MPLS
Internet-Draft
Intended status: Standards Track
Expires: October 31, 2021

H. Song, Ed.
Futurewei Technologies
Z. Li
T. Zhou
Huawei
L. Andersson
Bronze Dragon Consulting

April 29, 2021

MPLS Extension Header
draft-song-mpls-extension-header-04

Abstract

Motivated by the need to support multiple in-network services and functions in an MPLS network, this document describes a generic and extensible method to encapsulate extension headers into MPLS packets. The encapsulation method allows stacking multiple extension headers and quickly accessing any of them as well as the original upper layer protocol header and payload. We show how the extension header can be used to support several new network applications and optimize some existing network services.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

Status of This Memo

Copyright Notice

Table of Contents

1. Motivation

Some applications require adding instructions and/or metadata to user packets within a network. Such examples include In-situ OAM (IOAM) [I-D.ietf-ippm-ioam-data] and Service Function Chaining (SFC) [RFC7665]. New applications are emerging. It is possible that the instructions and/or metadata for multiple applications are stacked together in one packet to support a compound service.

Conceivably, such instructions and/or metadata would be encoded as new headers and encapsulated in user packets. Such headers may require to be processed in fast path or in slow path. Moreover, such headers may require being attended at each hop on the forwarding path (i.e., hop-by-hop or HBH) or at designated end nodes (i.e., end-to-end or E2E).

The encapsulation of the new header(s) poses some challenges to MPLS networks, because the MPLS protocol header contains no explicit indicator for the upper layer protocols by design. We leave the discussion on the indicator of new header(s) in an MPLS packet to another companion document [I-D.song-mpls-eh-indicator]. In this document, we focus on the encode and encapsulation of new headers in an MPLS packet.

The similar problem has been tackled for some particular application before. However, these solutions have some drawbacks:

o The solutions rely on either the built-in next-protocol indicator in the header or the knowledge of the format and size of the header to access the following packet data. The node is required to be able to parse the new header, which is unrealistic in an incremental deployment environment.

o These works only provide piecemeal solution which assumes the new header is the only extra header and its location in the packet is fixed by default. It is impossible or difficult to support multiple new headers in one packet due to the conflicted assumption.

o Some previous work such as G-ACH [RFC5586] was explicitly defined for control channel only but what we need is the user packet service.

To solve these problems, we introduce extension header as a general and extensible means to support new in-network functions and applications in MPLS networks. The idea is similar to IPv6 extension headers which offer a huge innovation potential (e.g, network security, SRv6 [RFC8754], network programming [I-D.ietf-spring-srv6-network-programming], SFC [I-D.xu-clad-spring-sr-service-chaining], etc.). Thanks to the existence of extension headers, it is straightforward to introduce new in-network services into IPv6 networks. For example, it has been proposed to carry IOAM header [I-D.brockners-inband-oam-transport] as a new extension header option in IPv6 networks.

Nevertheless, IPv6 EH is not perfect either. It has three issues:

o IPv6's header is large compared to MPLS, claiming extra bandwidth overhead and complicating the packet processing. We prefer to retain the header compactness in MPLS networks.

o IPv6's extension headers are chained with the original upper layer protocol headers in a flat stack. One must scan all the extension headers to access the upper layer protocol headers and the

payload. This is inconvenient and raises some performance concerns for some applications(e.g., DPI and ECMP). The new scheme for MPLS header extension needs to address these issues too.

- [RFC8200] enforces many constraints to IPv6 extension headers (e.g., EH can only be added or deleted by the end nodes specified by the IP addresses in the IPv6 header, and there is only one Hop-by-Hop EH that can be processed on the path nodes), which are not suitable for MPLS networks. For example, MPLS label stacks are added and changed in network, and there could be tunnel within tunnel, so the extension headers need more flexibility.

2. MPLS Extension Header

From the previous discussion, we have laid out the design requirements to support extension headers in MPLS networks:

Performance: Unnecessary label stack scanning for a label and the full extension header stack scanning for the upper layer protocol should be avoided. The extension headers a node needs to process should be located as close to the MPLS label stack as possible. Each extension header is better to serve only one application to avoid the need of packing multiple TLV options in one extension header.

Scalability: New applications can be supported by introducing new extension headers. Multiple extension headers can be easily stacked together to support multiple services simultaneously.

Backward Compatibility: Legacy devices which do not recognize the extension header option should still be able to forward the packets as usual. If a device recognize some of the extension headers but not the others in an extension header stack, it can process the known headers only while ignoring the others.

Flexibility: A node can be configured to process or not process any EH. Any tunnel end nodes in the MPLS domain can add new EH to the packets which shall be removed on the other end of the tunnel.

