Workgroup: MPLS
Internet-Draft:
draft-song-mpls-extension-header-07
Published: 8 July 2022
Intended Status: Standards Track
Expires: 9 January 2023
Authors: H. Song, Ed. Z. Li T. Zhou
Futurewei Technologies Huawei Huawei
L. Andersson Z. Zhang
Bronze Dragon Consulting Juniper Networks
R. Gandhi J. Rajamanickam J. Bhattacharya
Cisco Systems Cisco Systems Cisco Systems

MPLS Post-Stack Extension Header

Abstract

Motivated by the need to support multiple in-network services and functions in an MPLS network (a.k.a. MPLS Network Actions (MNA)), this document describes a generic and extensible method to encapsulate extension headers into MPLS packets. The encapsulation method allows stacking multiple post-stack extension headers and quickly accessing any of them as well as the original upper layer protocol header and payload. We show how the post-stack extension header can be used to support several new MNAs and provide a generic alternative for some existing ones.

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

This Internet-Draft will expire on 9 January 2023.

Table of Contents

1. Motivation

Some applications require adding sizable instructions and/or metadata to user packets within a network. Such examples include In-situ OAM (IOAM) [RFC9197] 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 due to performance considerations. 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 need and requirements to support such applications in MPLS networks, i.e., MPLS Network Actions (MNA), are described in [I-D.ietf-mpls-mna-usecases] and [I-D.ietf-mpls-mna-requirements]. It is clear that some headers should be located after the MPLS label stack. We call such a header "post-stack extension header". The encapsulation of post-stack extension header(s) poses some challenges to MPLS networks, because the MPLS label stack contains no explicit indicator for the upper layer protocols by design. The indicator for the presence of post-stack extension header is defined in [I-D.jags-mpls-mna-hdr] as POST-Stack Network Action Presence Indicator (PNI) in the MNA Label (Special Purpose Label Value TBA by IANA). We summarize the other possible indicator options in [I-D.song-mpls-eh-indicator]. In this document, we focus on the encode and encapsulation of new post-stack extension headers in an MPLS packet as Post-Stack Data (PSD) [I-D.andersson-mpls-mna-fwk].

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

* 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.

* These works provide only piecemeal solutions which assume 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.

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

To solve these problems, we introduce post-stack extension header as a general and extensible means to support new MNAs which involve instructions and/or meta data. The concept is similar to IPv6 extension headers which offer a huge potential for extending IPv6's capability (e.g, network security, SRv6 [RFC8754], network programming [RFC8986], SFC [I-D.ietf-spring-sr-service-programming], etc.). Thanks to the existence of extension headers, it is straightforward to introduce new network services into IPv6 networks. For example, it has been proposed to carry IOAM header [I-D.ietf-ippm-ioam-ipv6-options] as a new extension header option in IPv6 networks.

Nevertheless, when applying the extension header to MPLS, some issues of the IPv6 EH should be avoided:

* 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 improve this.

* [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

The concept and design of the MPLS post-stack extension header comply with the requirements laid out in [I-D.ietf-mpls-mna-requirements]. We highlight some specific design requirements for the post-stack 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 recognizes 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.

