Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 3, 2018

C. Filsfils, Ed.
S. Previdi, Ed.
Cisco Systems, Inc.
J. Mitchell
Unaffiliated
E. Aries
Juniper Networks
P. Lapukhov
Facebook

October 30, 2017

BGP-Prefix Segment in large-scale data centers
draft-ietf-spring-segment-routing-msdc-06

Abstract

This document describes the motivation and benefits for applying segment routing in BGP-based large-scale data-centers. It describes the design to deploy segment routing in those data-centers, for both the MPLS and IPv6 dataplanes.

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].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Segment Routing (SR), as described in [I-D.ietf-spring-segment-routing] leverages the source routing paradigm. A node steers a packet through an ordered list of instructions, called segments. A segment can represent any instruction, topological or service-based. A segment can have a local semantic to an SR node or global within an SR domain. SR allows to enforce a flow through any topological path while maintaining per-flow state only at the ingress node to the SR domain. Segment Routing can be applied to the MPLS and IPv6 data-planes.

The use-cases described in this document should be considered in the context of the BGP-based large-scale data-center (DC) design described in [RFC7938]. This document extends it by applying SR both with IPv6 and MPLS dataplane.

2. Large Scale Data Center Network Design Summary

This section provides a brief summary of the informational document [RFC7938] that outlines a practical network design suitable for data-centers of various scales:

- Data-center networks have highly symmetric topologies with multiple parallel paths between two server attachment points. The well-known Clos topology is most popular among the operators (as described in [RFC7938]). In a Clos topology, the minimum number of parallel paths between two elements is determined by the "width" of the "Tier-1" stage. See Figure 1 below for an illustration of the concept.

- Large-scale data-centers commonly use a routing protocol, such as BGP-4 [RFC4271] in order to provide endpoint connectivity. Recovery after a network failure is therefore driven either by local knowledge of directly available backup paths or by distributed signaling between the network devices.

- Within data-center networks, traffic is load-shared using the Equal Cost Multipath (ECMP) mechanism. With ECMP, every network device implements a pseudo-random decision, mapping packets to one of the parallel paths by means of a hash function calculated over certain parts of the packet, typically a combination of various packet header fields.

The following is a schematic of a five-stage Clos topology, with four devices in the "Tier-1" stage. Notice that number of paths between Node1 and Node12 equals to four: the paths have to cross all of Tier-1 devices. At the same time, the number of paths between Node1 and Node2 equals two, and the paths only cross Tier-2 devices. Other topologies are possible, but for simplicity only the topologies that have a single path from Tier-1 to Tier-3 are considered below. The rest could be treated similarly, with a few modifications to the logic.

2.1. Reference design

```
                                   Tier-1
                                  +-----+
                                  |NODE |
                               +->|  5  |--+
                               |  +-----+  |
                       Tier-2  |           |   Tier-2
                      +-----+  |  +-----+  |  +-----+
        +------------>|NODE |--+->|NODE |--+--|NODE |-------------+
        |       +-----|  3  |--+  |  6  |  +--|  9  |-----+       |
        |       |     +-----+     +-----+     +-----+     |       |
        |       |                                         |       |
        |       |     +-----+     +-----+     +-----+     |       |
        | +-----+---->|NODE |--+  |NODE |  +--|NODE |-----+-----+ |
        | |     | +---|  4  |--+->|  7  |--+--|  10 |---+ |     | |
        | |     | |   +-----+  |  +-----+  |  +-----+   | |     | |
        | |     | |            |           |            | |     | |
      +-----+ +-----+          |  +-----+  |          +-----+ +-----+
      |NODE | |NODE | Tier-3   +->|NODE |--+   Tier-3 |NODE | |NODE |
      |  1  | |  2  |             |  8  |             | 11  | |  12 |
      +-----+ +-----+             +-----+             +-----+ +-----+
        | |     | |                                     | |     | |
        A O     B O            <- Servers ->            Z O     O O
```

Figure 1: 5-stage Clos topology

In the reference topology illustrated in Figure 1, It is assumed:

o Each node is its own AS (Node X has AS X). 4-byte AS numbers are recommended ([RFC6793]).

