SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 29, 2017

S. Litkowski
Orange
Mustapha Aissaoui
Nokia

October 31, 2016

# Implementing non protected paths using SPRING
# draft-litkowski-spring-non-protected-paths-00

## Abstract

Segment Routing (SR) leverages the source routing paradigm. A node can steer a packet on a specific path by prepending the packet with an SR header. In the framework of traffic-engineering use cases, a customer may request its service provider to implement some non protected paths. This means that in case of a failure within the network, fast-reroute (or similar) techniques should not be activated for those paths. This document analyzes the different options to implement a non protected path with Segment Routing and in a future release will provide a recommandation on the best option.

## 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 [RFC2119].

## Status of This Memo

Table of Contents



## 1. Problem statement

In some cases, a customer may prefer to manage to react on network failures using its own mechanism. In such cases, the customer usually has two disjoint paths, so a path can take over the traffic in case of failure of the other. The disjoint paths can be provided by a single provider or by multihoming to different providers as displayed in the figure below.

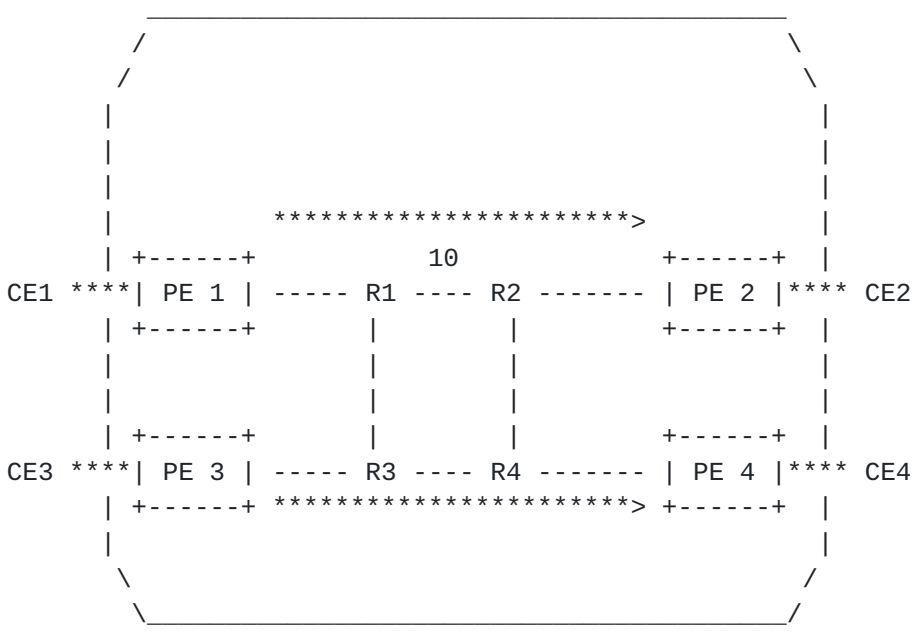


Figure 1 - Disjoint paths provided by a single provider

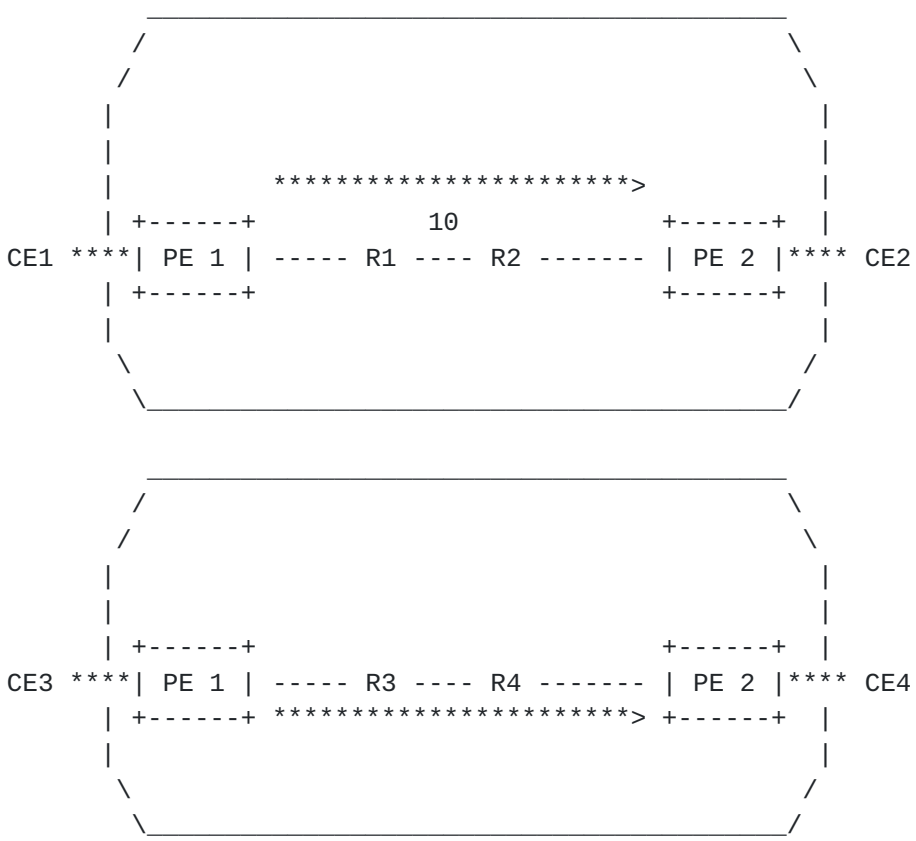


Figure 2 - Disjoint paths provided by using two providers

As the traffic protection is ensured by an end-to-end mechanism at the customer level, the customer requests the service provider to not protect the paths. However the service provider is allowed to restore the service automatically when the primary path is failing by computing and installing a new path in the network. A variant to this scheme is for the service provider to provision a unprotected secondary path between PE1 and PE2 which is also disjoint from the primary path. In the latter case, the end-to-end service protection at the CE becomes the last resort. How the end-to-end protection is handled is out of scope of this document and will be under the customer responsibility.

A segment-routing path is expressed as a list of segment identifiers (SID) from different types (Node-SID, Adj-SID, Binding-SID ...). In order to ensure that the segment routing path is not protected, we need to ensure that it does not contain any segment representing a protected path. As an example, in the Figure 1, we consider a path from PE1 to PE2 expressed with the following segment list: {Adj_R1R3,Node_R2,Adj_R2PE2}. If we want to ensure that this path is non protected, we need to ensure that the segment represented by

Adj_R1R3 represents a non protected, as well as the segment Node_R2 and Adj_R2PE2.

The segment routing path may be computed by a Path Computation Element (PCE). In order to fulfil the non protected path constraint, the PCE needs to be aware of the available SIDs in the network and their protection status.

Several techniques may be used to represent a non protected path with a segment identifier. We propose to analyze the different options.

