Abstract

This document specifies an architectural framework for the application of the MPLS Indicator and Ancillary Data (MIAD) technologies. MIAD technologies are used to indicate actions (I) for LSPs and/or packets and to transfer data needed for these actions (AD).

The document describes a common set of protocol functions and information elements - the MPLS Indicator and Ancillary Data - supporting additional operational models and capabilities of MPLS networks that support these functions. Some of these functions are defined in existing MPLS specifications, while others require extensions to existing specifications to meet the requirements in the MIAD requirement specification.

This document is the result of work started in MPLS Open Design Team, with participation by the MPLS, PALS and DETNET working groups.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 4, 2022.
Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction .................................................. 3
  1.1. Requirement Language .................................... 3
2. Normative Definitions .......................................... 3
  2.1. Ancillary Data (AD) ....................................... 3
  2.2. Ancillary Data Indicator (ADI) ............................ 3
3. Terminology .................................................. 4
  3.1. Abbreviations .............................................. 4
   3.1.1. Process Note on E2E .................................. 6
  3.2. Concepts used in this Framework .......................... 6
4. Methods to carry Ancillary Data Indicators .................... 6
  4.1. existing base SPL ......................................... 6
  4.2. new base SPL ............................................... 6
  4.3. new extend SPL ............................................. 7
5. Method chosen by the working group .......................... 7
6. Types of AD .................................................. 7
  6.1. In Stack Data (ISD) ....................................... 7
  6.2. Post Stack Data (PSD) .................................... 7
  6.3. Implicit Data .............................................. 7
7. ADI details .................................................. 7
  7.1. Hop by Hop (HBH) .......................................... 7
  7.2. End to End (E2E) .......................................... 7
  7.3. Initiate action at a specific single node ............... 7
8. Flag Field .................................................. 7
  8.1. Unused SPL bits ........................................... 7
    8.1.1. RFC 3032 LSE definition ............................... 8
    8.1.2. Carrying the flag field starting in the first LSE .. 8
    8.1.3. Carrying the flag field starting in the second LSE . 8
    8.1.4. Using two bSPLs ...................................... 9
  8.2. Process Note flags and flag field ....................... 9
9. ADI specification rules ...................................... 9
10. Ancillary Data ............................................... 9
1. Introduction

This document discusses how flag fields ancillary data and IANA registries for the data could be designed.

Maybe expand the abstract a bit and give some of the development we have seen in the MPLS forwarding model, e.g. original model, first nibble, Pseudowire ACH, GACH and MIAD.

1.1. Requirement 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 BCP14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document is intended to become an Informational RFC, it is not clear that we will need Section 1.1. We will leave it in for the time being and take the decision to remove or not closer to the Publication Request.

2. Normative Definitions

Text to be added, prime candidates to be discussed are AD and ADI, not least explaining the differences.

2.1. Ancillary Data (AD)

2.2. Ancillary Data Indicator (ADI)
3. Terminology

3.1. Abbreviations
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Meaning</th>
<th>Reference</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD</td>
<td>Ancillary Data</td>
<td>draft-bocci-mpls-miad-adi-requirements</td>
<td>place holder</td>
</tr>
<tr>
<td>ADI</td>
<td>Ancillary Data Indicator</td>
<td>draft-bocci-mpls-miad-adi-requirements</td>
<td></td>
</tr>
<tr>
<td>ECMP</td>
<td>Equal Cost Multipath</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E2E</td>
<td>End to end</td>
<td>In the MIAD context this document.</td>
<td></td>
</tr>
<tr>
<td>HBH</td>
<td>Hop by hop</td>
<td>In the MIAD context this document.</td>
<td></td>
</tr>
<tr>
<td>ISD</td>
<td>In stack data</td>
<td>draft-bocci-mpls-miad-adi-requirements</td>
<td></td>
</tr>
<tr>
<td>LSE</td>
<td>Label Stack Entry</td>
<td>RFC 3032 [RFC3032]</td>
<td></td>
</tr>
<tr>
<td>MIAD</td>
<td>MPLS Indicators and Ancillary Data</td>
<td>MPLS Open DT wiki and this document</td>
<td>MIAD is the name of both this technology and the project developing this part of the MPLS architecture</td>
</tr>
<tr>
<td>PSD</td>
<td>Post stack data</td>
<td>draft-bocci-mpls-miad-adi-requirements</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Abbreviations
3.1.1. Process Note on E2E

There has been some discussion on the of the E"E abbreviation. 1. In a mail to the MPLS Working group mailing list Joel Halpern pointed out that the abbreviation E2E has been used in several different meanings. Joel suggested to use another abbreviation.

1. Some variants has been proposed, for example.
   * Ingress to Egress (I2E); alternative abbreviation (I2e)
   * Egress
   * LSP Ingress to LSP Egress (LI2LE)

In a few days (counting from the publication date of this document) the working group chairs will take an initiative to poll the working groups for consensus on this.

3.2. Concepts used in this Framework

<table>
<thead>
<tr>
<th>Concept</th>
<th>Meaning</th>
<th>Reference</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>E2E</td>
<td>E2E in MIAD context is defined</td>
<td>this</td>
<td>-</td>
</tr>
<tr>
<td>concept</td>
<td>in...</td>
<td>document</td>
<td></td>
</tr>
<tr>
<td>concept</td>
<td>free text</td>
<td>this</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>document</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Concepts

Not complete, help appreciated.

4. Methods to carry Ancillary Data Indicators

Several possibilities to carry ADI’s has been discussed in MIAD drafts and in the MPLS Open DT.

4.1. existing base SPL

4.2. new base SPL
4.3. new extend SPL

5. Method chosen by the working group
   o confused, but we need to write something

6. Types of AD

   6.1. In Stack Data (ISD)

   a bit of explanatory text

   6.2. Post Stack Data (PSD)

   a bit of explanatory text

   6.3. Implicit Data

Note: We are changing the earlier "No Data" (NoD) to implicit without
creating an abbreviation [I-D.bocci-mpls-miad-adi-requirements]. ID
would be to close o ISD.

7. ADI details

   7.1. Hop by Hop (HBH)

   7.2. End to End (E2E)

   7.3. Initiate action at a specific single node

   We are looking to see if this is needed.

8. Flag Field

The MIAD flag field is carried in a Base Special Purpose Lavel
(bSPL). Different style of bSPLs can as discussed in (#carry) issues
around which bSPL to use for the flag field are discussed in this
section. This section discussed how the flag field itself is set up.

8.1. Unused SPL bits

In a SPL only the 20 bits of Label Value and the Bottom of Stack bit
are significant, the TC field (3 bits) and the TTL (8 bits) are not
used. This leaves 11 bits that could be used for the MIAD flag
fields (carrying ADIs) in the first LSE..

The flag field may also be carried in an LSE (second LSE) immediately
following the LSE that carries the Label Value
8.1.1. RFC 3032 LSE definition

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| Label: Label Value, 20 bits |
| TC: Traffic Class, 3 bits |
| S: Bottom of Stack, 1 bit |
| TTL: Time to Live, 8 bits |

8.1.2. Carrying the flag field starting in the first LSE

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| Label: Label Value, 20 bits |
| f: MIAD flag (ADI) |
| S: Bottom of Stack, 1 bit |
| x: Extension bit |

8.1.3. Carrying the flag field starting in the second LSE

An alternative would be to not carrying flags in the "spare" 11 bits of the first LSE, but to start the flag field in the second LSE.

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| Label: Label Value, 20 bits |
| TC: Traffic Class, 3 bits |
| S: Bottom of Stack, 1 bit |
| TTL: Time To Live |
| f: MIAD flag (ADI) |
| x: Extension bit |
8.1.4. Using two bSPLs

The possibility to use two bSPLs to carry flag fields has been suggested. For example action indicators (ADIs) using Implicit and ISD could use one bSPL and the action indicators using PSD could use the other. The mapping of the flag field into the bSPLs would be the same as above.

8.1.4.1. Interpreting flags and bits

- Label - As in RFC 3032.
- TC - Traffic Class, as in the updated RFC 3032.
- S - Bottom of Stack bit, as in RFC 3032.
- TTL - Time To Live, as in RFC 3032.
- f-bits (f) - Flags (ADIs) see Section 7. It is possible to have flags of more than one bit.
- x-bit (x) - Extension bit, if the x bit is 0 (zero) the next field is an LSE carrying MIAD flags, if it is 1 (one) there are no more flag field LSEs for that bSPL.

8.2. Process Note flags and flag field

1. It seem obvious that the working group would want to produce one single consolidated solution for how to carry the flags and flag field.

2. The decision on which method to use for carrying flags and flag fields will be taken by a consensus call in the MPLS, PALS and DETNET working groups, and be documented here.

9. ADI specification rules

Guidance what to include in an IANA section

10. Ancillary Data

  Structure, encoding, allocation of identifiers. Matters of sequence / priority. Instances of a single type of AD.
11. Packet Structure

12. Security Considerations

13. Management Considerations

14. MPLS Forwarding model

This is section here to basically to have a place holder where to discuss the development of the MPLS forwarding model. It might be removed.

14.1. Original Model

---

![Diagram of MPLS Original Forwarding Model]

LSE = Label Stack Entry (what many people call a label)
FIB = Forwarding Information (date)Base

Figure 1: MPLS Original Forwarding Model

15. IANA Considerations

This document does not make any allocations of code points from IANA registries.

As long as the "does not make any allocations ..." from IANA is true, this paragraph should be removed by the RFC-Editor. If it turns out

that we will need to do IANA allocation, a proper IANA section will be added.

16. The First Nibble considerations

The first nibble after the label stack has been used to convey information in certain cases.

For example, in [RFC4928] this nibble is investigated to find out if it has the value "4" or "6", if it is not, it is assumed that the packet payload is not IPv4 or IPv6 and Equal Cost Multipath (ECMP) is not performed.

It should be noted that this is an inexact method, for example an Ethernet Pseudowire without a control word might have "4" or "6" in the first nibble and thus will be ECMP’ed.

Nevertheless, the method is implemented and deployed, it is used to day and will be for the foreseeable future.

The use of the first nibble for BIER is specified in [RFC8296]. Bier sets the first nibble to 5. The same is true for BIER payload, as for any use of the first nibble, it is not possible from the first nibble itself being set to 5, conclude that the payload is BIER. However, it achieves the design goal of [RFC8296, to exclude that the payload is IPv4, IPv6 or a pseudowire.

There is possible more examples, they will be added if we find that they further highlights the issue with using the first nibble.

17. Acknowledgements

18. References

18.1. Normative References

[I-D.bocci-mpls-miad-adi-requirements]
Bocci, M. and S. Bryant, "Requirements for MPLS Label Stack Indicators and Ancillary Data", draft-bocci-mpls-miad-adi-requirements-02 (work in progress), March 2022.

[I-D.decrane-mpls-slid-encoded-entropy-label-id]
18.2. Informative References

[I-D.kbbma-mpls-1stnibble]


Authors’ Addresses

Loa Andersson
Bronze Dragon Consulting
Email: loa@pl.nu

Stewart Bryant
University of Surrey 5GIC
Email: sb@stewartbryant.com
Matthew Bocci
Nokia

Email: matthew.bocci@nokia.com
Requirements for MPLS Network Action Indicators and MPLS Ancillary Data
draft-bocci-mpls-miad-adi-requirements-04

Abstract

This draft specifies requirements for indicators in the MPLS label stack to support ancillary data in the packet and high level requirements on that ancillary data. This work is the product of the IETF MPLS Open Design Team. Requirements are derived from a number of new proposals for additions to the MPLS label stack to allow forwarding or other processing decisions to be made, either by a transit or terminating LSR (i.e. the LER), based on application data that may be in or below the bottom of the label stack.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 13, 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect
1. Introduction

There is significant interest in developing the MPLS data plane to address the requirements of new applications [I-D.saad-mpls-miad-usecases]. These applications typically require the inclusion of ancillary data in the MPLS packet. This data may be encoded either in the label stack or below the bottom of the label stack. This data is then either intercepted and processed, or some other forwarding decision is taken by routers processing the packet. The ancillary data is added by the ingress LSR, and is then processed using mechanisms implemented by intermediate and/or egress LSRS that comply with the MPLS base architecture and potentially its extensions, including (but not limited to) [RFC3031], [RFC3032], [RFC6790].

This draft specifies requirements for indicators in the MPLS label stack to support these applications, as well as the encoding and use of the ancillary data.

1.1. Terminology

- Ancillary Data (AD): Data relating to the MPLS packet that may be used to affect the forwarding or other processing of that packet, either at a Label Edge Router (LER) [RFC4221] or Label Switching Router (LSR). This data may be encoded within a network action

Table of Contents

1. Introduction .................................................. 2
  1.1. Terminology ............................................. 2
  1.2. Background .............................................. 3
2. Requirements Language ......................................... 5
3. MPLS Network Action Indicator Requirements ................. 5
  3.1. General Requirements ................................... 5
  3.2. Requirements on Network Action Indicators ............... 6
  3.3. Requirements on Ancillary Data ......................... 7
4. IANA Considerations .......................................... 8
5. Security Considerations ..................................... 8
6. Acknowledgements ........................................... 8
7. References .................................................. 9
  7.1. Normative References .................................... 9
  7.2. Informative References .................................. 9
Authors’ Addresses .............................................. 10
sub-stack (see below) (in-stack data), and/or after the bottom of the label stack (post-stack data).

