                        MPLS Extension Header
                  draft-song-mpls-extension-header-01

Abstract

   Motivated by the need to support multiple in-network services and
   functions in an MPLS network, this document describes a 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.

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

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.brockners-inband-oam-requirements] 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.

   However, the encapsulation of the new header(s) poses some challenges
   to the ubiquitous MPLS networks. The MPLS protocol header contains
   no explicit indicator for the upper layer protocols by design. The
   compact MPLS header, while beneficial to forwarding, allows little
   room for any extra information. Moreover, the backward compatibility
   issue discourages any attempts trying to overload the semantics of
   the existing MPLS header fields. While it is possible to designate
   "service labels" with special semantics by an operator, the non-

standard approach burdens the control plane and impairs the
interoperability.

The similar "header extension" requirement for MPLS has led to
several proposals. A special Generic Associated Channel Label (GAL)
[RFC5586] is assigned to support the identification of an Associated
Channel Header (ACH). Later, it was proposed to use GAL to indicate
the presence of a Metadata Channel Header (MCH)
[I-D.guichard-sfc-mpls-metadata] as well.

GAL has several limitations:

o It must be located at the bottom of a label stack for its chief
   use case of MPLS-TP. An LSR needs to search the entire label
   stack for the BoS bit and check if the corresponding label is GAL.
   This can impact the performance when the label stack is deep.

o When GAL is present, the first nibble of the word following the
   GAL needs to be checked to determine the header type. Since the
   value of the nibble cannot be greater than 3 to avoid any possible
   confliction with IP version numbers, this approach is not
   scalable.

o By design, GAL can only indicate the presence of a single header.
   Therefore, the solution alone is not sufficient to support adding
   multiple headers at the same time.

o The presence of GAL makes the network load balancing or deep
   packet inspection based on upper layer protocol headers and
   payload difficult.

In addition to the above limitations, it is not desirable to keep
overloading GAL with new semantics. Instead of trying to patch on
existing schemes, we propose a new mechanism to solve the above
mentioned issues and create new innovation opportunities, similar to
the way that IPv6 supports extension headers, which offer a huge
innovation potential (e.g, network security, SRv6
[I-D.ietf-spring-segment-routing], network programming
[I-D.filsfils-spring-srv6-network-programming], SFC
[I-D.xu-clad-spring-sr-service-chaining], etc.). Thanks to the
existing 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]
and NSH as new extension headers in IPv6 networks.

Nevertheless, IPv6 is not perfect either. It has two main issues.
First, 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. Second, 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.

2. MPLS Extension Header

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

Performance: If possible, unnecessary label stack scanning for a
   label and extension header stack scanning for the upper layer
   protocol should be avoided.

Scalability: New applications can be easily 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.

Extension headers can be be supported in several ways in an MPLS
network, we have outlined some of them in this document. However, it
should be noted that we do not intend to close this discussion yet,
and are prepared to listen to arguments why and how any other methods
could be used. One interesting line of thought is whether it would
be possible to let a label that is received on top of the stack
indicate whether there are extension headers beneath the stack.

Our investigations indicate that a special purpose label and/or an
extended special purpose label will be the optimal way to go. A new
special purpose label, namely the Extension Header Label (EHL), can
be used to indicate EHs. So far 8 special purpose label values are
left unsigned by IANA (which are 4 to 6 and 8 to 12). Alternatively,
a two label scheme with the use of the extension label (XL) plus an
EHL is possible, but it does use one more label. It is also possible
to use FEC labels to indicate the presence of extension headers.
Although this approach avoid the need of a new special purpose label,
it introduces a good deal of complexity into the control plane. In
the remaining of the document, we assume a special EHL is assigned
and use it as an example to illustrate the scheme. A formal

recommendation on the Extension Header (EH) indicator approach will
be given in a future revision of this document.

The format of the MPLS packets with extension headers is shown in
Figure 1. An EHL can be located in anywhere in an MPLS label stack.
However, if there are legacy devices which do not recognize the EHL
in the network, then for backward compatibility, the EHL must be
located at the bottom of the stack (i.e., only the MPLS tunnel ends
and EHL-aware nodes will look up and process it). Otherwise, the EHL
can be located close to the top of the stack for better lookup
performance.