## 2. Options to create a non protected path with Segment Routing

### 2.1. Using only non protected adjacency segments

The adjacency segment was created so a node can advertise multiple adjacency segments for a particular link with different properties. The non-protected property is already defined as part of the protocol encodings ([I-D.ietf-isis-segment-routing-extensions], [I-D.ietf-ospf-segment-routing-extensions] and [I-D.gredler-idr-bgp-ls-segment-routing-extension]) through the B flag. However, from an implementation perspective, advertising a protected adjacency segment, a non protected adjacency segment or both for each link is optional.

It is important to note that even if an adjacency segment has the B flag set (protected), it remains up to a local policy of the advertising router to implement the protection or not.

If both protected and non protected Adj-SID are advertised, every node in the network and a PCE can be aware of the adjacency segments protection property. When a non protected path is requested, the path computation module can choose to encode the path with a list a non protected adjacency segment only.

One of the advantage of using only adjacency segments is the insurance that the traffic will never go transiently outside the path defined by the computation module reponsible of the path.

One of the drawbacks of using only adjacency segments is the resulting label stack depth as each hop should require a segment in the stack: crossing 15 nodes, means stacking 15 labels for the transport. Having such a deep stack may be a problem for current hardwares and softwares for either pushing the stack (because the head end is limited in the number of labels it can push) or loadbalancing flows on transit nodes (as deep packet inspection or entropy label look up may be difficult with a deep label stack). Another drawback of advertising both protected and non protected adjacency segments is the additional controlplane and dataplane resource consumption used in the network.

### 2.2. Using a combination of node segments and adjacency segments

Using a combination of node segments and adjacency segments is the usual way when creating a segment routing path. However the well known Node-SID (algorithm type Shortest Path) may be protected by a local-repair mechanism by any transit node or may have multiple ECMP next-hops which may be a problem when used for a non protected path. Protecting a particular Node-SID is a matter of a local policy configuration on every node. The following discusses a number of possible approaches.

#### 2.2.1. Using Strict SPF Node SID

[I-D.ietf-spring-segment-routing] defines a Strict Shortest Path algorithm which mandates that the packet is forwarded according to ECMP-aware SPF algorithm and instructs any router in the path to ignore any possible local policy overriding SPF decision. The provided definition in this document does not clearly state that the Strict Shortest Path Node-SID cannot use a protection path but makes think that the path cannot be overriden by a local policy including a fast-reroute policy. If the scope of the Strict SPF Node-SID is clarified to mean one or more primary next-hops is selected with no local-repair, combining Strict SPF Node-SID with non protected Adj-SID may be a viable option to encode a non protected path. The selection of the primary next-hop(s) may be left to the local SPF calculation otherwise an adjacency SID can be used to exercise a specific next-hop or set of next-hops.

