

MPLS Working Group
Internet-Draft
Intended status: Informational
Expires: October 13, 2022

M. Bocci
Nokia
S. Bryant
University of Surrey 5GIC
April 11, 2022

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

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	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

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

- o 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 an Label Edge Router (LER) [RFC4221] or Label Switching Router (LSR). This data may be encoded within a network action

sub-stack (see below) (in-stack data), and/or after the bottom of the label stack (post-stack data).

- o 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.
- o 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.
- o 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.
- o 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).
- o 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).
- o 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 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 [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.

13. A mechanism **MUST** be defined for control / management planes in use to determine the ability of downstream LSRs/LEs 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 LEs, but **MAY** be processed at LSRs and LEs. 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

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

7.2. Informative References

- [I-D.bryant-mpls-dev-primer] Bryant, S., "A Primer on the Development of MPLS", draft-bryant-mpls-dev-primer-01 (work in progress), December 2021.
- [I-D.saad-mpls-miad-usecases] Saad, T., Makhijani, K., Song, H., and G. Mirsky, "Use Cases for MPLS Function Indicators and Ancillary Data", draft-saad-mpls-miad-usecases-01 (work in progress), March 2022.
- [I-D.song-mpls-extension-header] Song, H., Li, Z., Zhou, T., Andersson, L., and Z. Zhang, "MPLS Extension Header", draft-song-mpls-extension-header-06 (work in progress), January 2022.
- [RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, DOI 10.17487/RFC3031, January 2001, <<https://www.rfc-editor.org/info/rfc3031>>.
- [RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, DOI 10.17487/RFC3032, January 2001, <<https://www.rfc-editor.org/info/rfc3032>>.
- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <<https://www.rfc-editor.org/info/rfc3552>>.

- [RFC4221] Nadeau, T., Srinivasan, C., and A. Farrel, "Multiprotocol Label Switching (MPLS) Management Overview", RFC 4221, DOI 10.17487/RFC4221, November 2005, <<https://www.rfc-editor.org/info/rfc4221>>.
- [RFC5586] Bocci, M., Ed., Vigoureux, M., Ed., and S. Bryant, Ed., "MPLS Generic Associated Channel", RFC 5586, DOI 10.17487/RFC5586, June 2009, <<https://www.rfc-editor.org/info/rfc5586>>.
- [RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, DOI 10.17487/RFC6790, November 2012, <<https://www.rfc-editor.org/info/rfc6790>>.
- [RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <<https://www.rfc-editor.org/info/rfc6973>>.
- [RFC9088] Xu, X., Kini, S., Psenak, P., Filsfils, C., Litkowski, S., and M. Bocci, "Signaling Entropy Label Capability and Entropy Readable Label Depth Using IS-IS", RFC 9088, DOI 10.17487/RFC9088, August 2021, <<https://www.rfc-editor.org/info/rfc9088>>.

Authors' Addresses

Matthew Bocci
Nokia

Email: matthew.bocci@nokia.com

Stewart Bryant
University of Surrey 5GIC

Email: sb@stewartbryant.com

MPLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: 29 October 2022

K. Kompella, Ed.
Juniper Networks
S. Bryant
University of Surrey 5GIC
M. Bocci
Nokia
G. Mirsky
Ericsson
L. Andersson
Bronze Dragon Consulting
27 April 2022

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.

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 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	2
1.1. Conventions and Definitions	3
2. Rationale	6
2.1. Why Look at the First Nibble	6
2.1.1. Load Balancing	6
2.1.2. Requirement	8
2.1.3. Recommendation	8
2.1.4. Parsing the Post-stack Header	9
2.2. Why Create a Registry	9
2.3. Caveat	9
3. IANA Considerations	10
3.1. MPLS First Nibble Registry	10
3.1.1. Allocation Policy	10
4. References	11
4.1. Normative References	11
4.2. Informative References	12
Authors' Addresses	12

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].