We assume the MPLS label stack has included some indicator of the extension header(s). The actual extension headers are inserted between the MPLS label stack and the original upper layer packet header. The format of the MPLS packets with extension headers is shown in Figure 1.

```
      0                                  31
      +--------+--------+--------+--------+  \
      |                                   |  |
      ~     MPLS Label Stack              ~  |
      |                                   |  |
      +--------+--------+--------+--------+  |
      |     EH Indicator (TBD)            |   > MPLS Label Stack
      +--------+--------+--------+--------+  |  (extended with EHI)
      |                                   |  |
      ~     MPLS Label Stack              ~  |
      |                                   |  |
      +--------+--------+--------+--------+ <
      | Header of Extension Headers (HEH) |  |
      +--------+--------+--------+--------+  |
      |                                   |  |
      ~     Extension Header (EH) 1       ~  |
      |                                   |  |
      +--------+--------+--------+--------+   > MPLS EH Fields
      ~                                   ~  |  (new)
      +--------+--------+--------+--------+  |
      |                                   |  |
      ~     Extension Header (EH) N       ~  |
      |                                   |  |
      +--------+--------+--------+--------+ <
      |                                   |  |
      ~    Upper Layer Headers/Payload    ~   > MPLS Payload
      |                                   |  |  (as is)
      +--------+--------+--------+--------+  /
```

Figure 1: MPLS with Extension Headers

Following the MPLS label stack is the 4-octet Header of Extension Headers (HEH), which indicates the total number of extension headers in this packet, the overall length of the extension headers, and the type of the next header. The format of the HEH is shown in Figure 2.

```
       0          1          2          3
       0123 45678901 234567890123 45678901
      +----+--------+------------+--------+
      | R  | EHCNT  |   EHTLEN   |   NH   |
      +----+--------+------------+--------+
```

Figure 2: HEH Format

The meaning of the fields in an HEH is as follows:

R: 4-bit reserved.

EHCNT: 8-bit unsigned integer for the Extension Header Counter. This field keeps the total number of extension headers included in this packet. It does not count the original upper layer protocol headers.

EHTLEN: 12-bit unsigned integer for the Extension Header Total Length in 4-octet units. This field keeps the total length of the extension headers in this packet, not including the HEH itself.

NH: 8-bit selector for the Next Header. This field identifies the type of the header immediately following the HEH.

The value of the reserved nibble needs further consideration. The EHCNT field can be used to keep track of the number of extension headers when some headers are inserted or removed at some network nodes. The EHLEN field can help to skip all the extension headers in one step if the original upper layer protocol headers or payload need to be accessed.

The format of an Extension Header (EH) is shown in Figure 3.

```
     0          1          2          3
     01234567 89012345 6789012345678901
    +--------+--------+----------------+
    |  NH    |  HLEN  |                |
    +--------+--------+                +
    |                                  |
    ~        Header Specific Data      ~
    |                                  |
    +--------+--------+----------------+
```

Figure 3: EH Format

The meaning of the fields in an EH is as follows:

NH: 8-bit indicator for the Next Header. This field identifies the type of the EH immediately following this EH.

HLEN: 8-bit unsigned integer for the Extension Header Length in 4-octet units, not including the first 4 octets.

Header Specific Data: Variable length field for the specification of the EH. This field is 4-octet aligned.

The extension headers as well as the first original upper layer protocol header are chained together through the NH field in HEH and EHs. The encoding of NH can share the same value registry for IPv4/IPv6 protocol numbers. Values for new EH types shall be assigned by IANA.

Specifically, the NH field of the last EH in a chain can have two special values, which shall be assigned by IANA as well:

NONE (No Next Header): Indicates that there is no other header and payload after this header. This can be used to transport packets with only extension header(s), for example, the control packets for control or the probe packets for measurements. Note that value 59 was reserved for "IPv6 No Next Header" indicator. It may be possible for MPLS EH to share this value.

UNKNOWN (Unknown Next Header): Indicates that the type of the header after this header is unknown. This is intended to be compatible with the original MPLS design in which the upper layer protocol type is unknown from the MPLS header alone.

3. Type of MPLS Extension Headers

Basically, there are two types of MPLS EHs: HBH and E2E. E2E means that the EH is only supposed to be inserted/removed and processed at the MPLS tunnel end points where the MPLS header is inserted or removed. The EHs that need to be processed on path nodes within the MPLS tunnel are of the HBH type. However, any node in the tunnel can be configured to ignore an HBH EH, even if it is capable of processing it.

If there are two types of EHs in a packet, the HBH EHs must take precedence over the E2E EHs.

Making a distinction of the EH types and ordering the EHs in a packet help improve the forwardidng performance. For example, if a node within an MPLS tunnel finds only E2E EHs in a packet, it can avoid scanning the EH list.

4. Operation on MPLS Extension Headers

When the first EH X needs to be added to an MPLS packet, an EH indicator is inserted into the proper location in the MPLS label stack. A HEH is then inserted after the MPLS label stack, in which EHCNT is set to 1, EHTLEN is set to the length of X in 4-octet units, and NH is set to the header value of X. At last, X is inserted after the HEH, in which NH and HELN are set accordingly. Note that if this operation happens at a PE device, the upper layer protocol is known

before the MPLS encapsulation, so its value can be saved in the NH field if desired. Otherwise, the NH field is filled with the value of "UNKNOWN".

