Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 2, 2015

S. Kini, Ed.
Ericsson
K. Kompella
Juniper
S. Sivabalan
Cisco
S. Litkowski
Orange
R. Shakir
B.T.
X. Xu
Huawei
W. Hendrickx
Alcatel-Lucent
J. Tantsura
Ericsson

September 29, 2014

Entropy labels for source routed stacked tunnels
draft-kini-mpls-spring-entropy-label-01

Abstract

Source routed tunnel stacking is a technique that can be leveraged to provide a method to steer a packet through a controlled set of segments. This can be applied to the Multi Protocol Label Switching (MPLS) data plane. Entropy label (EL) is a technique used in MPLS to improve load balancing. This document examines and describes how ELs are to be applied to source routed stacked tunnels.

Table of Contents

1. Introduction

The source routed stacked tunnels paradigm is leveraged by techniques such as Segment Routing (SR) [I-D.filsfils-spring-segment-routing] to steer a packet through a set of segments. This can be directly applied to the MPLS data plane, but it has implications on label stack depth.

Clarifying statements on label stack depth have been provided in [RFC7325] but they do not address the case of source routed stacked MPLS tunnels as described in [I-D.gredler-spring-mpls] or

[I-D.filsfils-spring-segment-routing] where deeper label stacks are more prevalent.

Entropy label (EL) [RFC6790] is a technique used in the MPLS data plane to provide entropy for load balancing. When using LSP hierarchies there are implications on how [RFC6790] should be applied. One such issue is addressed by [I-D.ravisingh-mpls-el-for-seamless-mpls] but that is when different levels of the hierarchy are created at different LSRs. The current document addresses the case where the hierarchy is created at a single LSR as required by source stacked tunnels.

A use-case requiring load balancing with source stacked tunnels is given in Section 3. A recommended solution is described in Section 4. Options that were considered to arrive at the recommended solution are documented for historical purposes in Section 5.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Abbreviations and Terminology

EL - Entropy Label

ELI - Entropy Label Identifier

ELC - Entropy Label Capability

SR - Segment Routing

ECMP - Equal Cost Multi Paths

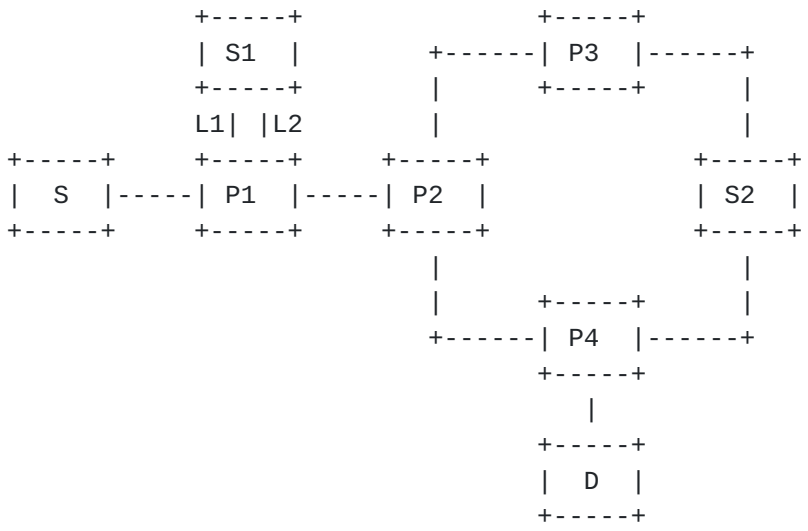
MPLS - Multiprotocol Label Switching

SID - Segment Identifier

3. Use-case for multipath load balancing in source stacked tunnels

Source stacked tunnels have several use-cases, one of which is service chaining [I-D.filsfils-spring-segment-routing-use-cases]. Consider the service-chaining network in Figure 1 that has MPLS as the data plane. The requirement of the use-case is to create a LSP from source LSR S, apply the services S1, S2 and finally terminate the LSP at destination LSR D. Local load balancing is required across the parallel links between P1 and S1. Local load balancing is

also required between the ECMP paths from S1 to S2 i.e., between the paths S1-P1-P2-P3-S2 and S1-P1-P2-P4-S2. Segment routing can be used to achieve this. A segment to S1 is stacked above the segment to S2 which in turn is stacked above the segment to D. Labels for service instructions are also inserted in the stack at appropriate depths so that services S1 and S2 are executed. To achieve local load balancing the SIDs of specific interfaces is not specified. Since entropy label is a standardized [RFC6790] mechanism defined for MPLS it can be adapted to the case of source stacked tunnels. Multiple ways to apply entropy labels exist and a recommended solution is described in Section 4 and all the options considered are listed in Section 5 along with their tradeoffs. We denote SN to be the node SID of LSR N and SN{L1,L2,...} to denote the SID of the adjacency for the set for links {L1,L2,...} of LSR N and S-SvcN to denote the SID for a service at service LSR N. The label stack that the source LSR S uses for the LSP can be <SS1, S-SvcS1, SS2, S-SvcS2, SD> or <SP1, SP1{L1,L2}, S-SvcS1, SS2, S-SvcS2, SD>.



