Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: December 16, 2021

Yimin Shen
Zhaohui Zhang
Juniper Networks
Rishabh Parekh
Cisco Systems
Hooman Bidgoli
Nokia
Yuji Kamite
NTT Communications

June 14, 2021

Point-to-Multipoint Transport Using Chain Replication in Segment Routing

draft-shen-spring-p2mp-transport-chain-04

Abstract

This document specifies a point-to-multipoint (P2MP) transport mechanism based on chain replication. It can be used in segment routing to achieve traffic optimization for multicast.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 2
2. Specification of Requirements 3
3. Applicability 3
4. P2MP Transport Using Chain Replication 4
4.1. P2MP Chain 5
4.2. Bud Segment 6
4.3. Forwarding Behaviors 6
4.3.1. SR-MPLS 6
4.3.2. SRv6 7
4.4. Example 7
5. Path Computation for P2MP Chains 9
6. Special Purpose Bud Segments 10
7. IANA Considerations 10
8. Security Considerations 10
9. Acknowledgements 11
10. Contributors 11
11. References 11
11.1. Normative References 11
11.2. Informative References 12
Authors' Addresses 12

1. Introduction

The Segment Routing Architecture [RFC8402] describes segment routing (SR) and its instantiation in two data planes, i.e. MPLS and IPv6. In SR, point-to-multipoint (P2MP) transport is currently achieved by using two approaches. The first approach is ingress replication, where a dedicated point-to-point (P2P) SR tunnel is set up from a root node to each leaf node, and the root node replicates and sends packets via a bundle of such P2P SR tunnels to all the leaf nodes. Although this approach provides P2MP reachability, it does not consider traffic optimization across the tunnels.

The second approach is to use P2MP trees. This approach can achieve maximum traffic optimization, but it relies a controller or path computation element (PCE) to provision and manage "replication segments" on branch nodes. The replication segments are essentially P2MP-tree state (i.e. transport tunnel state) on transit routers. Therefore, this approach is not fully aligned with SR's principles of single-point provisioning (at ingress routers) and stateless core network.

This document introduces a new solution for P2MP transport in SR, based on "chain replication". In this solution, P2MP transport is achieved by constructing a set of "P2MP chain tunnels" (or simply "P2MP chains") from a root node to leaf nodes. Each P2MP chain is a single-path tunnel, with a leaf node at tail end and some transit leaf nodes along the path, resembling a chain. The leaf node at the tail end behaves as a normal receiver. Each transit leaf node behaves as a receiver and a transit router, by replicating incoming packets once for local processing off the chain, and forwarding the original packets down the chain. The root node sends packets via the set of P2MP chains to all the leaf nodes.

As each P2MP chain can reach multiple leaf nodes via a single flow of packets, this solution is considered to be more optimal than ingress replication. Compared to the P2MP-tree based approach, this solution can retain the simplicity of SR, including single-point provisioning and statelessness in the core of a network.

2. Specification of Requirements

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] and [RFC8174].

3. Applicability

The P2MP transport mechanism in this document is generally applicable to all networks. However, it may benefit more for certain types of topologies than others. These topologies include ring topologies, linear topologies, topologies with leaf nodes concentrated in geographical sites which can be modeled as leaf groups (or clusters), etc.

The mechanism does not create any state of P2MP tunnel or P2MP tree on routers. It is transparent to all transit routers. Leaf nodes intended to take advantage of the mechanism will need to support the new forwarding behaviors specified in this document. For other leaf nodes, the mechanism has a backward compatibility to allow them to be reached by P2P tunnels of ingress replication. Path computation and P2MP chain construction will need to be supported by controllers or root nodes, depending on the location of the computation.

The mechanism is applicable to both SR-MPLS [RFC8660] and SRv6 [SRv6-SRH], [SRv6-Programming].

In this mechanism, if a leaf node needs to know the service level context (e.g. source, VPN) of a P2MP stream, it must rely on the

information contained in payload headers (i.e. inner headers) of packets. In SR-MPLS, service labels may be allocated from a domain-wide common block (DCB) to serve as globally unique context indicators. In SRv6, a root node's IP address or an upstream-assigned context indicator may be encoded in the source address of IPv6 header, or a downstream-assigned context indicator may be encoded in the ARG portion of a service SID.

This document introduces a new type of segments, called bud segments (Section 4.2). The segments are generic in nature. They may be used in cases other than P2MP transport, such as traffic mirroring and monitoring, OAM, etc. These use cases are out of the scope of this document.

