Abstract

This draft specifies requirements for indicators in the MPLS label stack of ancillary data that exists below the label stack. 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, based on application data that may be in or below the bottom of the label stack.

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 RFC 2119 [RFC2119].

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 April 17, 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.
1. Introduction

There is significant interest in developing the MPLS data plane to address the requirements of new applications. These applications typically include ancillary data that is contained in or below the label stack. There is a requirement for this data to be either intercepted and processed, or some other forwarding decision to be made. This makes use of mechanisms implemented by an intermediate or egress label switching router (LSR) that complies 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.

1.1. Terminology

- Ancillary Data: Data relating to the MPLS packet that may be used to affect the forwarding or other processing of that packet, either at the LER or LSR. This data may be implicit (i.e. context-specific), encoded within the label stack (in-stack data), after the bottom of the label stack but not considered a part of the payload, or within the payload.
o In-Stack: Any location within the MPLS label stack including the outer label and the bottom of stack (the label with the S-bit set).

o Ancillary Data Indicator (ADI): A indicator in the MPLS label stack that ancillary data exists in this packet. It may also indicate the specific type of the ancillary data.

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 as 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 [Reference to MIAD use cases]. [I-D.song-mpls-extension-header] summarises some of the issues with existing solutions to address these new applications:

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.

2. MPLS Ancillary Data Indicator Requirements

This document specifies requirements of MPLS Indicators for Ancillary Data (MIAD). The requirements are for the behavior of the protocol mechanisms and procedures that constitute building blocks out of which mechanisms for indicating ancilliary data that exists in the MPLS payload using the MPLS label stack (so-called in-stack indicators) are constructed. It does not specify the detailed processing that may be required by an application of that ancilliary data by an LSR. The requirements in this document do not describe what functions MIAD implementation supports. 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.

2.1. General Requirements

- 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.

- Any solutions to these requirements MUST not restrict the generality of MPLS architecture.
o Any solution MUST respect the principle that Special Purpose Labels are the mechanism of last resort.

o Solutions MUST be able to coexist with and not obsolete existing MPLS mechanisms.

o Neither an ADI or ancilliary data must be delivered to a node that is not capable of processing it.

o Care needs to be taken in the coexistence of ancillary data and existing post-stack data mechanisms.

o Mechanisms are required to determine that all nodes that need to process the ancillary data can read the required distance into the packet at that node.

o A mechanism is REQUIRED for Ancilliary Data Indicators to indicate the presence of ancilliary data in the MPLS label stack (Ed. note: This is similar to ELI).

o A mechanism is REQUIRED for Ancilliary Data Indicators to indicate the presence of ancilliary data below the MPLS label stack (Ed. note: this is similar to GAL/G-ACH).

o The mechanism to indicate that Ancilliary Data is present MUST operate in the context of the top of stack LSE.

o Ancilliary data may be associated with control or maintenance information for traffic carried by an LSP, or it may be associated with the user traffic itself.

o Ancilliary Data Indicators (ADIs) SHOULD make use of existing MPLS data plane operations. If extensions to the MPLS data plane are required, they MUST NOT be inconsistent with the MPLS architecture.

o A mechanism is REQUIRED to enable an LER inserting ADIs to determine whether LSRs along the path can parse the label stack and process the ADI at the location it is inserted.

o A mechanism is REQUIRED to enable an LER inserting ADIs to determine if the ADI will be processed by LSRs along the path.

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

o ADIs SHOULD be supported for both P2P and P2MP paths, but any specific ADI may only be supported for one or the other.
Data plane mechanisms for ADIs MUST be independent of the control plane type (LDP, RSVP, BGP, static, IGP, etc).

A mechanism MUST be defined for control planes (LDP, RSVP, BGP, static, IGP, etc) to determine the ability of downstream LSRs/LERs to accept/process a given ADI.

A mechanism is REQUIRED to enable an LSR to efficiently determine if an ADI is present in a packet.

ADIs can only be inserted at LERs, but may be processed at LSRs and LERs. If it is required to insert an ADI at a transit router on an LSP, then a new label stack must be pushed.

It SHOULD be possible to include indicators for ancillary data for multiple applications in the same packet, but each ADI only supports one application.

It MUST be possible to insert new ADIs for new applications on the same LSP. [Ed note: neet to clarify]

The solution must allow ADI and non-ADI packets to coexist on the same LSP.