   * For simple and efficient route propagation filtering, Node5, Node6, Node7 and Node8 use the same AS, Node3 and Node4 use the same AS, Node9 and Node10 use the same AS.

   * In case of 2-byte autonomous system numbers are used and for efficient usage of the scarce 2-byte Private Use AS pool, different Tier-3 nodes might use the same AS.

   * Without loss of generality, these details will be simplified in this document and assume that each node has its own AS.

o Each node peers with its neighbors with a BGP session. If not specified, eBGP is assumed. In a specific use-case, iBGP will be used but this will be called out explicitly in that case.

o Each node originates the IPv4 address of its loopback interface into BGP and announces it to its neighbors.

   * The loopback of Node X is 192.0.2.x/32.

In this document, the Tier-1, Tier-2 and Tier-3 nodes are referred to respectively as Spine, Leaf and ToR (top of rack) nodes. When a ToR node acts as a gateway to the "outside world", it is referred to as a border node.

3. Some open problems in large data-center networks

The data-center network design summarized above provides means for moving traffic between hosts with reasonable efficiency. There are few open performance and reliability problems that arise in such design:

o ECMP routing is most commonly realized per-flow. This means that large, long-lived "elephant" flows may affect performance of smaller, short-lived "mouse" flows and reduce efficiency of per-flow load-sharing. In other words, per-flow ECMP does not perform efficiently when flow lifetime distribution is heavy-tailed. Furthermore, due to hash-function inefficiencies it is possible to have frequent flow collisions, where more flows get placed on one path over the others.

o Shortest-path routing with ECMP implements an oblivious routing model, which is not aware of the network imbalances. If the network symmetry is broken, for example due to link failures, utilization hotspots may appear. For example, if a link fails between Tier-1 and Tier-2 devices (e.g. Node5 and Node9), Tier-3 devices Node1 and Node2 will not be aware of that, since there are other paths available from perspective of Node3. They will continue sending roughly equal traffic to Node3 and Node4 as if the failure didn't exist which may cause a traffic hotspot.

- The absence of path visibility leaves transport protocols, such as TCP, with a "blackbox" view of the network. Some TCP metrics, such as SRTT, MSS, CWND and few others could be inferred and cached based on past history, but those apply to destinations, regardless of the path that has been chosen to get there. Thus, for instance, TCP is not capable of remembering "bad" paths, such as those that exhibited poor performance in the past. This means that every new connection will be established obliviously (memory-less) with regards to the paths chosen before, or chosen by other nodes.

- Isolating faults in the network with multiple parallel paths and ECMP-based routing is non-trivial due to lack of determinism. Specifically, the connections from HostA to HostB may take a different path every time a new connection is formed, thus making consistent reproduction of a failure much more difficult. This complexity scales linearly with the number of parallel paths in the network, and stems from the random nature of path selection by the network devices.

Further in this document (Section 7), it is demonstrated how these problems could be addressed within the framework of Segment Routing.

First, it will be explained how to apply SR in the DC, for MPLS and IPv6 data-planes.

4. Applying Segment Routing in the DC with MPLS dataplane

4.1. BGP Prefix Segment (BGP-Prefix-SID)

A BGP Prefix Segment is a segment associated with a BGP prefix. A BGP Prefix Segment is a network-wide instruction to forward the packet along the ECMP-aware best path to the related prefix.

The BGP Prefix Segment is defined as the BGP-Prefix-SID Attribute in [I-D.ietf-idr-bgp-prefix-sid] which contains an index. Throughout this document the BGP Prefix Segment Attribute is referred as the BGP-Prefix-SID and the encoded index as the label-index.

In this document, the network design decision has been made to assume that all the nodes are allocated the same SRGB (Segment Routing Global Block), e.g. [16000, 23999]. This provides operational simplification as explained in Section 9, but this is not a requirement.

For illustration purpose, when considering an MPLS data-plane, it is assumed that the label-index allocated to prefix 192.0.2.x/32 is X.