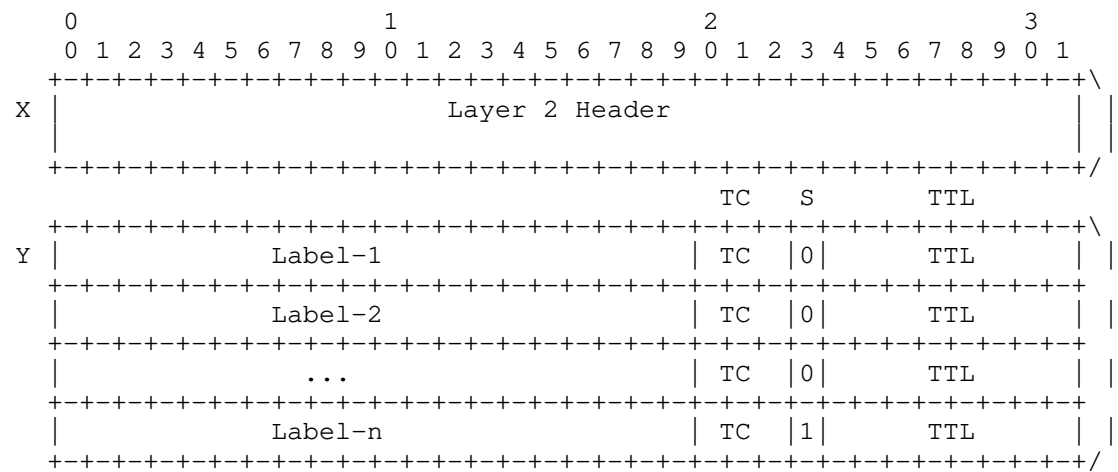


Figure 1: Example of an MPLS Packet With Label Stack

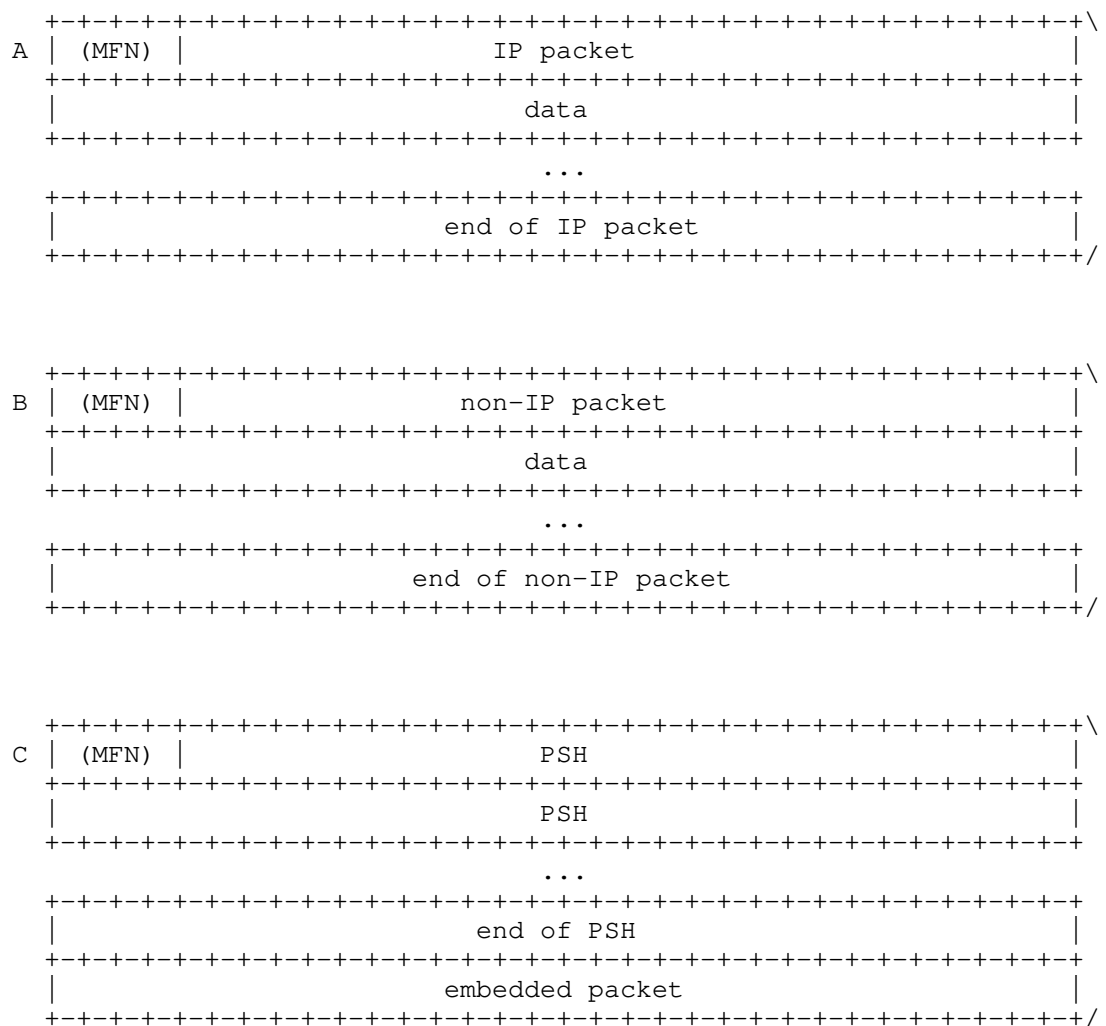


Figure 2: Three Examples of MPLS Payloads

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 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:

