MPLS WG K. Kompella
Internet-Draft V. Beeram
Intended status: Standards Track T. Saad
Expires: August 26, 2021 Juniper Networks
I. Meilik
Broadcom
February 22, 2021

Multi-purpose Special Purpose Label for Forwarding Actions
draft-kompella-mpls-mspl4fa-00

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 will expire on August 26, 2021.

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. 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.:

o to treat a packet according to its slice, given its GISS;

o to load balance a packet, given its entropy;

o whether or not to perform fast reroute on a failure [I-D.kompella-mpls-nffrr];

o whether or not the packet has a Flow ID;

o to update statistics based on the path identifier [I-D.hegde-spring-traffic-accounting-for-sr-paths];

- o to view/update OAM metadata; [I-D.cheng-mpls-inband-pm-encapsulation], [I-D.gandhi-mpls-ioam-sr], other approaches.

- o to reassemble a fragmented packet [I-D.zzhang-tsvwg-generic-transport-functions];

- o 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: for labels not at the top of the label stack, the only bits that a forwarding engine looks at 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 bits of labels below the top of stack, please contact the authors.] This means that for a bSPL that will never appear at the top of stack, 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 FAD words compact and efficient.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
                                            TC   S       TTL
                                           -----   --------------
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         (previous forwarding label    | TC  |0|      TTL      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         Forwarding Actions Indicator  |H|E G|S|N|F|Q|O A M|P f|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         LS Forwarding Actions Data          |S|               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         More LS FAD and/or other labels     |S|               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         Bottom of stack label               |1|               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |b b b b| Payload (potentially, PL FAD)                         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         Payload                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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.

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).

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

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

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

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

Q: if set, the payload contains Opaque data.

OAM: a 3-bit field that specifies what type(s) of OAM is carried in the in the label stack and/or payload.

P: If set, the PL FAD contains a Path Identifier.

F: If set, the payload contains a Fragmentation Header.

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.

Here's how the LS FAD is parsed. One must keep track of the S bit to know when the stack ends. It is an error if the label stack ends while there are more PL FAD words to process.

1. Set NL ("next label") to the first 4-octet word of the LS FAD. Set PL ("payload") to the first 4-octet word of the payload.

2. Process H: if set, (TBD); otherwise, NL is unchanged.

3. Process EG:

    1. If EG is 00, NL is unchanged.

    2. If EG is 01 or 10, NL contains both GISS and Entropy. Increment NL.

    3. If EG is 11, NL contains GISS; NL+1 contains Entropy. Increment NL by 2.

4. Process N. NL is unchanged.

5. Process F:

    1. If F is set, NL contains the Flow ID; increment NL.

6. Process Q:

    1. If set, (TBD); otherwise, NL is unchanged.

7. NL now points at next label in the stack.

A similar procedure applies to parsing the PL; details will be forthcoming when the OAM field is better defined.

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
                                            TC   S        TTL
                                          -----   ---------------
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      Tunnel Label-1                   | MBZ |0|      MBZ      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      Tunnel Label-2                   | MBZ |0|      MBZ      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      Forwarding Actions Indicator     |0|0 1|0|1|0|0|0 0 0|0 1|
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      Entropy                |    GISS ...  |0|      ...      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      VPN Label                        |MBZ  |1|      MBZ      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |b b b b|              Fragmentation Header ...                 |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 | real payload starts ...

H  =  0; ignore.
EG = 01: LS FAD 0 contains Entropy + GISS.
N  =  1: NFFRR is set.
F  =  0: No Flow ID.
Q  =  0: ignore.
OAM = 0; ignore.
P  =  0: no Path Identifier in payload.
F  =  1: Fragmentation Header is present.
```

Figure 2: Example of FAI + LS FAD + PL FAD

The real payload starts after the Fragmentation Header.

3. Issues to be Resolved

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 FAD with GE set to 00 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. First Nibble Issues

The first nibble of the first word of the payload SHOULD NOT be 0x4 or 0x6, as legacy LSRs may use the heuristic that this indicates a payload of IPv4/IPPv6. iOAM data has a first nibble of 0x1. However, if there is no iOAM data, the first nibble of the Path Identifier, if any, else that of the Fragmentation Header, MUST NOT be 0x4 or 0x6. However, it is inefficient to have to address this issue for every type of PL FAD, as it may be the first word in the payload. A future version of this document will propose an alternative solution.

It is unclear when a Control Word may be present as the first word of the payload; this is sometimes signaled and sometimes configured. When it is present, the above issue is moot.

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.

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., Wen, B., Ceccarelli, D., Halpern, J., Peng, S., Chen, R., and X. Liu, "Realizing Network Slices in IP/MPLS Networks", draft-bestbar-teas-ns-packet-01 (work in progress), December 2020.

[I-D.cheng-mpls-inband-pm-encapsulation]
Cheng, W., Min, X., Zhou, T., Dong, X., and Y. Peleg, "Encapsulation For MPLS Performance Measurement with Alternate Marking Method", draft-cheng-mpls-inband-pm-encapsulation-04 (work in progress), September 2020.

[I-D.gandhi-mpls-ioam-sr]
Gandhi, R., Ali, Z., Filsfils, C., Brockners, F., Wen, B., and V. Kozak, "MPLS Data Plane Encapsulation for In-situ OAM Data", draft-gandhi-mpls-ioam-sr-05 (work in progress), January 2021.

[I-D.hegde-spring-traffic-accounting-for-sr-paths]
Hegde, S., "Traffic Accounting for MPLS Segment Routing Paths", draft-hegde-spring-traffic-accounting-for-sr-paths-02 (work in progress), October 2018.

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

[I-D.zzhang-tsvwg-generic-transport-functions]
Zhang, Z., Bonica, R., and K. Kompella, "Generic Transport Functions", draft-zzhang-tsvwg-generic-transport-functions-00 (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 Transport Network Slices", draft-nsdt-teas-ns-framework-04 (work in progress), July 2020.

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