Workgroup: Network Working Group
Internet-Draft: draft-chen-pim-adaptive-te-02
Published: 21 October 2023
Intended Status: Standards Track
Expires: 23 April 2024
Authors: H. Chen M. McBride Y. Fan Z. Li
Futurewei Futurewei Casa Systems Huawei
X. Geng M. Toy G. Mishra Y. Liu
Huawei Verizon Verizon China Mobile
A. Wang L. Liu X. Liu
China Telecom Fujitsu Alef Edge

Adaptive Stateless TE Multicast

Abstract

This document describes a multicast solution which provides adaptive stateless traffic engineering along an explicit P2MP path/tree. Each portion of the tree is encoded by a most efficient encoding method. A tree portion includes all or some of the links/branches/subtrees from any number of nodes on the tree. An IPv6 extension header is used to encapsulate a packet which is to be multicast at the ingress. The overhead of the encapsulated packet is minimal. The packet is delivered to the egresses along the tree. 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

Table of Contents

1. Introduction

A data center node may have hundreds (or even thousands) of adjacencies/links. A point-to-multipoint (P2MP) path/tree, used to multicast traffic, may be big and sporadic. One portion of the tree may be scattered along a large range of node links. Another portion of the tree may include only a few links of a node at one end.

Encoding the tree, using only one particular method, may not be the most efficient.

This document describes adaptive stateless traffic engineering (TE) multicast along an explicit P2MP path/tree. Each portion of the tree is encoded by a most efficient encoding method. A tree portion may include all, or only a few, of the links/branches from any node on the tree.

The links/branches from a node are split into groups when a combination of encoding each group is more efficient than encoding all the links/branches from the node using only one method.

An new IPv6 extension header, a TE multicast routing header (MRH) with the tree encoded, is used to encapsulate a multicast packet at the ingress. The overhead of the encapsulated packet is minimal. The packet is delivered to the leaves/egresses along the tree. There is no state stored in the core of the network.

2. Overview of Adaptive TE Multicast

This section illustrates Adaptive TE Multicast through an example.

2.1. Example Network with P2MP Path/Tree

Figure 1 shows an example network having nodes P1 to P4 and PE1 to PE20 (other nodes not shown). For each of the links connected to a node, a number called local link number (or link number for short) is assigned to it. For example, PE1 has three links: link from PE1 to P1, link from PE1 to PE20, and link from PE1 to CE1. These three links have link numbers 4, 5 and 6 respectively on PE1. P1 has 16 links (other links not shown): a special Split Branch (SB) link, links from P1 to P2 - P3, P1 to PE8 - PE19, and P1 to PE1. These 16 links have link numbers 3, 4 - 5, 108 - 119 and 120 respectively on P1. P2 has 3 links: P2 to PE2, PE3 and P1. These 3 links have link numbers 4, 6 and 8 respectively on P2. P3 has two links: P3 to P1 and P3 to P4. These two links have link numbers 4 and 5 respectively on P3. P4 has 5 links (other links not shown): P4 to PE4 - PE7 and P4 to P3. These 5 links have link numbers 54 - 57 and 58 respectively on P4.

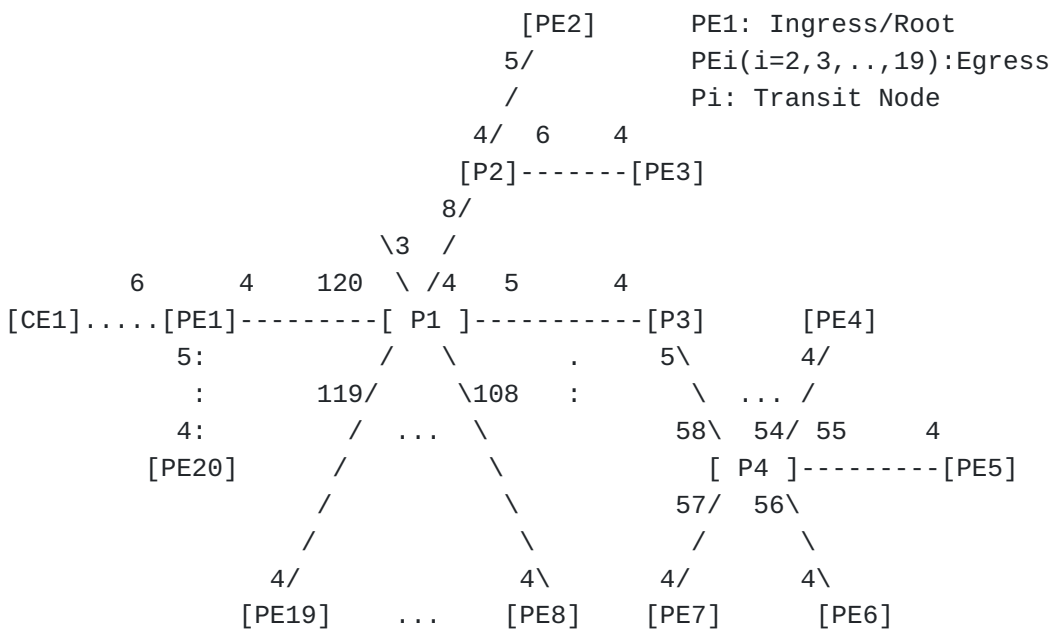

Figure 1: Network with Path/Tree from PE1 to egresses PE2 - PE19

The P2MP path/tree from ingress PE1 via P1 to egresses PE2 - PE19 in Figure 1 is represented by the link numbers along the path: PE1's link number 4; P1's link numbers 4, 5, 108 - 119; P2's link numbers 4 and 6; P3's link number 5; and P4's link numbers 54, 55, 56 and 57.

2.2. Encoding Links/Branches by Link Numbers

The link/branch from PE1 to P1 on the tree is encoded using link number as shown in Figure 2.

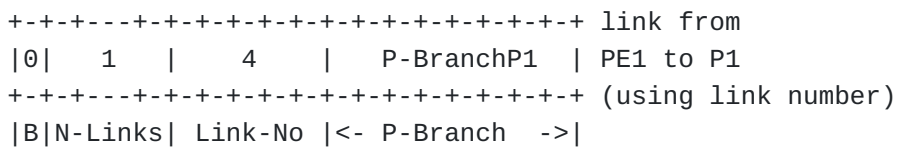

Figure 2: Encoding link/branch from PE1 using link number

*B flag with value 0 (i.e., B = 0) indicates that the link from PE1 is encoded by its link number.

*N-Links (number of links) field following the B flag has a value of 1, indicating there is 1 link from PE1.

