equal to StartIndex plus the bit number. Note that the bit number is counted from left to right and from 0.

For example, the P2MP path/tree from ingress PE1 to egresses PE2 - PE6 (i.e., PE2, PE3, PE4, PE5 and PE6) in Figure 1 is represented in Figure 2 using a flexible bitstring.

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 0 1 2 3 4 5 6 7 1 2 3 4 5 6 7 8
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|             2               |       1       |1|1|1|1|1|0|0|0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|B|         StartIndex          |  S-BitString  |   BitString   |
```

Figure 2: Encoding tree from PE1 to PE2 - PE6 by a flexible bitstring

The indexes of egress nodes PE2 to PE6 are represented by four fields: B flag with value of 1, StartIndex of 15 bits with value of 2, the S-BitString byte with value of 1, and the BitString of 1 byte (i.e., 8 bits) with value 0b11111000.

S-BitString = 1 indicates BitString occupies 1 byte. BitString = 0b11111000 combined with StartIndex = 2 indicates five node indexes

2, 3, 4, 5 and 6. BitString's first bit (bit 0) with value 1 indicates the first node index 2 equal to 2 + 0; the BitString's second bit (bit 1) with value 1 indicates the second node index 3 equal to 2 + 1, and so on.

In this case, the encoding of the P2MP tree uses 4 bytes.

2.2. Explicit Nodeindex

An explicit nodeindex has two fields: B flag with value of 0 and node index (Nodeindex) representing a node index directly/ explicitly.

Suppose that the indexes of egress nodes PE2 to PE6 of the P2MP tree in Figure 1 are 2 to 6 respectively. Figure 3 illustrates the encoding of the tree (i.e., PE2, PE3, PE4, PE5 and PE6) using explicit nodeindex.

```
  Size  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  ---+
   10  |0| NodeIndex = 2 (PE2's Index) |     |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+     |
    8  |0| NodeIndex = 3 (PE3's Index) |   Encoding indexes
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   of PE2 to PE6
    6  |0| NodeIndex = 4 (PE4's Index) |   by Explicit NodeIndex
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+     |
    4  |0| NodeIndex = 5 (PE5's Index) |     |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+     |
    2  |0| NodeIndex = 6 (PE6's Index) |     |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  ---+
       |B|<-------- NodeIndex -------->|
```

Figure 3: Encoding tree from PE1 to PE2 - PE6 by explicit nodeindex

The node index of PE2 is represented by B = 0 and NodeIndex of 15 bits with value of 2 (i.e., NodeIndex = 2); The node index of PE3 is represented by B = 0 and NodeIndex of 15 bits with value of 3 (i.e., NodeIndex = 3); ans so on.

In this case, the encoding of the P2MP tree uses 10 bytes. Using flexible bitstring is more efficient than using explicit nodeIndex.

2.3. More Efficient Encoding

We encode a tree more efficiently using flexible bitstring and/or explicit nodeindex. That is that we encode some egress nodes of the tree using flexible bitstring and the others using explicit nodeindex.

For the tree from PE1 towards PE2 to PE6 in Figure 1, if PE2 to PE6 have their indexes 2 to 6 respectively, we encode the tree using flexible bitstring as shown in Figure 2. Using flexible bitstring to encode the tree is more efficient than using explicit nodeindex as shown in Figure 3.

If PE2 to PE6 have their indexes 102, 503, 904, 905 and 906 respectively, we encode the tree using flexible bitstring and explicit nodeindex as shown in Figure 4.