- **Network Action**: An operation to be performed on a packet. A network action may affect router state, packet forwarding, or it may affect the packet in some other way. A network action is said to be present if there is an indicator in the packet that invokes the action.

- **Network Action Indication (NAI)**: An indication in the packet that a certain network action is to be performed. There may be associated ancillary data in the packet.

- **Network Action Sub-Stack (NAS)**: A set of related, contiguous Label Stack Entries (LSEs). The first LSE contains the NAI. The TC and TTL values in the sub-stack may be redefined. The label field in the second and following LSE may be redefined. Solutions MUST NOT redefine the S bit. See Section 3.1 through Section 3.5.

- **In-Stack Data**: Any data within the MPLS label stack including the outer LSE and the bottom of stack (the LSE with the S-bit set).

- **Post-Stack Data**: Any data beyond the LSE with the S-Bit set, but before the first octet of the user payload. This document does not prescribe whether post-stack data precedes or follows any other protocol structure such as a control word or associated channel header (ACH).

- **Scope**: The set of nodes that should perform a given action.

### 1.2. Background

The **MPLS architecture** is specified in [RFC3031] and provides a mechanism for forwarding packets through a network without requiring any analysis of the packet payload’s network layer header by intermediate nodes (Label Switching Routers - LSRs). Formally, inspection may only occur at network ingress (the Label edge router - LER) where the packet is assigned to a forwarding equivalence class (FEC).

MPLS uses switching based on a label pushed on the packet to achieve efficient forwarding and traffic engineering of flows associated with the FEC. While originally used for IP traffic, MPLS has been extended to support point-to-point, point-to-multipoint and multipoint-to-multipoint layer 2 and layer 3 services. An overview
of the development of MPLS is provided in [I-D.bryant-mpls-dev-primer].

A number of applications have emerged which require LSRs to make forwarding or other processing decisions based on inspection of the network layer header, or some other ancillary information in the protocol stack encapsulated deeper in the packet. An early example of this was generation of a hash of the payload header to be used for load balancing over Equal Cost Multipath (ECMP) or Link Aggregation Group (LAG) next hops. This is based on an assumption that the network layer protocol is IP. MPLS was extended to avoid the need for LSRs to perform this operation if load balancing was needed based on the payload and instead use only the MPLS label stack, using the Entropy Label / Entropy Label Indicator [RFC6790] which are inserted at the LER. Other applications where the intermediate LSRs may need to inspect and process a packet on an LSP include OAM, which can make use of mechanisms such the Router Alert Label [RFC3032] or the Generic Associated Channel Label (GAL) [RFC5586] to indicate that an intercepted packet should be processed locally. See [I-D.bryant-mpls-dev-primer] for detailed list of such applications.

There have been a number of new proposals for how ancillary data is carried in MPLS and how its presence is indicated to the LSR or egress LER, for example In-situ OAM and Service Function Chaining (SFC). A summary of these proposals is contained in [I-D.bryant-mpls-dev-primer], an overview of use cases is provided in [I-D.saad-mpls-miad-usecases]. [I-D.song-mpls-extension-header] summarises some of the issues with existing solutions to address these new applications (note that this document draws on the requirements and issues without endorsing a specific solution from [I-D.song-mpls-extension-header]):

These 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.

A piecemeal solution often 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. An example of this is that the GAL/G-ACH mechanism assumes that if the GAL is present, only a single G-ACH header follows.

New applications therefore require the definition of extensions to the MPLS architecture and label stack operations that can be used
across these applications in order to minimise implementation complexity and promote interoperability and extensibility.

2. 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.

3. MPLS Network Action Indicator Requirements

This document specifies requirements of MPLS Network Action Indicators, and the associated Ancillary Data. The requirements are for the behavior of the protocol mechanisms and procedures that constitute building blocks out of which indicators for network actions and associated ancillary data are constructed. It does not specify the detailed actions and processing that may be required by an application for any ancillary data by an LSR or LER. The requirements in this document do not describe what functions an implementation must support. The purpose of this document is to identify the toolkit and any new protocol work that is required. This new protocol work MUST be based on the existing MPLS architecture.

3.1. General Requirements

1. MPLS combines extensibility, flexibility and efficiency by using control plane context combined with a simple data plane mechanism to allow the network to make forwarding decisions about a packet. Any solution MUST maintain these properties of MPLS.

2. Any solutions to these requirements MUST NOT restrict the generality of MPLS architecture [RFC3031], [RFC3032].

3. If extensions to the MPLS data plane are required, they MUST NOT be inconsistent with the MPLS architecture [RFC3031], [RFC3032].

4. Solutions meeting the requirements set out in this document MUST be able to coexist with and MUST NOT obsolete existing MPLS mechanisms.

5. The design of any mechanism SHOULD be such that an LSR is able to efficiently parse the label stack.

6. Mechanisms MUST NOT increase the size of the MPLS label stack more than is necessary.
7. The design of solutions MUST NOT expose confidential information [RFC6973] [RFC3552] to the LSRs.

8. Solution specifications MUST document any changes to the existing MPLS data plane security model that they introduce.

3.2. Requirements on Network Action Indicators

1. When an MPLS Network Action is required, and indicator is REQUIRED in the label stack.

2. An MPLS Network Action MUST specify whether ancillary data is required in the label stack and/or post-stack data.

3. Any solution MUST respect the principle that Special Purpose Labels are the mechanism of last resort and therefore must minimise the number of new SPLs that are allocated.

4. Insertion, parsing, processing and disposition of Network Action Indicators SHOULD make use of existing MPLS data plane operations.

5. An NAI MUST NOT be delivered to a node that is not capable of processing in the way in a way that is acceptable to the imposing LER.

6. NAI MUST NOT become top of stack at a node that does not understand how to perform a disposition operation on it. Disposition includes both processing and ignoring.

7. The NAI design MUST support scoping of network actions.

8. A given NAI specification MUST specify if the scope is end-to-end, hop-by-hop, or directed at one or more selected nodes.

9. If a design allows more than one scope, a mechanism MUST be provided to specify the precedence of the scopes.

10. A mechanism is REQUIRED to enable an LER inserting NAIs to determine if the far-end LER can accept and process a packet containing a given NAI.

11. NAIs SHOULD be supported for both P2P and P2MP paths, but any specific NAI may only be supported for one or the other.

12. Data plane mechanisms for NAIs MUST be consistent across different control plane protocol types.
Internet-Draft              MNA Requirements                  April 2022

13. A mechanism MUST be defined for control / management planes in use to determine the ability of downstream LSRs/LERs to accept/ process a given NAI.

14. A mechanism is REQUIRED to enable an LSR to determine if an NAI is present in a packet.

15. NAIs can only be inserted at LERs, but MAY be processed at LSRs and LERs. If it is required to insert an NAI at a transit LSR on an LSP, then a new label stack MUST be pushed.

16. It SHOULD be possible to include indicators for multiple network actions in the same packet.

17. The solution MUST allow NAI-carrying and non-NAI-carrying packets to coexist on the same LSP.

18. The solution MUST support the processing of a subset of the NAIs on a packet.

19. Any specification of a solution that inserts or modifies the NAI MUST discuss the possible ECMP consequences.

3.3. Requirements on Ancillary Data

1. Solutions for in-stack ancillary data MUST be able to coexist with and MUST NOT obsolete existing MPLS mechanisms.

2. A common preamble for ancillary data MUST be defined so that a node receiving the ancillary data can determine whether to process, ignore, skip over or discard it according to network or local policies.

3. Any specification of a mechanism MUST describe whether it can coexist with existing post-stack data mechanisms e.g. control words and G-ACH, and if so how this coexistence operates.

4. A mechanism MUST be defined for an LER inserting ancillary data to determine that each node that needs to process the ancillary data can read the required distance into the packet at that node, for example [RFC9088].

5. Ancillary data MAY be associated with control or maintenance information for traffic carried by an LSP, and/or it MAY be associated with the user traffic itself.

6. For scoped ancillary data, a mechanism is REQUIRED to enable an LER inserting NAIs whose network actions make use of that
ancillary data, to determine if the NAI and ancillary data will be processed by LSRs within the scope along the path. Such a mechanism MAY need to determine if LSRs along the path can process a specific type of AD implied by the NAI at the depth in the stack that it will be presented to the LSR.

7. Network action specifications MUST specify if the ancillary data needs to be processed as a part of the immediate forwarding operation and whether packet mis-ordering is allowed to occur as a result of the time taken to process the ancillary data.

8. In order to prevent unnecessary scanning of the packet, care needs to be taken in the location of post stack ancillary data, for example it SHOULD be located as close to the bottom of the label stack as possible.

9. A solution MUST be provided to verify the authenticity of ancillary data processed to LSRs [RFC3552].

10. The design of the ancillary data MUST NOT expose confidential information [RFC6973] [RFC3552] to the LSRs.

4. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

5. Security Considerations

The mechanisms required by this document introduce new security considerations to MPLS. Individual solution specifications meeting these requirements MUST address any security considerations.

6. Acknowledgements

The authors gratefully acknowledge the contributions from Greg Mirsky, Yingzhen Qu, Haoyu Song, Tarek Saad, Loa Andersson, Tony Li, John Drake and Bruno Decraene.

The authors also gratefully acknowledge the input of the members of the MPLS Open Design Team.
7. References

7.1. Normative References


7.2. Informative References


Authors’ Addresses

Matthew Bocci
Nokia

Email: matthew.bocci@nokia.com

Stewart Bryant
University of Surrey 5GIC

Email: sb@stewartbryant.com
draft-decraene-mpls-slid-encoded-entropy-label-id-03

Abstract

This document defines a solution to encode a slice identifier in MPLS in order to distinguish packets that belong to different slices, to allow enforcing per network slice policies (e.g, Qos).

The slice identification is independent of the topology. It allows for QoS/DiffServ policy on a per slice basis in addition to the per packet QoS/DiffServ policy provided by the MPLS Traffic Class field.

In order to minimize the size of the MPLS stack and to ease incremental deployment the slice identifier is encoded as part of the Entropy Label.

This document also extends the use of the TTL field of the Entropy Label in order to provide a flexible set of flags called the Entropy Label Control field.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.
1. Introduction

Segment Routing (SR) [RFC8402] leverages the source-routing paradigm. A node steers a packet through a controlled set of instructions, called segments, by prepending the packet with an SR header. In the SR-MPLS data plane [RFC8660], the SR header is instantiated through a label stack.
This document defines a solution to encode a slice identifier in MPLS in order to provide QoS on a per slice basis. It allows for QoS/DiffServ policy on a per slice basis in addition to the per packet QoS/DiffServ policy provided by the MPLS Traffic Class field. The slice identification is independent of the topology and the QoS of the network, thus enabling scalable network slicing.

This document encodes the slice identifier in a portion of the MPLS Entropy Label (EL) defined in [RFC6790]. This has advantages in SR-MPLS networks as it avoids the use of additional label which would increase the size of the label stack. This also reuses the data plane processing of the Entropy Label on the egress LSR, the signaling of the Entropy Label capability from the egress to the ingress [I-D.ietf-isis-mpls-elc] [I-D.ietf-ospf-mpls-elc], and the signaling capability of transit routers to read this label [RFC8491] which allows for an easier and faster incremental deployment.

2. Entropy Label Control field

[RFC6790] defines the MPLS Entropy Label. [RFC6790] section 4.2 defines the use of the Entropy Label Indicator (ELI) followed by the Entropy Label (EL) and the MPLS header fields (Label, TC, S, TTL) in each. [RFC6790] also specifies that the TTL field of the EL must be set to zero by the ingress LSR.

Following the procedures of [RFC6790] EL is never used for forwarding and its TTL is never looked at nor decremented:

* An EL capable Egress LSR performs a lookup on the ELI and as a result pop two labels: ELI and EL.

* An EL non-capable Egress LSR performs a lookup on the ELI and as a result must drop the packet as specified in [RFC3031] for the handling of an invalid incoming label.

Hence essentially the TTL field of the EL behaves as a reserved field which must be set to zero when sent and ignored when received.

This document extends the TTL field of the EL and calls it the Entropy Label Control (ELC) field. The ELC is a set of eight flags: ELC0 for bit 0, ELC1 for bit 1, ..., ELC7 for bit 7.
Given that the MPLS header is very compact (32 bits) with no reserved bits and that MPLS is used within a trusted administrative domain, the semantic of these bits is not standardized but defined on a per administrative domain basis. This allows for increased re-use and flexibility of this scarce resource. As a consequence, an application using one of those buts MUST allow the choice of the bit by configuration by the network operator.

3. Slice Identifier

Each network slice in an MPLS domain is uniquely identified by a Slice Identifier (SLID) [I-D.bestbar-teas-ns-packet] . This section encodes the SLID in a portion of the MPLS Entropy Label defined in [RFC6790] .

The number of bits to be used for encoding the SLID in the EL is governed by a local policy and uniform within a network slice policy domain.

3.1. Ingress LSR

When an ingress LSR classifies that a packet belongs to the slice and that the egress has indicated via signaling that it can process EL for the tunnel, the ingress LSR pushes an Entropy Label with the:

* SLID encoded in the most significant bits of the Entropy Label.