*Link-No (Link Number) field with value 4 indicates the link number of the link from PE1 to P1. The link number of the link from PE1 to P1 is 4.

*P-Branch (pointer to branch) field following the Link-No field with value P-BranchP1 indicates ("points" to) the start of the branches/links/subtree from P1.

Encoding the link from PE1 by link number uses 17 bits when B, N-Links, Link-No and B-Branch fields occupy 1, 3, 5 and 8 bits respectively.

Encoding the link/branch from PE1 by half flexible bitstring is shown in Figure 3.

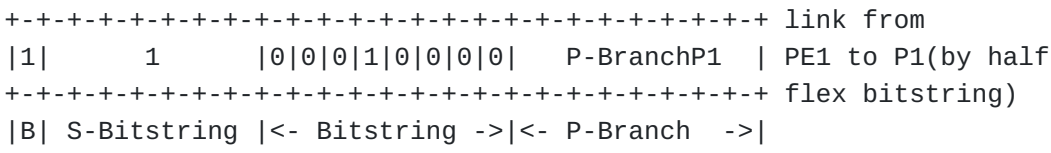

Figure 3: Encoding link/branch from PE1 by half flexible bitstring

*B flag with value 1 (i.e., B = 1) indicates that the link from PE1 is encoded by bitstring.

*S-Bitstring (size of bitstring) field with value of 1 indicates the size of the Bitstring field is 1 byte.

*Bitstring field with the 4-th bit having value of 1 indicates the link from PE1 to P1 with link number 4. The link number of the link from PE1 to P1 is 4.

*P-Branch field is the same as that in Figure 2.

Encoding the link/branch from PE1 by half flexible bitstring uses 24 bits when B, S-Bitstring, Bitstring and B-Branch fields occupy 1, 7, 8 and 8 bits respectively.

For encoding the link/branch from PE1, using link number is more efficient than using half flexible bitstring. The former is selected.

Note: Encoding the link/branch from PE1 by fix bitstring is shown in Figure 4. The size of bitstring is fixed and is the maximum link number (bits), which is 6 (bits).

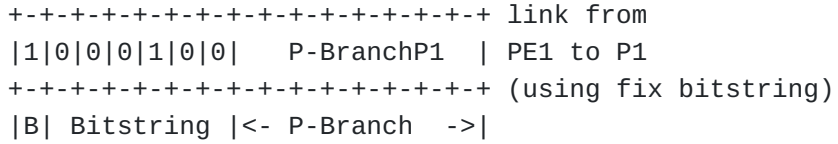

Figure 4: Encoding link/branch from PE1 using fix bitstring

*B flag with value 1 (i.e., B = 1) indicates that the link from PE1 is encoded by fix bitstring.

*Bitstring field with the 4-th bit having value of 1 indicates the link from PE1 to P1 with link number 4.

*P-Branch field is the same as that in Figure 2.

This encoding uses 15 bits and is more efficient than the one by link number (which needs 17 bits). However, if there are changes on links of PE1 (e.g., adding a link to PE1), the maximum link number (e.g., 6) used in the fix bitstring is different from the one (e.g., 7) on PE1 and encoding the link from PE1 by fix bitstring can not be used. This document will not use fix bitstring.

2.3. Encoding Links/Branches by Different Bitstrings

When different bitstrings can be selected to encode a portion of a tree, a Bitstring Identifier (B-I) field is defined to indicate which bitstring is used. Value 0 and 1 in B-I field indicate flexible and half flexible bitstring respectively.

Encoding the links/branches from P4 by half flexible bitstring is shown in Figure 5.

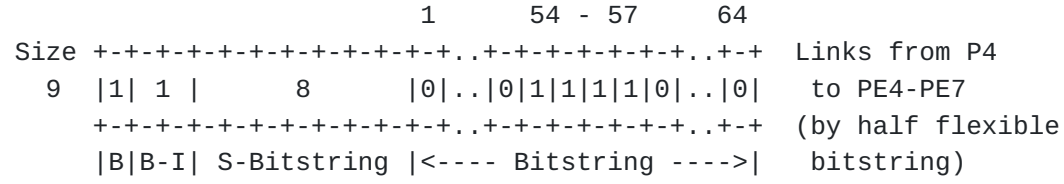

Figure 5: Encoding links/branches from P4 by half flexible bitstring

*B flag with value 1 (i.e., B = 1) indicates that the links from P4 are encoded by bitstring.

*B-I field with value 1 (i.e., B-I = 1) indicates that the links from P4 are encoded by half flexible bitstring.

*S-Bitstring (size of bitstring) field with value of 8 indicates the size of the Bitstring field is 8 bytes (i.e., 64 bits).

*Bitstring field with each of the 54-th - 57-th bit having value of 1 indicates that the links from P4 with link numbers 54 - 57 (i.e., 54, 55, 56, 57) are on the path/tree.

Encoding the links/branches from P4 by half flexible bitstring uses 9 bytes.

Encoding the links/branches from P4 by flexible bitstring is shown in Figure 6.

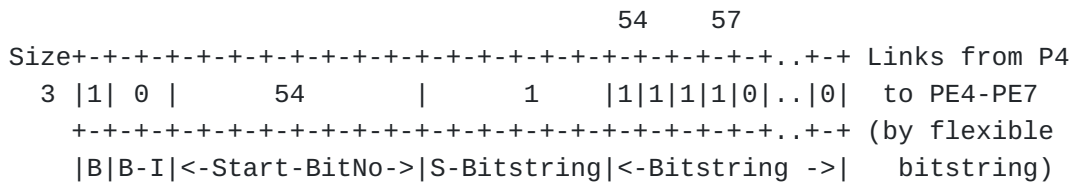

Figure 6: Encoding links/branches from P4 by flexible bitstring

*B flag with value 1 (i.e., B = 1) indicates that the links from P4 are encoded by bitstring.

*B-I field with value 0 (i.e., B-I = 0) indicates that the links from P4 are encoded by flexible bitstring.

*Start-BitNo (Start Bit Number) field has value of 54, indicating the first bit in the Bitstring field is bit 54, the second bit is bit 55, the third bit is bit 56, and so on.

*S-Bitstring (size of bitstring) field with value of 1 indicates the size of the Bitstring field is 1 byte (i.e., 8 bits).

*Bitstring field with each of the 54-th - 57-th bit having value of 1 indicates that the links from P4 with link numbers 54 - 57 (i.e., 54, 55, 56, 57) are on the path/tree.

Encoding the links/branches from P4 by flexible bitstring uses 3 bytes and is more efficient than the one by half flexible bitstring, which uses 9 bytes.

Encoding the links/branches from P1 by half flexible bitstring is shown in Figure 7.