```
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  ---+
 |0|NodeIndex = 102 (PE2's Index)|   Encoding PE2 and PE3
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   by Explicit NodeIndex
 |0|NodeIndex = 503 (PE3's Index)|     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  ---+          0 1 2 3 4 5 6 7  PE4-PE
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ by
 |1|            904             |       1       |1|1|1|0|0|0|0|0| Bit
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ string
 |B|<-------  StartIndex ------->|<-S-BitString->|<--BitString-->|
```

Figure 4: Encoding tree to PE2 - PE6 by flexible bitstring and explicit index

We encode egress nodes PE2 and PE3 of the tree using two explicit nodeindexes. PE2's index 102 is represented by the first explicit nodeindex with B = 0 and NodeIndex of 15 bits with value of 102 (i.e., NodeIndex = 102). PE3's index 503 is represented by the second explicit nodeindex with B = 0 and NodeIndex of 15 bits with value of 503 (i.e., NodeIndex = 503).

We encode egress nodes PE4 - PE6 (i.e., PE4, PE5 and PE6) of the tree using a flexible bitstring. The indexes of PE4 to PE6 are represented by the flexible bitstring with four fields: B flag of 1 bit with value of 1, StartIndex of 15 bits with value of 904, S-BitString of 8 bits with value of 1, and BitString of 1 byte (i.e., 8 bits) with value 0b11100000. S-BitString = 1 indicates BitString occupies 1 byte. BitString = 0b11100000 combined with StartIndex = 904 indicates thsat the indexes of PE4 to PE6 are 904 to 906 respectively.

The BitString's first bit (i.e., bit 0) = 1 indicates PE4's index 904 (i.e., 904 = 904 + 0); The BitString's second bit (i.e., bit 1) = 1 indicates PE5's index 905 (i.e., 905 = 904 + 1); and The BitString's third bit (i.e., bit 2) = 1 indicates PE6's index 906 (i.e., 906 = 904 + 2).

3. Node Index Forwarding Table

Every node in a network has a Node Index IPv6 Forwarding Table (NIFT). The table has a row for the index of each egress node. The row contains the index of the egress node, the IPv6 address and the index of the next hop on the shortest path to the egress node, and node index bit mask (BM) of the same next hop node (BM-SNH). This table indicates the shortest IGP path to each egress, i.e., the next hop of the shortest path to each egress. This is similar to a unicast forwarding table but organized by exact match node index rather than longest match IP address or the like.

Figure 5 shows an example Node Index IPv6 Forwarding Table of PE1 in Figure 1.

Node Index	IPv6 Address of next hop	Index of next hop	BM of same next hop
1 (PE1)	NULL	NULL	NULL
2 (PE2)	P1's IPv6 address	11 (P1)	0b0111111110
3 (PE3)	P1's IPv6 address	11 (P1)	0b0111111110
4 (PE4)	P1's IPv6 address	11 (P1)	0b0111111110
5 (PE5)	P1's IPv6 address	11 (P1)	0b0111111110
6 (PE6)	P1's IPv6 address	11 (P1)	0b0111111110
7 (PE7)	P1's IPv6 address	11 (P1)	0b0111111110
8 (PE8)	P1's IPv6 address	11 (P1)	0b0111111110
9 (PE9)	P1's IPv6 address	11 (P1)	0b0111111110
10 (PE10)	PE10's IPv6 address	10 (PE10)	0b0000000001

Figure 5: Node Index IPv6 Forwarding Table of PE1

The table has 10 rows/entries of node index, next hop IPv6 address, next hop index, and BM of the same next hop. The next hop to PE1 itself is NULL. The next hop to each of PE2 to PE9 is P1. The next hop to PE10 is PE10. Note: The information such as port number or interface used to forward a packet to the next hop is not shown in the figure, which is the same as the corresponding information in the forwarding table (FIB) of PE1.

For example, the second row/entry contains node index 2 of egress PE2, next hop node P1's IPv6 address, next hop node P1's index 11, and the same next hop P1's bit mask (BM-SNH) 0b0111111110 indicating node indexes 2 to 9 of PE2 to PE9 have the same next hop P1. The tenth row/entry contains node index 10 of egress PE10, next hop node PE10's IPv6 address, next hop node PE10's index 10, and the same next hop PE10's bit mask (BM-SNH) 0b0000000001 indicating node index 10 of PE10 has the same next hop PE10.

Figure 6 shows an example Node Index IPv6 Forwarding Table of P1 in Figure 1. The table has 10 rows/entries of node index, next hop IPv6 address, next hop node index, and BM of the same next hop (BM-SH).

Node Index	IPv6 Address of next hop	Index of next hop	BM of same next hop
1 (PE1)	PE1's IPv6 address	1 (PE1)	0b1000000001
2 (PE2)	P2's IPv6 address	12 (P2)	0b0110000000
3 (PE3)	P2's IPv6 address	12 (P2)	0b0110000000
4 (PE4)	P5's IPv6 address	15 (P5)	0b0001111000
5 (PE5)	P5's IPv6 address	15 (P5)	0b0001111000
6 (PE6)	P5's IPv6 address	15 (P5)	0b0001111000
7 (PE7)	P5's IPv6 address	15 (P5)	0b0001111000
8 (PE8)	PE8's IPv6 address	8 (PE8)	0b0000000100
9 (PE9)	PE9's IPv6 address	9 (PE9)	0b0000000010
10 (PE10)	PE1's IPv6 address	1 (PE1)	0b1000000001

Figure 6: Node Index IPv6 Forwarding Table of P1

For example, since the next hop to PE1 and PE10 is PE1, the first row/entry contains has node index 1 of PE1, next hop PE1's IPv6 address, next hop PE1's index 1, and the same next hop PE1's bit mask (BM-SH) 0b1000000001 indicating node indexes 1 and 10 of PE1 and PE10 have the same next hop PE1.

The next hop to PE2 and PE3 is P2. The second row/entry contains node index 2 of egress PE2, next hop P2's IPv6 address, next hop P2's index 12, and the same next hop P2's bit mask (BM-SH) 0b0110000000 indicating node indexes 2 and 3 of PE2 and PE3 have the same next hop P2.

The next hop to PE4 - PE7 is P5. The fourth row/entry contains node index 4 of egress PE4, next hop P5's IPv6 address, next hop P5's index 15, and the same next hop P5's bit mask (BM-SH) 0b0001111000 indicating node indexes 4 to 7 of PE4 to PE7 have the same next hop P5.

4. IPv6 Multicast Routing Header (MRH)

Figure 7 shows a Multicast Routing Header (MRH) in an IPv6 packet. The IPv6 packet has an IPv6 header with a destination address (DA) and source address (SA) of IPv6, a routing header with Routing type (TBD) indicating MRH and an IP multicast datagram. The routing header is indicated by the Next Header in the IPv6 header.