When an EH Y needs to be added to an MPLS packet which already contains extension header(s), the EHCNT and EHTLEN in the HEH are updated accordingly (i.e., EHCNT is incremented by 1 and EHTLEN is incremented by the size of Y in 4-octet units). Then a proper location for Y in the EH chain is located. Y is inserted at this location. The NH field of Y is copied from the previous EH's NH field (or from the HEH's NH field, if Y is the first EH in the chain). The previous EH's NH value, or, if Y is the first EH in the chain, the HEH's NH, is set to the header value of Y.

Deleting an EH simply reverses the above operation. If the deleted EH is the last one, the EH indicator and HEH can also be removed.

When processing an MPLS packet with extension headers, the node needs to scan through the entire EH chain and process the EH one by one. The node should ignore any unrecognized EH or the EH that is configured as "No Processing".

The EH can be categorized into HBH or E2E. Since EHs are ordered based on their type(i.e., HBH EHs are located before E2E EHs), a node can avoid some unnecessary EH scan.

5. Use Cases

In this section, we show how MPLS extension header can be used to support several new network applications.

In-situ OAM: In-situ OAM (IOAM) records flow OAM information within user packets while the packets traverse a network. The instruction and collected data are kept in an IOAM header [I-D.ietf-ippm-ioam-data]. When applying IOAM in an MPLS network, the IOAM header can be encapsulated as an MPLS extension header.

Network Telemetry and Measurement: A network telemetry and instruction header can be carried as an extension header to instruct a node what type of network measurements should be done. For example, the method described in [RFC8321] can be implemented in MPLS networks since the EH provides a natural way to color MPLS packets.

Network Security: Security related functions often require user packets to carry some metadata. In a DoS limiting network architecture, a "packet passport" header is used to embed packet authentication information for each node to verify.

Segment Routing and Network Programming: MPLS extension header can support the implementation of a new flavor of the MPLS-based segment routing, with better performance and richer functionalities. The details will be described in another draft.

With MPLS extension headers, multiple in-network applications can be stacked together. For example, IOAM and SFC can be applied at the same time to support network OAM and service function chaining. A node can stop scanning the extension header stack if all the known headers it can process have been located. For example, if IOAM is the first EH in a stack and a node is configured to process IOAM only, it will stop searching the EH stack when the IOAM EH is found.

6. Security Considerations

TBD

7. IANA Considerations

This document requests IANA to assign two new Internet Protocol Numbers from the "Protocol Numbers" Registry to indicate "No Next Header" and "Unknown Next Header".

This document does not create any other new registries.

8. Contributors

The other contributors of this document are listed as follows.

- James Guichard
- Stewart Bryant
- Andrew Malis

9. Acknowledgments

TBD.

10. References

10.1. 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>.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <https://www.rfc-editor.org/info/rfc7665>.

[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>.

[RFC8321] Fioccola, G., Ed., Capello, A., Cociglio, M., Castaldelli, L., Chen, M., Zheng, L., Mirsky, G., and T. Mizrahi, "Alternate-Marking Method for Passive and Hybrid Performance Monitoring", RFC 8321, DOI 10.17487/RFC8321, January 2018, <https://www.rfc-editor.org/info/rfc8321>.

[RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <https://www.rfc-editor.org/info/rfc8754>.

10.2. Informative References

[I-D.brockners-inband-oam-transport]
Brockners, F., Bhandari, S., Govindan, V., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Mozes, D., Lapukhov, P., and R. Chang, "Encapsulations for In-situ OAM Data", draft-brockners-inband-oam-transport-05 (work in progress), July 2017.

[I-D.ietf-ippm-ioam-data]
Brockners, F., Bhandari, S., and T. Mizrahi, "Data Fields for In-situ OAM", draft-ietf-ippm-ioam-data-12 (work in progress), February 2021.

[I-D.ietf-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-ietf-spring-srv6-network-programming-28 (work in progress), December 2020.

[I-D.song-mpls-eh-indicator]
Song, H., Li, Z., Zhou, T., and L. Andersson, "Options for MPLS Extension Header Indicator", draft-song-mpls-eh-indicator-01 (work in progress), March 2021.

[I-D.xu-clad-spring-sr-service-chaining]
Clad, F., Xu, X., Filsfils, C., daniel.bernier@bell.ca, d., Decraene, B., Yadlapalli, C., Henderickx, W., Salsano, S., and S. Ma, "Segment Routing for Service Chaining", draft-xu-clad-spring-sr-service-chaining-00 (work in progress), December 2017.

[RFC5586] Bocci, M., Ed., Vigoureux, M., Ed., and S. Bryant, Ed., "MPLS Generic Associated Channel", RFC 5586, DOI 10.17487/RFC5586, June 2009, <https://www.rfc-editor.org/info/rfc5586>.

[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>.

Authors' Addresses

Haoyu Song (editor)
Futurewei Technologies
2330 Central Expressway
Santa Clara
USA

Email: haoyu.song@futurewei.com

Zhenbin Li
Huawei
156 Beiqing Road
Beijing, 100095
P.R. China

Email: lizhenbin@huawei.com

Tianran Zhou
Huawei
156 Beiqing Road
Beijing, 100095
P.R. China

Email: zhoutianran@huawei.com

Loa Andersson
Bronze Dragon Consulting

Stockholm
Sweden

Email: loa@pi.nu