Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: October 3, 2020

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

April 1, 2020

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

draft-shen-spring-p2mp-transport-chain-02

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.

Status of This Memo

Copyright Notice

Table of Contents

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 ingress replication, where a point-to-point (P2P) SR tunnel is constructed from a root node to each leaf node, and every ingress packet is replicated and sent 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, as the path of each tunnel is computed or decided independently.

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

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 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 replicates a packet once for local processing off the chain, and also forwards the original packet down the chain. The root node replicates and sends packets via the set of P2MP chains to all the leaf nodes.

As a P2MP chain can reach multiple leaf nodes, it is considered to be more efficient than the multiple P2P tunnels which would be needed in ingress replication to reach these leaf nodes. Compared with ingress replication and the P2MP-tree based approach, this solution provides a middle ground by achieving a certain level of traffic optimization, while aligning with the fundamental principles of SR, including single-point provisioning and stateless core. The solution can be used to improve P2MP transport efficiency in general, and to achieve maximum traffic optimization in certain types of topologies.

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 benefits 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, etc.

The mechanism is transparent to all transit routers. Leaf nodes intended to take advantage of the mechanism will need to support the new forwarding behavior specified in this document. For other leaf nodes, the mechanism has a backward compatibility to allow them to be reached by P2P tunnels using ingress replication. Path computation and P2MP chain construction will need to be supported by a controller or root nodes, depending on where they are performed.

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

The mechanism does not create any state of P2MP tunnel or P2MP tree on routers. Therefore, if leaf nodes need to know the service level context (e.g. source, VPN) of a P2MP stream, they must rely on the information contained in an inner header. 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.