Figure 7: Encoding links/branches from P1 by half flexible bitstring

*B flag with value 1 (i.e., B = 1) indicates that the links from P1 are encoded by bitstring.

*B-I field with value 1 (i.e., B-I = 1) indicates that the links from P1 are encoded by half flexible bitstring.

*S-Bitstring (size of bitstring) field with value of 15 indicates the size of the Bitstring field is 15 bytes (i.e., 120 bits).

*Bitstring field with each of the 4-th, 5-th, 108-th - 119-th bit having value of 1 indicates that the links from P1 with link numbers 4, 5, 108 - 119 are on the path/tree.

*P-Branch field contains the "Pointers" to the starts of the branches/links from the next hop nodes of P1. The first "Pointer" has value P-BranchP2 to the start of the branches/links from P2. The second "Pointer" has value P-BranchP3 to the start of the branch/link from P3.

Encoding the links/branches from P1 by half flexible bitstring uses 18 bytes.

2.4. Encoding Links/Branches from a Node in Multiple Ways

Encoding the links/branches from P1 in multiple ways is shown in Figure 8. The links/branches from P1 on the P2MP path/tree is split into two groups (named G1 and G2) by Split Branch (SB) links. The first group has links/branches from P1 to P2 - P3. The second group has links/branches from P1 to PE8 - PE19. The first group is encoded by link numbers. The second group is encoded by flexible bitstring.

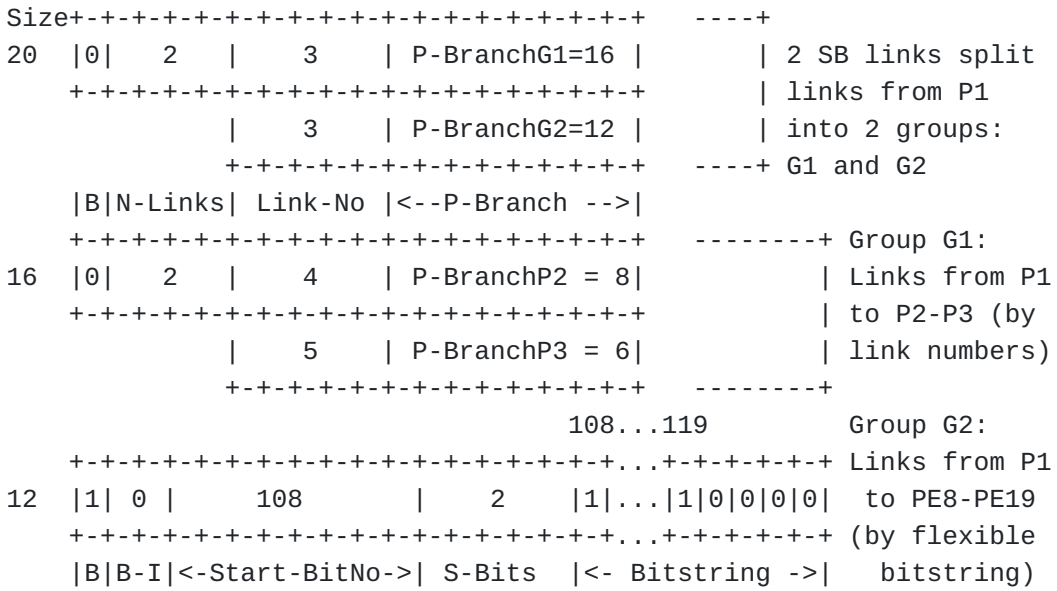

Figure 8: Encoding links from P1 by link number and flexible bitstring

Two SB links with link number 3 split the links/branches of P1 into two groups: G1 and G2. The first SB link with link number 3 is followed by P-BranchG1=16, which "points" to the start (byte 16) of the first group of links/branches from P1. The second SB link with link number 3 is followed by P-BranchG2=12, which "points" to the start (byte 12) of the second group of links/branches from P1. The encoding of the links/branches from P1 to P2 - P3 is similar to the encoding of the links/branches from PE1 to P1 by link number in Figure 2. The encoding of the links/branches from P1 to PE8 - PE19 is similar to the encoding of the links/branches from P4 to PE4 - PE7 by flexible bitstring in Figure 6.

In this case, encoding the links from P1 by SB links, link numbers and flexible bitstring needs 12 bytes (4 bytes (byte 20 - 17) for two SB links by link numbers (padding not shown), 4 bytes (byte 16 - 13) for the first group using link numbers, and 4 bytes (byte 12 - 9) for the second group by flexible bitstring). Encoding links from P1 by any one encoding method needs more space. For example, encoding links from P1 by half flexible needs 18 bytes.

2.5. Encoding P2MP Path/Tree

Encoding the P2MP path/tree is shown in Figure 9.