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

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

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, DOI 10.17487/RFC3032, January 2001, <<https://www.rfc-editor.org/info/rfc3032>>.
- [RFC4385] Bryant, S., Swallow, G., Martini, L., and D. McPherson, "Pseudowire Emulation Edge-to-Edge (PWE3) Control Word for Use over an MPLS PSN", RFC 4385, DOI 10.17487/RFC4385, February 2006, <<https://www.rfc-editor.org/info/rfc4385>>.
- [RFC4928] Swallow, G., Bryant, S., and L. Andersson, "Avoiding Equal Cost Multipath Treatment in MPLS Networks", BCP 128, RFC 4928, DOI 10.17487/RFC4928, June 2007, <<https://www.rfc-editor.org/info/rfc4928>>.
- [RFC6391] Bryant, S., Ed., Filsfils, C., Drafz, U., Kompella, V., Regan, J., and S. Amante, "Flow-Aware Transport of Pseudowires over an MPLS Packet Switched Network", RFC 6391, DOI 10.17487/RFC6391, November 2011, <<https://www.rfc-editor.org/info/rfc6391>>.
- [RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, DOI 10.17487/RFC6790, November 2012, <<https://www.rfc-editor.org/info/rfc6790>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

- [RFC8296] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Tantsura, J., Aldrin, S., and I. Meilik, "Encapsulation for Bit Index Explicit Replication (BIER) in MPLS and Non-MPLS Networks", RFC 8296, DOI 10.17487/RFC8296, January 2018, <<https://www.rfc-editor.org/info/rfc8296>>.
- [RFC8469] Bryant, S., Malis, A., and I. Bagdonas, "Recommendation to Use the Ethernet Control Word", RFC 8469, DOI 10.17487/RFC8469, November 2018, <<https://www.rfc-editor.org/info/rfc8469>>.
- [RFC8964] Varga, B., Ed., Farkas, J., Berger, L., Malis, A., Bryant, S., and J. Korhonen, "Deterministic Networking (DetNet) Data Plane: MPLS", RFC 8964, DOI 10.17487/RFC8964, January 2021, <<https://www.rfc-editor.org/info/rfc8964>>.

4.2. Informative References

- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <<https://www.rfc-editor.org/info/rfc4364>>.
- [RFC4446] Martini, L., "IANA Allocations for Pseudowire Edge to Edge Emulation (PWE3)", BCP 116, RFC 4446, DOI 10.17487/RFC4446, April 2006, <<https://www.rfc-editor.org/info/rfc4446>>.
- [RFC4761] Kompella, K., Ed. and Y. Rekhter, Ed., "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling", RFC 4761, DOI 10.17487/RFC4761, January 2007, <<https://www.rfc-editor.org/info/rfc4761>>.
- [RFC4762] Lasserre, M., Ed. and V. Kompella, Ed., "Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling", RFC 4762, DOI 10.17487/RFC4762, January 2007, <<https://www.rfc-editor.org/info/rfc4762>>.
- [RFC7432] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <<https://www.rfc-editor.org/info/rfc7432>>.

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

MPLS WG
Internet-Draft
Intended status: Standards Track
Expires: 12 August 2022

K. Kompella
V.P. Beeram
T. Saad
Juniper Networks
I. Meilik
Broadcom
8 February 2022

Multi-purpose Special Purpose Label for Forwarding Actions
draft-kompella-mpls-mspl4fa-02

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.