S=Source LSR, D=Destination LSR, S1,S2=service-LSRs, L1,L2=links, P1,P2,P3,P4=Transit LSRs

Figure 1: Service chaining use-case

4. Recommended EL solution for SPRING

The solution described in this section follows [RFC6790].

An LSR may have a limitation in its ability to read and process the label stack in order to do multipath load balancing. This limitation

expressed in terms of the number of label stack entries that the LSR can read and is henceforth referred to as the Readable Label Depth (RLD) capability. In order for the EL to occur within the RLD of LSRs along the path corresponding to a label stack, multiple <ELI, EL> pairs MAY be inserted. The recommendations for inserting <ELI, EL> pairs are:

- <ELI, EL> pairs MUST be inserted below those labels that are advertised with ELC.
- An LSR that is limited in the number of <ELI, EL> pairs that it can insert SHOULD prefer to insert such pairs deeper in the stack.
- An LSR SHOULD try to insert an <ELI, EL> pair within the RLD of the maximum number of LSRs along the path as it can.
- An LSR SHOULD try to insert the minimum number of such pairs while trying to satisfy the above criteria.

A sample algorithm to insert ELs is shown below. Implementations can choose any algorithm as long as it follows the above recommendations.

```
set current EL insertion point to the bottommost EL-capable location
while local-node can push more labels or top of stack has been reached {
    insert an ELI+EL at current insertion point
    move insertion point up until current EL is out of RLD
                                                    AND
                                        insertion point is EL-capable
    set current insertion point to new insertion point
}
```

Figure 2: Algorithm to insert <ELI, EL> pairs in a label stack

The RLD can be advertised via protocols and those extensions would be described in a separate document.

5. Options considered

5.1. Single EL at the bottom of the stack of tunnels

In this option a single EL is used for the entire label stack. The source LSR S encodes the entropy label (EL) below the labels of all the stacked tunnels. In Figure 1 label stack at LSR S would look like <SP1, SS1, S-SvcS1, SS2, S-SvcS2, SD, ELI, EL> <remaining packet header>. Note that the notation in [RFC6790] is used to describe the label stack. An issue with this approach is that as the label stack grows due an increase in the number of SIDs, the EL correspondingly goes deeper in the label stack. As a result, intermediate LSRs (such

as P1) that have to walk the label stack at least until the EL (if found) to perform load balancing decisions have to access a larger number of bytes in the packet header when making forwarding decisions. A load balanced network design using this approach must ensure that all intermediate LSRs have the capability to traverse the maximum label stack depth in order to do effective load balancing. The use-case for which the tunnel stacking is applied would determine the maximum label stack depth.

In the case where the hardware is capable of pushing a single <ELI, EL> pair at any depth, this option is the same as the recommended solution in Section 4.

This option was discounted since there exist a number of hardware implementations which have a low maximum readable label depth. Choosing this option can lead to a loss of load-balancing using EL in a significant part of the network but that is a critical requirement in a service provider network.

5.2. An EL per tunnel in the stack

In this option each tunnel in the stack can be given its own EL. The source LSR pushes an <ELI, EL> before pushing a tunnel label when load balancing is required to direct traffic on that tunnel. For the same Figure 1 above, the source LSR S encoded label stack would be <SS1, ELI, EL1, S-SvcS1, SS2, ELI, EL2, SD> where all the ELs can have the same value. Accessing the EL at an intermediate LSR is independent of the depth of the label stack and hence independent of the specific use-case to which the stacked tunnels are applied. A drawback is that the depth of the label stack grows significantly, almost 3 times as the number of labels in the label stack. The network design should ensure that source LSRs should have the capability to push such a deep label stack. Also, the bandwidth overhead and potential MTU issues of deep label stacks should be accounted for in the network design.

In the case where the RLD is the minimum value (3) for all LSRs, all LSRs are EL capable and the LSR that is inserting <ELI, EL> pairs has no limit on how many it can insert then this option is the same as the recommended solution in Section 4.

This option was discounted due to the existence of hardware implementations that can push a limited number of labels on the label stack. Choosing this option would result in a hardware requirement to push two additional labels per tunnel label. Hence it would restrict the number of tunnels that can form a LSP and constrain the types of LSPs that can be created. This was considered unacceptable.

5.3. A re-usable EL for a stack of tunnels

In this option an LSR that terminates a tunnel re-uses the EL of the terminated tunnel for the next inner tunnel. It does this by storing the EL from the outer tunnel when that tunnel is terminated and re-inserting it below the next inner tunnel label during the label swap operation. The LSR that stacks tunnels SHOULD insert an EL below the outermost tunnel. It SHOULD NOT insert ELs for any inner tunnels. Also, the penultimate hop LSR of a segment MUST NOT pop the ELI and EL even though they are exposed as the top labels since the terminating LSR of that segment would re-use the EL for the next segment.