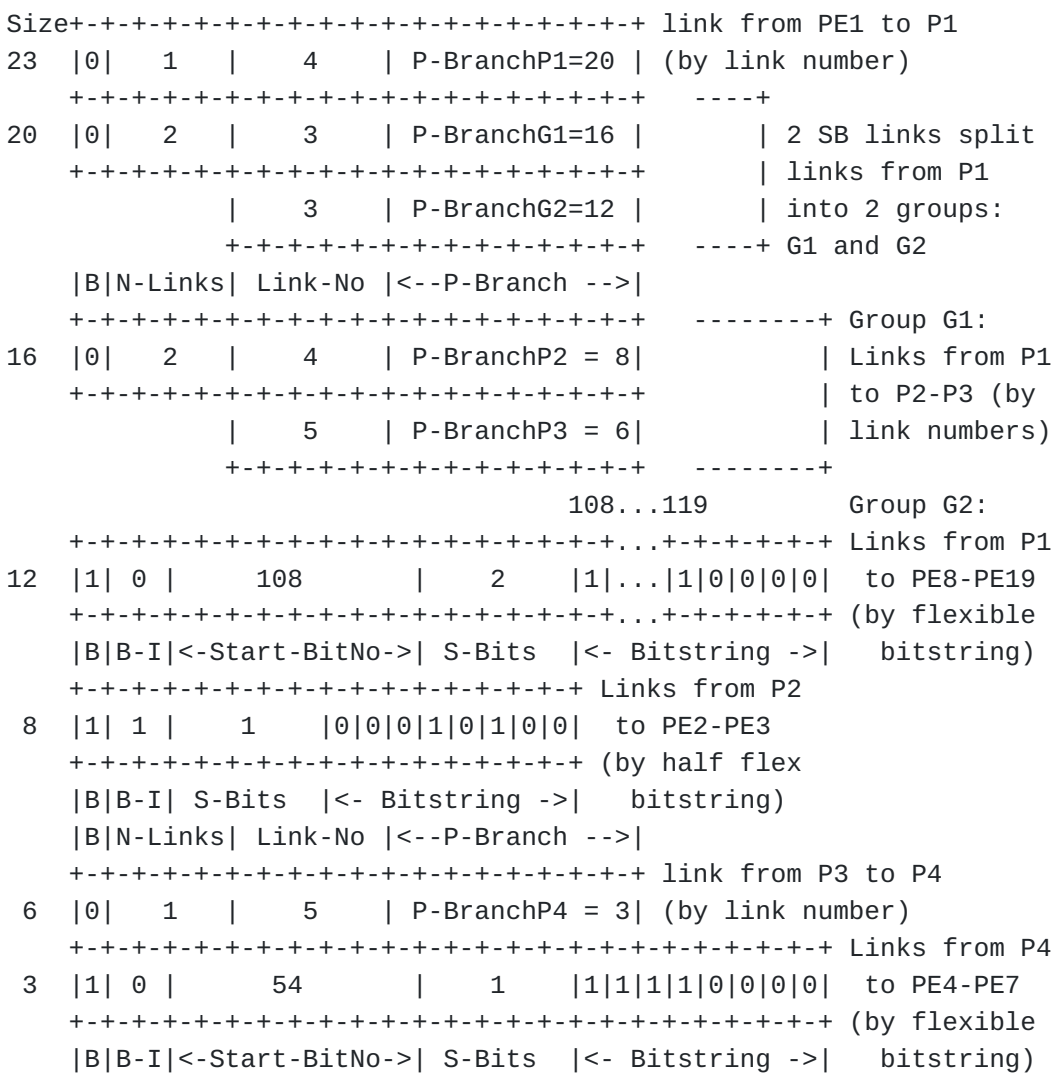

Figure 9: Encoding P2MP path/tree

The link from PE1 to P1 is encoded by link number in 3 bytes (byte 23 - 21 (i.e., 23, 22 and 21), padding not shown). The Link-No field with value of 4 indicates the link with link number 4 from PE1 to P1. The value 20 in P-Branch field (i.e., P-BranchP1 = 20) "points" to the start (byte 20) of the links/branches from P1. Encoding the link from PE1 using link number is more efficient than other encodings when there are changes on links of PE1.

The links/branches from P1 are split into two small groups (G1 and G2) of links/branches using SB links in 4 bytes (byte 20 - 17). Links from P1 to P2 - P3 are in group G1 and are encoded using link numbers in 4 bytes (byte 16 - 13). Links from P1 to PE8 - PE19 are in group G2 and are encoded using flexible bitstring in 4 bytes (byte 12 - 9). The first SB link with link number 3 is followed by P-BranchG1=16, which "points" to the start (byte 16) of the first group of links/branches from P1. The second SB link with link number 3 is followed by P-BranchG2=12, which "points" to the start (byte

12) of the second group of links/branches from P1. The encoding of the SB links from P1 to G1 and G2 contains two links with link number 3 and two P-Branch fields with values 16 and 12 as pointers pointing to the links from P1 to P2 - P3 and the links from P1 to PE8 - PE19 respectively.

The links from P2 to PE2 - PE3 are encoded by half flexible bitstring in 2 bytes (byte 8 - 7), wherein B-I = 1 indicates half flexible bitstring.

The links/branches from P1 to P2 - P3 are encoded by link numbers in 4 bytes (byte 16 - 13). The link from P3 to P4 is encoded by link number in 3 bytes (byte 6 - 4).

The links/branches from P1 to PE8 - PE19 are encoded by flexible bitstring in 4 bytes (byte 12 - 9), wherein B-I = 0 indicates flexible bitstring.

The links/branches from P4 to PE4 - PE7 are encoded by flexible bitstring in 3 bytes (byte 3 - 1), wherein B-I = 0 indicates flexible bitstring.

The encoding of the tree by selecting a most efficient encoding method for each portion of the tree is optimal. It uses 23 bytes in total. The encoding of the tree by half flexible bitstring uses 35 bytes (3 bytes for link from PE1 to P1, 18 bytes for links from P1 to P2-P3 and PE8-PE19, 2 bytes for link from P2 to PE2-PE3, 3 bytes for link from P3 to P4, and 9 bytes for link from P4 to PE4-PE7). The encoding of the tree by flexible bitstring uses 33 bytes (4 bytes for link from PE1 to P1, 19 bytes for links from P1 to P2-P3 and PE8-PE19, 3 bytes for link from P2 to PE2-PE3, 4 bytes for link from P3 to P4, and 3 bytes for link from P4 to PE4-PE7). The encoding of the tree by link number uses 38 bytes (assuming 1.5 bytes for a big link number such as 108 and 54, 3 bytes for link from PE1 to P1, 22 bytes for links from P1 to P2-P3 and PE8-PE19, 3 bytes for link from P2 to PE2-PE3, 3 bytes for link from P3 to P4, and 7 bytes for link from P4 to PE4-PE7).

A controller such as PCE as a controller has the P2MP path and the link numbers of the links on every node. The controller can send the ingress the P2MP path encoded.

After receiving a packet from traffic source CE1, ingress PE1 encapsulates the packet in a MRH with the P2MP path encoded. The packet in the MRH is transmitted along the path to the egresses of the path.

When a node such as P1 receives a packet with the MRH, the node gets each of its link numbers, finds the address of the next hop from an

neighbor address table using the link number, and sends the packet to the next hop.

2.6. Neighbor IPv6 Address Table

Figure 10 shows the neighbor IPv6 address table of P1. The table has 16 rows of link number, link type and IPv6 address of next hop node. The first row has link number 3 and link type SB (Split Branch) for the split branch link of P1. The other columns in the first row are NULL (empty). The 2nd row has link number 4, link type P2P (Point to Point) for link from P1 to next hop P2 and IPv6 address of next hop P2. The 3rd row has link number 5, link type P2P for link from P1 to next hop P3 and P3's IPv6 address. The 4-th row has link number 108, link type P2P2E (P2P to Egress) for link from P1 to next hop egress PE8 and PE8's IPv6 address. The 15-th row has link number 119, link type P2P2E for link from P1 to next hop egress PE19 and PE19's IPv6 address. The 16-th row has link number 120, link type P2P for link from P1 to next hop PE1 and PE1's IPv6 address.