```
  |<--IPv6 header-->|<-Routing header->|
  +-----------------+------------------+------------------------+
  |  Next Header =  | Next Header      | (an extension header)  |
  |  Routing header |                  | IP multicast datagram  |
  | SA=IPv6 Address | Routing Type =   |                        |
  | DA=IPv6 Address |   TBD (MRH)      |                        |
  |                 | SL, SE, Sub-tree |                        |
  +-----------------+------------------+------------------------+
                    |<----   MRH  ---->|
```

Figure 7: Multicast Routing Header (MRH) in IPv6 packet

The format of the MRH is shown in Figure 8.

```
  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
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | Next Header   |  Hdr Ext Len  |RoutingType=TBD|Version| Flags |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | SL(Sub-tree Left) | SE (Sub-tree End) |       Reserved        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           Sub-tree (encoding of egresses of Sub-tree)         |
 :                                                               :
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8: Format of Multicast Routing Header (MRH)

The MRH has the following fields:

Next Header:
The type of the header after the MRH. Either another extension header or the type of IP multicast datagram in the packet.

Hdr Ext Len: Its value indicates the length of the MRH in a unit of 64 bits (i.e., 8 bytes) excluding the first 8 bytes.

Routing Type: Its value TBD indicates that the routing header is a Multicast Routing Header (MRH).

Version: The Version of the MRH. This document specifies Version one.

Flags: No flag is defined yet.

Sub-tree Left (SL): Its value points to the sub-tree (the start of the subtree).

Sub-tree End (SE): Its value indicates the end of the sub-tree.

Sub-tree: Its value encodes the egress nodes of the sub-tree. A node index MUST NOT occur more than once. The node indexes in sub-tree are ordered.

For the P2MP path/tree from PE1 via P1 to PE2, PE3, PE4, PE5 and PE6 as shown in Figure 1, we select and use the encoding of the tree by flexible bitstring as illustrated in Figure 2. For an IP multicast datagram/packet to be transmitted by the P2MP path/tree, PE1 constructs an IPv6 packet for each sub-tree of the tree and sends the packet containing a MRH and the IP multicast datagram/packet to the next hop along the sub-tree.