The solution must support the processing of a subset of the ADIs on a packet.

The solution MUST support slow path processing of ancilliary data.

The solution MUST support fast path processing of ancilliary data.

The solution MUST support hop-by-hop processing of ancilliary data.

The solution MUST support end-to-end processing of ancilliary data.

If both hop-by-hop and end-to-end ancilliary data indicators are present together, the precedence must be specified in the design.

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

A solution must be provided to verify the authenticity of ancillary data processed to LSRs.
The design of the ADIs and ancillary data must not expose confidential information to the LSRs.

3. IANA Considerations

This document makes no request of IANA.

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

4. Security Considerations

The mechanisms required by this document introduce new security considerations to MPLS. It is expected that individual solutions meeting these requirements will address any security considerations.

5. Acknowledgements

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

6. References

6.1. Normative References


6.2. Informative References

[I-D.bryant-mpls-dev-primer]
Bryant, S., "A Primer on the Development of MPLS", draft-bryant-mpls-dev-primer-00 (work in progress), March 2021.

[I-D.song-mpls-extension-header]


Authors’ Addresses

Matthew Bocci
Nokia

Email: matthew.bocci@nokia.com

Stewart Bryant
University of Surrey 5GIC

Email: sb@stewartbryant.com
IANA Registry for the First Nibble Following a Label Stack
draft-kbbma-mpls-1stnibble-00

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 April 28, 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 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 to be 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
USA

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
Multi-purpose Special Purpose Label for Forwarding Actions
draft-kompella-mpls-mspl4fa-01

Abstract

A Slice Selector is packet metadata that dictates the packet’s forwarding handling in order to conform to its slice requirements. There are multiple proposals for carrying slice selectors in MPLS networks. One of the more practical proposals is the "Global Identifier for Slice Selector" (GISS). Global uniqueness requires the GISS label be identified as such, via a special purpose label (ideally a base special purpose label (bSPL)). However, bSPLs are a precious commodity, and there are many requests for them. This document serves two purposes: to define a bSPL for carrying a GISS, and to show how this bSPL can consolidate many current requests for special purpose labels while carrying associated data compactly and efficiently.

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 January 13, 2022.
1. Introduction

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).

Base Special Purpose Labels 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.  Therefore, this document also suggests a method whereby a single bSPL can be used for all the purposes currently documented.  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 and efficient means of carrying associated data.

1.1.  Terminology

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.

1.2.  Revision History

This section (to be removed before publication) highlights 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.

1.3. Slice Selector

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 a Flow ID;
- whether or not a packet has metadata relevant to intermediate hops along the path;
- a faster way of finding the End of Stack;
- and perhaps other functions in the future.

This bSPL is called the "Forwarding Actions Indicator" (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, the Forwarding Actions Data (FAD). The FAD may be carried in the Label Stack (LS FAD) or in the payload (PL FAD).

2.1. The FAI bSPL

The design of the bSPL hinges on a key insight: 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). 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 LS FAD, 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 LS FAD words compact and efficient.

2.1.1. LS FAD vs PL FAD

A pertinent question is whether one should put non-label data in the label stack. The alternative is to put all such data in the PL FAD. However, this would mean that accessing such information would require finding the End of Stack, and parsing the PL FAD. 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), the GISS (needed for accurate packet forwarding) and the Flow ID (needed for telemetry). Having any of this data in the PL FAD would hurt forwarding performance.

This memo will document criteria for when data should be in the LS FAD versus in the PL FAD.

2.2. Format of the FAI bSPL
The FAI's label value MUST be the IANA allocated value. The S bit MUST be reflect whether the label stack ends at this label or not.

2.2.1. Definitions of the FAI Flag Bits

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

H: if set, the FAI is followed by a Forwarding Actions Header (FAH).

A: Associated data (LS FAD) is present (1) or not (0).

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

S: MUST be set if the FAI is the end of stack, and clear otherwise.

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

F: If set, the LS FAD has a Flow ID.

h: If set, the PL FAD 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.
edist: ("distance to End of Stack") a 4-bit field that indicates how many 4-octets labels to skip to reach End of Stack.

The FAH consists of a single 4-octet word, and is used if more FAD flags are needed. As these bits are defined, processing of the associated data MUST also be defined. The format of the FAH is TBD.