4. P2MP Transport Using Chain Replication

In this document, a P2MP stream associated with a root node and a set of leaf nodes is denoted as {root node, leaf nodes}. It is achieved by using a bundle of P2MP chains covering all the leaf nodes. Each P2MP chain is a single-path tunnel starting from the root node and reaching one or multiple leaf nodes along the path. The tail-end node of the P2MP chain is a leaf node, called a "tail-end" leaf node. Each leaf node traversed by the P2MP chain is called a "transit" leaf node. As a special case, a P2MP chain may have only a tail-end leaf node and no transit leaf node, essentially becoming a P2P tunnel, but it is not the focus of this document.

```
    R ------ R1 ------ R2 ------ L1 ------ R3 ------ L2 ------ L3



                       R  : root node
                       Li : leaf node
                       Ri : transit router
```

Figure 1

A tail-end leaf node and a transit leaf nodes have different behaviors when processing an incoming packet. In particular, a tail-end leaf node processes the packet as a normal receiver. A transit leaf node not only processes the packet as a receiver, but also forwards it downstream along the P2MP chain, hence acting as a "bud node". To achieve this, the transit leaf node needs to replicate the packet, producing two packets, one for forwarding and the other for local processing. Such packet replication happens on every transit

leaf node along a P2MP chain. Therefore, it is called "chain replication".

This document introduces a new type of segments, called "bud segments", to model the above packet replication and processing on transit leaf nodes. The segment ID (SID) of a bud segment is a "bud-SID".

4.1. P2MP Chain

Construction of P2MP chains for a P2MP stream is performed by a controller or the root node based on configuration or path computation (Section 5). This generates the set of P2MP chains to use, and decides the set of leaf nodes that each P2MP chain reaches. In general, if not all leaf nodes can be covered by using a single P2MP chain, multiple P2MP chains MUST be used, and the root node MUST replicate ingress packets over the P2MP chains.

The path of a P2MP chain is a single path traversing one or multiple transit leaf nodes and terminating at a tail-end leaf node. Between the root node and the first transit leaf node, and between two consecutive leaf nodes, there may be none, one, or multiple transit routers.

The path is then translated to a SID list to be programmed on the root node. In the SID list, each transit leaf node has its bud-SID in a corresponding position. Given a P2MP chain to a set of leaf nodes in the order of L1, L2, ..., Ln, the SID list may be represented as below:

<SID_11, SID_12, ...>, L1's bud-SID, ..., <SID_i1, SID_i2, ...>, Li's bud-SID, ..., <SID_n1, SID_n2, ...>

Where:

o <SID_11, SID_12, ...> is the sub-path from the root node to L1.

o <SID_i1, SID_i2, ...> is the sub-path from Li-1 to Li.

o <SID_n1, SID_n2, ...> is the sub-path from Ln-1 to Ln. There is no need for Ln's bud-SID to be at the end of the SID list, because the tail-end leaf node does not perform a chain replication.

Each of the above sub-paths is a regular point-to-point path, and its SIDs are regular SIDs, such as adjacency-SIDs, node-SIDs, binding-SIDs, etc. As a special case, a sub-path from Li-1 to Li may have an empty SID list, if the sub-path takes the shortest path represented by Li's bud-SID.

4.2. Bud Segment

On a transit leaf node, a bud segment represents the forwarding instructions below. They are applied to an incoming packet P when the packet's active SID is the bud-SID of this bud segment.

[1] Replicate the packet P to generate a copy P1.

[2] For P, perform a NEXT operation on the bud-SID, make the next SID active, and forward the packet based on that SID.

[3] For P1, perform a sequence of NEXT operations on the bud-SID and all the subsequent SIDs of the P2MP chain, and process the packet locally as an endpoint.

Bud segments are global segments of leaf nodes. They are routable segments via the shortest topological paths. Bud-SIDs are allocated from SRGB (SR global block). Only one bud segment is needed per leaf node, and per SR-MPLS or SRv6 dataplane. It is used only when the leaf node is a transit leaf node on a P2MP chain.

Bud segments are shared by all P2MP streams, i.e. all instances of {root node, leaf nodes}. A leaf node SHOULD advertise a bud segment for SR-MPLS if its forwarding hardware supports the SR-MPLS behavior (Section 4.3.1), and a bud segment for SRv6 if its forwarding hardware supports the above SRv6 behavior (Section 4.3.2). The advertisement may be via a protocol, e.g. ISIS, OSPF, or BGP. The advertisement allows the leaf node to be considered as a transit leaf node on a P2MP chain. If a leaf node does not advertise a bud segment, it can only be considered as a tail-end leaf node on a P2MP chain, or reached via a P2P tunnel using ingress replication. The extensions of the protocols are out of the scope of this document.