The number of sub-trees from PE1 is the number of different next hop nodes from PE1 to the egress nodes (i.e., PE2 to PE6). PE1 gets the next hops to the egress nodes using its Node Index IPv6 Forwarding Table as shown in Figure 5 with the node indexes of the egress nodes, which are 2, 3, 4, 5 and 6. The next hops are the same, which are P1. Thus, there is one sub-tree from PE1 via P1 towards PE2 to PE6.

PE1 sets DA of the IPv6 packet to P1's IPv6 address (P1's IPv6 for short) and SA of the packet to PE1's IPv6 address (PE1's IPv6 for short). PE1 builds the MRH based on the encoding of the tree through including the sub-tree from P1 and setting SL to 4 as a pointer pointing to the sub-tree and setting SE to 4, which is the size of the sub-tree and indicates the end of the sub-tree. Figure 9 shows the packet to be sent to P1, which is received by P1.

```
|  IPv6 Header   | <-------   MRH   -------> |
+----------------+--------------------------+-------------+
|DA = P1's IPv6  |RoutingType=TBD,SL=4,SE=4  |IP multicast |
|SA = PE1's IPv6 |sub-tree from P1 to PE2-PE6|datagram     |
+----------------+--------------------------+-------------+
Size  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5                0 1 2 3 4 5 6 7
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  4  |1|             2               |       1       |1|1|1|1|1|0|0|0|
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |B|<-------  StartIndex ------->|<-S-BitString->|<--BitString-->|
```

Figure 9: IPv6 packet with MRH received by P1

After receiving the IPv6 packet from PE1, P1 determines whether the packet's next header is a MRH through checking if the next header is a routing header, and if so, whether the routing type in the routing header is TBD for MRH. When the next header is the MRH, P1 duplicates the packet for each sub-tree from P1 and sends the packet copy with an updated MRH to the next hop along the sub-tree.

P1 gets the next hops to the egress nodes using its Node Index IPv6 Forwarding Table as shown in Figure 6 with the node indexes of the egress nodes, which are 2, 3, 4, 5 and 6. PE2 and PE3 have the same next hop P2 according to the table. PE4 to PE6 have the same next hop P5.

There are 2 sub-trees from P1. One sub-tree is from P1 via next hop P2 to PE2 and PE3. The other is from P1 via next hop P5 to PE4, PE5 and PE6. P1 duplicates the packet for each of these two sub-trees and sends the packet copy to the next hop along the sub-tree.

P1 sets the DA of one packet copy to P2's IPv6 address. P1 updates the MRH based on the encoding of the tree in Figure 9 through logically anding the BitString of 8 bits with the corresponding 8 bits (i.e., bits 2 to 9) in BM-SNH of PE2 (or PE3) (i.e., removing the egress nodes PE4 to PE6, which are not on the sub-tree from P2 to PE2 and PE3). Figure 10 shows the IPv6 packet to be sent to P2, which is received by P2.

```
|  IPv6 Header    | <-------   MRH   -------> |
+----------------+---------------------------+-------------+
|DA = P2's IPv6  |RoutingType=TBD,SL=4,SE=4  |IP multicast |
|SA = PE1's IPv6 |sub-tree from P2 to PE2-PE3|datagram     |
+----------------+---------------------------+-------------+
Size  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5                0 1 2 3 4 5 6 7
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  4  |1|             2              |       1       |1|1|0|0|0|0|0|0|
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |B|<-------  StartIndex ------->|<-S-BitString->|<--BitString-->|
```