As a result, a local label (16000+x) is allocated for prefix 192.0.2.x/32 by each node throughout the DC fabric.

When IPv6 data-plane is considered, it is assumed that Node X is allocated IPv6 address (segment) 2001:DB8::X.

4.2. eBGP Labeled Unicast (draft-ietf-mpls-rfc3107bis)

Referring to Figure 1 and [RFC7938], the following design modifications are introduced:

- o Each node peers with its neighbors via a eBGP session with extensions defined in [I-D.ietf-mpls-rfc3107bis] (named "eBGP3107" throughout this document) and with the BGP-Prefix-SID attribute extension as defined in [I-D.ietf-idr-bgp-prefix-sid].

- o The forwarding plane at Tier-2 and Tier-1 is MPLS.

- o The forwarding plane at Tier-3 is either IP2MPLS (if the host sends IP traffic) or MPLS2MPLS (if the host sends MPLS-encapsulated traffic).

Figure 2 zooms into a path from server A to server Z within the topology of Figure 1.

```
                   +-----+     +-----+     +-----+
       +---------->|NODE |     |NODE |     |NODE |
       |           |  4  |--+->|  7  |--+--|  10 |---+
       |           +-----+     +-----+     +-----+   |
       |                                             |
   +-----+                                         +-----+
   |NODE |                                         |NODE |
   |  1  |                                         | 11  |
   +-----+                                         +-----+
     |                                              |
     A                    <- Servers ->             Z
```

Figure 2: Path from A to Z via nodes 1, 4, 7, 10 and 11

Referring to Figure 1 and Figure 2 and assuming the IP address with the AS and label-index allocation previously described, the following sections detail the control plane operation and the data plane states for the prefix 192.0.2.11/32 (loopback of Node11)

4.2.1. Control Plane

Node11 originates 192.0.2.11/32 in BGP and allocates to it a BGP-Prefix-SID with label-index: index11 [I-D.ietf-idr-bgp-prefix-sid].

Node11 sends the following eBGP3107 update to Node10:

. IP Prefix: 192.0.2.11/32
. Label: Implicit-Null
. Next-hop: Node11's interface address on the link to Node10
. AS Path: {11}
. BGP-Prefix-SID: Label-Index 11

Node10 receives the above update. As it is SR capable, Node10 is able to interpret the BGP-Prefix-SID and hence understands that it should allocate the label from its own SRGB block, offset by the Label-Index received in the BGP-Prefix-SID (16000+11 hence 16011) to the NLRI instead of allocating a non-deterministic label out of a dynamically allocated portion of the local label space. The implicit-null label in the NLRI tells Node10 that it is the penultimate hop and must pop the top label on the stack before forwarding traffic for this prefix to Node11.

Then, Node10 sends the following eBGP3107 update to Node7:

. IP Prefix: 192.0.2.11/32
. Label: 16011
. Next-hop: Node10's interface address on the link to Node7
. AS Path: {10, 11}
. BGP-Prefix-SID: Label-Index 11

Node7 receives the above update. As it is SR capable, Node7 is able to interpret the BGP-Prefix-SID and hence allocates the local (incoming) label 16011 (16000 + 11) to the NLRI (instead of allocating a "dynamic" local label from its label manager). Node7 uses the label in the received eBGP3107 NLRI as the outgoing label (the index is only used to derive the local/incoming label).

Node7 sends the following eBGP3107 update to Node4:

. IP Prefix: 192.0.2.11/32
. Label: 16011
. Next-hop: Node7's interface address on the link to Node4
. AS Path: {7, 10, 11}
. BGP-Prefix-SID: Label-Index 11

Node4 receives the above update. As it is SR capable, Node4 is able to interpret the BGP-Prefix-SID and hence allocates the local

(incoming) label 16011 to the NLRI (instead of allocating a "dynamic" local label from its label manager). Node4 uses the label in the received eBGP3107 NLRI as outgoing label (the index is only used to derive the local/incoming label).