The EG field is used as follows:

00: No Entropy or GISS present

01: LS FAD 0 contains 16 bits of Entropy in the high order 16 bits and 15 bits of GISS in the low order 16 bits (S bit excepted).

10: LS FAD 0 contains 20 bits of Entropy in the high order 20 bits and 11 bits of GISS in the low order 12 bits (S bit excepted).

11: LS FAD 0 contains the 31-bit Entropy; LS FAD 1 contains the 31-bit GISS. In LS FAD 0, the S bit MUST be 0; the packet forwarding engine may choose to use this as part of the Entropy, as it doesn’t affect the outcome. In LS FAD 1, the S bit may be 0 or 1.

2.2.2. Processing the FAI Flags and the LS FAD

Here’s how the LS FAD is parsed. One must keep track of the S bit to know when the stack ends. The LS FAD data appears in the order of the corresponding flags.

It is an error if the label stack ends while there are more LS FAD words to process. In particular, it is an error if the FAI’s S bit is set, but the H flag is set, or the A flag is set and any of EG or F or edist is non-zero.

It is not an error if H, A, N, EG, F and h are all zero; however, in that case, it’s not clear what purpose the FAI serves.

1. 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.

2. Process H:
   1. If set, increment CL; process the FAH.
   2. Otherwise, CL is unchanged.

3. If A is 0, done: there is no associated LS FAD.
4. Process N. CL is unchanged.

5. Process EG:
   1. If EG is 00, CL is unchanged.
   2. If EG is 01 or 10, increment CL. CL now contains both GISS and Entropy.
   3. If EG is 11, CL+1 contains Entropy; CL+2 contains GISS.
      Increment CL by 2.

6. Process F:
   1. If F is set; increment CL. CL contains the Flow ID.

7. Process edist:
   1. LL = CL + edist
   2. while LL's S bit == 0, LL++
   3. PL = LL+1

The edist field is used to expedite reaching the PL FAD (e.g., to process hop-by-hop information). A forwarding engine can skip forward edist 4-octet words, i.e., CL += edist. This word MUST be a label, which may or may not have S = 1. If not, keep parsing until a label with S = 1 is hit; this is the End of Stack. PL FAD follows this label.

Details for parsing the PL FAD are outside the scope of this memo, and will be addressed in the document describing its format.

2.2.3. Example of the FAI
### 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 encounters an FAI, X can pop the FAI and all LS FAD words. To do that, it must be able to parse the FAI to determine how many LS FAD words there exist. This can be used in conjunction with Section 3.2. However, there are cases when it is desired to preserve the FAI+FAD until the egress. In this case, X should push an explicit NULL (label value 0 or 2) onto the stack above the FAI, with the correct TC and TTL values.

---

### Figure 2: Example of FAI + LS FAD + hop-by-hop PL FAD

The real payload starts after the PL FAD.
Other options will be pursued.

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", say every 4 labels. In the above case, the FAI+LS FAD can be repeated every 4 labels; or a truncated version, just the FAI with A set to 0 can be used. Only the last FAI would contain the full information, reducing the burden on 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. PL FAD

The format of the PL FAD, 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 PL FAD, besides the single flag on whether or not the PL FAD contains information relevant to (most) intermediate hops. It is assumed that another memo will document the format of the PL FAD, and that this memo will provide a means of parsing the PL FAD (e.g., a TLV structure) and thus determining its contents.

The PL FAD memo should also comment on the impact of processing the PL FAD 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
Deterministic Networking (DetNet): OAM Functions for The Service Sub-Layer
draft-varga-detnet-service-sub-layer-oam-01

Abstract

Operation, Administration, and Maintenance (OAM) tools are essential for a deterministic network. The DetNet architecture [RFC8655] has defined two sub-layers: (1) DetNet service sub-layer and (2) DetNet forwarding sub-layer. OAM mechanisms exist for the DetNet forwarding sub-layer. Nonetheless, OAM for the service sub-layer might require new extensions to the existing OAM protocols. This draft presents an analysis of OAM procedures for the DetNet service sub-layer functions.

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 25 April 2022.