* the entropy information encoded in the remaining lower bits of the Entropy Label as described in section 4.2 of [RFC6790] .

* SPI bit (SLID Presence Indicator) set in one bit of the ELC field.

The choice of the ELC field used for SPI, and the number of bits to be used for encoding the SLID MUST be configurable by the network operator.

The slice classification method is outside the scope of this document.

The encoding of the Slide ID in the Entropy Label is in line with the specification of the Flow Label as the slide identification _is_ a property of the flow:

* For a given flow it is constant in all packets.

* It’s a property specific to the flow so would typically be used to determine the Entropy Label.
3.2. Transit LSR

Any router within the SR domain that forwards a packet with the SPI bit set MUST use the SLID to select a slice and apply per-slice policies.

There are many different policies that could define a slice for a particular application or service. The most basic of these is bandwidth-allocation, an implementation complying with this specification SHOULD support the bandwidth-allocation slice as defined in the next section.

3.3. Bandwidth-Allocation Slice

A per-slice policy is configured at each interface of each router in the SR domain, with one traffic shaper per SLID. The bit rate of each shaper is configured to reflect the bandwidth allocation of the per-slice policy.

If shapers are not available, or desirable, an implementation MAY configure one scheduling queue per SLID with a guaranteed bandwidth equal to the bandwidth-allocation for the slice. This option allows a slice to consume more bandwidth than its allocation when available.

Per-slice shapers or queues effectively provides a virtual port per slice. This solution MAY be complemented with a per-virtual-port hierarchical DiffServ policy. Within the context of one specific slice, packets are further classified into children DiffServ queues which hang from the virtual port. The Traffic Class value in the MPLS header SHOULD be used for queue selection.

3.4. Backward Compatibility

The Entropy Label usage described in this document is consistent with [RFC6790] as ingress LSRs freely chooses the EL of a given flow, and transit LSRs treat the EL as an opaque set of bits.

As per [RFC6790] an ingress LSR that does not support this extension has the SPI bit cleared, and thus does not enable the SLID semantic of the Entropy bits. Hence, SLID-aware transit LSRs will not classify these packets into a slice.

3.5. Benefits

From a Segment Routing architecture perspective, this network slice identifier for SR-MPLS is inline with the network slice identifier for SRv6 proposed in [I-D.filipsil-spring-srv6-stateless-slice-id].
From an SR-MPLS perspective, using the EL to carry the network slice identifier has multiple benefits:

* This limits the number of labels pushed on the MPLS stack compared to using a pair of labels (ELI+EL) for flow entropy plus two or three labels for the slice indicator and the slice identifier. This is beneficial for the ingress LSR which may have limitations with regards to the number of labels pushed, for the transit LSR which may have limitations with regards to the label stack depth to be examined during transit in order to read both the entropy and the SLID. This presents additional benefit to network operators by reducing the packet overhead for traffic carried through the network;

* This avoids defining new extensions for the signaling of the egress capability to support the slice indicator and the slice identifier;

* This improves incremental deployment as all egress LSRs supporting EL can be sent the slice identifier from day one, allowing slice classification on transit LSRs.

4. End to end absolute loss measurements

This section describes the usage of an ELC flag to enable packet loss measurements, as described in section 3.1 of [RFC8321], for SR-MPLS networks.

TBD

5. Programmed sampling of packets

This section describes the usage of an ELC flag to detect end to end packet loss.

TBD

6. Changes / Authors Notes

[RFC Editor: Please remove this section before publication]

00: Initial version.

01: New co-author

02: editorial precision that the slice ID is a component of flow entropy hence inline with the use of entropy label.
7. References

7.1. Normative References


7.2. Informative References


Authors’ Addresses

Bruno Decraene (editor)
Orange

Email: bruno.decreene@orange.com

Clarence Filsfils
Cisco Systems, Inc.
Belgium

Email: cf@cisco.com

Wim Henderickx
Nokia
Copernicuslaan 50
95134 Antwerp 2018
Belgium

Email: wim.henderickx@nokia.com
Abstract

This document uses the Multiprotocol Label Switching (MPLS) Entropy Label (EL) extensions defined in draft-decraene-mpls-slid-encoded-entropy-label-id or a new Special Purpose Label to indicate the presence of MPLS Extension Header (MEH) in an MPLS label stack. It defines different MPLS Extension Header encoding formats to carry additional data in the MPLS label stack that can influence forwarding decision and to carry additional data after the Bottom of the MPLS label stack.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 3 September 2022.
Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the
document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal
license-info) in effect on the date of publication of this document.
Please review these documents carefully, as they describe your rights
and restrictions with respect to this document. Code Components
extracted from this document must include Revised BSD License text as
described in Section 4.e of the Trust Legal Provisions and are
provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction ............................... 3
  1.1. Requirements ........................... 4
2. Conventions Used in This Document .......... 4
  2.1. Requirements Language ................. 4
  2.2. Terminology ............................ 4
3. Overview ................................ 6
  3.1. Option 1 - ELC as MPLS Extension Header Indicator ... 7
  3.1.1. Advantages with ELC ................ 8
  3.2. Option 2 - New SPL as MPLS Extension Header Indicator ... 9
  3.3. Option 3 - NPL as MPLS Extension Header Indicator ... 9
4. In-Stack MPLS Extension Header Encoding ........ 10
5. Bottom Of Stack MPLS Extension Header Encoding .... 12
6. MPLS Extension Header Encoding Example Use-case-1.a - Carrying
  FI without data in the MPLS label stack .......... 14
7. MPLS Extension Header Encoding Example Use-case-1.b - Carrying
  FI with data in the MPLS label stack .......... 15
8. MPLS Extension Header Encoding Example Use-case-2 - Carrying FI
  with data after the MPLS label stack .......... 16
9. MPLS Extension Header Encoding Example Use-case-3 - Carrying
  use-case 1.a, 1.b and 2 in MPLS packet .......... 17
10. Node Capability Signaling ................ 18
11. Security Considerations ................ 18
12. Backward Compatibility ................ 18
  12.1. Backward Compatibility With ELC as MEH Indicator ... 18
  12.2. Backward Compatibility With SPL as MEH Indicator ... 19
13. Processing In-Stack MPLS Extension Header .... 19
14. Processing BOS MPLS Extension Header .... 20
15. IANA Considerations ................ 20
  15.1. IANA Considerations for Forwarding Instruction Flags ... 20
  15.2. IANA Considerations for IS-FI Opcode ........ 21
  15.3. IANA Considerations for BOS-FI Opcode .......... 21
  15.4. IANA Considerations for New Special Purpose Label ... 22
1. Introduction

[RFC3032] defines MPLS Header for carrying a stack of MPLS labels which are used to forward packets in an MPLS network. Today’s new applications require the MPLS packets to carry some additional indicators and associated ancillary data that would be used in MPLS packet forwarding decision or for OAM purpose.

Each application requires a separate Extended Special Purpose Label (eSPL) to address its problem that adds 2 extra labels (extension label 15 + eSPL) in the MPLS label stack. This approach does not scale, as it increases the label stack depth with multiple eSPLs that need to be imposed by the encapsulation node and scanned by the intermediate nodes. Also, currently there are no solutions defined to add ancillary data in a label stack or add multiple ancillary data after the Bottom Of Stack (BOS) in an MPLS packet. Ancillary data can be used to carry additional information, for example, a network slice identifier, In-Situ OAM (IOAM) data presence indicator, etc. Some of these use-cases are described in [I-D.saad-mpls-miad-usecases].

This document defines a new MPLS data plane extension header format to efficiently encode forwarding and OAM instructions those are easy to process in hardware. The instructions are encoded in the form of flags and opcodes and can be carried without associated ancillary data or with short in-stack ancillary data or with one or more ancillary data after the BOS.
MPLS Entropy Label (EL) standard is defined in [RFC6790]. This document uses the Entropy Label extensions defined in [I-D.decrane-mpls-slid-encoded-entropy-label-id] or a new Special Purpose Label (SPL) to indicate the presence of MPLS Extension Header (MEH) in an MPLS label stack. It defines different MPLS Extension Header encoding formats to carry additional data in the MPLS label stack that can influence forwarding decision and to carry additional data after the Bottom of the MPLS label stack.

1.1. Requirements

This document defines different MPLS Extension Header encoding formats to support the following requirements:

1. MPLS packet to carry additional data in the MPLS label stack to influence forwarding. This can be of two types:
   1a. Forwarding Instruction Flags (FIF) that does not use additional data.
   1b. Forwarding Instruction (FI) that needs additional data.

2. MPLS packet to carry additional data after the Bottom of the MPLS Label Stack.

3. Any combination of (1) and (2) in the same MPLS packet.

When MPLS Extension Header is added in an MPLS Label stack, the extension header MUST NOT contain the label field that can conflict with any previously allocated reserved label value. [I-D.bocci-mpls-miad-adi-requirements] describes additional requirements for MPLS Extension Header.

2. Conventions Used in This Document

2.1. Requirements Language

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] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.2. Terminology

BOS (Bottom Of Stack): Bottom of the MPLS label stack.

BOS-FI (Bottom Of Stack Forwarding Instruction): This is the Forwarding Instruction that is encoded after Bottom of MPLS Stack.
BPI (Bottom of the Stack MPLS Extension Header Presence Indicator): This is the flag to indicate the presence of MPLS Extension Header after the bottom of the MPLS label stack.

E2E (Edge-To-Edge): Edge to Edge.

EL (Entropy Label): Entropy Label defined as per [RFC6790].

ELC (Entropy Label Control): EL TTL field re-purposed to carry Entropy Label control bits defined in [I-D.decraene-mpls-slid-encoded-entropy-label-id].

FI (Forwarding Instruction): Forwarding Instruction is the instruction that expresses the forwarding behaviour. This can result in changing the forwarding decision or adding some information or important data to the packet.

FIF (Forwarding Instruction Flags): A bitwise flag that influences the forwarding behaviour. This flag does not need any additional data to execute its FI.

FIOC (Forwarding Instruction Opcode): A Opcode value that refers to a specific Forwarding Instruction.

HBI (Hop-By-Hop Bottom of the Stack MPLS Extension Header Presence Indicator): This is the flag to indicate the presence of MPLS Extension Header after the bottom of the MPLS label stack that require Hop-By-Hop processing.

IS-FI (In-Stack Forwarding Instruction): This is the Forwarding Instruction that is encoded in the MPLS label stack.

IPI (In-Stack MPLS Extension Header Presence Indicator): This is the flag to indicate the presence of MPLS Extension Header in the MPLS label stack.

MEI (MPLS Extension Indicator): This is the Indicator MPLS Label which indicates the presence of MPLS Extension Header in the MPLS Label stack.

MEH (MPLS Extension Header): MPLS Extension Header encoding carried in the MPLS Label stack.

MPLS (Multiprotocol Label Switching): Multiprotocol Label Switching.

NPL (Network Programming Label): Network Programming Label provisioned by user.
SPI (Slice ID Presence Indicator): This is the flag to indicate the presence of Slice ID in the Entropy Label field.

SPL (Special Purpose Label): IANA Allocated Special Purpose Label in the range of 0-15. Extended Special Purpose Label (eSPL) uses label value 15.

TC (Traffic Class): Traffic Class.

TTL (Time-To-Live): Time To Live.

3. Overview

Extending existing MPLS Header needs two main parts.

* MPLS Extension Header Indicator (MEI) - This is a way to indicate the presence of MPLS Extension Header in the packet. This could be done using two different methods. Each method has its own advantages and disadvantages. This document describes both options of MEI. The encoding formats defined in this document are compatible with both options of MEI.

Option 1. MEI by extending ELI/EL

Option 2. MEI by using a New Special Purpose Label (SPL) allocated by IANA

Option 3. MEI by using New Network Programming Label (NPL) provisioned by user

* MPLS Extension Header Format - The format in which the MPLS Extension Header could be carried in the MPLS packet. This includes both In-stack Extension Header and BOS Extension Header.

```
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Label | TC | S | TTL |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: MPLS Label Format

New In-Stack (IS) MPLS Extension Header format is defined in this document to carry the In-stack Forwarding Instruction and corresponding data in the MPLS label stack.

* It uses MPLS Label field to carry the Forwarding Instruction Opcode.
* It uses Traffic Class (TC) field to identify the Length of the MPLS In-Stack Extension Header size.

* It uses MPLS Label and Time-To-Live (TTL) fields to carry the In-Stack data (can be Flags or data).

A new Bottom Of Stack (BOS) MPLS Extension Header format is defined in this document to carry the BOS Forwarding Instruction and corresponding data after the MPLS Label stack.

The MPLS Extension Header encoding formats defined in this document are flexible and allow to stack multiple In-Stack and BOS MPLS Extension Headers in a desired order in the same MPLS packet.

3.1. Option 1 - ELC as MPLS Extension Header Indicator

As described in [I-D.decraene-mpls-slid-encoded-entropy-label-id], the EL’s 8-bit TTL field is re-purposed as Entropy Label Control (ELC) field. One bit from ELC is requested for the Slice ID Presence Indicator (SPI) and the 7 bits are available for use. From the ELC, 3 bits (for IPI, BPI and HBI) are allocated to indicate the presence of MPLS Extension Header.

