                  Implementing non protected paths using SPRING
                  draft-litkowski-spring-non-protected-paths-02

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

Copyright Notice

Table of Contents

1. Problem statement

   In some cases, a customer may prefer 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.

```
        _____
       /                                                \
      /                                                  \
      |                                                  |
      |                                                  |
      |       ********************>                      |
      |  +------+           10           +------+        |
 CE1 ****| PE 1 | ----- R1 ---- R2 ------- | PE 2 |**** CE2
      |  +------+       |         |        +------+        |
      |                 |         |                       |
      |                 |         |                       |
      |  +------+       |         |        +------+        |
 CE3 ****| PE 3 | ----- R3 ---- R4 ------- | PE 4 |**** CE4
      |  +------+  ********************>   +------+        |
      |                                                  |
      \                                                  /
       _____/
```

        Figure 1 - Disjoint paths provided by a single provider

```
              _____
             /                                        \
            /                                          \
           |                                            |
           |     **********************>                |
           |  +------+         10              +------+  |
      CE1 ****| PE 1 | ----- R1 ---- R2 ------- | PE 2 |**** CE2
           |  +------+                         +------+  |
           |                                            |
            \                                          /
             _____/


              _____
             /                                        \
            /                                          \
           |                                            |
           |  +------+                         +------+  |
      CE3 ****| PE 1 | ----- R3 ---- R4 ------- | PE 2 |**** CE4
           |  +------+ **********************>  +------+  |
           |                                            |
            \                                          /
             _____/
```
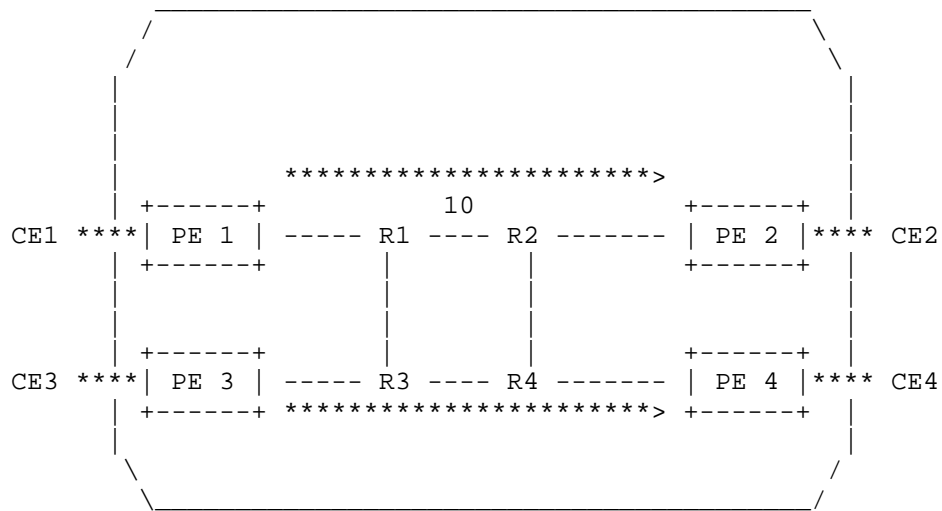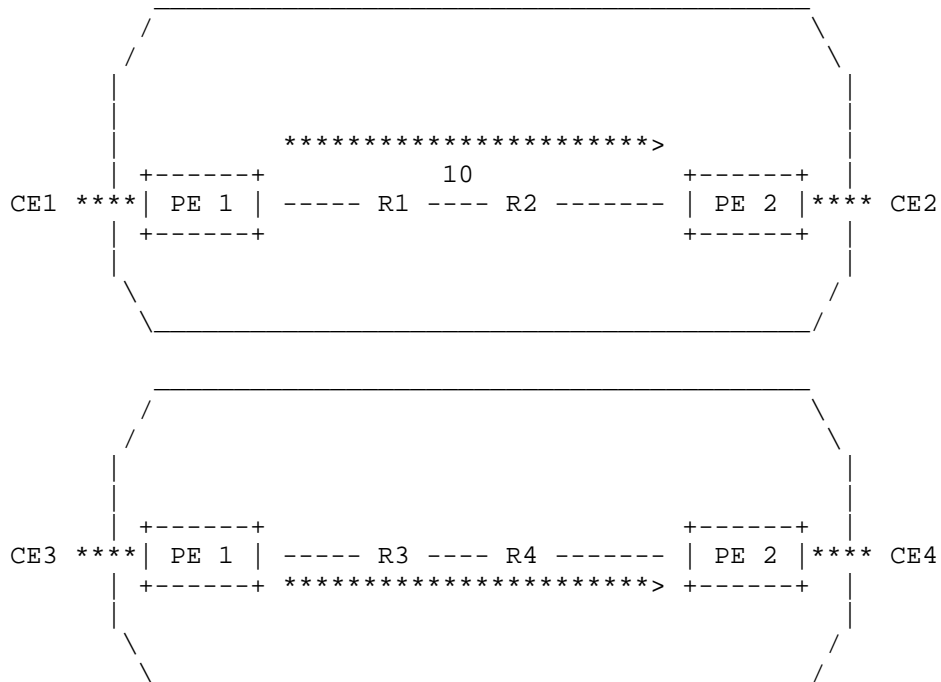
         Figure 2 - Disjoint paths provided by using two providers