Copyright Notice

Copyright (c) 2021 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
1. Introduction

The DetNet Working Group has defined two sub-layers: (1) DetNet service sub-layer, at which a DetNet service (e.g., service protection) is provided and (2) DetNet forwarding sub-layer, which optionally provides resource allocation for DetNet flows over paths provided by the underlying network. In [RFC8655] new DetNet-specific functions have been defined for the DetNet service sub-layer, namely PREOF (a collective name for Packet Replication, Elimination, and Ordering Functions).
Framework of Operations, Administration and Maintenance (OAM) for Deterministic Networking (DetNet) is described in [I-D.ietf-detnet-oam-framework]. OAM for the DetNet MPLS data plane is described in [I-D.ietf-detnet-mpls-oam] and OAM for the DetNet IP data plane is described in [I-D.ietf-detnet-mpls-oam].

This draft has been submitted as an individual contribution to OAM discussions, in particular, to kick-off Working Group discussions on introducing OAM functions for the DetNet service sub-layer. It is also up to the Working Group discussions to which draft parts of this contribution may go, if any.

The OAM functions for the DetNet service sub-layer allow, for example, to recognize/discover DetNet relay nodes, to get information about their configuration, and to check their operation or status. Furthermore, the OAM functions for the DetNet service sub-layer need to meet new challenges (see section Section 3) and requirements (see section Section 4) introduced by PREOF.

An approach described in this draft introduces a new OAM shim layer to achieve OAM for the DetNet service sub-layer. In the rest of the draft, this approach is referred to as "DetNet PING", which is an in-band OAM approach, i.e., the OAM packets follow precisely the same path as the data packets of the corresponding DetNet flow(s). The OAM packets provide DetNet service sub-layer specific information, like:

* Identity of a DetNet service sub-layer node.
* Discover Ingress/Egress flow-specific configuration of a DetNet service sub-layer node.
* Detect the status of the flow-specific service sub-layer function.

DetNet PING applies both to IP and MPLS DetNet data planes.

2. Terminology

2.1. Terms Used in This Document

This document uses the terminology established in the DetNet architecture [RFC8655], and the reader is assumed to be familiar with that document and its terminology.

2.2. Abbreviations

The following abbreviations are used in this document:

DetNet Deterministic Networking.
2.3. 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. DetNet Service Sub-layer OAM Challenges

3.1. Illustrative example

This section introduces an example that is used to explain the DetNet Service Sub-layer OAM challenges. Figure 1 shows a DetNet flow on which PREOF functions are applied during forwarding from source to destination.
DetNet service sub-layer nodes are interconnected by DetNet forwarding sub-layer paths. DetNet forwarding sub-layer path (e.g., P1 = R1->E1 path, P4 = E1->R3 path) may contain multiple transit nodes. A DetNet forwarding sub-layer path is used by a member flow and terminated by relay nodes (see [RFC8655] for relay node definition).

A DetNet service sub-layer graph includes all relay nodes and the interconnecting forwarding sub-layer paths. This graph can be also called as "PREOF graph" and it describes the compound flow as a whole.

3.2. DetNet Service Sub-layer Specifics for OAM

Several DetNet Service Sub-layer specifics have to be considered for OAM.

1. The service sub-layer graph is segmented into multiple parts, as forwarding sub-layer paths are terminated at DetNet relay nodes.
2. These are particular characteristics of DetNet PW:

1. PREOF acts as per-packet protection. PEF is a brand-new functionality at network layer, due to the per-packet merge action.

2. All paths are active and forward traffic. These paths may have a different number of hops.

3. Mandatory usage of a sequence number.

The above specifics have to be considered in combination with the requirement that DetNet OAM and DetNet data flows MUST receive the same treatment. OAM packets MUST follow precisely the same graph as the monitored DetNet flow(s).

3.3. Information Needed during DetNet OAM Packet Processing

This section collects some questions that have been already discussed by the DetNet WG and/or require further discussions by the WG. The section is structured in the form of a question list.

Question-1: Injecting OAM traffic in a DetNet flow? A DetNet data flow has a continuous Sequence Number. In order not to spoil that, the injected OAM packets require OAM-specific Sequence Number added. (See also Section Section 5.)

Question-2: How to process OAM packets by DetNet service sub-layer nodes? In order to cover the DetNet forwarding graph by OAM, PREOF has to be executed in an OAM specific manner (i.e., PREOF uses a separate SeqNum space for OAM. See details in Section 5.

Note: the question list is non-exhaustive.