```
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Entropy Label Indicator (7)       | TC  |S|      TTL      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Slice ID     |  Entropy Label        | IL  |S|  ELC (SPI=1)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

**Figure 2: ELI/EL Packet Format**

TTL (in ELC) bit allocations are defined by user as follows:
<table>
<thead>
<tr>
<th>Bit Position</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD0</td>
<td>SPI - Slice ID Presence Indicator: Indicate the presence of Slice ID in the Entropy label as defined in [I-D.decreaene-mpls-slid-encoded-entropy-label-id].</td>
</tr>
<tr>
<td>TBD1</td>
<td>IPI - In-Stack Extension Header Presence Indicator: Indicate the presence of In-Stack MPLS Extension Header after this label.</td>
</tr>
<tr>
<td>TBD2</td>
<td>BPI - Bottom Of Stack Extension Header Presence Indicator: Indicates the presence of MPLS Extension Header after the Bottom Of Stack (BOS).</td>
</tr>
<tr>
<td>TBD3</td>
<td>HBI - Hop-By-Hop Bottom Of Stack Extension Header Indicator: Indicates the MPLS Extension Header after the Bottom Of Stack requires Hop-By-Hop processing.</td>
</tr>
<tr>
<td>TBD4 - TBD7</td>
<td>Unassigned Bits.</td>
</tr>
</tbody>
</table>

Table 1: Bit Fields

**IL - In-Stack Extension Header Length** - The 3-bit TC field in the EL is used to indicate the length of the In-Stack MPLS Extension Header (excluding the ELI and EL labels) in terms of number of 32-bit labels. If more than 7 labels are needed in an MPLS extension header, the node can either use a BOS extension header to carry the data or use an additional In-stack MPLS Extension Header with MEI Label.

For backwards compatibility, an intermediate and decapsulating nodes MUST only read the length from the TC field when the IPI (In-Stack Extension Presence Indicator) is set to "1".

### 3.1.1. Advantages with ELC

Faster deployment in an existing network that has EL already deployed with an incremental benefit (e.g., incremental signaling extension for ELI capability).

Single label for Entropy in the MPLS header which helps with keeping label stack size smaller.
When EL is already enabled in the network, the proposed scheme does not require hardware to support an additional SPL indicator.

Save a new Special Purpose Label and related protocol extensions to signal its capability in LDP, RSVP-TE, BGP, IS-IS, OSPF, BGP-LS, etc.

An intermediate node can compute ECMP hash with the EL field and avoid inconsistent load-balancing of traffic flow that can happen when MPLS Extension Header alters the label stack.

Reduce MPLS Label stack size when EL is enabled for ECMP hashing when MPLS Extension Header is also used. As there is only one field for EL in the MPLS Header, it simplifies the MPLS header processing.

3.2. Option 2 - New SPL as MPLS Extension Header Indicator

The MPLS Extension Header encoding formats defined in this document is equally applicable when using a new Special Purpose Label (SPL) (value TBA1) allocated by IANA.

```
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     MEI=SPL (TBA1)                    | IL  |S| IPI,BPI,HBI   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: New SPL as MPLS Extension Header Indicator

The TTL field in the SPL (value TBA1) is used to encode FI Flags including IPI, HBI and BPI flags defined in this document. The definition and meaning of these flags and IL field are exactly the same as those in ELC field.

3.3. Option 3 - NPL as MPLS Extension Header Indicator

The MPLS Extension Header encoding formats defined in this document is equally applicable when using a Network Programming Label (NPL) configured by an operator.

```
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     MEI=NPL                            | IL  |S| IPI,BPI,HBI   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: NPL as MPLS Extension Header Indicator
The TTL field in the NPL is used to encode FI Flags including IPI, HBI and BPI flags defined in this document. The definition and meaning of these flags and IL field are exactly the same as those in ELC field.

4. In-Stack MPLS Extension Header Encoding

This section describes the encoding format of the MPLS Extension Header carried as part of the MPLS label stack. The encoding format defined is flexible (e.g., stackable opcodes in desired order), extensible (by defining new Opcodes) and ASIC friendly (by using Extension Header Length, Opcode+Data in the same field).

```
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Entropy Label or SPL or NPL      | IL=1|S| IPI=1         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  IS-FI Opcode |    In-Stack Data      |R|D|E|S| In-Stack Data |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: In-Stack Extension Header Format

IPI flag is set to "1" to indicate the presence of In-Stack MPLS Extension Header.

Since In-Stack MPLS Extension Header is present as part of the MPLS Label stack, the 32-bit MPLS Label is redefined to encode the MPLS Extension Header as follows:

Label Field:

The first 8 bits are used to define the In-Stack Forwarding Instruction (IS-FI) Opcode. Next 12 bits in the Label field and the 8 bits from the TTL field are used to carry In-Stack data corresponding to the IS-FI opcode. This opcode ranges from 1 to 255. IS-FI Opcode value of 0 is marked as invalid to avoid the label value aliasing with the reserved SPLs.

* IS-FI Opcode Value:1 - IANA Allocated to carry the Forwarding Instruction Flags (FIF).

* IS-FI Opcode Value:2 - IANA Allocated to indicate the offset in terms of number of bytes for the start of the BOS data after the MPLS Label Bottom of the Stack. This can allow to carry Generic Control Word (0000b) [RFC4385] and G-ACh (0001b) [RFC5586] fields immediately after the BOS. Adding of this opcode is not required when the BOS data starts immediately after the Bottom of the Label Stack (i.e. when offset is 0).
* IS-FI Opcode Value: 3-254 - MUST be assigned by IANA.

* IS-FI Opcode Value: 255 - IANA Allocated for IS-FI Opcode range extension. This gives the extensibility for opcode range beyond 255.

IS-FI Opcode MUST define the following procedure before it can be used:

1. Define the Data format encoded in the MPLS extension header.

2. Define the Hop-By-Hop or Edge-To-Edge (only on the decapsulation node) processing scope.

3. The Hop-By-Hop IS-FI opcodes MUST be placed before the Edge-To-Edge IS-FI Opcodes in the MPLS Extension Header of the packet to optimize the Hop-By-Hop processing in hardware.

TC Field:

This field is used to indicate the MPLS Extension Header stacking and In-Stack Data stacking.

E (E2E-Bit): MPLS Extension Header In-Stack Data requires E2E processing. If this is set to "1", then this 4-byte MPLS Extension Header requires Edge-To-Edge processing. If this is set to 0, then this 4-byte MPLS Extension Header requires Hop-By-Hop processing. Note that E2E-Bit is not used with the Entropy Label TC field.

D (DS-Bit): Data Stacking Bit. This is used to encode more than 20 bits of data for this IS-FI Opcode. If this is set to "1", then this is the end of the data for the IS-FI Opcode.

R (Reserved Bit): MUST be set to "0" on transmit and ignored on receive.

TTL Field:

This 8-bit field is used to carry In-Stack data apart from the 12 bits in the Label field.

NOTE:
An intermediate node may use the full MPLS label stack for ECMP hash computation hence the In-Stack MPLS extension header MUST NOT change the Label Field part of the IS-FI data within the same traffic flow. But the TTL part of IS-FI data can change for the same traffic flow without affecting the ECMP hash. The In-Stack Extension Header encoding defined above ensures this.

5. Bottom Of Stack MPLS Extension Header Encoding

This section describes the encoding format of the MPLS Extension Header which is present after the bottom of the MPLS label stack.

```
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Entropy Label or SPL or NPL      | TC  |1|  BPI=1, HBI   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0 0 1 0|Reserve|  BOS-FI Opcode| Length=1(word)|   BOS-Flags   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        BOS-Data                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Payload                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Figure 6: BOS Extension Header Format
```

BPI flag is set to "1" to indicate the presence of MPLS Extension Header after the bottom of MPLS label stack.

HBI flag is set to "1" to indicate the MPLS Extension Header after the Bottom Of Stack that requires Hop-By-Hop processing.

A new generic 4-byte header is defined to carry the information about the Forwarding Instruction and its corresponding data that is carried after the bottom of label stack. This generic header is added to each Forwarding Instruction that is encoded after the MPLS bottom of the stack. This generic header gives the flexibility to add multiple Forwarding Instruction after the BOS in any desired order.

The 4-byte BOS Extension Header is described below:
<table>
<thead>
<tr>
<th>Bit Position</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - 3</td>
<td>This 4-bit nibble MUST be set to &quot;0010b&quot;. This is to avoid aliasing with an IPv4/IPv6 header.</td>
</tr>
<tr>
<td>4 - 7</td>
<td>This 4-bit nibble defines the version of the generic header format. The current version value is &quot;0&quot;.</td>
</tr>
<tr>
<td>8 - 15</td>
<td>This 8-bit field indicates the BOS FI Opcode value. This opcode values will be allocated by IANA.</td>
</tr>
<tr>
<td>16 - 23</td>
<td>This 8-bit field indicates the length of the data encoded in units of 4 bytes excluding the current header.</td>
</tr>
<tr>
<td>24 - 31</td>
<td>This 8-bit field carries the BOS-Flags. 0 - NH bit (Next-Header Presence Bit): Indicates the presence of next BOS extension header. 1 - H bit (Hop-By-Hop Bit): Hop-By-Hop processing is required for this Bottom Of Stack data. 7 - 2 bits: Unassigned bits.</td>
</tr>
</tbody>
</table>

Table 2: BOS MPLS Extension Header Format

BOS-FI Opcode value of 0 is marked as invalid.

BOS-FI Opcode Value:1-254 - MUST be assigned by IANA.

BOS-FI Opcode Value:255 - IANA Allocated for BOS-FI Opcode range extension. This gives the extensibility for opcode range beyond 255.

If an application requires to add its own data TLV, then the TLV can be added as part of BOS-Data.

BOS-FI Opcode MUST define the following procedure before it can be used:

1. Define the Data format encoded in the MPLS extension header.

2. Define the Hop-By-Hop or Edge-To-Edge (only on the decapsulation node) processing scope.

3. The Hop-By-Hop BOS-FI opcodes MUST be placed before the Edge-To-Edge BOS-FI Opcodes in the MPLS Extension Header of the packet to optimize the Hop-By-Hop processing in hardware.
6. MPLS Extension Header Encoding Example Use-case-1.a - Carrying FI without data in the MPLS label stack

The TTL field can support only up to 8-bit flags. This is the use-case to extend the TTL flags and carry additional Forwarding Instruction Flags (FIF) in the MPLS label stack. These forwarding instructions do not require any additional data to be carried with this FI.

```
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Entropy Label or SPL or NPL | IL=1|0| IPI=1 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|IS-FI Opcode=1 | Flags                |R|1|E|1| Flags        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: Example In-Stack Extension Header Carrying Forwarding Instruction Flags

IPI flag is set to "1" to indicate the presence of In-Stack MPLS Extension Header.

Label Field:

In this case the FI opcode value is set to "1". FI Opcode value "1" is reserved for extending the TTL flags. This indicates the presence of additional flags in the Label field and TTL fields

TC Field:

DS-Bit - This bit is set to "1" to indicate that the flags are not extended further.

TTL Field:

8-bit field is used to encode the Forwarding Instruction Flags apart from 12 bits Label field.

The FIF bit position and its meaning MUST be defined by IANA.

```
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Entropy Label or SPL or NPL | IL=2|0| IPI=1 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|IS-FI Opcode=1 | Flags                |R|1|E|1| Flags        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8: Example In-Stack Extension Header carrying more than 20 bits FI Flags

More than 20 bits of data can be encoded as part of IS-FI opcode. In this specific case, the FI flags which are more than 20 bits are encoded in next 4 bytes of the MPLS header.

While encoding the additional data, the Most Significant bit of the Label Field MUST be set to "1" to prevent from aliasing with the reserved SPLs in the case of legacy devices.

7. MPLS Extension Header Encoding Example Use-case-1.b - Carrying FI with data in the MPLS label stack

This is the use-case where the MPLS Label stack to carry the Forwarding Instruction with a corresponding data.

```
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Entropy Label or SPL or NPL      | IL=1|0|   IPI=1       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| IS-FI Opcode  |        Data           |R|1|E|1|   Data        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9: Example In-Stack Extension Header Carrying FI with the data

IPI flag is set to "1" to indicate the presence of In-Stack MPLS Extension Header.

Label Field:

First 8 bits encodes the In-Stack forwarding opcode. In this case the FI opcode value ranges from 1 to 254. This value is assigned by IANA. This opcode value defines data format carried in the Label field and the TTL field.

TC Field:

DS-Bit - This bit is set to "1" to indicate that the data is encoded in the 19-bit Label field and does not exceed 19 bits.

R-Bit - Reserved bit and MUST be set to "0" on transmit and ignored when received.

TTL Field:

8-bit field is used to encode the In-Stack data apart from 12-bit Label field.
Figure 10: Example In-Stack Extension Header Carrying FI with the data more than 20 bits

More than 20 bits of data can be encoded as part of IS-FI opcode. In this specific case, the In-Stack data which are more than 20 bits are encoded in next 4 bytes of the MPLS header.

While encoding the additional data, the Most Significant bit of the Label Field MUST be set to "1" to prevent from aliasing with the reserved SPLs in the case of legacy devices.

