MPLS WG
Internet-Draft
Intended status: Standards Track
Expires: January 13, 2022

K. Kompella
V. Beeram
T. Saad
Juniper Networks
I. Meilik
Broadcom

July 12, 2021

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.

Table of Contents

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.

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

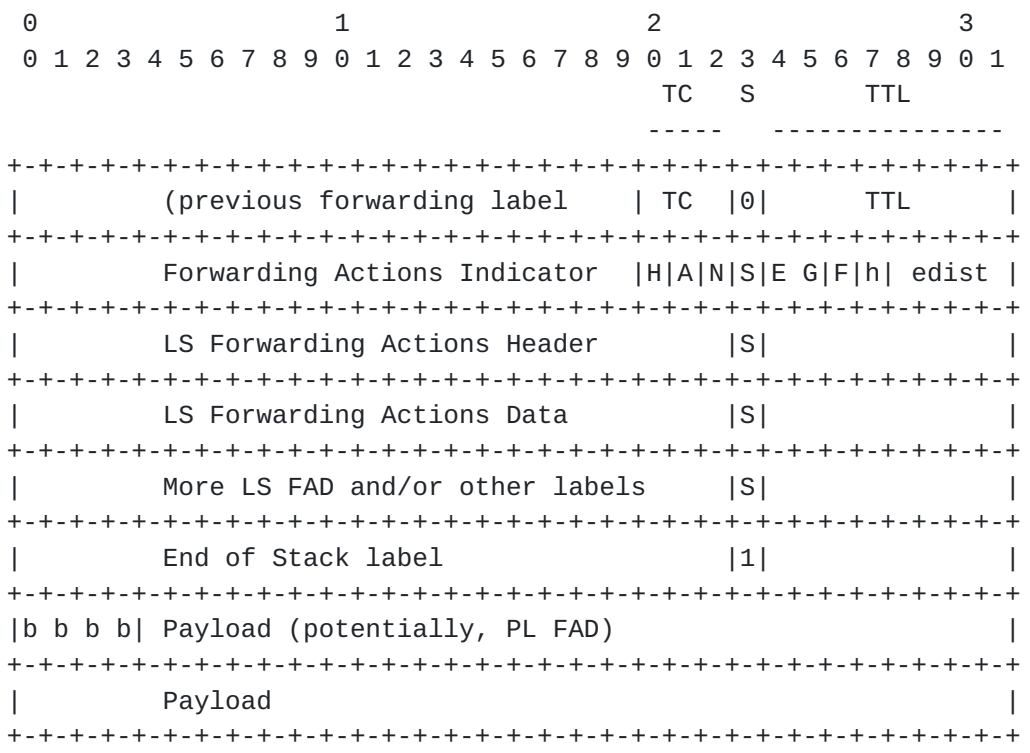


Figure 1: Format for FAI, LS FAD and PL FAD

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

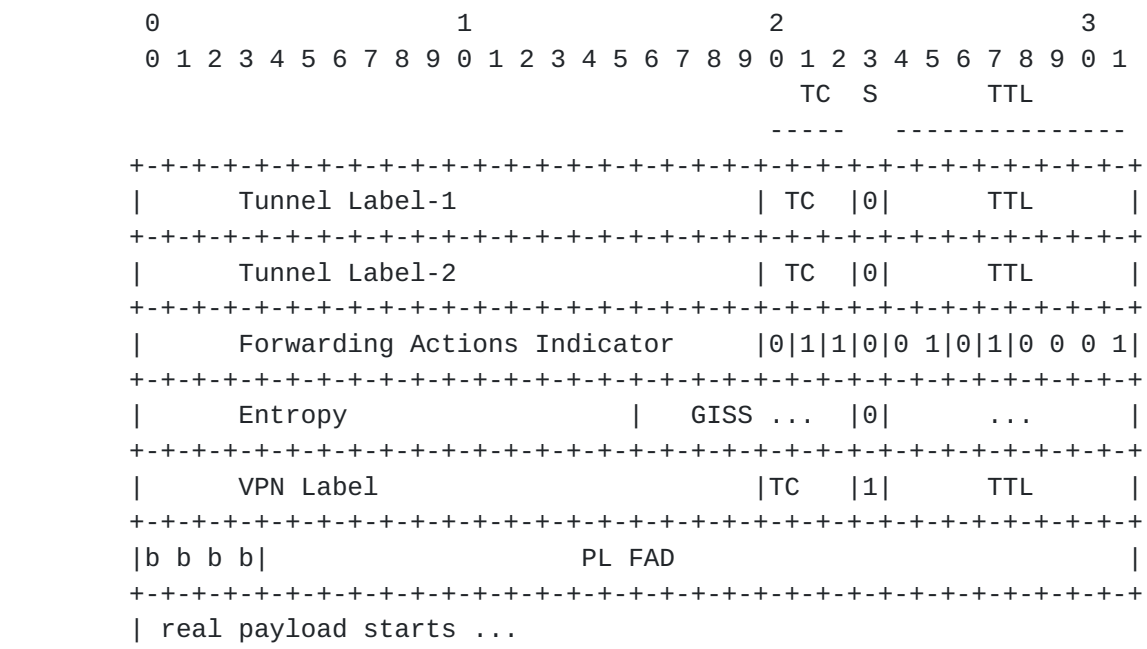
    2. while LL's S bit == 0, LL++

    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



H = 0; no FAH.
A = 1: there is LS FAD.
N = 1: NFFRR is set.
EG = 01: LS FAD 0 contains Entropy + GISS.
F = 0: No Flow ID.
h = 1: There is hop-by-hop PL FAD.
edist = 1: one more label to End of Stack.

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

The real payload starts after the PL FAD.

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.

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]
Saad, T., Beeram, V. P., Wen, B., Ceccarelli, D., Halpern, J., Peng, S., Chen, R., Liu, X., and L. M. Contreras, "Realizing Network Slices in IP/MPLS Networks", draft-bestbar-teas-ns-packet-02 (work in progress), February 2021.

[I-D.kompella-mpls-nffrr]
Kompella, K. and W. Lin, "No Further Fast Reroute", draft-kompella-mpls-nffrr-01 (work in progress), November 2020.

[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.nsdt-teas-ns-framework]
Gray, E. and J. Drake, "Framework for IETF Network Slices", draft-nsdt-teas-ns-framework-05 (work in progress), February 2021.

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