For the same Figure 1 above, the source LSR S encoded label stack would be <SS11, ELI, EL, S-SvcS1, SS2, SD>. At P1 the outgoing label stack would be <SS1, ELI, EL, S-SvcS1, SS2, SD> after it has load balanced to one of the links L1 or L2. At S1 the outgoing label stack would be <SS2, ELI, EL, SD>. At P2 the outgoing label stack would be <SS2, ELI, EL, SD> and it would load balance to one of the nexthop LSRs P3 or P4. Accessing the EL at an intermediate LSR (e.g. P3) is independent of the depth of the label stack and hence independent of the specific use-case to which the stacked tunnels are applied.

This option was discounted due to the significant change in label swap operations that would be required for existing hardware.

5.3.1. EL at top of stack

A slight variant of the re-usable EL option is to keep the EL at the top of the stack rather than below the tunnel label. In this case each LSR that is not terminating a segment should continue to keep the received EL at the top of the stack when forwarding the packet along the segment. An LSR that terminates a segment should use the EL from the terminated segment at the top of the stack when forwarding onto the next segment.

This option was discounted due to the significant change in label swap operations that would be required for existing hardware.

5.4. ELs at readable label stack depths

In this option the source LSR inserts ELs for tunnels in the label stack at depths such that each LSR along the path that must load balance is able to access at least one EL. Note that the source LSR may have to insert multiple ELs in the label stack at different depths for this to work since intermediate LSRs may have differing capabilities in accessing the depth of a label stack. The label

stack depth access value of intermediate LSRs must be known to create such a label stack. How this value is determined is outside the scope of this document. This value can be advertised using a protocol such as an IGP. For the same Figure 1 above, if LSR P1 needs to have the EL within a depth of 4, then the source LSR S encoded label stack would be <SS1, S-SvcS1, ELI, EL2, SS2, SD> where all the ELs would typically have the same value.

In the case where the RLD has different values along the path and the LSR that is inserting <ELI, EL> pairs has no limit on how many pairs it can insert, and it knows the appropriate positions in the stack where they should be inserted, then this option is the same as the recommended solution in Section 4.

A variant of this solution was selected which balances the number of labels that need to be pushed against the requirement for entropy.

6. Acknowledgements

The authors would like to thank John Drake and Loa Andersson for their comments.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

9. References

9.1. Normative References

[I-D.filsfils-spring-segment-routing]
Filsfils, C., Previdi, S., Bashandy, A., Decraene, B., Litkowski, S., Horneffer, M., Milojevic, I., Shakir, R., Ytti, S., Henderickx, W., Tantsura, J., and E. Crabbe, "Segment Routing Architecture", draft-filsfils-spring-segment-routing-04 (work in progress), July 2014.

[I-D.filsfils-spring-segment-routing-use-cases]
Filsfils, C., Francois, P., Previdi, S., Decraene, B., Litkowski, S., Horneffer, M., Milojevic, I., Shakir, R., Ytti, S., Henderickx, W., Tantsura, J., Kini, S., and E. Crabbe, "Segment Routing Use Cases", draft-filsfils-spring-segment-routing-use-cases-00 (work in progress), March 2014.

[I-D.gredler-spring-mpls]
Gredler, H., Rekhter, Y., Jalil, L., Kini, S., and X. Xu, "Supporting Source/Explicitly Routed Tunnels via Stacked LSPs", draft-gredler-spring-mpls-06 (work in progress), May 2014.

[I-D.ravisingh-mpls-el-for-seamless-mpls]
Singh, R., Shen, Y., and J. Drake, "Entropy label for seamless MPLS", draft-ravisingh-mpls-el-for-seamless-mpls-02 (work in progress), July 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, November 2012.

[RFC7325] Villamizar, C., Kompella, K., Amante, S., Malis, A., and C. Pignataro, "MPLS Forwarding Compliance and Performance Requirements", RFC 7325, August 2014.

9.2. Informative References

[I-D.previdi-isis-segment-routing-extensions]
Previdi, S., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., and J. Tantsura, "IS-IS Extensions for Segment Routing", draft-previdi-isis-segment-routing-extensions-05 (work in progress), February 2014.

[I-D.psenak-ospf-segment-routing-extensions]
Psenak, P., Previdi, S., Filsfils, C., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPF Extensions for Segment Routing", draft-psenak-ospf-segment-routing-extensions-05 (work in progress), June 2014.

Authors' Addresses

Sriganesh Kini (editor)
Ericsson

Email: sriganesh.kini@ericsson.com

Kireeti Kompella
Juniper

Email: kireeti@juniper.net

Siva Sivabalan
Cisco

Email: msiva@cisco.com

Stephane Litkowski
Orange

Email: stephane.litkowski@orange.com

Rob Shakir
B.T.

Email: rob.shakir@bt.com

Xiaohu Xu
Huawei

Email: xuxiaohu@huawei.com

Wim Hendrickx
Alcatel-Lucent

Email: wim.henderickx@alcatel-lucent.com

Jeff Tantsura
Ericsson

Email: jeff.tantsura@ericsson.com