8. MPLS Extension Header Encoding Example Use-case-2 - Carrying FI with data after the MPLS label stack

This is the use-case where the Forwarding Instruction with a corresponding data is carried after the MPLS bottom of label stack.

Figure 11: Example BOS Extension Header Carrying FI with data

BPI flag is set to "1" to indicate the presence of BOS MPLS Extension Header. Also, HBI flag is set to 1 to indicate the presece of BOS MPLS Extension Header that requires Hop-By-Hop processing.
In this case, the MPLS packet is encoding two different types of BOS FI (Opcode 1 and Opcode 2) after the bottom of MPLS label stack.

The first BOS MPLS Extension Header has the Length value as "1", this indicates that the data corresponding to this FI opcode "Type1" is 4 bytes following this header. Also the Next-Header (NH) flag in BOS-Flags is set to "0x1", this indicates the presence of next BOS MPLS Extension Header. The H flag is set to "0x1" that indicates the Hop-By-Hop processing is required.

The second BOS MPLS Extension Header has the Length value as "2", this indicates that the data corresponding to the FI opcode "Type2" is 8 bytes following this header. In this case the Next-Header flag in BOS-Flags is set to "0x0", this indicates that this is the last BOS MPLS Extension Header encoded. The H flag is set to "0x0" that indicates the Hop-By-Hop processing is not required.

9. MPLS Extension Header Encoding Example Use-case-3 - Carrying use-case 1.a, 1.b and 2 in MPLS packet

This is the use-case where the same MPLS packet carry the use-cases "1.a", "1.b" and "2".

```
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|      Entropy Label or SPL or NPL      | IL=3|0| IPI=BPI=HBI=1 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| IS-FI Opcode=1| Flags                |R|0|E|0| Flags         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| IS-FI Opcode=2| 0                    |R|1|E|0| Offset = 1    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| IS-FI Opcode=3| Data                 |R|1|E|1|      Data     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
                                      +---------------------+
                                      | 0 0 1 0|Reserve|BOS-FI Opcode=1| Length=1(word) Flags NH=1,H=1 |
                                      +---------------------+
                                      | BOS-Data1          |
                                      +---------------------+
                                      | 0 0 1 0|Reserve|BOS-FI Opcode=2| Length=2(word) Flags NH=0,H=0 |
                                      +---------------------+
                                      | BOS-Data2          |
                                      | BOS-Data2          |
                                      | Payload            |
```

IPI and BPI flags are set to "1" to indicate the presence of both In-Stack and BOS MPLS Extension Header as mentioned in the above use-cases. IS-FI Opcode 2 is added to indicate the offset of 1 word after the MPLS header BOS and start of the BOS Extension Header.

10. Node Capability Signaling

The node capability for the MPLS Extension Header must be signaled before the MPLS Encapsulating node can add the necessary MPLS Extension Header in the MPLS label stack. The capability signaling will be added in LDP, RSVP-TE, BGP, IS-IS, OSPF, BGP-LS, etc. This is outside the scope of this document.

11. Security Considerations

The security considerations in [RFC3032] also apply to the extensions defined in this document. The MPLS Extension header MUST NOT be exposed to the node which does not support the new MPLS Extension Header.

12. Backward Compatibility

12.1. Backward Compatibility With ELC as MEH Indicator

As specified in [RFC6790], the TTL field of the EL MUST be "0". On the Node which is capable of processing the MPLS Extension Header when it finds that this TTL value is non-zero, then only it will start processing the MPLS Extension header.

In addition, the TC field will be interpreted as the In-Stack MPLS Header Extension Length only when the TTL field’s IPI Flag is set to "1".

For the legacy node that does not advertise the MPLS Extension Header capability, the Encapsulating node MUST make sure that the MPLS Extension header is not at the top of the MPLS label stack to avoid misforwarding the packets by misinterpreting In-Stack Extension Header as a label.

The MPLS Extension Header Encoding format is designed to make sure that it does not alias with any reserved SPL.

The MPLS extension does not affect the existing GAL / G-ACh [RFC5586] based encoding of data in the MPLS packet. This MPLS extension can co-exist with the existing GAL / G-ACh based encoding of data.
12.2. Backward Compatibility With SPL as MEH Indicator

For the legacy node that does not advertise the MPLS Extension Header capability, the Encapsulating node MUST make sure that the MPLS Extension header is not at the top of the MPLS label stack to avoid dropping the packets.

The MPLS Extension Header Encoding format is designed to make sure that it does not alias with any reserved SPL.

The MPLS extension does not affect the existing GAL / G-ACh [RFC5586] based encoding of data in the MPLS packet. This MPLS extension can co-exist with the existing GAL / G-ACh based encoding of data.

13. Processing In-Stack MPLS Extension Header

Encapsulating Node:

* MUST NOT add In-Stack MPLS Extension header if the decapsulation node is not capable of In-Stack MPLS Extension header.

* SHOULD NOT change the IS-FI Opcode and the first 12 bits of the In-Stack Data for the same packet flow avoid ECMP path change.

* MAY change In-Stack data part present only in the TTL field for the same packet flow.

* MUST ensure that the penultimate node does not remove the MPLS extension header.

Intermediate Node:

* MUST ignore the IS-FI Opcode that are not supported.

* MUST NOT add In-Stack MPLS Extension header if the decapsulation node is not capable of In-Stack MPLS Extension header.

* SHOULD NOT change the IS-FI Opcode and the first 12 bits of the In-Stack Data for the same packet flow.

* MAY change In-Stack data part present only in the TTL field for the same packet flow.

* MAY remove the IS-FI opcode and its corresponding data for all matching packet flow.

Decapsulating Node:
* MUST remove the In-Stack MPLS Extension header.

14. Processing BOS MPLS Extension Header

Encapsulating Node:

* MUST NOT add BOS MPLS Extension header if the decapsulation node is not capable of BOS MPLS Extension header.

* MUST ensure that the penultimate node does not remove the MPLS extension header.

Intermediate Node:

* MAY add additional data to the existing BOS-FI encoded.

* MAY add a new BOS-FI and its corresponding data if the decapsulation node supports BOS MPLS Extension header.

Decapsulating Node:

* MUST remove the BOS MPLS Extension header.

15. IANA Considerations

Below are the IANA actions which this document is requesting.

15.1. IANA Considerations for Forwarding Instruction Flags

IANA is requested to create a new registry to assign the bit position and the meaning to the Forwarding Instruction Flags based on the user request.

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>19-0</td>
<td>Unassigned</td>
<td>This document</td>
</tr>
</tbody>
</table>

Table 3: Forwarding Instruction Flags Registry
15.2. IANA Considerations for IS-FI Opcode

IANA is requested to create a new registry to assign IS-FIOC opcode values. All code-points in the range 1 through 175 in this registry shall be allocated according to the "IETF Review" procedure as specified in [RFC8126]. Code points in the range 176 through 239 in this registry shall be allocated according to the "First Come First Served" procedure as specified in [RFC8126]. Remaining code-points are allocated according to Table 4:

| Value | Description       | Reference     |
|-------+------------------|---------------|
| 1 - 175 | IETF Review       | This document |
| 176 - 239 | First Come First Served | This document |
| 240 - 251 | Experimental Use | This document |
| 252 - 254 | Private Use       | This document |

Table 4: In-Stack Forwarding Instruction Opcode Registry

Following IS-FIOC Opcode values are assigned from this registry.

| Value | Description                  | Reference     |
|-------+==============================|---------------|
| 0     | Invalid value                | This document |
| 1     | Forwarding Instruction Flags | This document |
| 2     | Offset of start of Bottom Of Stack Data after BOS Label | This document |
| 255   | Opcode Range Extension Beyond 255 | This document |

Table 5: In-Stack Forwarding Instruction Opcode Values

15.3. IANA Considerations for BOS-FI Opcode

IANA is requested to create a new registry to assign BOS-FIOC opcode values.
Following BOS-FIOC Opcode values are assigned from this registry.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Invalid value</td>
<td>This document</td>
</tr>
<tr>
<td>255</td>
<td>Opcode Range Extension Beyond 255</td>
<td>This document</td>
</tr>
</tbody>
</table>

The application that requires an Opcode for the Forwarding Instruction (IS-FIOC or BOS-FIOC) or a Flag must request the code-point and its meaning from IANA.

15.4. IANA Considerations for New Special Purpose Label

IANA is requested to allocate a value TBA1 for the MEI SPL label from the "Base Special-Purpose MPLS Label Values" registry to indicate the presence of MPLS Header Extension.

16. Appendix

16.1. Alternate approach for In-Stack Extension Header Encoding

In the above In-Stack Extension Header Encoding the Label field is used to encode the FI Opcode. So just for completeness, here is the alternate way of In-Stack Extension Header Encoding is provided.
Figure 13: Alternate In-Stack Extension Header Format

IPI flag is set to "1" to indicate the presence of In-Stack MPLS Extension Header.

Since In-Stack MPLS Extension Header is present as part of the MPLS Header, the MPLS Header is redefined to encode the MPLS Extension Header.

Label Field:

Most significant bit is always set to "1" to avoid aliasing with the reserved SPLs.

Rest of the 19 bits and the "R" bit from the TC bit can be used by the application. So total of 20 bits can be used to carry the data corresponding to IS-FI opcode.

TC Field:

This carries data stacking bits. They are as follows:

D (DS-Bit): Data Stacking Bit. This is used to encode more than 19 bits of extended data in the MPLS Label stack. If this is set to "1", then this is the end of extended data.

R (Reserved Bit): This is used to encode the IS-FI data.

TTL Field:

This carries In-Stack Forwarding Instruction opcode.

16.2. MPLS Extension Header Example for Entropy Label using New SPL

The MPLS Extension Header encoding formats defined in this document is applicable when using a new Special Purpose Label (SPL) or using a Network Programming Label (NPL) configured by an operator.

The TTL field in the SPL (value TBA1) is used to encode FI Flags including IPI, HBI and BPI flags defined in this document.
16.3. MPLS BOS Extension Header Example with IOAM Data Fields

The Bottom Of Stack (BOS) Extension Header is used with BOS Opcode for IOAM.