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 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 no transit leaf node, but only a tail-end leaf node, essentially becoming a P2P tunnel of ingress replication.

```
    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 a received 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 facilitate the above packet processing on transit leaf nodes. The segment ID (SID) of a bud segment is a "bud-SID".

4.1. Bud Segment

On a transit leaf node, a bud segment represents the following instructions for forwarding hardware to execute on a received packet P. They apply when the active SID of the packet P 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. (The SIDs of the P2MP chain are not useful for processing P1 locally. Hence, they are removed before the processing.)

Bud segments are global segments of leaf nodes. They are routable segments via topological shortest-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. It is used only when the leaf node is a transit leaf node on a P2MP chain.

In SR-MPLS, bud-SIDs are labels, and penultimate hop popping (PHP) MUST be disabled for bud-SID labels. In SRv6, bud-SIDs are IPv6 addresses explicitly associated with bud segments. Therefore, the above instructions [1] to [3] are achieved in different ways in SR-MPLS and SRv6:

(a) In SR-MPLS, the packet may have a service label(s) after P2MP chain labels in MPLS header, e.g. a VPN label, a bridge domain label, a source Ethernet segment label, etc. Therefore, the bud segment MUST have a way to identify the position of the last P2MP chain label, in order to execute [3] above. This document introduces an "end-of-chain" (EoC) label to facilitate the process. The EoC label is an extended special-purpose label (ESPL) [RFC 7274] with value TDB. When a root node constructs an MPLS header for a packet, if the packet has a service label(s), the root node MUST push the Extension Label (XL, value 15) and the EoC label, after pushing the service label(s) and before push P2MP chain labels. Hence, [XL, EoC] serves as a recognizable pattern to indicate the end of the P2MP chain labels. If the packet does not have a service label(s), the root node SHOULD NOT push [XL, EoC] to the MPLS header. In any case, in [3] above, the bud segment MUST pop labels until [XL, EoC] are popped or all labels have been popped.

(b) In SRv6, the packet is encapsulated with an outer IPv6 header corresponding to the P2MP chain, optionally followed by a segment routing header (SRH) containing the SIDs of the P2MP chain, and followed by an inner header (of IPv4, IPv6, MPLS, layer-2, etc.) associated with a service. In [3] above, the bud segment SHOULD simply remove the outer IPv6 header and the SRH (if any), and leave the packet with the inner header to local processing.

Bud segments are shared by all P2MP streams, i.e. all combinations of {root node, leaf nodes}. A leaf node SHOULD advertise a bud segment for SR-MPLS, if its forwarding hardware supports the above SR-MPLS processing. Likewise, it SHOULD advertise a bud segment for SRv6, if its forwarding hardware supports the above SRv6 processing. The advertisement may be via IGP (ISIS, OSPF) or BGP-LS. 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.

Bud segments are generic purpose segments. They may also be used in cases other than P2MP transport, such as traffic monitoring. These use cases are out of the scope of this document.

4.2. P2MP Chain

Construction of P2MP chains for a P2MP stream is performed by a controller or the root node based on path computation (Section 5). This decides the number of P2MP chains to use, and the set of leaf nodes that each P2MP chain reaches. In general, if the leaf nodes of the P2MP stream cannot 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:

<SID_11, SID_12, ...>, bud-SID of L1, ..., <SID_i1, SID_i2, ...>, bud-SID of Li, ..., <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. The SIDs in the sub-path are regular SIDs, such as adjacency-SIDs, node-SIDs, binding-SIDs, etc. There is no SID specific to the given P2MP chain. A sub-path from Li-1 to Li may have an empty SID list, if the sub-path takes the shortest path indicated by the bud-SID of Li.

The root node then uses the SID list in packet encapsulation. Note that in the SR-MPLS case where the EoC label is needed, [XL, EoC] MUST be pushed to an MPLS header, before the SID list is pushed.

4.3. 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

bud-SID 4000 - L4

5. Path Computation for P2MP Chains

Path computation for the P2MP chains of a P2MP stream {root node, leaf nodes} lies in the responsibility of a controller or the root node. This document does not enforce a particular computation algorithm. In general, any P2P path computation algorithm may be extended to serve the purpose.

The path computation may consider general metric for shortest paths, or traffic engineering (TE) constraints for TE paths. This document recommends the following constraints to be considered as well:

- The maximum hop count of path. This SHOULD be based on the maximum delay allowed for a packet to accumulate before reaching a tail-end leaf node. It may be used to restrict the length of each P2MP chain.

- The maximum length of SID list. This SHOULD be based on the maximum header size which a root node may apply to a packet. This is typically a limit of forwarding hardware. Note that a SID list is translated from a computed path. Hence, the length of the SID list and the hop count of the path are generally not the same.

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

- Maximum hops between consecutive leaf nodes on a P2MP chain. This may be used prevent a P2MP chain from attempting leaf nodes which should ideally be reached by separate P2MP chains.

- Maximum times that a node or link may be traversed by a P2MP chain. This may be used to prevent a P2MP chain from congesting a node or link.

As an example, the path computation may start with forming a path from the root node to the closest leaf node, and extend the path to a second leaf node, a third leaf node, and so on. When any of the above limits is hit, the current computation SHOULD end, the path

SHOULD be saved as a completed P2MP chain, and a new computation SHOULD be performed for the rest leaf nodes. This process SHOULD repeat until all the leaf nodes are covered, where a set of paths have been computed.

The path computation is generally deterministic in a ring or linear topology. In an arbitrary topology, deterministic path computation may be achieved by dividing leaf nodes into groups based on their location, and computing a separate path for each group. A group may even define its leaf nodes as an ordered list of loose hops, so that a path will traverse the leaf nodes in the specified order. During the computation of a group, if any of the above limits is hit, the computation SHOULD end, the path SHOULD be saved as a completed P2MP chain, and a new computation SHOULD be performed for the rest leaf nodes of the group. This process SHOULD repeat until all the leaf nodes of the group are covered. In this case, the group will end up using multiple P2MP chains.

6. IGP and BGP-LS Extensions for Bud Segment

The protocol extensions of IGP (ISIS and OSPF) and BGP-LS for bud segment advertisement will be specified in the next version of this document.

7. Bud Segments for Special Processing

So far, the discussion in this document has been focusing on bud segments that are created on a per SR-MPLS or SRv6 basis on each leaf node. These bud segments indicate generic local processing which is based on the inner header of a packet. They are applicable to most of the common cases of P2MP transport, and hence are viewed as the default bud segments of leaf nodes.

The concept of bud segment can also be extended to other cases, where a transit leaf node needs to perform a special kind of local processing for packets, but cannot derive the context of the processing from their inner headers. For example, the node may need to forward the packets over one or more interfaces or tunnels to downstream device(s), or to process the packets based on a particular forwarding table or policy, and so on. In such cases, a dedicated bud segment SHOULD be created for the special kind of local processing. It will serve the general purpose of a bud segment, and additionally indicate the context of the special processing. Note that scaling of such bud segments per leaf node SHOULD be a consideration in network design, as well as the requirement for a controller or ingress router to have the knowledge of various special processing scenarios on leaf nodes and use the corresponding bud segments in P2MP chain construction.

8. 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-LS extensions for bud segment advertisement. The details will be provided in the next version of this document.

9. 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.

10. Acknowledgements

This document leverages work done by Alexander Arseniev and Ron Bonica.

11. Contributors

Alexander Arseniev

Juniper networks

Email: aarseniev@juniper.net

Ron Bonica

Juniper networks

Virginia

USA

Email: rbonica@juniper.net

12. References

12.1. 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.

12.2. 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