3.4. A Possible Format of DetNet Associated Channel Header (d-ACH)

Figure 2 shows a possible format of the DetNet Associated Channel Header (d-ACH).
The d-ACH encodes the following fields:

* Bits 0..3 MUST be 0b0001. This value of the first nibble allows the packet to be distinguished from an IP packet [RFC4928] and a DetNet data packet [RFC8964].

* Version: this is the version number of the d-ACH. This specification defines version 0x1.

* Sequence Number: this is an unsigned eight-bit field. The sequence number space is a circular one with no restriction on the initial value. The originator DetNet node MUST set the value of the Sequence Number field before the transmission of a packet. The originator node MUST increase the value of the Sequence Number field by 1 for each active OAM packet.

* Channel Type: encodes the value of DetNet Associated Channel Type is one of the values defined in the IANA PW Associated Channel Type registry.

* Node ID: it is an unsigned 20-bits field. The value of the Node ID field identifies the DetNet node that originated the packet. Methods to distribute Node ID are outside the scope of this specification.

* Level: is a 3-bit field.

* Flags: is a 5-bit field. Section 7.1 creates an IANA registry for new flags to be defined.

* Session ID: is a 4-bit field.

The DetNet flow, according to [RFC8964], is identified by the S-label that MUST be at the bottom of the stack. An active DetNet OAM packet MUST include d-ACH immediately following the S-label.

4. Requirements on OAM for DetNet Service Sub-layer

[Editor’s note: The content of this section has been discussed and the outcome of the discussion has been documented in [I-D.ietf-detnet-oam-framework].]

The requirements on OAM for a DetNet relay node are:

1. to provide OAM functions for the DetNet service sub-layer.
2. to discover DetNet relay nodes in a DetNet network.

3. to collect DetNet service sub-layer specific (e.g., configuration/operation/status) information from DetNet relay nodes.

4. to work for both DetNet data planes: (1) MPLS and (2) IP.

5. DetNet PING

5.1. Overview

The "DetNet PING" approach uses two types of OAM packets: (1) DetNet-Echo-Request and (2) DetNet-Echo-Reply. Their encapsulation is identical to that of the corresponding DetNet data flow, so they follow precisely the same path as the packets of the corresponding DetNet data flow. They target DetNet service sub-layer entities, so they may not be recognized as OAM packets by entities not implementing DetNet service sub-layer for a packet flow (e.g., transit nodes). Other entities treat them as packets belonging to the corresponding DetNet data flow.

The following relay node roles can be distinguished:

1. DetNet PING originator node,

2. Intermediate DetNet service sub-layer node,

3. DetNet PING targeted node.

An originator node sends (generates) DetNet-Echo-Request packet(s). DetNet-Echo-Request packet contains an OAM specific "PINGSeqNum", which can be used by the DetNet service sub-layer of relay nodes. Note that "PINGSeqNum" is originator specific.

An intermediate DetNet service sub-layer node executes DetNet flow-specific service sub-layer functionality. Packet processing may be done in an OAM specific manner (see details in Section 5.2).

A targeted node answers with DetNet-Echo-Reply packet for each DetNet-Echo-Request. DetNet-Echo-Reply packet provides DetNet service sub-layer specific information on (i) identities of DetNet service sub-layer node (e.g., Node-ID, local Flow-ID), (ii) ingress/egress flow related configuration (e.g., in/out member flow specific information (including forwarding sub-layer specifics)), and (iii) status of service sub-layer function (e.g., local PxF-ID, ActionType=x, operational status, value of key state variable(s), function related related counters).
5.2. OAM processing at the DetNet service sub-layer

Detailed OAM packet processing rules of various DetNet relay nodes are described in the following sections.

5.2.1. Relay node with PRF

A DetNet relay node with PRF processes DetNet OAM packets in a stateless manner.

If the relay node with PRF is the target of a DetNet-Echo-Request packet, then the DetNet-Echo-Request packet MUST NOT be further forwarded, and a DetNet Echo-Reply packet MUST be generated. If the relay node with PRF is not the target of a DetNet Echo-Request packet, then the DetNet Echo-Request packet MUST be sent over all DetNet flow specific member flow paths (i.e., it is replicated).

A DetNet Echo-Reply packet MUST contain the following information:

* Identities related to the DetNet service sub-layer node (e.g., Node-ID, local Flow-ID),

* Ingress/Egress flow related configuration (e.g., in/out member flow specific information (including forwarding sub-layer specifics)),

* Status of service sub-layer function (e.g., local PRF-ID, Action-Type=Replication, operational status, value of the flow related key state variable (e.g., "GenSeqNum" in [IEEE8021CB]).