Bottom Of Stack Presence Indicator (BPI) flag in TTL is set to "1" to indicate the presence of BOS Extension Header. HBI flag in TTL is set to "1" to indicate the BOS Extension Header requires Hop-By-Hop processing.

```
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Entropy Label or SPL or NPL | TC |1| BPI=1, HBI |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 0 0 1 0|Reserve|BOS Opcode=IOAM|Length (words) | Flags (NH, H) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| IOAM-OPT-Type | IOAM HDR Len | Block Number | Reserved |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
IOAM Option and Data Space

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Optional Payload + Padding
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 15: Example MPLS Encapsulation for IOAM using BOS Extension Header
17. References

17.1. Normative References


17.2. Informative References


Acknowledgments

The authors of this document would like to thank the MPLS Working Group Design Team for the discussions and comments on this document.

Contributors

TBD

Authors’ Addresses

Jaganbabu Rajamanickam (editor)
Cisco Systems, Inc.
Canada
Email: jrajaman@cisco.com

Rakesh Gandhi (editor)
Cisco Systems, Inc.
Canada
Email: rgandhi@cisco.com

Jisu Bhattacharya
Cisco Systems, Inc.
Email: jisu@cisco.com

Bruno Decraene
Orange
Email: bruno.decraene@orange.com

IANA Registry for the First Nibble Following a Label Stack
draft-kbbma-mpls-1stnibble-01

Abstract

The goal of this memo is to create a new IANA registry (called the
MPLS First Nibble registry) for the first nibble (4-bit field)
immediately following an MPLS label stack. The memo offers a
rationale for such a registry, describes how the registry should be
managed, and provides some initial entries. Furthermore, this memo
sets out some documentation requirements for registering new values.
Finally, it provides some recommendations that makes processing MPLS
packets easier and more robust.

There is an important caveat on the use of this registry versus the
IP version number registry.

This memo, if published, would update [RFC4928] and [RFC8469].

Status of This Memo

This Internet-Draft is submitted in full conformance with the
provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering
Task Force (IETF). Note that other groups may also distribute
working documents as Internet-Drafts. The list of current Internet-
Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months
and may be updated, replaced, or obsoleted by other documents at any
time. It is inappropriate to use Internet-Drafts as reference
material or to cite them other than as "work in progress."

This Internet-Draft will expire on 29 October 2022.
1. Introduction

An MPLS packet consists of a label stack, an optional "post-stack header" and an optional embedded packet (in that order). However, in the data plane, there are scant clues regarding the post-stack header, and no clue as to the type of embedded packet; this information is communicated via other means, such as the routing protocols that signal the labels in the stack. Nonetheless, in order to better handle an MPLS packet in the data plane, it is common practice for network equipment to "guess" the type of embedded packet. Such equipment may also need to process the post-stack header. Both of these require parsing the data after the label stack. To do this, the "first nibble" (the top four bits of the first octet following the label stack) is often used.
The semantics and usage of the first nibble is not well documented, nor are the assignments of values. This memo serves three purposes:

* To document the assignments already made

* To provide for the clear documentation of future assignments through the creation of an "MPLS First Nibble registry"

* Provide a method to tracking usage by requiring more consistent documentation

1.1. Conventions and Definitions

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 BCP14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

LSR: label switching router.

MPLS packet: one whose Layer 2 header declares the type to be MPLS. For Ethernet, that means the Ethertype is 0x8847 or 0x8848.

Label Stack: (of an MPLS packet) all labels (four octet fields) after the Layer 2 header, up to and including the label with the BoS bit set ([RFC3032]).

MPLS First Nibble (MFN): the most significant four bits of the first octet following the label stack.

MPLS Payload: all data after the label stack, including the MFN, an optional post-stack header and the embedded packet.

Post-stack Header (PSH): optional field of interest to the egress LSR (and possibly to transit LSRs). Examples include a control word or an associated channel. The PSH MUST indicate its length, so that a parser knows where the embedded packet starts.

Embedded Packet: All octets beyond the PSH (if any). This could be an IPv4 or IPv6 packet (e.g., for traffic engineering of IP packets, or for a Layer 3 VPN [RFC4364]), an Ethernet packet (for VPLS ([RFC4761], [RFC4762]) or EVPN [RFC7432]), or some other type of Layer 2 frame [RFC4446].
Figure 1: Example of an MPLS Packet With Label Stack
Figure 1 shows an MPLS packet with Layer 2 header X and a label stack Y ending with Label-n. Figure 2 shows three examples of an MPLS payload, A, B and C. The full MPLS packets thus are: [X, Y, A], [X, Y, B], and [X, Y, C].

A. The first payload is a bare IP packet, i.e., no PSH. The MFN (MPLS First Nibble) in this case overlaps with the IP version number.

B. The next payload is a bare non-IP packet; again, no PSH. The MFN here is the first nibble of the payload, whatever it happens to be.
C. The last example is an MPLS Payload that starts with a PSH followed by the embedded packet.

2. Rationale

2.1. Why Look at the First Nibble

An MPLS packet can contain many types of embedded packet. The most common types are:

1. An IPv4 packet (whose IP header has version number 4).
2. An IPv6 packet (whose IP header has version number 6).
3. A Layer 2 Ethernet frame (i.e., not including the Preamble or the Start frame delimiter), starting with the destination MAC address.

Many other packet types are possible, and in principle, any Layer 2 embedded packet is permissible; indeed, in the past, PPP, Frame Relay and ATM packets were reasonably common.

In addition, there may be a post-stack header ahead of the embedded packet, and this needs be to parsed. The MPLS First Nibble is currently used for both of these purposes.

2.1.1. Load Balancing

There are four common ways to load balance an MPLS packet:

1. One can use the top label alone.
2. One can do better by using all the (non-SPL) labels in the stack.
3. One can do even better by "divining" the type of embedded packet, and using fields from the guessed header.
4. One can do best by using either an Entropy Label [RFC6790] or a FAT Pseudowire Label [RFC6391]; see Section 2.1.3.)
Load balancing based on just the top label means that all packets with that top label will go the same way -- this is far from ideal. Load balancing based on the entire label stack (not including SPLs) is better, but may still be uneven. If, however, the embedded packet is an IP packet, then the combination of (<source IP address>, <dest IP address>, <transport protocol>, <source port>, and <dest port>) from the IP header of the embedded packet forms an excellent basis for load balancing. This is what is typically used for load balancing IP packets.

An MPLS packet doesn’t, however, carry a payload type identifier. There is a simple heuristic that is commonly used to guess the type of the embedded packet. The first nibble, i.e., the four most significant bits of the first octet, of an IP header contains the IP version number. This in turn indicates where to find the relevant fields for load balancing. The heuristic goes roughly as follows:

2.1.1.1. Heuristic for Load Balancing

1. If the MFN is 0x4 (0100b), treat the payload as an IPv4 packet, and find the relevant fields for load balancing on that basis.

2. If the MFN is 0x6 (0101b), treat the payload as an IPv6 packet, and find the relevant fields for load balancing on that basis.

3. If the MFN is anything else, the MPLS payload is not an IP packet; fall back to load balancing using the label stack.

This heuristic has been implemented in many (legacy) routers, and performs well in the case of Figure 1, A. However, this heuristic can work very badly for Figure 1, B. For example, if payload B is an Ethernet frame, then the MFN is the first nibble of the OUI of the destination MAC address, which can be 0x4 or 0x6, and if so would lead to very bad load balancing. This behavior can happen to other types of non-IP payload as well.

This in turn led to the idea of inserting a PSH (e.g., a pseudowire control word [RFC4385], a DetNet control word [RFC8964] or a BIER header [RFC8296]) where the MPLS First Nibble is NOT 0x4 or 0x6, to explicitly prevent forwarding engines from confusing the MPLS payload with an IP packet. [RFC8469] recommends the use of a control word when the embedded packet is an Ethernet frame. RFC 8469 was published at the request of the operator community and the IEEE RAC as a result of operational difficulties with pseudowires that did not contain the control word.
This memo introduces a requirement and a recommendation, the first building on the above; the second deprecating the use of the heuristic in Section 2.1.1.1. The intent of both of these is that legacy routers continue to operate as they have, with no new problems introduced as a result of this memo. However, new implementations SHOULD follow these recommendations for more robust operation.

2.1.2. Requirement

Going forward, network equipment MUST use a post-stack header with an MPLS First Nibble value that is not 0x4 or 0x6 in all cases when the MPLS payload is not an IP packet. Effectively, Figure 1, B is disallowed. [AGREED??]

This replaces the following text from [RFC4928], section 3, paragraph 3:

"It is REQUIRED, however, that applications depend upon in-order packet delivery restrict the first nibble values to 0x0 and 0x1. This will ensure that their traffic flows will not be affected if some future routing equipment does similar snooping on some future version(s) of IP."

This also replaces the following text from [RFC8469], section 4, paragraph 1:

"This document updates [RFC4448] to state that both the ingress provider edge (PE) and the egress PE SHOULD support the Ethernet PW CW and that, if supported, the CW MUST be used."

2.1.3. Recommendation

It is RECOMMENDED that, going forward, if good load balancing of MPLS packets is desired, either an Entropy Label or a FAT Pseudowire Label SHOULD be used; furthermore, going forward, the heuristic in Section 2.1.1.1 MUST NOT be used. [AGREED??]

A consequence of Recommendation 2 is that, while legacy routers may look for a MPLS First Nibble of 0x4 or 0x6, no router will look for a MPLS First Nibble of 0x7 (or whatever the next IP version number will be) for load balancing purposes. This means that the values 0x4 and 0x6 are used to (sometimes incorrectly) identify IPv4 and IPv6 packets, but no other First Nibble values will be used to identify IP packets.

This obviates the need for paragraph 4, section 3 in [RFC4928]:

"This behavior implies that if in the future an IP version is defined with a version number of 0x0 or 0x1, then equipment complying with this BCP would be unable to look past one or more MPLS headers, and loadsplit traffic from a single LSP across multiple paths based on a hash of specific fields in the IPv0 or IPv1 headers. That is, IP traffic employing these version numbers would be safe from disturbances caused by inappropriate loadsplitting, but would also not be able to get the performance benefits."

This also expands the MFN Registry to all 16 possible values, not just 0x0 and 0x1.

2.1.4. Parsing the Post-stack Header

Given the above recommendations on the use of a post-stack header and future non-use of the heuristic (Section 2.1.1.1) via the use of Entropy or FAT Pseudowire Labels, the main reason for creating a First Nibble registry is to document the types of post-stack headers that may follow a label stack, and to simplify their parsing.

2.2. Why Create a Registry

The MPLS WG is currently engaged in updating the MPLS architecture; part of this work involves the use of post-stack headers. This is not possible if post-stack header values are allocated on an ad hoc basis, and their parsing and semantics is ill-specified. Consider that the MPLS First Nibble value of 0x0 has two different formats, depending on whether the post-stack header is a pseudowire control word or a DetNet control word; disambiguation requires the context of the service label. This was a considered decision; documenting this would be helpful to future implementors.

With a registry, post-stack headers become easier to parse; the values are unique, not needing means outside the data plane to interpret them correctly; and their semantics and usage are documented. (Thank you, IANA!)

2.3. Caveat

The use of the MPLS First Nibble stemmed from the desire to heuristically identify IP packets for load balancing purposes. It was then discovered that non-IP packets, misidentified as IP when the heuristic failed, were being badly load balanced, leading to [RFC4928]. This situation may confuse some as to relationship between the MPLS First Nibble Registry and the IP Version Numbers registry. These registries are quite different:
1. The IP Version Numbers registry’s explicit purpose is to track IP version numbers in an IP header.

2. The MPLS First Nibble registry’s purpose is to track post-stack header types.

The only intersection points between the two registries is for values 0x4 and 0x6 (for backward compatibility). There is no need to track future IP version number allocations in the MPLS First Nibble registry.

3. IANA Considerations

3.1. MPLS First Nibble Registry

This memo recommends the creation of an IANA registry called "The MPLS First Nibble Registry" with the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
<th>Reference</th>
<th>Allocation Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>PW Control Word</td>
<td>RFC 4385</td>
<td></td>
</tr>
<tr>
<td>0x0</td>
<td>DetNet Control Word</td>
<td>RFC 8964</td>
<td></td>
</tr>
<tr>
<td>0x1</td>
<td>PW Assoc Channel</td>
<td>RFC 4385</td>
<td></td>
</tr>
<tr>
<td>0x2</td>
<td>Unallocated</td>
<td></td>
<td>Standards Action</td>
</tr>
<tr>
<td>0x3</td>
<td>Unallocated</td>
<td></td>
<td>Standards Action</td>
</tr>
<tr>
<td>0x4</td>
<td>IPv4 header</td>
<td>RFC 791</td>
<td></td>
</tr>
<tr>
<td>0x5</td>
<td>BIER header</td>
<td>RFC 8296</td>
<td></td>
</tr>
<tr>
<td>0x6</td>
<td>IPv6 header</td>
<td>RFC 8200</td>
<td></td>
</tr>
<tr>
<td>0x7-e</td>
<td>Unallocated</td>
<td></td>
<td>Standards Action</td>
</tr>
<tr>
<td>0xf</td>
<td>Reserved for expansion</td>
<td></td>
<td>Standards Action</td>
</tr>
</tbody>
</table>

Table 1: MPLS First Nibble Values

3.1.1. Allocation Policy

All new values registered here MUST use the Standards Action policy [RFC8126].
4. References

4.1. Normative References


4.2. Informative References


Authors’ Addresses
Kireeti Kompella (editor)
Juniper Networks
1133 Innovation Way
Sunnyvale, 94089
United States of America
Phone: +1-408-745-2000
Email: kireeti.ietf@gmail.com

Stewart Bryant
University of Surrey 5GIC
Email: sb@stewartbryant.com

Matthew Bocci
Nokia
Email: matthew.bocci@nokia.com

Greg Mirsky
Ericsson
Email: gregimirsky@gmail.com

Lars Olaf (Loa) Andersson
Bronze Dragon Consulting
Email: loa@pi.nu
Abstract

The MPLS architecture introduced Special Purpose Labels (SPLs) to indicate special forwarding actions and offered a few simple examples, such as Router Alert. In the two decades since the original architecture was crafted, the range, complexity and sheer number of such actions has grown; in addition, there now is need for "associated data" for some of the forwarding actions. Likewise, the capabilities and scale of forwarding engines has also improved vastly over the same time period. There is a pressing need to match the needs with the capabilities to deliver the next generation of MPLS architecture.