Link number	Link type	Address of next hop
3	SB	NULL
4	P2P	IPv6 address of P2
5	P2P	IPv6 address of P3
108	P2P2E	IPv6 address of PE8
~ :	:	: ~
119	P2P2E	IPv6 address of PE19
120	P2P	IPv6 address of PE20

Figure 10: Neighbor IPv6 address table of P1

Using link number 4, P1 gets P2's IPv6 address from the table; using link number 5, P1 gets P3's IPv6 address from the table; using link number 108, P1 knows that the link with link number 108 is a P2P2E link from the table and gets PE8's IPv6 address from the table; and so on.

Using link number 3, P1 knows that the link with link number 3 is a split branch (SB) link from the table. P1 gets the P-Branch for the SB link pointing to a group of links, duplicates the packet received

for each of the links in the group and sends a packet copy to the next hop of the link.

3. Multicast Routing Header (MRH)

Figure 11 shows a Multicast Routing Header (MRH) in an IPv6 packet. The IPv6 packet comprises 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.

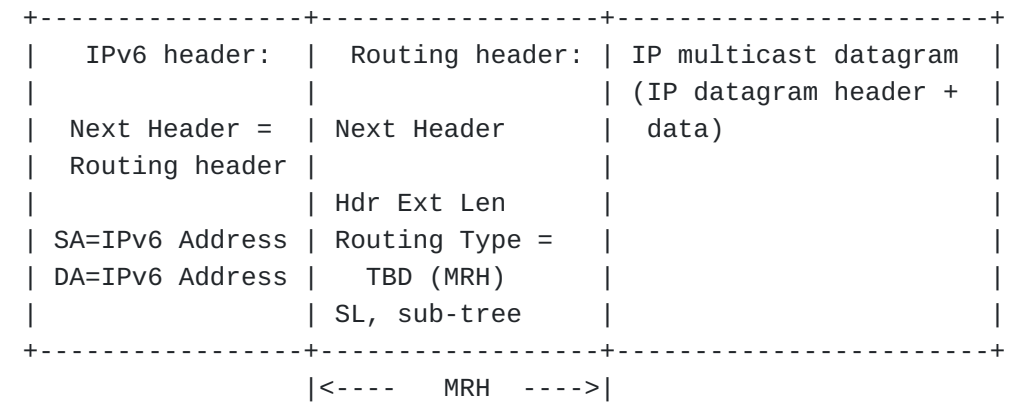

Figure 11: Multicast Routing Header (MRH) in IPv6 packet

The format of the MRH is shown in Figure 12.

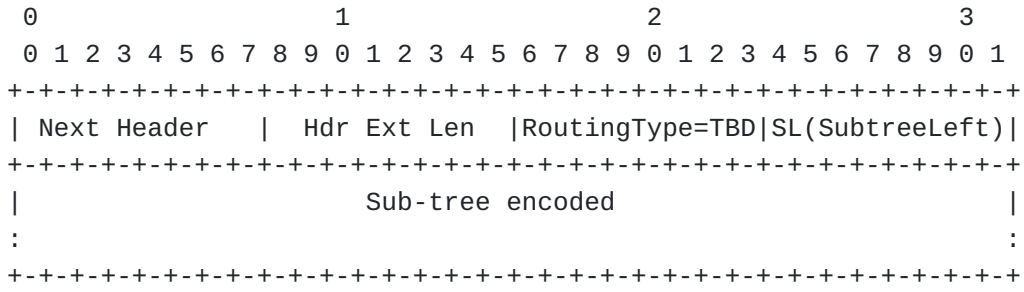

Figure 12: Format of Multicast Routing Header (MRH)

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) for TE multicast.

Sub-tree Left (SL): Its value as a pointer points to the sub-tree.

Sub-tree: Its value encodes the TE multicast sub-tree/tree.

The P2MP path/tree from PE1 via P1 to PE2 - PE19 in Figure 1 is encoded as illustrated in Figure 9. For an IP multicast packet to be transmitted by the P2MP path/tree, PE1 constructs an IPv6 packet for each sub-tree/branch from PE1 and sends the packet containing a MRH and the IP multicast packet to the next hop along the sub-tree.

The P2MP path/tree has one sub-tree from PE1 via P1 to PE2 - PE19. PE1 finds P1's IPv6 address from PE1's neighbor IPv6 address table using the link number 4 of the link from PE1 to P1 and sets DA (destination address) of the IPv6 packet to P1's IPv6 address and SA (source address) of the IPv6 packet to PE1's IPv6 address. PE1 builds the MRH based on the encoding of the tree in Figure 9, sets SL to 20 (P-BranchP1 = 20 for the link from PE1 to P1). Figure 13 shows the IPv6 packet to be sent to P1, which is received by P1. Note that some details are not shown.