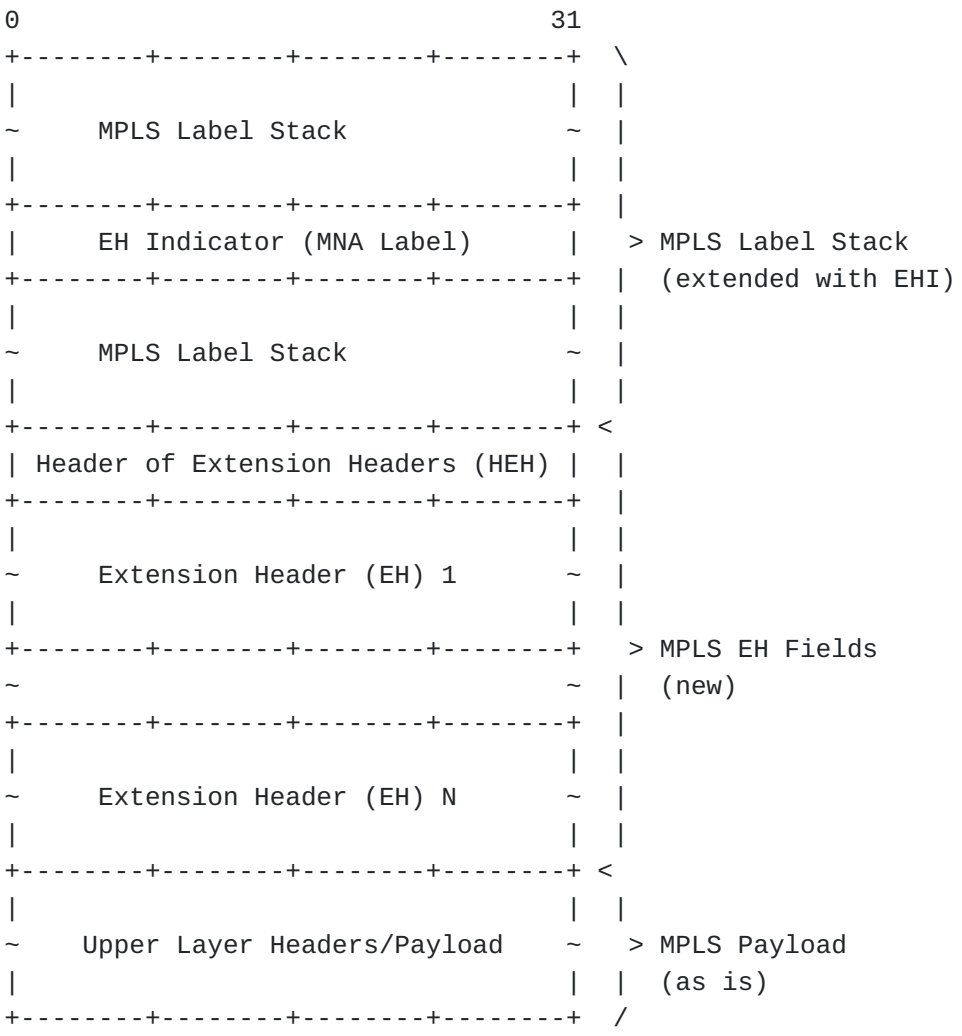

Figure 1: MPLS with Extension Headers

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, the type of the original upper layer header, and the type of the next header. The format of the HEH is shown in Figure 2.

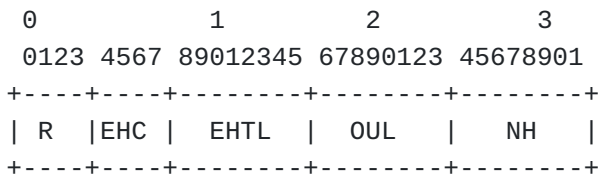

Figure 2: HEH Format

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

R: 4-bit reserved. The nibble value means to avoid conflicting with IP version numbers and other well-defined semantics [I-D.kbbma-mpls-1stnibble].

EHC: 4-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. At most 15 EHs are allowed in one packet.

EHTL: 8-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.

OUL: 8-bit Original Upper Layer protocol number indicating the original upper layer protocol type. It can be set to "UNKNOWN" if unknown.

NH: 8-bit indicator 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 EHC 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 EHTL 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 OUL field can help identify the type of the original upper layer protocol.

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

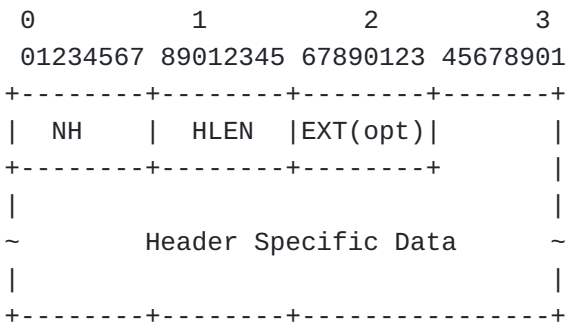

Figure 3: EH Format

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.

EXT: 8-bit optional type extension. To save the Next Header numbers and extend the number space, it is possible to use one "Next Header" code to cover a set of sub-types. Each sub-type is assigned a new code in a different name space. This field is optional and it is only specified for some specific EH type.

Header Specific Data: Variable length field for the specification of the EH. This field is in length such that the EH 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 (i.e., NH number) shall be assigned by IANA from the same registry as for the ipv4 and ipv6 protocol numbers (https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml).

Specifically, the NH field of the last EH in a chain can have some 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.

MPLS: Indicates that the original upper layer protocol is still MPLS and another MPLS label stack follows.

Note that the original upper layer protocol can be of type "MPLS", which implies that in a packet there might be multiple label stacks separated by EHs. Having more than one independent label stack is not new. For example, A Bier header could separate the transport/ bier labels and the payload labels; An MPLS Pseudo Wire (PW) network could be implemented on the top of another infrastructure MPLS network. In such cases, we have the flexibility to apply different services to different label stacks.

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.

The type of an EH (i.e., HBH or E2E) is an intrinsic property of the EH. In other words, EH type indicates if it needs to be processed on each hop or only on edge node.

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 [RFC9197]. 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.

Details on some of these use cases and discussions on some other use cases are covered in [I-D.ietf-mpls-mna-usecases].

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. New registries for protocol numbers and type extension numbers should be requested by each MNA document.