   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. This is particularly required to avoid both
   protection mechanisms (customer level and provider level) to be
   activated at the same time which may lead to unpredictable side
   effects. However the service provider is allowed to restore the end-
   to-end path automatically when the primary path is failing by
   computing and installing a new primary path at the head-end. How the
   end-to-end protection is handled is out of scope of this document and
   will be under the customer responsibility.

   Another use case could be a service provider selling the traffic
   protection as a service option. So by default, the provided IP/MPLS
   path is not protected by any fast-reroute mechanism but the customer
   can subscribe to an option to activate fast-reroute for its traffic.
   In the figure 3, the Customer1 service between PE1 and PE2 is
   protected, in case of failure between R1 and R2, the LSP can use a
   bypass through R3-R4 nodes until the convergence occurs. The
   Customer2 did not subscribe to the traffic protection option. If

R3-R4 fails, the traffic between CE3 and CE4 will be disrupted until
the convergence occurs.

```
         _____
        /                                               \
       /                                                 \
      |                 Protected LSP                      |
      |      ********************>                          |
      | +------+          10               +------+        |
CE1 ****| PE 1 | ----- R1 ---- R2 ------- | PE 2 |**** CE2   Customer1
      | +------+          |*       *|       +------+        |
      |                   |*       *|                       |
      |                   |*       *|                       |
      |                   |*       *|                       |
      | +------+          |*******|         +------+        |
CE3 ****| PE 3 | ----- R3 ---- R4 ------- | PE 4 |**** CE4   Customer2
      | +------+ ********************> +------+              |
      |               Non protected LSP                     |
      |  \                                               /
       \  _____/
        _____/
```
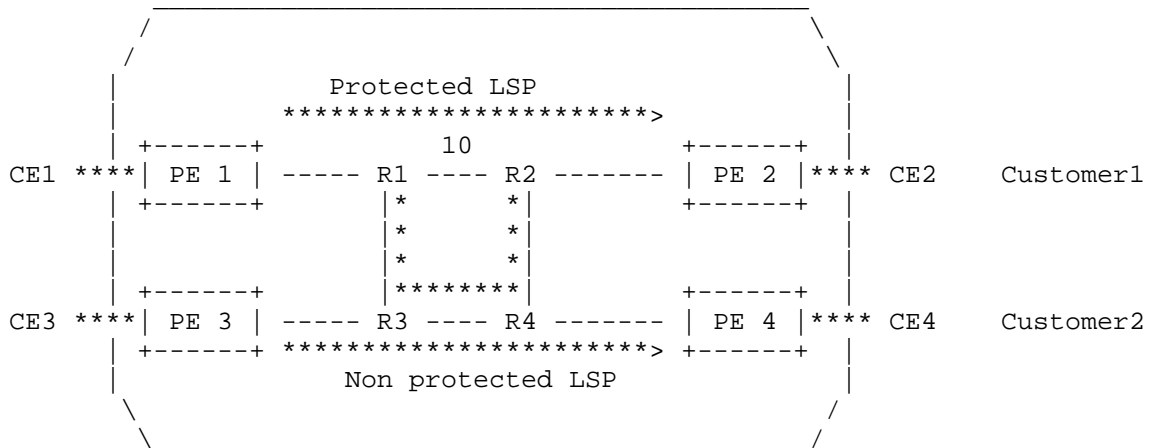
        Figure 3 - Provider selling traffic protection as an option