Figure 10: IPv6 packet with MRH received by P2

P1 sets the DA of the other packet copy to P5's IPv6 address. P1 updates the MRH based on the encoding of the tree in Figure 9 through logically anding the BitString of 8 bits with the corresponding 8 bits (i.e., bits 2 to 9) in BM-SNH of PE4 (or PE5 or PE6) (i.e., removing the egress nodes PE2 and PE3, which are not on the sub-tree from P5 to PE4, PE5 and PE6). Figure 11 shows the IPv6 packet to be sent to P5, which is received by P5.

```
|  IPv6 Header    | <-------   MRH   -------> |
+----------------+---------------------------+-------------+
|DA = P5's IPv6  |RoutingType=TBD,SL=4,SE=4  |IP multicast |
|SA = PE1's IPv6 |sub-tree from P5 to PE4-PE6|datagram     |
+----------------+---------------------------+-------------+
Size  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5                0 1 2 3 4 5 6 7
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  4  |1|             2              |       1       |0|0|1|1|1|0|0|0|
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |B|<-------  StartIndex ------->|<-S-BitString->|<--BitString-->|
```

Figure 11: IPv6 packet with MRH received by P5

After receiving the IPv6 packet from P1, P5 determines whether the packet's next header is an MRH. When the next header is an MRH, P5 duplicates the packet for each sub-tree from P5 and sends the packet copy with an updated MRH to the next hop along the sub-tree.

P5 gets the next hops to the egress nodes using its Node Index IPv6 Table with the node indexes of the egress nodes, which are 4, 5 and 6. PE4, PE5 and PE6 have the same next hop P4 according to the table.

P5 sets the DA of the packet copy to P4's IPv6 address. P5 updates the MRH based on the encoding of the tree in Figure 11. Figure 12 shows the packet to be sent to P4, which is received by P4.

```
 |  IPv6 Header   | <-------   MRH   -------> |
 +----------------+--------------------------+-------------+
 |DA = P4's IPv6  |RoutingType=TBD,SL=4,SE=4  |IP multicast |
 |SA = PE1's IPv6 |sub-tree from P4 to PE4-PE6|datagram     |
 +----------------+--------------------------+-------------+
 Size  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5                0 1 2 3 4 5 6 7
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   4  |1|             2               |       1       |0|0|1|1|1|0|0|0|
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |B|<-------  StartIndex ------->|<-S-BitString->|<--BitString-->|
```

Figure 12: IPv6 packet with MRH received by P4

After receiving the IPv6 packet from P5, P4 determines whether the packet's next header is an MRH. When the next header is the MRH, P4 duplicates the packet for each sub-tree from P4 and sends the packet copy with an updated MRH to the next hop along the sub-tree.

P4 gets the next hops to the egress nodes using its Node Index IPv6 Table with the node indexes of the egress nodes, which are 4, 5 and 6. PE4, PE5 and PE6 are the next hops PE4, PE5 and PE6 themselves according to the table.

P4 sends the copy with MRH containing SL = 0 to each of PE4, PE5 and PE6. The packet received by PE4 is shown in Figure 13.

```
      |  IPv6 Header   | <-------   MRH   -------> |
      +----------------+--------------------------+-------------+
      |DA = PE4's IPv6 |RoutingType=TBD,SL=0,SE    |IP multicast |
      |SA = PE1's IPv6 |                          |datagram     |
      +----------------+--------------------------+-------------+
```

Figure 13: IPv6 packet with MRH received by PE4

When a leaf/egress such as PE4 receives an IPv6 packet with MRH having SL = 0, the leaf/egress sends the IP multicast packet to the multicast layer of the leaf/egress.

5. Procedures at Nodes

This section describes the procedure at the ingress of a P2MP path/tree, and the BE multicast forwarding procedure which can be used at every node (i.e., ingress, transit and egress) of the tree.

5.1. Procedure at Ingress Node

In one implementation, for a packet to be transported by a P2MP Path/tree, the ingress of the tree duplicates the packet for each sub-tree of the tree branching from the ingress, encapsulates the packet copy in a MRH containing the sub-tree and sends the encapsulated packet copy to the next hop node along the sub-tree.

For example, there is one sub-tree branching from the ingress of the tree from ingress PE1 via next hop node P1 towards PE2 to PE6 in Figure 1. The sub-tree is from ingress PE1 via next hop node P1 towards PE2 to PE6. Ingress PE1 sends P1 the packet as illustrated in Figure 9.

In another implementation, for a packet to be transported by a P2MP Path/tree, the ingress of the tree encapsulates the packet in a MRH containing the tree and "sends" the encapsulated packet to the ingress itself through calling the BE multicast forwarding procedure of the ingress as shown in Figure 14. This procedure duplicates the encapsulated packet for each sub-tree of the tree branching from the ingress and sends the copy to the next hop node along the sub-tree.