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>.

[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>.

10.2. Informative References

[I-D.andersson-mpls-mna-fwk] Andersson, L., Bryant, S., Bocci, M., and T. Li, "MPLS Network Actions Framework", Work in Progress, Internet-Draft, draft-andersson-mpls-mna-fwk-04, 27 June 2022, <https://www.ietf.org/archive/id/draft-andersson-mpls-mna-fwk-04.txt>.

[I-D.ietf-ippm-ioam-ipv6-options] Bhandari, S. and F. Brockners, "In-situ OAM IPv6 Options", Work in Progress, Internet-Draft, draft-ietf-ippm-ioam-ipv6-options-08, 16 June 2022, <https://www.ietf.org/archive/id/draft-ietf-ippm-ioam-ipv6-options-08.txt>.

[I-D.ietf-mpls-mna-requirements] Bocci, M., Bryant, S., and J. Drake, "Requirements for MPLS Network Action Indicators and MPLS Ancillary Data", Work in Progress, Internet-Draft, draft-ietf-mpls-mna-requirements-01, 21 June 2022, <https://www.ietf.org/archive/id/draft-ietf-mpls-mna-requirements-01.txt>.

[I-D.ietf-mpls-mna-usecases] Saad, T., Makhijani, K., Song, H., and G. Mirsky, "Use Cases for MPLS Network Action Indicators and MPLS Ancillary Data", Work in Progress, Internet-Draft, draft-ietf-mpls-mna-usecases-00, 19 May 2022, <https://www.ietf.org/archive/id/draft-ietf-mpls-mna-usecases-00.txt>.

[I-D.ietf-spring-sr-service-programming]
Clad, F., Xu, X., Filsfils, C., Bernier, D., Li, C., Decraene, B., Ma, S., Yadlapalli, C., Henderickx, W., and S. Salsano, "Service Programming with Segment Routing", Work in Progress, Internet-Draft, draft-ietf-spring-sr-service-programming-06, 9 June 2022, <https://www.ietf.org/archive/id/draft-ietf-spring-sr-service-programming-06.txt>.

[I-D.jags-mpls-mna-hdr]
Rajamanickam, J., Gandhi, R., Bhattacharya, J., Decraene, B., Zigler, R., Cheng, W., Jalil, L., and H. Song, "MPLS Network Action (MNA) Header Encodings", Work in Progress, Internet-Draft, draft-jags-mpls-mna-hdr-00, 6 July 2022, <https://www.ietf.org/archive/id/draft-jags-mpls-mna-hdr-00.txt>.

[I-D.kbbma-mpls-1stnibble]
Kompella, K., Bryant, S., Bocci, M., Mirsky, G., and L. O. (. Andersson, "IANA Registry for the First Nibble Following a Label Stack", Work in Progress, Internet-Draft, draft-kbbma-mpls-1stnibble-01, 27 April 2022, <https://www.ietf.org/archive/id/draft-kbbma-mpls-1stnibble-01.txt>.

[I-D.song-mpls-eh-indicator] Song, H., Li, Z., Zhou, T., and L. Andersson, "Options for MPLS Extension Header Indicator", Work in Progress, Internet-Draft, draft-song-mpls-eh-indicator-05, 27 June 2022, <https://www.ietf.org/archive/id/draft-song-mpls-eh-indicator-05.txt>.

[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>.

[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>.

[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>.

[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>.

[RFC8986] Filsfils, C., Ed., Camarillo, P., Ed., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming", RFC 8986, DOI 10.17487/RFC8986, February 2021, <https://www.rfc-editor.org/info/rfc8986>.

[RFC9197] Brockners, F., Ed., Bhandari, S., Ed., and T. Mizrahi, Ed., "Data Fields for In Situ Operations, Administration,

and Maintenance (IOAM)", RFC 9197, DOI 10.17487/RFC9197,
May 2022, <https://www.rfc-editor.org/info/rfc9197>.

Authors' Addresses

Haoyu Song (editor)
Futurewei Technologies
Santa Clara,
United States of America

Email: haoyu.song@futurewei.com

Zhenbin Li
Huawei
Beijing
P.R. China

Email: lizhenbin@huawei.com

Tianran Zhou
Huawei
Beijing
P.R. China

Email: zhoutianran@huawei.com

Loa Andersson
Bronze Dragon Consulting
Stockholm
Sweden

Email: loa@pi.nu

Zhaohui Zhang
Juniper Networks
Boston,
United States of America

Email: zzhang@juniper.net

Rakesh Gandhi
Cisco Systems
Canada

Email: rgandhi@cisco.com

Jaganbabu Rajamanickam
Cisco Systems
Canada

Email: jrajaman@cisco.com

Jisu Bhattacharya
Cisco Systems

Email: jisu@cisco.com