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 12 August 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 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. 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

7. Security Considerations	12
8. References	12
8.1. Normative References	12
8.2. Informative References	13
Authors' Addresses	13

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.nsdtd-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

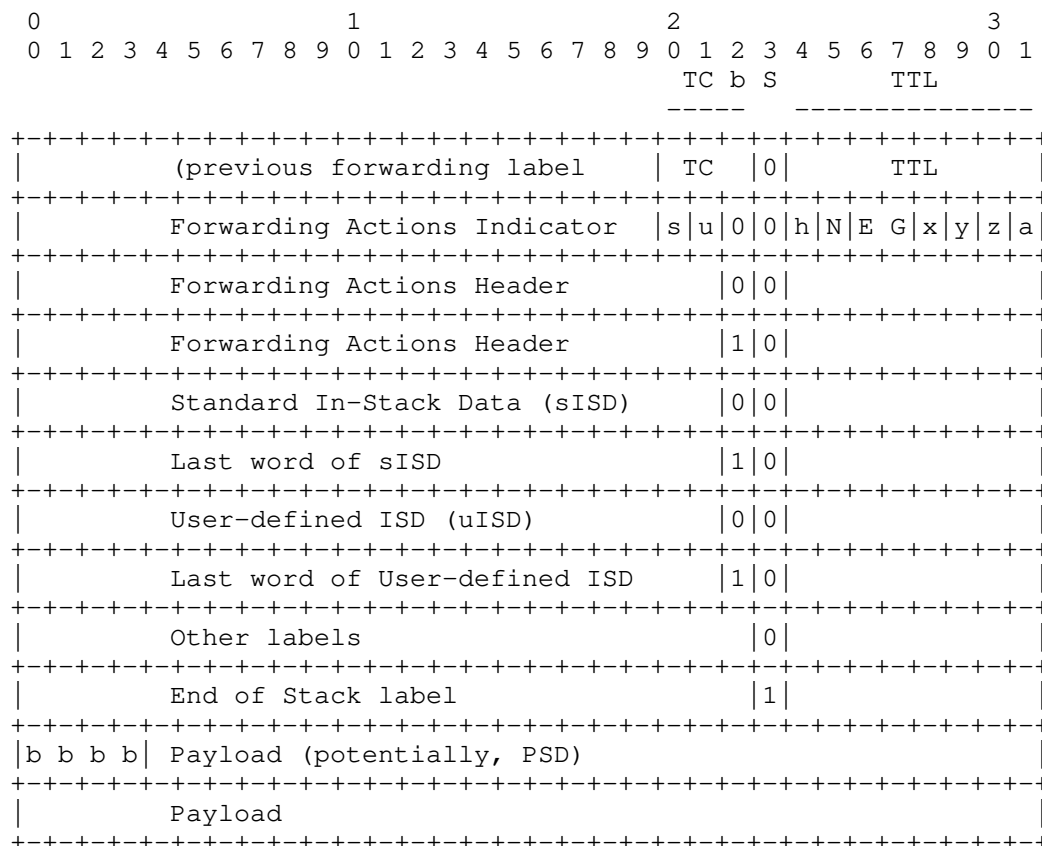


Figure 1: Format for FAI, ISD and PSD

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:

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

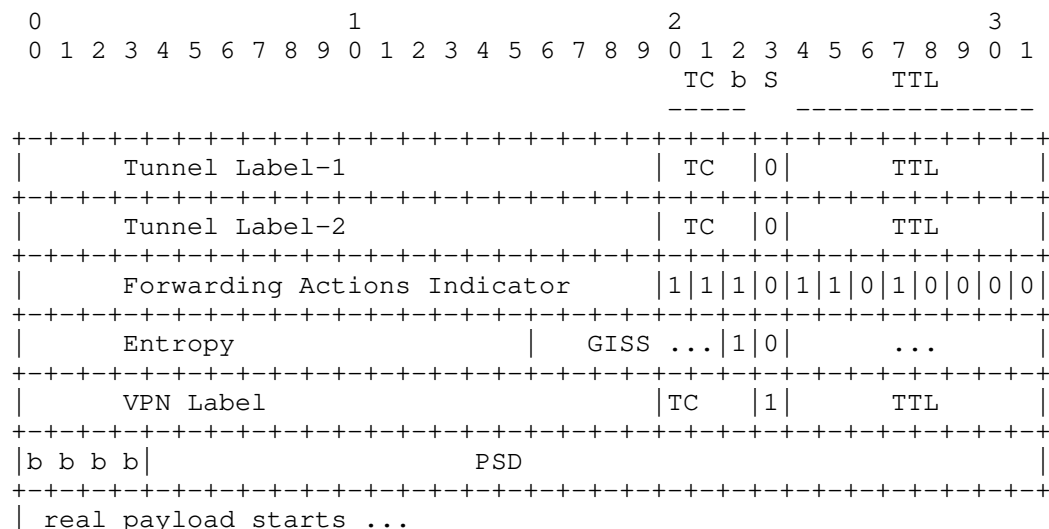
Here's how the Standard ISD is parsed. One must keep track of the s bit to know when the Standard ISD Block ends, and the S bit to know when the stack ends. The Standard ISD data appears in the order of the corresponding flags.