For example, suppose that there is a P2MP path/tree from ingress PE1 to egresses PE2, PE3, PE4, PE5 and PE10 in Figure 1. There are two sub-trees branching from the ingress PE1 of the tree. One is from ingress PE1 via next hop node P1 towards PE2 to PE5; the other is from ingress PE1 to egress PE10. For a packet to be transported by the tree, ingress PE1 encapsulates the packet in a MRH containing the tree and calls the BE multicast forwarding procedure of PE1. The procedure duplicates the encapsulated packet for each of these two sub-trees branching from PE1 and sends the copy to the next hop node along the sub-tree.

5.2. BE Multicast Forwarding Procedure

When receiving an IPv6 packet with a MRH containing a tree/sub-tree, a node duplicates the packet for each sub-tree branching from the node and sends the packet copy with a updated MRH to the next hop along the sub-tree. The number of sub-trees branching from the node is the number of different next hop nodes from the node to the egress nodes of the tree. The node determines the different next hops to the egress nodes using the Node Index Forwarding Table of the node with the node indexes of the egress nodes.

Figure 14 shows a BE Multicast Forwarding Procedure. The execution of the procedure for an IPv6 packet with a MRH at a node duplicates the packet for each sub-tree branching from the node and sends the packet copy with a updated MRH to the next hop along the sub-tree.

```
Pkt-p = the packet received by N;
1. IF the tree from N in Pkt-p's MRH has no egress index,
   THEN discard Pkt-p and return;
2. Find the first egress node index J in the tree in Pkt-p's MRH;
3. IF J is the node index of N itself, THEN copy Pkt-p to Pkt-c,
   decapsulate Pkt-c, send the decapsulated packet to upper layer,
   clear J in the tree in Pkt-p's MRH and go to step 1;
4. Get the next hop IPv6 address (NH-IPv6) and BM-SNH from N's NIFT
   using J as the "index" into the NIFT;
5. Copy Pkt-p to Pkt-c, remove the egress node indexes from the tree
   in Pkt-c's MRH that do not have the same next hop as node index J,
   set DA of Pkt-c to NH-IPv6, send Pkt-c to DA (i.e., the next hop);
6. Remove the egress node indexes having the same next hop as node
   index J from the tree in Pkt-p's MRH. Go to step 1.
```

Figure 14: BE Multicast Forwarding Procedure at Node N

Initially, Pkt-p is the IPv6 packet received by node N.

At step 1, the procedure checks if the tree from N in Pkt-p's MRH does not have any egress node index. If the tree does not have any egress node index, the procedure discards Pkt-p and return; otherwise (i.e., the tree has some egress node indexes), the procedure proceeds to next step (i.e., step 2).

SL and SE in the MRH indicates the start and end of the tree from N respectively. If each NodeIndex and BitString in the tree are zeros, the tree does not have any egress node index. In one option, for a flexible bitstring with a StartIndex and a BitString, when the BitString becomes zeros, the StartIndex is set to zero (0). In this case, if each NodeIndex and StartIndex in the tree are zeros, the tree does not have any egress node index.

At step 2, the procedure finds the first egress node index J in the tree from N in Pkt-p's MRH. J is the first node index represented by a NodeIndex with value J or represented indirectly by a flexible bitstring.

At step 3, the procedure checks if node index J is the index of node N itself. If so, the procedure duplicates Pkt-p to Pkt-c, decapsulates the packet copy (i.e., Pkt-c), sends the decapsulated packet copy (i.e., IP multicast datagram/packet) to the IP multicast forwarding module, clears node index J in the tree from N in Pkt-p's MRH, and go to step 1; otherwise (i.e., node index J is not the node index of N), the procedure proceeds to next step (i.e., step 4).

Clearing node index J in the tree is setting NodeIndex to 0 when node index J is represented by NodeIndex with value J, or setting the bit for the node index J to 0 in the BitString when node index J is represented by the BitString.