If this solution is viable, when a network wants to implement both protected and non protected paths, the network requires the advertisement of two Node SIDs per node (one with SPF algorithm for protected paths, and one with Strict SPF algorithm for non protected paths).

#### 2.2.2. Adding a protection flag in the Node SID

As for adjacency segments, a new flag may be added in the Prefix-SID to encode the willigness of protection. Each node will then advertise two Node-SID (using SPF algorithm), one with the protection flag set, the other without the protection flag set. The same discussion regarding ECMP is also applicable here.

#### 2.2.3. Using two Node-SIDs with different local policies

Having two instances of the Node-SID (protected and not protected) is a requirement when using Node-SID in protected and non protected paths. The protection of a Node-SID is a matter of a local policy configuration on every node in the network. A service provider may configure two Node-SIDs per node and may adjust the local-repair on

every node to protect one Node-SID but not the other. As the protection of the Node-SID is inherited from the protection of the associated prefix, the service provider will need to deploy a new set of prefixes to all nodes to deploy the new set of Node-SIDs. Then it will need to maintain the local-repair policy on every node to ensure

that the prefixes associated to the non protected Node-SID are not using the local-repair.

### 2.2.4. Advantages and drawbacks

One advantage of combining adjacency and node segments is the reduction of the label stack size.

The drawbacks are the increase of the controlplane and dataplane resource consumption, leading possibly to higher convergence time. One of the other drawback is that a Node-SID may be transiently rerouted on a path that does not fit the constraints anymore if a transit node converges faster than the head-end: this concern is not new and applies to all traffic-engineering use cases.

## 2.3. Using a combination of adjacency segments and binding-SID

[I-D.ietf-spring-segment-routing] defines the binding segment with multiple use cases. One of the use case of the binding segment is to advertise a tunnel as a segment. When a computation engine computes a non protected path and if the resulting label stack using only non protected adjacency segments is too deep for the network, an external component may create shortcuts in the network by creating a binding segment representing a list of non protected adjacency segments.

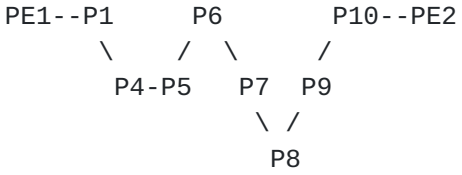


Figure 3 - Use of Binding SID

In the example above, the path from PE1 to PE2 must be expressed with the stack: {Adj_P1P4,Adj_P4P5,Adj_P5P6,Adj_P6P7,Adj_P7P8,Adj_P8P9,Adj_P9P10,Adj_P10PE2}. This stack is too deep due to the limitations of the network. An external component may create a binding Binding1 on P5 that represents the non protected path (P5->P6->P7->P8->P9->P10). When the binding is created and advertised in the topology, the computation engine can use this binding SID in a path, resulting for a PE1 to PE2 path to the stack: {Adj_P1P4,Adj_P4P5,Binding1,Adj_P10PE2}. The usage of the binding SID in the stack allowed to reduce its size to an acceptable value.

One advantage of combining adjacency and binding segments is the reduction of the label stack size.

The drawbacks are the increase of the controlplane and dataplane resource consumption. This controlplane and dataplane resource consumption will be linked to the intelligence of the external controller and computation engines and especially how the placement of the bindings is done to maximize the sharing between LSPs. Moreover any optimization try in the binding segment may introduce churn in the network controlplane (Make Before Break can be used to ensure that dataplane is not affected).

## 3. Recommended option(s)

TBD.

## 4. Security Considerations

TBD.

## 5. Acknowledgements

## 6. IANA Considerations

N/A

## 7. Normative References

[I-D.gredler-idr-bgp-ls-segment-routing-extension]
Gredler, H., Ray, S., Previdi, S., Filsfils, C., Chen, M., and J. Tantsura, "BGP Link-State extensions for Segment Routing", draft-gredler-idr-bgp-ls-segment-routing-extension-02 (work in progress), October 2014.

[I-D.ietf-isis-segment-routing-extensions]
Previdi, S., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., Decraene, B., and j. jefftant@gmail.com, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-08 (work in progress), October 2016.

[I-D.ietf-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-ietf-ospf-segment-routing-extensions-09 (work in progress), July 2016.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-09 (work in progress), July 2016.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

## Author's Address

Stephane Litkowski
Orange

Email: stephane.litkowski@orange.com

Mustapha Aissaoui
Nokia

Email: mustapha.aissaoui@nokia.com