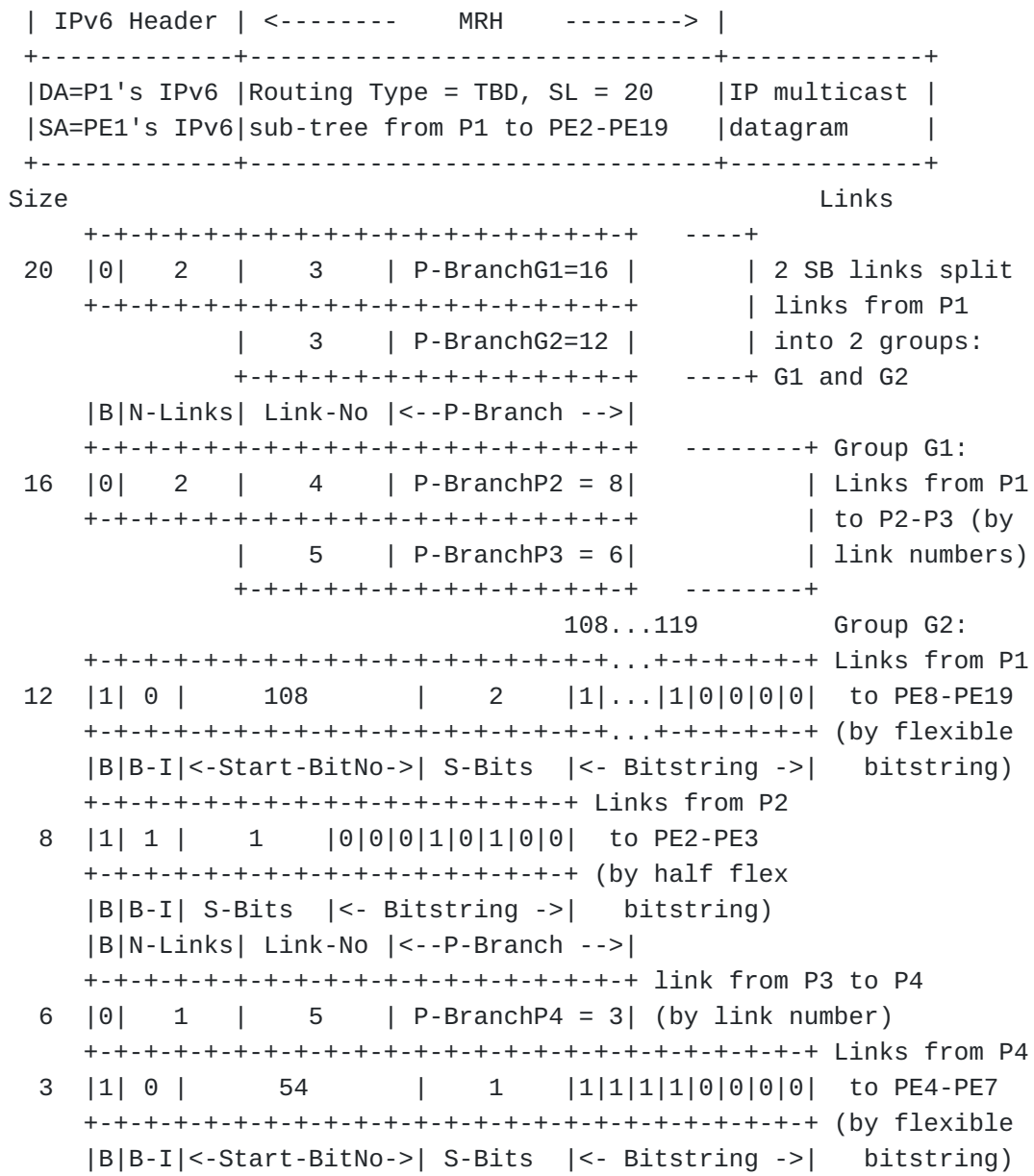

Figure 13: IPv6 packet with MRH received by P1

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 routing header and routing type in the routing header is TBD for MRH. When the next header is the MRH, P1 duplicates the packet for each link/branch/sub-tree from P1 on the path/tree and sends the packet copy with an updated MRH (note: only SL is updated) to the next hop along the link.

The number of links from P1 is 2, which is the value in N-Links field pointed by SL = 20. These 2 links are 2 SB links spliting 14 links from P1 on the path/tree into two groups G1 and G2. The first SB link has its P-Branch field (P-BranchG1 = 16), which "points" to the start (byte 16) of group G1 of the 2 links from P1 to P2-P3 on

the path/tree. The second SB link has its P-Branch field (P-BranchG2 = 12), which "points" to the start (byte 12) of group G2 of the 12 links from P1 to PE8-PE19 on the path/tree.

For each of these 2 links in group G1, P1 duplicates the packet received for the link. For the link from P1 to P2 in group G1, P1 finds IPv6 address of P2 from P1's neighbor table using link number 4 of the link from P1 to P2, sets DA of a packet copy to P2's IPv6 address, SL in the copy to P-BranchP2 = 8, and sends the copy to P2. Figure 14 shows the IPv6 packet to be sent to P2, which is received by P2. Note that some details are not shown.

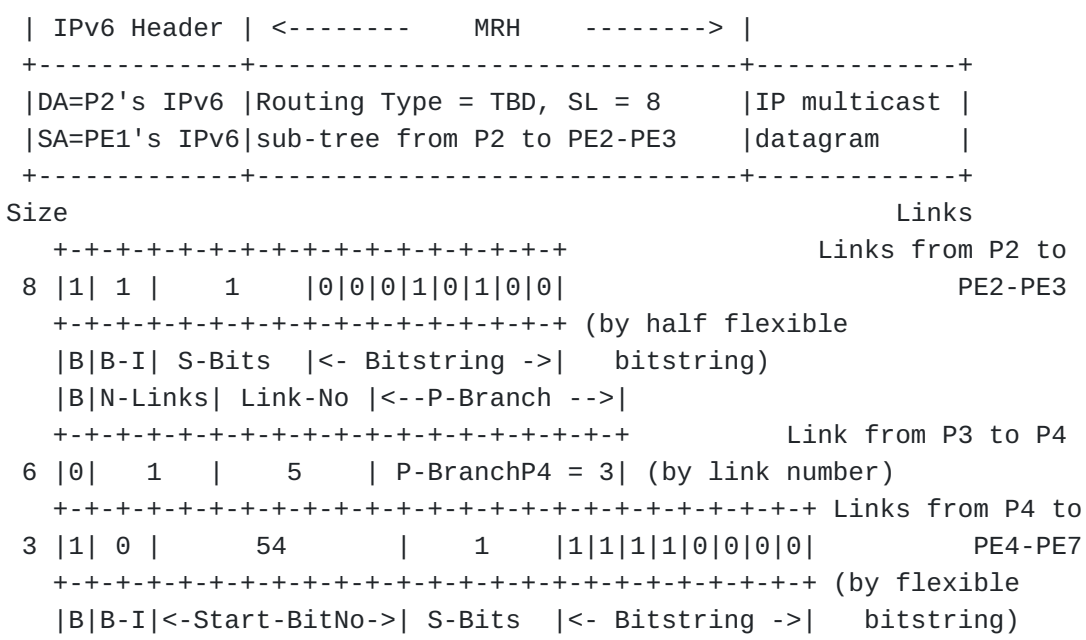

Figure 14: IPv6 packet with MRH received by P2

Similarly, for the link from P1 to P3 in group G1, P1 finds IPv6 address of P3 from P1's neighbor table using link number 5 of the link from P1 to P3, sets DA of a packet copy to P3's IPv6 address, SL in the copy to P-BranchP3 = 6, and sends the copy to P3.

For each of the 12 links in group G2, P1 duplicates the packet received for the link. For the link from P1 to egress PE8 in group G2, P1 finds IPv6 address of PE8 from P1's neighbor table using link number 108 of the link from P1 to PE8, sets DA of a packet copy to PE8's IPv6 address, SL in the copy to 0 since egress PE8 has no link/branch, and sends the copy to PE8. Figure 15 shows the IPv6 packet to be sent to PE8, which is received by PE8. Note that some details are not shown.

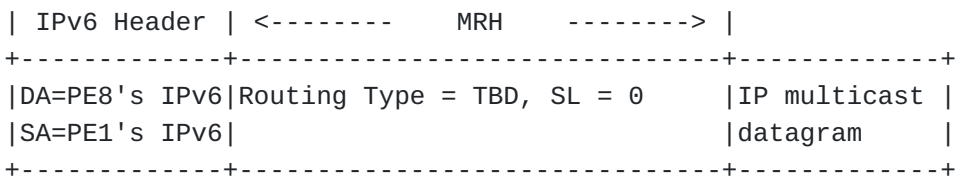

Figure 15: IPv6 packet with MRH received by PE8

Similarly, for the other 11 links from P1 to PE9 - PE119 in group G2, P1 sends a packet copy to each of the egress nodes PE9 - PE19.