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



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]

Saad, T., Beeram, V. P., Wen, B., Ceccarelli, D., Halpern, J., Peng, S., Chen, R., Liu, X., Contreras, L. M., Rokui, R., and L. Jalil, "Realizing Network Slices in IP/MPLS Networks", Work in Progress, Internet-Draft, draft-bestbar-teas-ns-packet-07, 11 January 2022, <<https://www.ietf.org/archive/id/draft-bestbar-teas-ns-packet-07.txt>>.

[I-D.kompella-mpls-nffrr]

Kompella, K. and W. Lin, "No Further Fast Reroute", Work in Progress, Internet-Draft, draft-kompella-mpls-nffrr-02, 12 July 2021, <<https://www.ietf.org/archive/id/draft-kompella-mpls-nffrr-02.txt>>.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

8.2. Informative References

- [I-D.nsd-t-teas-ns-framework]
Gray, E. and J. Drake, "Framework for IETF Network Slices", Work in Progress, Internet-Draft, draft-nsd-t-teas-ns-framework-05, 2 February 2021, <<https://www.ietf.org/archive/id/draft-nsd-t-teas-ns-framework-05.txt>>.

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

DetNet
Internet-Draft
Intended status: Informational
Expires: 23 July 2022

B. Varga
J. Farkas
G. Mirsky
Ericsson
19 January 2022

Deterministic Networking (DetNet): OAM Functions for The Service Sub-
Layer
draft-varga-detnet-service-sub-layer-oam-02

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 23 July 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 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	2
2. Terminology	3
2.1. Terms Used in This Document	3
2.2. Abbreviations	3
2.3. Requirements Language	4
3. DetNet Service Sub-layer OAM Challenges	4
3.1. Illustrative example	4
3.2. DetNet Service Sub-layer Specifics for OAM	5
3.3. Information Needed during DetNet OAM Packet Processing .	6
3.4. A Possible Format of DetNet Associated Channel Header (d-ACH)	6
4. Requirements on OAM for DetNet Service Sub-layer	6
5. DetNet PING	6
5.1. Overview	7
5.2. OAM processing at the DetNet service sub-layer	7
5.2.1. Relay node with PRF	7
5.2.2. Relay node with PEF	8
5.2.3. Relay node with POF	9
5.2.4. Relay node without PREOF	9
6. Security Considerations	10
7. IANA Considerations	10
7.1. DetNet MPLS OAM Flags Registry	10
8. Acknowledgements	10
9. References	10
9.1. Normative References	10
9.2. Informative References	11
Authors' Addresses	12

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.

PEF Packet Elimination Function.

POF Packet Ordering Function.

PREOF Packet Replication, Elimination and Ordering Functions.

PRF Packet Replication Function.

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.