At step 4, the procedure gets the next hop IPv6 address (NH-IPv6 for short) and the BM-SNH from Node Index Forwarding Table of N using node index J as the "index" into the table.

At step 5, the procedure duplicates Pkt-p to Pkt-c, removes the egress node indexes from the tree from N in packet copy's MRH (i.e., Pkt-c's MRH) that do not have the same next hop as node index J, sets DA of the packet copy to NH-IPv6, sends the copy to DA (i.e., the next hop).

Removing the egress node indexes from the tree that do not have the same next hop as node index J is logically ANDing each BitString with the BM-SNH's bits corresponding to the BitString (i.e., BitString = BitString AND BM-SNH's bits corresponding to BitString), and setting each NodeIndex to 0 when node index in NodeIndex does not have the same next hop as node index J.

At step 6, the procedure removes the egress node indexes having the same next hop as node index J from the tree from N in Pkt-p's MRH, and then go to step 1.

Removing the egress node indexes from the tree that have the same next hop as node index J is logically ANDing each BitString with INVERSE of the BM-SNH's bits corresponding to the BitString (i.e., BitString = BitString AND ~BM-SNH's bits corresponding to BitString), and setting each NodeIndex field to 0 when node index in the field has the same next hop as node index J.

After or while changing the tree in the MRH, each of step 3, 5 and 6 also updates SL and SE to indicate the start and end of the tree/ sub-tree in the MRH respectively, wherein the updated SL points to the first flexible bitstring with a bit having value 1 or the first NodeIndex with a value greater than 0, and the updated SE is the size of the tree/sub-tree from the start pointed by the updated SL to the last flexible bitstring with a bit having value 1 or the last NodeIndex with a value greater than 0.

6. Security Considerations

For general IPv6 and IPv6 extension header security considerations, see [RFC8200]. More TBD

7. IANA Considerations

IANA is requested to assign a new Routing Type in the subregistry "Routing Types" under registry "Internet Protocol Version 6 (IPv6) Parameters" as follows:

Value	Description	Reference
TBD (8 suggested)	Multicast Routing Header	This document

8. Acknowledgements

TBD

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

9.2. Informative References

[I-D.chen-pim-srv6-p2mp-path] Chen, H., McBride, M., Fan, Y., Li, Z., Geng, X., Toy, M., Mishra, G. S., Wang, A., Liu, L., and X. Liu, "Stateless SRv6 Point-to-Multipoint Path", Work in Progress, Internet-Draft, draft-chen-pim-srv6-p2mp-path-06, 30 April 2022, <https://www.ietf.org/archive/id/draft-chen-pim-srv6-p2mp-path-06.txt>.

[I-D.ietf-pim-sr-p2mp-policy] (editor), D. V., Filsfils, C., Parekh, R., Bidgoli, H., and Z. Zhang, "Segment Routing Point-to-Multipoint Policy", Work in Progress, Internet-Draft, draft-ietf-pim-sr-p2mp-policy-05, 2 July 2022, <https://www.ietf.org/archive/id/draft-ietf-pim-sr-p2mp-policy-05.txt>.

Authors' Addresses

Huaimo Chen
Futurewei
Boston, MA,

United States of America

Email: Huaimo.chen@futurewei.com

Donald E. Eastlake 3rd
Futurewei
2386 Panoramic Circle
Apopka, FL, 32703
United States of America

Phone: +1-508-333-2270
Email: d3e3e3@gmail.com

Mike McBride
Futurewei

Email: michael.mcbride@futurewei.com

Yanhe Fan
Casa Systems
United States of America

Email: yfan@casa-systems.com

Gyan S. Mishra
Verizon
13101 Columbia Pike
Silver Spring, MD 20904
United States of America

Phone: 301 502-1347
Email: gyan.s.mishra@verizon.com

Yisong Liu
China Mobile

Email: liuyisong@chinamobile.com

Aijun Wang
China Telecom
Beiqijia Town, Changping District
Beijing
102209
China

Email: wangaj3@chinatelecom.cn

Xufeng Liu
IBM Corporation
United States of America

Email: xufeng.liu.ietf@gmail.com

Lei Liu
Fujitsu
United States of America

Email: liulei.kddi@gmail.com