A service provider may also propose a traffic protection service
based on path protection rather than local repair on each transit
node. In the figure 4, on PE1, two LSPs were created to ensure the
customer traffic protection between PE1 and PE2. The primary LSP is
used to carry the traffic in the nominal situation. The protection
LSP is built as disjoint from the primary LSP and may be
preestablished (from controlplane and/or dataplane point of view).
When the primary LSP fails, PE1 is responsible to switch the traffic
to the protection LSP. As the protection is provided by PE1, both
primary and protection LSPs should be setup as non protected so
transit nodes will not activate any local-repair mechanism for those
LSPs.

```
              _____
             /                                        \
            /                                          \
            |          Primary LSP                     |
            |    *********************>                 |
            | +------+        10            +------+    |
    CE1 ****| PE 1 | ----- R1 ---- R2 ------- | PE 2 |**** CE2     Customer1
            | +------+      |       |        +------+    |
            |     *\        |       |        />          |
            |      *\       |       |       /*           |
            |       *\      |       |      /*            |
            |        *+- R3 ---- R4 ----+*              |
            |         *******************               |
            |          Protection LSP                   |
            |                                           |
             \                                         /
              _____/
```
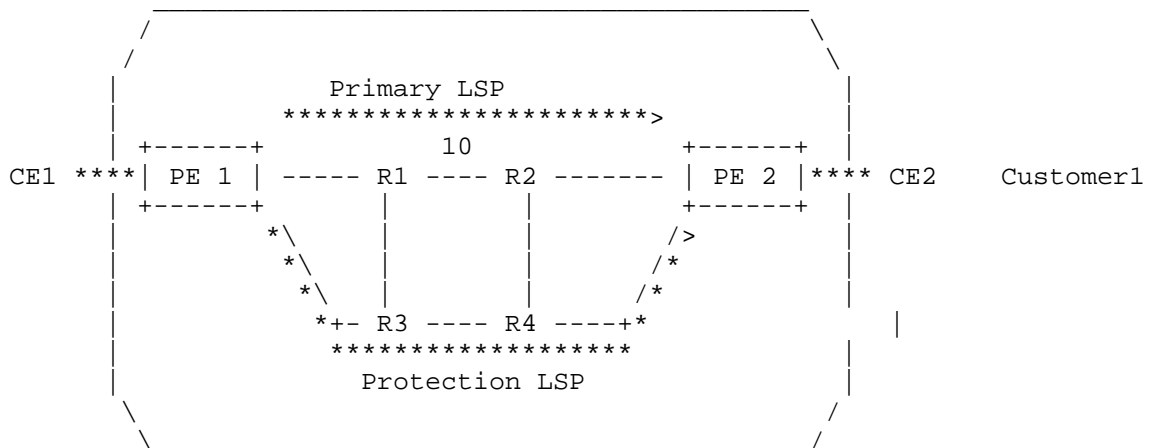
                Figure 4 - Provider selling traffic protection as an option

   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
   not protected, we need to ensure that the segment represented by
   Adj_R1R3 represents a non protected segment, as well as the segments
   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. Requirements for a non protected LSP

   o A non protected LSP SHOULD follow a primary path defined based on
      the constraints of the LSP. This path can be the shortest path
      (as per the IGP metric) or a more constrained path (explicit path)
      to fulfil for example a bandwidth, latency or disjointness
      requirement.