In this memo, we propose an alternate mechanism whereby a single SPL can encode multiple forwarding actions and carry associated data, some in the label stack and some after the label stack. This proposal also solves the problem of scarcity of base SPLs.

This approach can immediately address several use cases:

* to carry a Slice Selector for IETF network slicing;
* to signal that further fast reroute may have harmful consequences;
* to indicate that there is relevant data after the label stack;
* among others.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.
Table of Contents

1.  Introduction ........................................... 3
1.1.  Conventions and Definitions .......................... 3
1.2.  Revision History .................................... 3
   1.2.1.  Changes from -00 to -01 ...................... 4
   1.2.2.  Changes from -01 to -02 ...................... 4
1.3.  Slice Selector ...................................... 5
2.  Multi-purpose bSPL: the Forwarding Actions Indicator .... 5
   2.1.  The FAI bSPL .................................. 6
      2.1.1.  ISD vs PSD ............................... 6
   2.2.  Format of the FAI bSPL .......................... 6
      2.2.1.  Definitions of the FAI Flag Bits .......... 7
      2.2.2.  Processing the FAI Flags and the ISD .... 9
      2.2.3.  Example of the FAI ....................... 9
3.  Issues to be Resolved .................................. 10
   3.1.  Preventing FAI From Reaching Top of Stack ....... 10
   3.2.  Repeating the FAI at "Readable Stack Depth" .... 11
   3.3.  PSD ........................................... 11
4.  Contributors ........................................... 11
5.  Acknowledgments ....................................... 12
6.  IANA Considerations .................................... 12
1. Introduction

Base Special Purpose Labels (bSPLs) are a precious commodity; there are only 16 such values, of which 8 have already been allocated. There are currently five requests for bSPLs that the authors are aware of; this document proposes another use case for a bSPL, in all consuming nearly all the remaining values. This document suggests a method whereby a single bSPL can be used for all the purposes currently requested. This leads to perhaps the more valuable long-term contribution of this document: an approach to the definition and use of bSPLs (and SPLs in general) whereby a single value can be used for multiple purposes, and provide a flexible yet efficient means of carrying associated data.

1.1. Conventions and Definitions

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.

FAI: Forwarding Actions Indicator
FFB: Forwarding Flags Block
ISD: In-Stack Data
sISD: Standard ISD
uISD: User-Defined ISD
PSD: Post-Stack Data
SPL: Special-purpose label
bSPL: Base special-purpose label

1.2. Revision History

This section (to be removed before publication) offers highlights from the draft’s revision history.
1.2.1. Changes from -00 to -01

1. This section added.

2. Added a section discussing when data should be put in the LS FAD vs in the PL FAD.

3. Tweaked the bits in the FAI. Added a field "edist".

4. Elaborated on the use of the H bit and the FAH data.

5. Updated the processing of the LS FAD.

6. Added processing of edist.

7. Updated the FAI example.

8. Updated the Issues section.

1.2.2. Changes from -01 to -02

1. Updated Abstract and Introduction to focus on FAI; moved description of use cases to separate section.

2. Added terminology.

3. Changed terminology: LS FAD and PL FAD to ISD and PSD, respectively.

4. Updated text on criteria for putting associated data in ISD.

5. Introduced the terms FAI Block, FFB Block, sISD Block and uISD Block. Introduced an "end of block" bit, s. Updated flag bits; updated processing of ISD.

6. Removed field edist.

7. Updated the section on preventing the FAI from reaching the Top of Stack.

8. Updated the section on Readable Stack Depth.
1.3. Slice Selector

Network slicing is an important ongoing effort both for network design, as well as for standardization, in particular at the IETF [I-D.nsdt-teas-ns-framework]. A key issue is identifying which slice a packet belongs to, by means of a "slice selector" carried in the packet header. [I-D.bestbar-teas-ns-packet] describes several such methods for MPLS networks, of which the Global Identifier for Slice Selector (GISS) is one of the more practical solutions. This document shows how to realize the GISS using a base special purpose label (bSPL).

In MPLS networks, a GISS is a data plane construct identifying packets belonging to a slice aggregate (the set of packets that belong to the slice). The GISS dictates forwarding actions for the slice aggregate: QoS behavior and next hop selection. The purpose of the GISS is detailed in [I-D.bestbar-teas-ns-packet]. To embed a GISS in a label stack, one must preface it with a bSPL identifying it as such. For reasons that will become apparent, this bSPL is called the Forwarding Actions Indicator (FAI).

2. Multi-purpose bSPL: the Forwarding Actions Indicator

This document proposes the use of a single bSPL to tell routers one or more forwarding actions they should take on a packet, e.g.:

* to treat a packet according to its slice, given its GISS;
* to load balance a packet, given its entropy;
* whether or not to perform fast reroute on a failure [I-D.kompella-mpls-nffrr];
* whether or not a packet has metadata relevant to intermediate hops along the path;
* and perhaps other functions in the future.

This bSPL is called the "Forwarding Actions Indicator" (FAI). There are other suggestions for this name, including "Network Functions Indicator" and "Network Actions Indicator". We’ll let WG consensus determine the final choice of name, but for now, we’ll continue to use FAI.
The FAI uses the label’s TC bits and TTL field to inform the forwarding plane of the required actions. Each of these actions may have associated data. This data may be carried in the label stack as "In-Stack Data" (ISD) or after the label stack as "Post-Stack Data" (PSD).

2.1. The FAI bSPL

The design of the bSPL hinges on two key insights: forwarding engines do not interpret the TC bits or the TTL field for labels that are not at the top of the label stack (ToS); nor do they do so for SPLs. For non-ToS labels, the important bit fields are the label value field (to compute entropy and identify SPLs) and the End of Stack (S) bit (to know when the label stack ends). [If you know of a forwarding engine that looks at other bit fields of labels below the ToS, please contact the authors.] This means that for a bSPL that will never appear at the ToS, the TC bits and the TTL bits can be used to carry additional information. Furthermore, for the ISD, the entire 4-octet label word, the S bit excepted, can be used to carry data. We use this technique to make the FAI bSPL multipurpose, and to make the ISD words compact and efficient.

2.1.1. ISD vs PSD

A pertinent question is when one should put data in the ISD versus in the PSD. One alternative is to put all such data in the PSD. However, this would mean that accessing such information would require finding the End of Stack, and parsing the PSD. For certain types of data, this would be a severe burden on the packet forwarding engine. Examples of such data are the Entropy label (needed for efficient load balancing) and the GISS (needed for accurate packet forwarding). Having any of this data in the PSD would hurt forwarding performance.

This memo suggests that data that is required for accurate and optimal forwarding should be put in the ISD, and data that is optional from a forwarding point of view should be put in the PSD. Furthermore, each flag bit should have no more than one word of associated ISD. The EG flag can thus have up to 2 words of associated data.

By the above criteria, this memo suggests that in-situ OAM data and the Flow ID be carried in the PSD.

2.2. Format of the FAI bSPL
The FAI’s label value MUST be the IANA allocated value. The S bit MUST be set if the FAI is the end of stack, and clear otherwise.

2.2.1. Definitions of the FAI Flag Bits

The TC and TTL bits are used as flags, defined as follows:

- **s**: sISD is present (1) or not (0).
- **u**: uISD is present (1) or not (0).
- **b**: this is the "end of block" bit that indicates the end of the Forwarding Flags Block and the end of the ISD Block.
- **S**: MUST be set if the FAI is the end of stack, and clear otherwise.
h: If set, the PSD contains hop-by-hop information. Every node in the path SHOULD attempt to process the hop-by-hop information, but not at the expense of exceeding the processing time budget, which could cause this (or other) packets to be dropped. If clear, no hop-by-hop data exists in the PSD: either the PSD is empty, or it contains only end-to-end data (to be processed by the egress).

N: If set, do not do fast reroute (NFFRR).

EG: this is a 2-bit flag indicating whether the ISD carries Entropy and/or GISS information.

The FAI Block consists of a Forwarding Flags Block, an sISD Block and a uISD Block. The two ISD Blocks are optional; their presence is indicated by the s and u bits. Each of these three blocks end when the b bit is set.

The Forwarding Flags Block extends from the FAI bSPL up to (and including) the first label that has the b bit set. If the FFB consists of just the bSPL, then its b bit must be set.

The sISD Block extends from the label after the FFB up to (and including) the label with the b bit set. If there is no sISD, the s bit in the FFB MUST be clear.

The uISD Block extends from the label after the sISD Block up to (and including) the label with the b bit set. If there is no uISD, the u bit in the FFB MUST be clear.

The EG field is used as follows:

00: No Entropy or GISS present

01: ISD 0 contains 16 bits of Entropy in the high order 16 bits and 14 bits of GISS in the low order 16 bits (S and b bits excepted).

10: ISD 0 contains 20 bits of Entropy in the high order 20 bits and 10 bits of GISS in the low order 12 bits (S and b bits excepted).

11: ISD 0 contains the 30-bit Entropy; ISD 1 contains the 30-bit GISS. In ISD 0, the S and b bits MUST be 0; the packet forwarding engine may choose to use the S and b bits as part of the Entropy, as it doesn’t affect the outcome. In ISD 1, the S bit may be 0 or 1.
2.2.2. Processing the FAI Flags and the ISD

Here’s how the Standard ISD is parsed. One must keep track of the s bit to know when the Standard ISD Block end, and the S bit to know when the stack ends. The Standard ISD data appears in the order of the corresponding flags.

It is an error if the label stack ends while there are more ISD words to process. In particular, it is an error if the FAI’s S bit is set, but the b bit is clear.

1. If s and u are both 0, done: there is no associated ISD.

2. Set CL ("current label") to the FAI label. LL is the last label (End of Stack); PL ("payload") is the first 4-octet word of the payload.

3. While b is clear:
   1. increment CL

4. Process N. CL is unchanged.

5. If s is set, Standard ISD is present: process standard flags.
   1. Process EG:
      2. If EG is 00, CL is unchanged.
      3. If EG is 01 or 10, increment CL. CL now contains both GISS and Entropy.
      4. If EG is 11, CL+1 contains Entropy; CL+2 contains GISS. Increment CL by 2.
      5. Process other standard data-bearing flags; increment CL by 1 for each.

6. If u is set, uISD is present.
   1. Process uISD until b is set.

Note that how the uISD is used is not defined here; this is up to the user. All that is included here is how a forwarding engine can tell where the uISD block ends.

2.2.3. Example of the FAI
0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
TC b S TTL
-----   ---------------

+-----------------------------+TC |0| TTL
| Tunnel Label-1              |
+-----------------------------+TC |0| TTL

+-----------------------------+TC |0| TTL
| Tunnel Label-2              |
+-----------------------------+TC |0| TTL

+-----------------------------+1|1|0|1|0|1|0|0|0|0
| Forwarding Actions Indicator |
+-----------------------------+1|1|0|1|0|1|0|0|0|0

+-----------------------------+ Entropy [GISS ...1|0| ...
| VPN Label                   |
+-----------------------------+1| TTL
| b b b | PSD
+-----------------------------+ real payload starts ...

s = 1: there is standard ISD.
u = 0: there is no user-defined ISD.
N = 1: NFFRR is set.
EG = 01: ISD 0 contains Entropy + GISS.
h = 1: There is hop-by-hop PSD.

Figure 2: Example of FAI + ISD + hop-by-hop PSD

The real payload starts after the PSD.

3. Issues to be Resolved

This section captures issues to be resolved, in this memo and others. As the issues are fixed, they should be removed from here; ideally, this section should be empty before publication.

3.1. Preventing FAI From Reaching Top of Stack

As was said earlier, the FAI MUST NOT be at the top of stack, since its TC and TTL bits have been repurposed. There are two ways to prevent this. If an LSR X pops a label and the next label is the FAI, X can pop the FAI and all ISD words. This version of the memo introduces the "end-of-block" (s) bit, whereby a forwarding engine that knows the FAI can detect the entire FAI block, even if it doesn't know some of the flags. This can be used in conjunction with Section 3.2.
In case it is desired to preserve the FAI+FAD until the egress, X should push an explicit NULL (label value 0 or 2) onto the stack above the FAI, with the correct TC and TTL values.

Other options may be pursued; however, we believe this is an adequate resolution.

3.2. Repeating the FAI at "Readable Stack Depth"

For LSRs which cannot parse the entire label stack, or would prefer not to unless needed, it is possible to repeat the FAI at "readable stack depth" (rsd). Say the rsd is 10 labels, and the FAI block is 3 labels. Then, the FAI block can be repeated every 7 labels, allowing all forwarding engines in the path to process it. When a forwarding label is popped and the FAI block exposed, it is deleted in its entirety, since the same (or potentially different) FAI block is again within the rsd.

Note that the s or u bits set to 0 can be used to indicate that the corresponding ISD is absent. Only the last FAI would contain the full information, reducing the size of the label stack. However, in this case, LSRs that don’t process the whole stack may not load balance less effectively, and potentially not adhere to the slice service level objectives.

Other options will be described in future versions of this document.

3.3. PSD

The format of the PSD, whether or not a Control Word is present, and handling of the first nibble, is outside the scope of this document. The FAI will not contain details about the contents of the PSD, besides the single flag on whether or not the PSD contains information relevant to (most) intermediate hops. It is assumed that another memo will document the format of the PSD, and that that memo will provide a means of parsing the PSD (e.g., a TLV structure) and thus determining its contents.

The PSD memo should also comment on the impact of processing the PSD on forwarding performance, especially in the case of hop-by-hop info.

4. Contributors

Many thanks to Colby Barth, Chandra Ramachandran and Srihari Sangli for their contributions to this draft.
5. Acknowledgments

We’d like to acknowledge the helpful discussions with Swamy SRK and folks from the Broadcom team on the impacts to existing and future forwarding engines.

The edist field was added thanks to Haoyu Song, who suggested the optimization to find End of Stack.

6. IANA Considerations

If this draft is deemed useful and adopted as a WG document, the authors request the allocation of a bSPL for the FAI. We suggest the early allocation of label 8 for this.

7. Security Considerations

A malicious or compromised LSR can insert the FAI and associated data into a label stack, preventing (for example) FRR from occurring. If so, protection will not kick in for failures that could have been protected, and there will be unnecessary packet loss. Similarly, inserting or removing a Fragmentation Header means that a packet’s contents cannot be accurately reconstructed. Inserting or changing a GISS means that the packet will be misclassified, perhaps leaving or entering a high-value slice and causing damage.