The format of an EHL is the same as an MPLS label. The first 20-bit
label value will be assigned by IANA. The BoS bit is used to
indicate the location of the label. The other fields, CoS and TTL,
are unused in the context of EHL.

```
      0                               31
      +--------+--------+--------+--------+
      |                                  |
      ~        MPLS Label Stack          ~
      |                                  |
      +--------+--------+--------+--------+
      |      EHL                         |
      +--------+--------+--------+--------+
      |                                  |
      ~        MPLS Label Stack          ~
      |                                  |
      +--------+--------+--------+--------+
      | Header of Extension Headers (HEH) |
      +--------+--------+--------+--------+
      |                                  |
      ~       Extension Header (EH) 1     ~
      |                                  |
      +--------+--------+--------+--------+
      ~                                  ~
      +--------+--------+--------+--------+
      |                                  |
      ~       Extension Header (EH) N     ~
      |                                  |
      +--------+--------+--------+--------+
      |                                  |
      ~     Upper Layer Protocols/Payload  ~
      |                                  |
      +--------+--------+--------+--------+
```
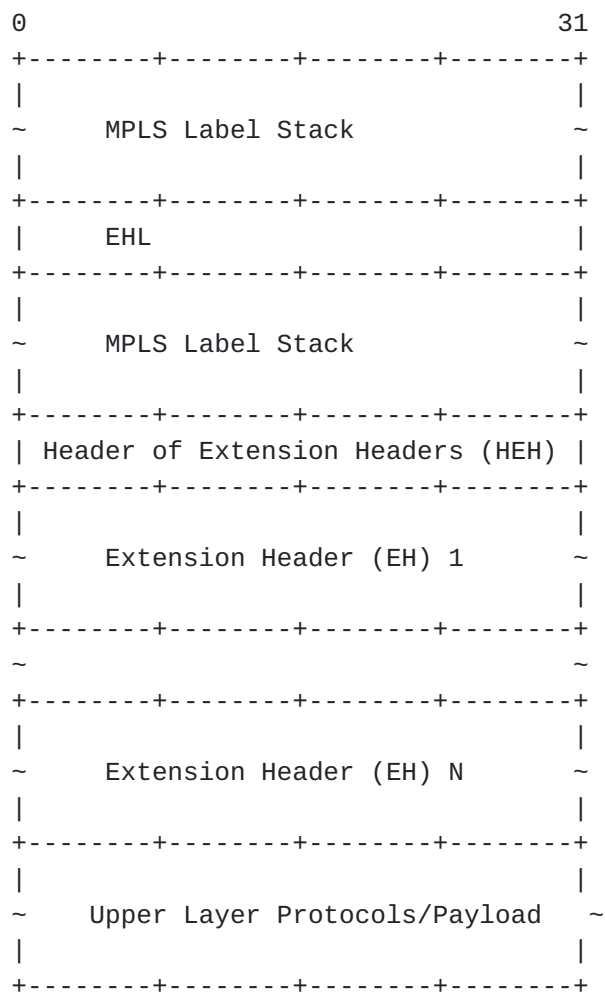
Figure 1: MPLS with Extension Header

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
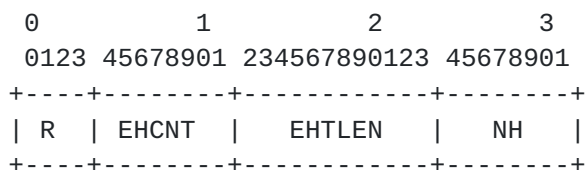   type of the header immediately following the HEH.

The 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

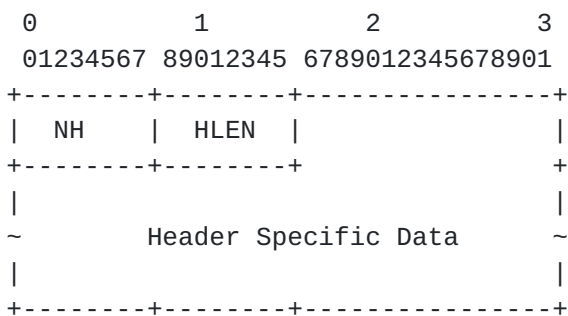The meaning of the fields in an EH is as follows:

NH: 8-bit selector 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 uses the same values as the IPv4 protocol
field. 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:

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).

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](). Operation on MPLS Extension Headers

When the first EH X needs to be added to an MPLS packet, an EHL 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 EHL and HEH can also be deleted.

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.

4. 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.