o Upon a failure, a non protected LSP SHOULD be reestablished over a
  new suitable non-protected path that still fulfils the constraints
  of the LSP.

o Upon a failure (link, node, srlg...), the traffic of a non
  protected LSP MUST NOT use any local-repair or any local-rerouting
  mechanism on transit nodes.

o The computation of a new primary path for the LSP will be handled
  by the computation node responsible of this LSP (it could be the
  head-end or a PCE).

o Upon any other traffic-engineering topology change (metric change,
  overload status change, bandwidth change, latency change...), the
  non protected LSP MAY be reoptimized to a better path.

2.1. ECMP considerations

When equal cost paths are available within the end-to-end path,
implementations may reuse a fast-reroute like mechanism in the
dataplane, so when one of the outgoing interface fails, the dataplane
switches traffic immediately to the remaining outgoing interfaces in
the ECMP set. This behavior is usually hardcoded and cannot be
disabled. Based on this assumption, a non protected LSP SHOULD avoid
ECMPs.

3. Options to create a non protected path with Segment Routing

3.1. Using only non protected adjacency segments

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 (including PCEs) 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 segments 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 responsible of the path. This
solution is fully compliant with the requirements sets in Section 2.

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 to encode the
SR tunnel. 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. As the adjacency SIDs have
a local significance, this resource consumption can be considered as
negligeable from a data plane point of view. From a control plane
point of view, this can also be considered as negligeable with the
current CPU and memory usually available on routers.

3.2. Using a combination of node segments and adjacency segments

Using a combination of node segments and adjacency segments is the
usual way of 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 use ECMPs 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.

3.2.1. 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 willingness of protection. Each node will then
advertise two Node-SIDs (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.

The remaining flag space in the Prefix-SID is small, so adding a new
flag requires analysis but this should not be considered as a
showstopper.

3.2.2. 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 use of
   a local-repair for a strict SPF Node-SID is allowed as long as the
   FRR mechanism enforces the post convergence path to the destination.

   This solution does not bring any benefit compared to the regular
   Node-SID (as it has similar properties).

3.2.3. 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.

   The path computation engine (head-end or PCE) must be aware of the
   policy defined by the service provider so it can select the right
   SIDs/prefixes when computing a path.

3.2.4. 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. Whereas having two adjacency SIDs introduces a
   negligeable impact, having two nodes SIDs increases controlplane and
   dataplane processing as each node in the network will have to install
   an MPLS->MPLS and IP->MPLS entry for each additional Node-SID. The
   regular IP convergence time of the network may be doubled in the
   worst case while the newly deployed node-SIDs are only used for
   traffic-engineering applications. 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. Note that there is a high chance for a

transit node to reroute faster than the head-end as it has usually
less computations to run (SPF+CSPFs) and less prefixes to rewrite; it
may also run less features leaving more CPU slots for IGP
reconvergence. The transient rerouting of the Node-SID may lead to
microloops in the network that may impact the customer traffic.
Node-SIDs are subject to ECMP and a local-repair mechanism may be
implemented for equal cost paths with no way to disable it. If the
requirement of preventing any local-repair or ECMP is strict, the
path computation engine needs to prevent the usage of all Node-SIDs
or needs to detect that a particular Node-SID will be subject to ECMP
and enforce the usage of additional adjacency SIDs to break the ECMP.
In any case, more adjacency-SIDs will be required in the stack to
avoid the ECMP, leading to a deeper label stack.

3.3. 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.

   PE1--P1   P6   P10--PE2
      \   / \   /
       P4-P5   P7 P9
                 \ /
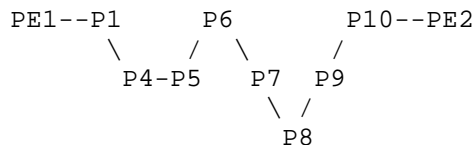   P8

                  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 label stack size can be

reduced to a small amount of labels at some price (creating some
states on transit nodes).