4.3. Forwarding Behaviors

4.3.1. SR-MPLS

In SR-MPLS, bud-SIDs are labels. A root node applies a stack of labels corresponding to a P2MP chain's SID list to ingress packets. These labels are called P2MP chain labels. A packet may have an inner service label(s), e.g. a VPN label, a bridge domain label, a source Ethernet segment label, etc. In this case, the root node must have a way to mark the end of P2MP chain labels in the MPLS header, in order for transit leaf nodes to process the packet as receivers. This document introduces an "end-of-chain" (EoC) label to facilitate this. The EoC label is an extended special-purpose label (ESPL) [RFC 7274] with value TDB. If an ingress packet has a service label(s), the root node MUST push the service label(s) first, then the

Extension Label (XL, value 15) and the EoC label, and finally the P2MP chain labels. Hence, the [XL, EoC] labels serve as a unique pattern to indicate the end of the P2MP chain labels. If a packet does not have a service label(s), the root node MUST NOT push the [XL, EoC] labels, but only the P2MP chain labels.

A transit leaf node will receive the packet (P) with the node's bud-SID label as top label. The node replicates P to generate a copy P1. For P, the node pops the bud-SID label and forwards the packet based on the next label in the MPLS header. For P1, the node removes all the P2MP chain labels, by popping labels until [XL, EoC] are popped or all labels are popped. The node then processes P1 locally as an SR-MPLS endpoint.

The tail-end leaf node will receives the packet with (1) no label; (2) the [XL, EoC] labels as top labels; or (3) the node's node-SID label as top label, followed by the [XL, EoC] labels. In case (3), the [XL, EoC] labels will be exposed to the top after the node-SID label is popped. In both cases (2) and (3), the node pops [XL, EoC] and continues to process the inner service label(s). This imposes a requirement on the node to be able to handle the [XL, EoC] labels as described. Ultimately, the node processes the packet with no label or an exposed service label(s), as an SR-MPLS endpoint.

4.3.2. SRv6

In SRv6, bud-SIDs are IPv6 addresses specifically assigned to bud segments. A root node constructs a packet with an IPv6 header corresponding to the P2MP chain, followed by a segment routing header (SRH) containing the SIDs of the P2MP chain, and followed by an inner IP header or service header.

A transit leaf node will receive the packet (P) with the node's bud-SID as active SID. The node replicates P to generate a copy P1. For P, the node performs NEXT operation on the bud-SID by adjusting the IPv6 header and SRH, and forwards the packet based on the new active SID. For P1, the node removes the IPv6 header and SRH, and processes the packet (with the inner header exposed) as an SRv6 endpoint.

The tail-end leaf node will receives the packet with the IPv6 header and optionally the SRH. The node processes the packet as an SRv6 endpoint.

4.4. Example

In the following example, P2MP transport is needed from the root node R, to leaf nodes L1, L2, L3 and L4.

```
                R ------ R1 -------------------- R2 ------- L1
                          |                       |      /
                          |                       |    /
                          |                       |  /
                         R3 -------------------- R4 ------- L2
                          |                       |
                          |                       |
                          |                       |
                         R5 -------------------- R6 ------- L3
                          |                       |      /
                          |                       |    /
                          |                       |  /
                         R7 -------------------- R8 ------- L4
```

Figure 2

Path computation results in two P2MP chains:

P2MP chain 1:

Path: R -> R1 -> R2 -> L1 -> R4 -> L2, where L1 is a transit leaf node, and L2 is the tail-end leaf node.

Assuming that the sub-path R -> R1 -> R2 -> L1 is not the shortest path from R to L1, so that an explicit sub-path must be used. Also assuming that the sub-path L1 -> R4 -> L2 is the shortest path from L1 to L2, so that the node-SID of L2 can be used to represent this sub-path. The segment list applied to packets on R is:

adj-SID 100 - link from R to R1

adj-SID 200 - link from R1 to R2

adj-SID 300 - link from R2 to L1

bud-SID 1000 - L1

node-SID 2000 - L2

P2MP chain 2:

Path: R -> R1 -> R3 -> R5 -> R6 -> L3 -> R8 -> L4, where L3 is a transit leaf node, and L4 is the tail-end leaf node.

Assuming that the sub-path R -> R1 -> R3 -> R5 -> R6 -> L3 is the shortest path from R to L3, so that the bud-SID of L3 can

be used to represent this sub-path. Also assuming that the sub-path L3 -> R8 -> L4 is not the shortest path from L3 to L4, so that an explicit sub-path must be used. The segment list applied to packets on R is:

bud-SID 3000 - L3

adj-SID 600 - link from L3 to R8

adj-SID 700 - link from R8 to L4

node-SID 4000 - L4

5. Path Computation for P2MP Chains

P2MP chain path computation for a P2MP stream {root node, leaf nodes} may be performed by a controller or the root node. Each P2MP chain is a single-path tunnel. In general, any P2P path computation algorithm may be extended to serve the purpose. This document does not enforce a particular algorithm.