After receiving the packet from P1, PE8 determines whether the packet's next header is a MRH. When the packet's next header is the MRH, PE8 checks if PE8 itself is an egress/leaf through checking whether SL is 0. When PE8 is an egress, PE8 decapsulates the packet and sends the IP multicast datagram to the IP multicast forwarding module.

Alternatively, after receiving the packet from PE1 and determining that the packet's next header is a MRH, P1 checks if each of its next hops is an egress/leaf. When the next hop is an egress, P1 decapsulates the packet and sends the IP multicast datagram to the next hop. Since 12 next hops PE8 - PE19 are egresses, P1 sends the IP multicast datagram to each of PE8 - PE19.

4. Procedures/Behaviors

This section describes the procedures or behaviors on the ingress, transit and egress/leaf node of a P2MP path to deliver a packet received from the path to its destinations.

4.1. Procedure/Behavior on Ingress Node

For a packet to be transported by a P2MP Path, the ingress of the P2MP path duplicates the packet for each link/branch/sub-tree of the P2MP path 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 link.

For example, there is one sub-tree branching from the ingress of the P2MP path/tree in Figure 1. The sub-tree is from ingress PE1 via next hop node P1 towards PE2 to PE19. The sub-tree in the MRH is the encoding of the subtree from P1 (i.e., branches from P1) as shown in Figure 13. The SL in the MRH is 20, which as a pointer points to the start (byte 20) of the subtree (i.e., the start of the links/ branches from P1).

4.2. Procedure/Behavior on Transit Node

When a transit node of a P2MP path/tree receives a packet transported by the P2MP path/tree, the node determines whether the current routing header is a MRH. After determining that it is a MRH, the node executes the procedure to duplicate the packet for each of the downstream links of the node on the P2MP path/tree and send the packet copy to next hop (i.e., downstream node) of the link.

Suppose that the transit node receives the packet from a upstream link from a upstream node to the transit node and there are n downstream links from the transit node on the P2MP path/tree (i.e., there are n branches/sub-trees from the transit node on the P2MP path/tree, where n is greater than zero).

The information about n downstream links from the transit node is pointed by SL. When B = 1, the number of links/branches from the transit node is the number of bits with value 1 in the Bitstring field. The information about n downstream links is encoded by the Bitstring field and the fields following the Bitstring field. When B = 0, the number of links/branches from the transit node is the value in the N-Links field. The information about n downstream links is encoded by the Link-No fields and the fields following the Link-No fields.

For each of the downstream links of the transit node on the path/tree, the transit node duplicates the packet for the link, sets SL in the packet copy accordingly.

When the link is a link to an egress/leaf (i.e., next hop node), the transit node sets SL in the packet copy to 0, DA to the IPv6 address of the next hop (i.e., the egress) and sends the packet copy to the next hop node.

When the link is a link to another transit node, the transit node sets SL in the packet copy to the value in the P-Branch field for the link. The transit node gets the IPv6 address of the next hop from the neighbor IP table of the transit node using the link number of the link, sets DA of the packet copy to the IPv6 address of the next hop and sends the packet copy to the DA (i.e., the next hop).

When the link is a SB link, the transit node sets SL in the packet copy to the value in the P-Branch field for the link and processes the links from the start of the links pointed by SL. The transit node duplicates the packet for each of the links, sets SL in the packet copy accordingly, sets DA of the packet copy to the IPv6 address of the next hop of the link, and sends the packet copy to the DA (i.e., the next hop).

4.3. Procedure/Behavior on Egress Node

When an egress node of a P2MP path receives a packet transported by the path, the DA of the packet is the IPv6 address of the egress node and there is an indication in the MRH for the egress/leaf. The egress node proceeds to process the next header in the packet.

For example, after receiving the IPv6 packet from P1, PE8 determines whether the packet's next header is a MRH. When the packet's next header is the MRH, PE8 checks if PE8 itself is an egress/leaf through checking whether SL is 0. When PE8 is an egress, PE8 decapsulates the packet and sends the IP multicast datagram to the IP multicast forwarding module.

5. Simplified Version

This section focuses on a simplified version of Adaptive Stateless TE Multicast and makes some comparisons.

5.1. Simplified Adaptive TE Multicast

Encoding a P2MP tree by selecting a most efficient encoding method from multiple methods for each portion of the tree is called full scan encoding. Encoding a P2MP tree by selecting a more efficient encoding method from two methods for each portion of the tree is called simplified encoding, where one method is link number and the other is flexible bitstring.

The simplified encoding is used to encode a P2MP tree in the Simplified Adaptive TE Multicast.

A flag of 1 bit called B flag indicates which encoding method is used for a portion of the tree.

- B flag with value 0 (i.e., B = 0) indicates that the links in the portion are encoded by link numbers.

- B flag with value 1 (i.e., B = 1) indicates that the links in the portion are encoded by flexible bitstring.

There is no Bitstring Identifier (B-I) field for indicating which bitstring is used since only flexible bitstring is used. It is indicated by B flag with value 1 (i.e., B = 1).

The Multicast Routing Header contains a sub-tree using simplified encoding. The procedures on the ingress, transit and egress/leaf nodes of a P2MP path/tree consider only sub-trees using simplified encoding.

Encoding the P2MP path/tree in Figure 1 by the simplified encoding is shown in Figure 16.