   NSH: Network Service Header (NSH) [RFC8300] provides a service plane
      for Service Function Chaining (SFC). NSH maintains the SFC
      context and metadata. If MPLS is used as the transport protocol
      for NSH, NSH 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: 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.

5. Summary

   Evidenced by the existing and emerging use cases, MPLS networks need
   a standard way to support extension headers. In Figure 4, we
   summarize the potential schemes that allow MPLS packets to carry
   extension headers and list the main issues for each scheme.

```
+---+-------------------------+-------------------------------+
|No.|    Description          |           Issues              |
+---+-------------------------+-------------------------------+
| 1 | GAL + MCH with multi-   |- Label location limitation lead |
|   | header extension        |  to performance concern       |
|   |                         |- Interfere with load balancing |
|   |                         |  and DPI functions            |
|   |                         |- Overload GAL semantics       |
|   |                         |- Need standard extension      |
+---+-------------------------+-------------------------------+
| 2 | GAL + another nibble value|- Same as above              |
|   | to encode the EHs (e.g., |                              |
|   | "0010")                 |                               |
+---+-------------------------+-------------------------------+
| 3 | FEC label to indicate EH |- Complex control plane       |
|   |                         |- Interoperability             |
+---+-------------------------+-------------------------------+
| 4 | XL(15) + EHL            |- One extra label              |
|   |                         |- Need standard extension      |
+---+-------------------------+-------------------------------+
| 5 | Special purpose EHL     |- Need standard extension      |
+---+-------------------------+-------------------------------+
```

Figure 4: Potential Schemes for MPLS Extension Headers

Through comprehensive considerations on the pros and cons of each scheme, we currently recommend the scheme No.5. The proposed MPLS extension header scheme provides a generic way to support in-network services and functions in MPLS networks.

6. Security Considerations

TBD

7. IANA Considerations

If the EHL approach is adopted to indicate the presence of MPLS extension header(s), this document requests IANA to assign a new Special-Purpose MPLS Label Value from the Special-Purpose Multiprotocol Label Switching (MPLS) Label Values Registry of "Extension Header Label (EHL)".

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

This document does not create any new registries.

8. Contributors

The other contributors of this document are listed as follows.

o James Guichard

o Stewart Bryant

o 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>.

[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>.

[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>.

[RFC8300]   Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed.,
            "Network Service Header (NSH)", RFC 8300,
            DOI 10.17487/RFC8300, January 2018,
            <https://www.rfc-editor.org/info/rfc8300>.

[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>.

10.2. Informative References

   [I-D.brockners-inband-oam-requirements]
              Brockners, F., Bhandari, S., Dara, S., Pignataro, C.,
              Gredler, H., Leddy, J., Youell, S., Mozes, D., Mizrahi,
              T., <>, P., and r. remy@barefootnetworks.com,
              "Requirements for In-situ OAM", draft-brockners-inband-
              oam-requirements-03 (work in progress), March 2017.

   [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.filsfils-spring-srv6-network-programming]
              Filsfils, C., Camarillo, P., Leddy, J.,
              daniel.voyer@bell.ca, d., Matsushima, S., and Z. Li, "SRv6
              Network Programming", draft-filsfils-spring-srv6-network-
              programming-05 (work in progress), July 2018.

   [I-D.guichard-sfc-mpls-metadata]
              Guichard, J., Pignataro, C., Spraggs, S., and S. Bryant,
              "Carrying Metadata in MPLS Networks", draft-guichard-sfc-
              mpls-metadata-00 (work in progress), September 2013.

   [I-D.ietf-ippm-ioam-data]
              Brockners, F., Bhandari, S., Pignataro, C., Gredler, H.,
              Leddy, J., Youell, S., Mizrahi, T., Mozes, D., Lapukhov,
              P., Chang, R., daniel.bernier@bell.ca, d., and J. Lemon,
              "Data Fields for In-situ OAM", draft-ietf-ippm-ioam-
              data-03 (work in progress), June 2018.

   [I-D.ietf-spring-segment-routing]
              Filsfils, C., Previdi, S., Ginsberg, L., Decraene, B.,
              Litkowski, S., and R. Shakir, "Segment Routing
              Architecture", draft-ietf-spring-segment-routing-15 (work
              in progress), January 2018.

   [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.

Authors' Addresses

   Haoyu Song (editor)
   Huawei
   2330 Central Expressway
   Santa Clara
   USA

   Email: haoyu.song@huawei.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