The path computation may consider topological metrics for shortest paths, or traffic engineering (TE) constraints for TE paths. In addition, this document also suggests the following constraints:

- Maximum hops per P2MP chain. This SHOULD be based on the maximum delay allowed for packets to accumulate before reaching a tail-end leaf node.

- Maximum length of SID list. This SHOULD be based on the maximum header size which may be applied to packets by a root node. This is typically a limit of forwarding hardware. Note that a SID list is translated from a computed path. Hence, the SID list's length and the path's hop count are not necessarily the same.

- Maximum leaf nodes per P2MP chain. This may be used to restrict the length of each P2MP chain.

- Maximum hops between two consecutive leaf nodes. This may be useful to avoid a sparse chain, where an excessive distance between two consecutive leaf nodes will cause a P2MP chain's efficiency to degrade.

- Maximum number of times that a node or link may be traversed by a P2MP chain. This may be useful to prevent a node or link from being congested by duplicate traffic.

The path computation tends to be deterministic in a ring or linear topology. In an arbitrary topology, the path computation can be made controllable by dividing leaf nodes into groups (or clusters) based on geographic locations or policies, and computing a separate path for each group. A leaf group may be defined as a sequence (i.e. ordered) or set (i.e. unordered) of leaf nodes, which are treated as loose hops in path computation.

6. Special Purpose Bud Segments

So far, the discussion has been focusing on nodal bud segments, which are per node and per SR-MPLS/SRv6. The local processing represented by these bud segments is completely based on a packet's inner header, i.e. after all the SIDs of a P2MP chain are removed by the instruction [3] in Section 4.2. These bud segments are applicable to common P2MP transport, and hence are considered as the default and general purpose bud segments.

The concept of bud segment is also applicable to other types of local processing on a transit leaf node, where the context of the local processing cannot be derived from a packet's inner header. For example, the node may want to forward packets over a particular interface or tunnel, or based on a particular forwarding table or policy. In such cases, a dedicated bud segment may be introduced to serve each distinct scenario and indicate a specific context. These bud segments are called special purpose bud segments.

The scale of special purpose bud segments per leaf node SHOULD be a consideration in network design, as well as the mechanisms for distributing these bud segments to controllers or all the root nodes.

7. IANA Considerations

This document requires IANA to allocate a value from the "Extended Special-Purpose MPLS Label Values" registry for the EoC label.

The document also requires IANA registration and allocation for the ISIS, OSPF and BGP extensions for bud segment advertisement. The details will be provided in the next version of this document.

8. Security Considerations

This document introduces bud segments for leaf nodes to act as both packet receivers and transit routers. A security attack may target on a leaf node by constructing malicious packets with the node's bud-SID. Such kind of attacks can be defeated by restricting bud segment distribution and P2MP chain construction within the scope of a controller and a given network.

9. Acknowledgements

This document leverages work done by Alexander Arseniev, Ron Bonica, and G Sri Karthik Goud.

10. Contributors

Alexander Arseniev

Juniper networks

Email: aarseniev@juniper.net

Ron Bonica

Juniper networks

Virginia

USA

Email: rbonica@juniper.net

11. References

11.1. Normative References

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

[RFC8660] Bashandy, A., Ed., Filsfils, C., Ed., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with the MPLS Data Plane", RFC 8660, DOI 10.17487/RFC8660, December 2019, <https://www.rfc-editor.org/info/rfc8660>.

[RFC7274] Kompella, K., Andersson, L., and A. Farrel, "Allocating and Retiring Special-Purpose MPLS Labels", RFC 7274, DOI 10.17487/RFC7274, June 2014, <https://www.rfc-editor.org/info/rfc7274>.

[SRv6-SRH]
Filsfils, C., Dukes, D., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header", draft-ietf-6man-segment-routing-header (work in progress), 2019.

[SRv6-Programming]
Filsfils, C., Garvia, P., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-ietf-spring-srv6-network-programming (work in progress), 2019.

11.2. Informative 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>.

Authors' Addresses

Yimin Shen
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
USA

Email: yshen@juniper.net

Zhaohui Zhang
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
USA

Email: zzhang@juniper.net

Rishabh Parekh
Cisco Systems
San Jose, CA
USA

Email: riparekh@cisco.com

Hooman Bidgoli
Nokia
Ottawa
Canada

Email: hooman.bidgoli@nokia.com

Yuji Kamite
NTT Communications
Tokyo
Japan

Email: y.kamite@ntt.com