8. References

8.1. Normative References

[I-D.bestbar-teas-ns-packet]

[I-D.kompella-mpls-nffrr]
8.2. Informative References

[I-D.nsdt-teas-ns-framework]

Authors’ Addresses

Kireeti Kompella
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
United States

Email: kireeti.ietf@gmail.com

Vishnu Pavan Beeram
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
United States

Email: vbeeram@juniper.net

Tarek Saad
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
United States

Email: tsaad@juniper.net
Israel Meilik
Broadcom

Email: israel.meilik@broadcom.com
Use Cases for MPLS Network Action Indicators and MPLS Ancillary Data

draft-saad-mpls-miad-usecases-02

Abstract

This document presents a number of use cases that have a common need for encoding network action indicators and associated ancillary data inside MPLS packets. There has been significant recent interest in extending the MPLS data plane to carry such indicators and ancillary data to address a number of use cases that are described in this document.

The use cases described in this document are not an exhaustive set, but rather the ones that are actively discussed by members of the IETF MPLS, PALS and DETNET working groups participating in the MPLS Open Design Team.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 22 October 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.
Table of Contents

1. Introduction ................................................. 2
   1.1. Terminology ............................................. 3
   1.2. Acronyms and Abbreviations .............................. 3
2. Use Cases ................................................... 3
   2.1. No Further Fastreroute .................................. 3
   2.2. In-situ OAM ............................................... 4
   2.3. Network Slicing .......................................... 4
       2.3.1. Global Identifier as Flow-Aggregate Selector .... 5
       2.3.2. Forwarding Label as a Flow-Aggregate Selector ... 5
   2.4. Delay Budgets for Time-Bound Applications ............... 6
       2.4.1. Stack Based Methods for Latency Control ......... 7
   2.5. NSH-based Service Function Chaining ..................... 7
   2.6. Network Programming ................................... 7
   2.7. Application Aware Networking ........................... 8
3. Co-existence of Usecases ..................................... 8
4. IANA Considerations ......................................... 8
5. Security Considerations ..................................... 8
6. Acknowledgement ............................................ 9
7. Contributors ............................................... 9
8. Informative References ..................................... 9
Authors’ Addresses ............................................ 12

1. Introduction

This document describes important cases that require carrying additional ancillary data within the MPLS packets, as well as means to indicate the ancillary data is present, and a specific action needs to be performed on the packet.

These use cases have been identified by the MPLS Working Group Open Design Team working on defining MPLS Network Actions for the MPLS data plane. The MPLS Ancillary Data (AD) can be classified as:

* implicit, or "no-data" associated with a Network Action (NA) indicator,
* residing within the MPLS label stack and referred to as In Stack Data (ISD), and
* residing after the Bottom of MPLS label Stack (BoS) and referred to as Post Stack Data (PSD).

The use cases described in this document will be used to assist in identifying requirements and issues to be considered for future resolution by the working group.

1.1. Terminology

The following terminology is used in the document:

**IETF Network Slice:**
a well-defined composite of a set of endpoints, the connectivity requirements between subsets of these endpoints, and associated requirements; the term 'network slice' in this document refers to 'IETF network slice' as defined in [I-D.ietf-teas-ietf-network-slices].

**Time-Bound Networking:**
Networks that transport time-bounded traffic.

1.2. Acronyms and Abbreviations

ISD: In-stack data
PSD: Post-stack data
MNA: MPLS Network Action
NAI: Network Action Indicator
AD: Ancillary Data

2. Use Cases

2.1. No Further Fast reroute

MPLS Fast Reroute (FRR) [RFC4090], [RFC5286] and [RFC7490] is a useful and widely deployed tool for minimizing packet loss in the case of a link or node failure.

Several cases exist where, once FRR has taken place in an MPLS network and resulted in rerouting a packet away from the failure, a second FRR that impacts the same packet to rerouting is not helpful, and may even be disruptive.
For example, in such a case, the packet may continue to loop until its TTL expires. This can lead to link congestion and further packet loss. Thus, the attempt to prevent a packet from being dropped may instead affect many other packets. A proposal to address this is presented in [I-D.kompella-mpls-nffrr].

2.2. In-situ OAM

In-situ Operations, Administration, and Maintenance (IOAM) may record operational and telemetry information within the packet while the packet traverses a particular path in a network domain.

The term "in-situ" refers to the fact that the IOAM data fields are added to the data packets rather than being sent within the probe packets specifically dedicated to OAM or Performance Measurement (PM).

IOAM can run in two modes Edge-to-Edge (E2E) and Hop-by-Hop (HbH). In E2E mode, only the encapsulating and decapsulating nodes will process IOAM data fields. In HbH mode, the encapsulating and decapsulating nodes as well as intermediate IOAM-capable nodes process IOAM data fields.

The IOAM data fields are defined in [I-D.ietf-ippm-ioam-data], and can be used for various use-cases of OAM and PM.

[I-D.gandhi-mpls-ioam-sr] defines how IOAM data fields are transported using the MPLS data plane encapsulations, including Segment Routing (SR) with MPLS data plane (SR-MPLS).

The IOAM data may be added after the bottom of the MPLS label stack. The IOAM data fields can be of fixed or incremental size as defined in [I-D.ietf-ippm-ioam-data]. [I-D.gandhi-mpls-ioam] describes the applicability of IOAM to MPLS dataplane. The encapsulating MPLS node needs to know if the decapsulating MPLS node can process the IOAM data before adding it in the packet. In HbH IOAM mode, nodes that are capable of processing IOAM will intercept and process the IOAM data accordingly. The presence of IOAM header and optional IOAM data will be transparent to nodes that do not support or do not participate in the IOAM process.

2.3. Network Slicing

[I-D.ietf-teas-ietf-network-slices] specifies the definition of an IETF Network Slice. It further discusses the general framework for requesting and operating IETF Network Slices, their characteristics, and the necessary system components and interfaces.
Multiple network slices can be realized on top of a single physical network.

In order to overcome scale challenges, IETF Network Slices may be aggregated into groups according to similar characteristics. The slice aggregate [I-D.bestbar-teas-ns-packet] is a construct that comprises of the traffic flows of one or more IETF Network Slices of similar characteristics.

A router that requires forwarding of a packet that belongs to a slice aggregate may have to decide on the forwarding action to take based on selected next-hop(s), and the forwarding treatment (e.g., scheduling and drop policy) to enforce based on the associated per-hop behavior.

The routers in the network that forward traffic over links that are shared by multiple slice aggregates need to identify the slice aggregate packets in order to enforce the associated forwarding action and treatment.

An IETF network slice MAY support the following key features:

1. A Slice Selector
3. A Path selection criteria
4. Verification that per slice Slice Level Objectives (SLOs) are being met. This may be done by active measurements (inferred) or by using hybrid measurement methods, e.g., IOAM.
5. Additionally, there is an on-going discussion on using Service Functions (SFs) with network slices. This may require insertion of an NSH.
6. For multi-domain scenarios, a packet that traverses multiple domains may encode different identifiers within each domain.

2.3.1. Global Identifier as Flow-Aggregate Selector

A Global Identifier as a Flow-Aggregate Selector (G-FAS) can be encoded in the MPLS packet as defined in [I-D.kompella-mpls-mspl4fa], [I-D.li-mpls-enhanced-vpn-vtn-id], and [I-D.decraene-mpls-slid-encoded-entropy-label-id]. The G-FAS is used to associate the packets belonging to Slice-Flow Aggregate to the underlying Network Resource Partition (NRP) as described in [I-D.bestbar-teas-ns-packet].
The G-FAS can be encoded within an MPLS label carried in the packet’s MPLS label stack. All packets that belong to the same flow aggregate MAY carry the same FAS in the MPLS label stack.

When MPLS packets carry a G-FAS, MPLS LSRs use the forwarding label to select the forwarding next-hop(s), and use the G-FAS in the MPLS packet to infer the specific forwarding treatment that needs to be applied on the packet.

2.3.2. Forwarding Label as a Flow-Aggregate Selector

[RFC3031] states in Section 2.1 that: ‘Some routers analyze a packet’s network layer header not merely to choose the packet’s next hop, but also to determine a packet’s "precedence" or "class of service"’.

It is possible by assigning a unique MPLS forwarding label to each flow aggregate (FEC) to distinguish the packets forwarded to the same destination from other flow aggregates. In this case, LSRs can use the top forwarding label to infer both the forwarding action and the forwarding treatment to be invoked on the packets.

2.4. Delay Budgets for Time-Bound Applications

The routers in a network can perform two distinct functions on incoming packets, namely forwarding (where the packet should be sent) and scheduling (when the packet should be sent). IEEE-802.1 Time Sensitive Networking (TSN) and Deterministic Networking provide several mechanisms for scheduling under the assumption that routers are time-synchronized. The most effective mechanisms for delay minimization involve per-flow resource allocation.

Segment Routing (SR) is a forwarding paradigm that allows encoding forwarding instructions in the packet in a stack data structure, rather than being programmed into the routers. The SR instructions are contained within a packet in the form of a First-in First-out stack dictating the forwarding decisions of successive routers. Segment routing may be used to choose a path sufficiently short to be capable of providing a bounded end-to-end latency but does not influence the queueing of individual packets in each router along that path.

When carried over the MPLS data plane, a solution is required to enable the delivery of such packets that can be delivered to their final destination by a given time budget.
2.4.1. Stack Based Methods for Latency Control

One efficient data structure for inserting local deadlines into the headers is a "stack", similar to that used in Segment Routing to carry forwarding instructions. The number of deadline values in the stack equals the number of routers the packet needs to traverse in the network, and each deadline value corresponds to a specific router. The Top-of-Stack (ToS) corresponds to the first router’s deadline while the Bottom-of-Stack (BoS) refers to the last’s. All local deadlines in the stack are later or equal to the current time (upon which all routers agree), and times closer to the ToS are always earlier or equal to times closer to the BoS.

The ingress router inserts the deadline stack into the packet headers; no other router needs to be aware of the requirements of the time-bound flows. Hence admitting a new flow only requires updating the information base of the ingress router.

MPLS LSRs that expose the Top of Stack (ToS) label can also inspect the associated "deadline" carried in the packet (either in MPLS stack as ISD or after BoS as PSD).

2.5. NSH-based Service Function Chaining

[RFC8595] describes how Service Function Chaining (SFC) can be realized in an MPLS network by emulating the NSH by using only MPLS label stack elements.

The approach in [RFC8595] introduces some limitations that are discussed in [I-D.1m-mpls-sfc-path-verification]. This approach, however, can benefit from the framework introduced with MNA [I-D.andersson-mpls-mna-fwk].

For example, it may be possible to extend NSH emulation using MPLS labels [RFC8595] to support the functionality of NSH Context Headers, whether fixed or variable-length. One of the use cases could support Flow ID [I-D.ietf-sfc-nsh-tlv] that may be used for load-balancing among Service Function Forwarders (SFFs) and/or the Service Function (SF) within the same SFP.

2.6. Network Programming

In SR, an ingress node steers a packet through an ordered list of instructions, called "segments". Each one of these instructions represents a function to be called at a specific location in the network. A function is locally defined on the node where it is executed and may range from simply moving forward in the segment list to any complex user-defined behavior.
Network Programming combines Segment Routing (SR) functions to achieve a networking objective that goes beyond mere packet routing.

It may be desirable to encode a pointer to function and its arguments within an MPLS packet transport header. For example, in MPLS we can encode the FUNC::ARGs within the label stack or after the Bottom of Stack to support the equivalent of FUNC::ARG in SRv6 as described in [RFC8986].

2.7. Application Aware Networking

Application-aware Networking (APN) as described in [I-D.li-apn-problem-statement-usecases] allows application-aware information (i.e., APN attributes) including APN identification (ID) and/or APN parameters (e.g., network performance requirements) to be encapsulated at network edge devices and carried in packets traversing an APN domain.

The APN data is carried in packets to facilitate service provisioning, and be used to perform fine-granularity traffic steering and network resource adjustment. To support APN in MPLS networks, mechanisms are needed to carry such APN data in MPLS encapsulated packets.

3. Co-existence of Usecases

Two or more of the aforementioned use cases MAY co-exist in the same packet. This may require the presence of multiple ancillary data (whether In-stack or Post-stack ancillary data) to be present in the same MPLS packet.

For example, IOAM may provide key functions along with network slicing to help ensure that critical network slice SLOs are being met by the network provider. In this case, IOAM is able to collect key performance measurement parameters of network slice traffic flows as it traverses the transport network.

4. IANA Considerations

This document has no IANA actions.

5. Security Considerations

This document introduces no new security considerations.
6. Acknowledgement

The authors gratefully acknowledge the input of the members of the MPLS Open Design Team.

7. Contributors

The following individuals contributed to this document:

Loa Andersson
Bronze Dragon Consulting
Email: loa@pi.nu

8. Informative References

[I-D.andersson-mpls-mna-fwk]

[I-D.bestbar-teas-ns-packet]

[I-D.decraene-mpls-slid-encoded-entropy-label-id]

[I-D.gandhi-mpls-ioam]
[I-D.gandhi-mpls-ioam-sr]

[I-D.ietf-ippm-ioam-data]

[I-D.ietf-sfc-nsh-tlv]

[I-D.ietf-teas-ietf-network-slices]

[I-D.kompella-mpls-mspl4fa]

[I-D.kompella-mpls-nffrr]

[I-D.li-apn-problem-statement-ucases]
Li, Z., Peng, S., Voyer, D., Xie, C., Liu, P., Qin, Z., and G. Mishra, "Problem Statement and Use Cases of

[I-D.li-mpls-enhanced-vpn-vtn-id]

[I-D.lm-mpls-sfc-path-verification]


Authors’ Addresses

Tarek Saad
Juniper Networks
Email: tsaad@juniper.net

Kiran Makhijani
Futurewei Technologies
Email: kiranm@futurewei.com

Haoyu Song
Futurewei Technologies
Email: haoyu.song@futurewei.com

Greg Mirsky
Ericsson
Email: gregimirsky@gmail.com