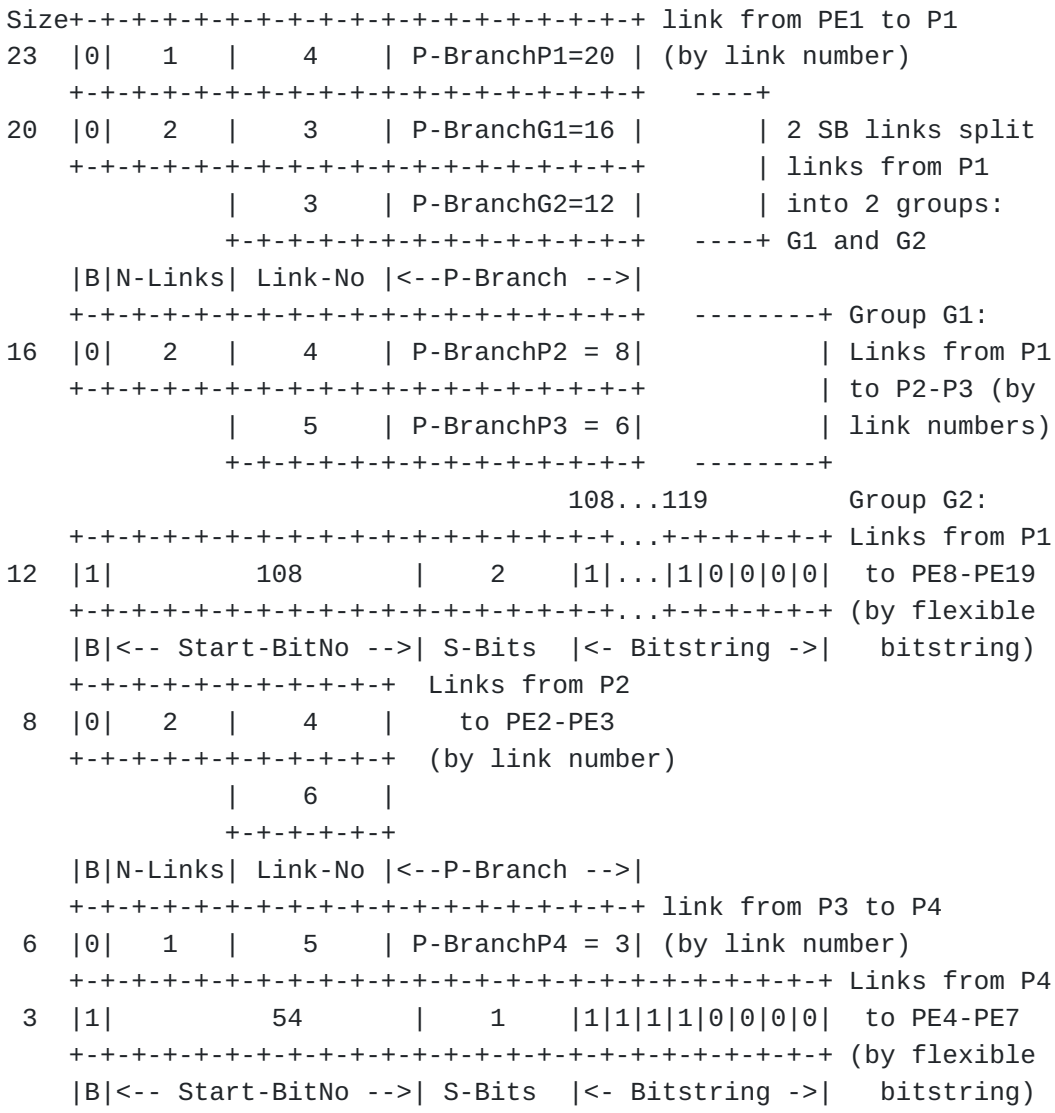

Figure 16: Encoding P2MP path/tree by simplified encoding

The link from PE1 to P1, 2 SB links from P1 to groups G1 and G2, 2 links (in group G1) from P1 to P2 and P3, and link from P3 to P4 are encoded by link numbers in the same way as the one in Figure 9.

The links from P2 to leaves PE2 and PE3 are encoded by link numbers in 2 bytes (byte 8 and 7). There is no P-Branch field for each of these two links since PE2 and PE3 are leaves.

The links/branches from P1 to PE8 - PE19 are encoded by flexible bitstring in 4 bytes (byte 12 - 9), wherein B = 1 indicates flexible bitstring.

The links/branches from P4 to PE4 - PE7 are encoded by flexible bitstring in 3 bytes (byte 3 - 1), wherein B = 1 indicates flexible bitstring.

The encoding of the tree using simplified encoding uses 23 bytes in total.

5.2. Comparisons

In general, the encoding of a P2MP tree using the simplified encoding is much more efficient than the encoding of the tree using any single encoding method. For example, the encoding of the tree in Figure 1 by simplified encoding uses 23 bytes in total. The encoding of the tree by half flexible bitstring uses 35 bytes. The encoding of the tree by flexible bitstring uses 33 bytes. The encoding of the tree by link number uses 38 bytes.

Overall, the encoding of a P2MP tree by the full scan encoding is optimal. The encoding of the tree by the simplified encoding is very close to optimal. For example, the encoding of the tree in Figure 1 by the full scan encoding uses 23 bytes. The encoding of the same tree by the simplified encoding uses 23 bytes too.

The Simplified Adaptive TE Multicast is simpler than the full version of Adaptive TE Multicast. The encoding of a P2MP tree in the simplified version is simpler. The procedures on the ingress and transit nodes of the tree in the simplified version are simpler.

6. IANA Considerations

This document requests assigning a new Routing Type in the subregistry "Routing Types" under registry "Internet Protocol Version 6 (IPv6) Parameters" as follows:

Value	Description	Reference
TBD (7 suggested)	Multicast Routing Header	This document

7. Security Considerations

TBD

8. Acknowledgements

The authors would like to thank Jeffrey Zhang and Toerless Eckert for the valuable comments and suggestions on this draft.

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-08, 24 April 2023, <https://datatracker.ietf.org/doc/html/draft-chen-pim-srv6-p2mp-path-08>.

[I-D.ietf-pim-sr-p2mp-policy] Voyer, D., Filsfils, C., Parekh, R., Bidgoli, H., and Z. J. Zhang, "Segment Routing Point-to-Multipoint Policy", Work in Progress, Internet-Draft, draft-ietf-pim-sr-p2mp-policy-07, 11 October 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-pim-sr-p2mp-policy-07>.

[I-D.liu-msr6-use-cases] Liu, Y., Yang, F., Wang, A., Zhang, Geng, X., and Z. Li, "MSR6(Multicast Source Routing over IPv6) Use Cases", Work in Progress, Internet-Draft, draft-liu-msr6-use-cases-01, 11 July 2022, <https://datatracker.ietf.org/doc/html/draft-liu-msr6-use-cases-01>.

Authors' Addresses

Huaimo Chen
Futurewei
Boston, MA,
United States of America

Email: Huaimo.chen@futurewei.com

Mike McBride
Futurewei

Email: michael.mcbride@futurewei.com

Yanhe Fan
Casa Systems
United States of America

Email: yfan@casa-systems.com

Zhenbin Li
Huawei

Email: lizhenbin@huawei.com

Xuesong Geng
Huawei

Email: gengxuesong@huawei.com

Mehmet Toy
Verizon
United States of America

Email: mehmet.toy@verizon.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

Lei Liu
Fujitsu
United States of America

Email: liulei.kddi@gmail.com

Xufeng Liu
Alef Edge
United States of America

Email: xufeng.liu.ietf@gmail.com