A DetNet Echo-Reply packet MAY contain the following information:

* PRF related local counters.

5.2.2. Relay node with PEF

A DetNet relay node with PEF processes DetNet OAM packets in a stateful manner.

If the relay node with PEF is the target of DetNet-Echo-Request packet, then the DetNet Echo-Request packet MUST NOT be further forwarded and an DetNet Echo-Reply packet MUST be generated. If the relay node with PEF is not the target of DetNet Echo-Request packet, then elimination MUST be executed on the DetNet Echo-Request packet(s) using the OAM specific "PINGSeqNum" in the packet. So only a single DetNet Echo-Request packet is forwarded and all further replicas (having the same originator's sequence number) MUST be discarded.
A DetNet-Echo-Reply packet MUST contain the following information:

* Identities related to the DetNet service sub-layer node (e.g., Node-ID, local Flow-ID),
* Ingress/Egress flow related configuration (e.g., in/out member flow specific information (including forwarding sub-layer specifics)),
* Status of service sub-layer function (e.g., local PEF-ID, Action-Type=Elimination, operational status, value of the flow related key state variable (e.g., "RecovSeqNum" in [IEEE8021CB]).

A DetNet Echo-Reply packet MAY contain the following information:

* PEF-related local counters.

5.2.3. Relay node with POF

A DetNet relay node with POF processes DetNet OAM packets in a stateless manner.

If the relay node with POF is the target of DetNet Echo-Request packet, then the DetNet Echo-Request packet MUST NOT be further forwarded and a DetNet Echo-Reply packet MUST be generated. If the relay node with POF is not the target of DetNet-Echo-Request packet, then the DetNet Echo-Request packet(s) MUST be forwarded without any POF-specific action.

A DetNet Echo-Reply packet MUST contain the following information:

* Identities of the DetNet service sub-layer node (e.g., Node-ID, local Flow-ID),
* Ingress/Egress flow related configuration (e.g., in/out member flow specific information (including forwarding sub-layer specifics)),
* Status of service sub-layer function (e.g., local POF-ID, Action-Type=Ordering, operational status, value of the flow related key state variable (e.g., "POFLastSent" in [I-D.varga-detnet-pof]).

A DetNet Echo-Reply packet MAY contain the following information:
* POF-related local counters.

5.2.4. Relay node without PREOF

A DetNet relay node without PREOF processes DetNet OAM packets in a stateless manner.

If the relay node without PREOF is the target of DetNet Echo-Request packet, then the DetNet Echo-Request packet MUST NOT be further forwarded and an DetNet Echo-Reply packet MUST be generated. If the relay node without PREOF is not the target of DetNet-Echo-Request packet, then the DetNet-Echo-Request packet(s) MUST be forwarded (as any data packets of the related DetNet flow).

A DetNet Echo-Reply packet MUST contain the following information:

* Identities of the DetNet service sub-layer node (e.g., Node-ID, local Flow-ID),

* Ingress/Egress flow-related configuration (e.g., in/out member flow specific information (including forwarding sub-layer specifics)) .

6. Security Considerations

Tbd.

7. IANA Considerations

7.1. DetNet MPLS OAM Flags Registry

This document describes a new IANA-managed registry to identify DetNet MPLS OAM Flags Bits. The registration procedure is "IETF Review" [RFC8126]. The registry name is "DetNet MPLS OAM Flags". There are five flags in the five-bit Flags field.

8. Acknowledgements

Authors extend their appreciation to Janos Szabo and Gyorgy Miklos for their insightful comments and productive discussion that helped to improve the document.

9. References

9.1. Normative References
9.2. Informative References

[I-D.ietf-detnet-ip-oam]

[I-D.ietf-detnet-mpls-oam]


Authors’ Addresses

Balázs Varga
Ericsson
Budapest
Magyar Tudosok krt. 11.
1117
Hungary

Email: balazs.a.varga@ericsson.com

János Farkas
Ericsson
Budapest
Magyar Tudosok krt. 11.
1117
Hungary

Email: janos.farkas@ericsson.com
Greg Mirsky
Ericsson

Email: gregimirsky@gmail.com