The drawbacks are the increase of the controlplane and dataplane
resource consumption. This controlplane and dataplane resource
consumption are variable and 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). Programming a
binding-SID on a transit node is feasible only if the programming
node has the necessary protocol sessions to do so. When a head-end
router is performing a path computation, it is usually not the case.
When a controller (PCE) is used, it may not have a session to all
LSRs in the network, as only edge nodes may require a path
computation. The controller may be limited for the placement of the
binding SID to the nodes it has a protocol session with (it cannot
setup a PCEP session by itself). A full deployment of protocol
sessions with the controller may not be feasible for technical
reasons (scaling, ...) or economical reasons. A potential mitigation
could be to allow protocol sessions to be setup dynamically (when
requirement comes) to an authorized subset of nodes in the network:
some protocol modifications may be necessary to allow this behavior.

4. Comparison

The following table tries to summarize the various solution pros/cons
within a comparison table:

o Solution 1: using adjacency-SIDs only

o Solution 2: using adjacency-SIDs + Node-SIDs with strict SPF
   algorithm

o Solution 3: using adjacency-SIDs + Node-SIDs with new protection
   flag

o Solution 4: using adjacency-SIDs + two regular Node-SIDs with a
   different policy

o Solution 5: using adjacency-SIDs + Binding-SIDs

We consider a network with N nodes and L links, with an average of l
links per node.

```
+----------+--------+-----------+------------+------------+---------+
| Criteria | Soluti |  Solution | Solution 3 | Solution 4 | Solutio |
```

	on 1	2			n 5
Label stack size	One label per hop	Reduced	Reduced	Reduced	Reduced
Controlp lane	Neglig ible	Potential additiona l computa tion + 2*N entries in RIB	+ 2*N entries in RIB	+ 2*N entries in RIB	Adds states in the LSRs
Dataplan e	+1 ent ries	+2*N entries	+2*N entries	+2*N entries	Variabl e
IP conve rgence time	None	Double	Double	Double	None
Computat ion engine	Needs to select Adj-SIDs with B=0	Needs to select Adj-SIDs with B=0 and Node-SIDs with strict SPF	Needs to select Adj-SIDs with B=0 and Node-SIDs with B=0	Needs to select Adj-SIDs with B=0 and needs to understand policy from the SP to select the right Node-SIDs	Needs to select Adj-SIDs with B=0 and place the binding SID in a smart way
Protocol	None	None	Need a new flag	None	None
ECMP avo idance	Suppor ted	Supported at the price of increasin g the label stack	Supported at the price of increasing the label stack	Supported at the price of increasing the label stack	Support ed
Requirem	Yes	Partially	Partially	Partially	Yes

ents ful filment		(allows ECMP+trans ient rerouting)	(allows ECMP+transie nt rerouting)	(allows ECMP+transie nt rerouting)	
Others	None	None	None	None	Requires a controller with sessions to all nodes (even transit)

Comparison of solutions

5. Recommended option(s)

Based on the analysis in Section 4, we only have two solutions that fulfill the requirements expressed in Section 2: usage of adjacency-SIDs only, usage of a combination of adjacency SIDs and binding SIDs.

As using only Adjacency-SIDs may reduce today the possibility of creating a path (due to the hardware/software limitations), authors would like to encourage the usage of a combination of adjacency-SIDs and binding-SIDs (Section 3.3) as a short-term solution.

However this approach has also several drawbacks, but authors think that these drawbacks can be reduced by enhancing existing protocols.

As a long term solution, authors would like to encourage vendors to support the ability for a node to push a significant number of labels, up to the full network diameter.

6. Security Considerations

TBD.

7. Acknowledgements

Authors would like to thank Bruno Decraene for his valuable comments.

8. IANA Considerations

N/A

9. 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-13 (work in progress), June
             2017.

   [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-18 (work in progress), July 2017.

   [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-12 (work in progress), June 2017.

   [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>.

Authors' Addresses

   Stephane Litkowski
   Orange

   Email: stephane.litkowski@orange.com

   Mustapha Aissaoui
   Nokia

   Email: mustapha.aissaoui@nokia.com