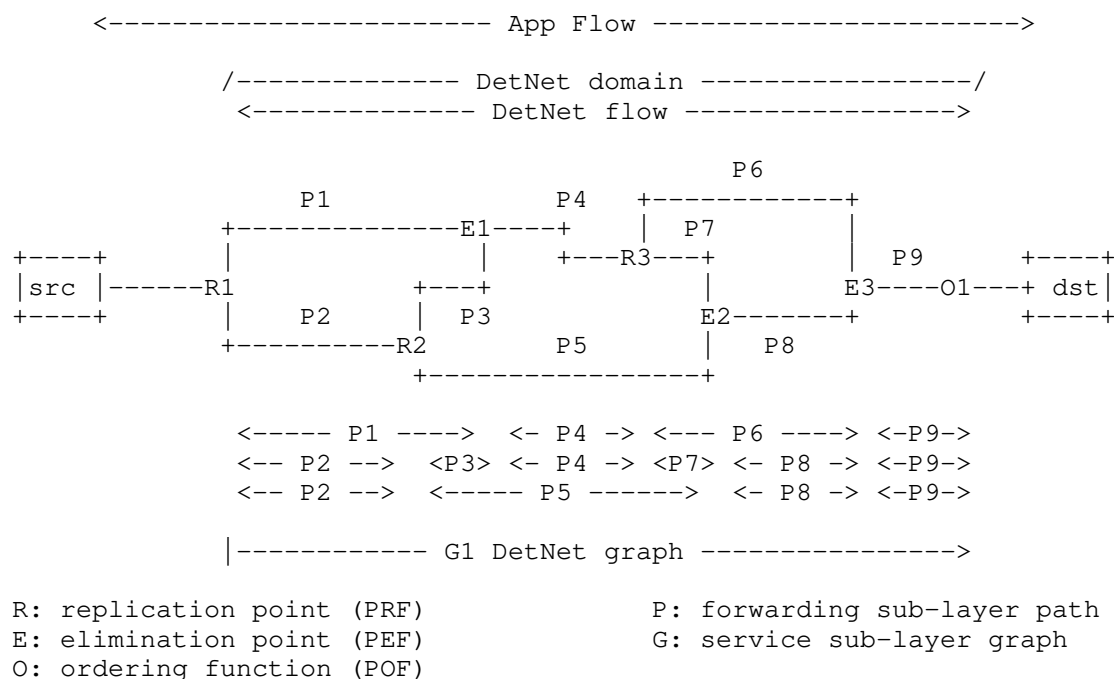


Figure 1: PREOF scenario in a DetNet network

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 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)

[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-mpls-oam].]

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].]

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, Action-Type=x, operational status, value of key state variable(s), function 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.

Note, PEF MAY use a simplified elimination algorithm for DetNet Echo-Request packets (e.g., "MatchRecoveryAlgorithm" in [IEEE8021CB]) as OAM is a slow protocol.

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

[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-mpls-oam].]

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

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4928] Swallow, G., Bryant, S., and L. Andersson, "Avoiding Equal Cost Multipath Treatment in MPLS Networks", BCP 128, RFC 4928, DOI 10.17487/RFC4928, June 2007, <<https://www.rfc-editor.org/info/rfc4928>>.

- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8655] Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", RFC 8655, DOI 10.17487/RFC8655, October 2019, <<https://www.rfc-editor.org/info/rfc8655>>.
- [RFC8964] Varga, B., Ed., Farkas, J., Berger, L., Malis, A., Bryant, S., and J. Korhonen, "Deterministic Networking (DetNet) Data Plane: MPLS", RFC 8964, DOI 10.17487/RFC8964, January 2021, <<https://www.rfc-editor.org/info/rfc8964>>.

9.2. Informative References

- [I-D.ietf-detnet-ip-oam]
Mirsky, G., Chen, M., and D. Black, "Operations, Administration and Maintenance (OAM) for Deterministic Networks (DetNet) with IP Data Plane", Work in Progress, Internet-Draft, draft-ietf-detnet-ip-oam-03, 19 September 2021, <<https://www.ietf.org/archive/id/draft-ietf-detnet-ip-oam-03.txt>>.
- [I-D.ietf-detnet-mpls-oam]
Mirsky, G., Chen, M., Varga, B., and J. Farkas, "Operations, Administration and Maintenance (OAM) for Deterministic Networks (DetNet) with MPLS Data Plane", Work in Progress, Internet-Draft, draft-ietf-detnet-mpls-oam-06, 10 December 2021, <<https://www.ietf.org/archive/id/draft-ietf-detnet-mpls-oam-06.txt>>.
- [I-D.ietf-detnet-oam-framework]
Mirsky, G., Theoleyre, F., Papadopoulos, G. Z., Bernardos, C. J., Varga, B., and J. Farkas, "Framework of Operations, Administration and Maintenance (OAM) for Deterministic Networking (DetNet)", Work in Progress, Internet-Draft, draft-ietf-detnet-oam-framework-05, 14 October 2021, <<https://www.ietf.org/archive/id/draft-ietf-detnet-oam-framework-05.txt>>.

[I-D.varga-detnet-pof]

Varga, B., Farkas, J., Kehrler, S., and T. Heer,
"Deterministic Networking (DetNet): Packet Ordering
Function", Work in Progress, Internet-Draft, draft-varga-
detnet-pof-02, 22 October 2021,
<<https://www.ietf.org/archive/id/draft-varga-detnet-pof-02.txt>>.

[IEEE8021CB]

IEEE, "IEEE Standard for Local and metropolitan area
networks -- Frame Replication and Elimination for
Reliability", DOI 10.1109/IEEESTD.2017.8091139, October
2017,
<https://standards.ieee.org/standard/802_1CB-2017.html>.

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