Node4 sends the following eBGP3107 update to Node1:

. IP Prefix: 192.0.2.11/32
. Label: 16011
. Next-hop: Node4's interface address on the link to Node1
. AS Path: {4, 7, 10, 11}
. BGP-Prefix-SID: Label-Index 11

Node1 receives the above update. As it is SR capable, Node1 is able to interpret the BGP-Prefix-SID and hence allocates the local (incoming) label 16011 to the NLRI (instead of allocating a "dynamic" local label from its label manager). Node1 uses the label in the received eBGP3107 NLRI as outgoing label (the index is only used to derive the local/incoming label).

4.2.2. Data Plane

Referring to Figure 1, and assuming all nodes apply the same advertisement rules described above and all nodes have the same SRGB (16000-23999), here are the IP/MPLS forwarding tables for prefix 192.0.2.11/32 at Node1, Node4, Node7 and Node10.

Incoming label or IP destination	outgoing label	Outgoing Interface
16011	16011	ECMP{3, 4}
192.0.2.11/32	16011	ECMP{3, 4}

Figure 3: Node1 Forwarding Table

Incoming label or IP destination	outgoing label	Outgoing Interface
16011	16011	ECMP{7, 8}
192.0.2.11/32	16011	ECMP{7, 8}

Figure 4: Node4 Forwarding Table

Incoming label or IP destination	outgoing label	Outgoing Interface
16011	16011	10
192.0.2.11/32	16011	10

Figure 5: Node7 Forwarding Table

Incoming label or IP destination	outgoing label	Outgoing Interface
16011	POP	11
192.0.2.11/32	N/A	11

Node10 Forwarding Table

4.2.3. Network Design Variation

A network design choice could consist of switching all the traffic through Tier-1 and Tier-2 as MPLS traffic. In this case, one could filter away the IP entries at Node4, Node7 and Node10. This might be beneficial in order to optimize the forwarding table size.

A network design choice could consist in allowing the hosts to send MPLS-encapsulated traffic based on the Egress Peer Engineering (EPE) use-case as defined in [I-D.ietf-spring-segment-routing-central-epe]. For example, applications at HostA would send their Z-destined traffic to Node1 with an MPLS label stack where the top label is 16011 and the next label is an EPE peer segment ([I-D.ietf-spring-segment-routing-central-epe]) at Node11 directing the traffic to Z.

4.2.4. Global BGP Prefix Segment through the fabric

When the previous design is deployed, the operator enjoys global BGP-Prefix-SID and label allocation throughout the DC fabric.

A few examples follow:

o Normal forwarding to Node11: a packet with top label 16011 received by any node in the fabric will be forwarded along the ECMP-aware BGP best-path towards Node11 and the label 16011 is penultimate-popped at Node10 (or at Node 9).

o Traffic-engineered path to Node11: an application on a host behind Node1 might want to restrict its traffic to paths via the Spine node Node5. The application achieves this by sending its packets with a label stack of {16005, 16011}. BGP Prefix SID 16005 directs the packet up to Node5 along the path (Node1, Node3, Node5). BGP-Prefix-SID 16011 then directs the packet down to Node11 along the path (Node5, Node9, Node11).

4.2.5. Incremental Deployments

The design previously described can be deployed incrementally. Let us assume that Node7 does not support the BGP-Prefix-SID and let us show how the fabric connectivity is preserved.

From a signaling viewpoint, nothing would change: even though Node7 does not support the BGP-Prefix-SID, it does propagate the attribute unmodified to its neighbors.

From a label allocation viewpoint, the only difference is that Node7 would allocate a dynamic (random) label to the prefix 192.0.2.11/32 (e.g. 123456) instead of the "hinted" label as instructed by the BGP-Prefix-SID. The neighbors of Node7 adapt automatically as they always use the label in the BGP3107 NLRI as outgoing label.

Node4 does understand the BGP-Prefix-SID and hence allocates the indexed label in the SRGB (16011) for 192.0.2.11/32.

As a result, all the data-plane entries across the network would be unchanged except the entries at Node7 and its neighbor Node4 as shown in the figures below.

The key point is that the end-to-end Label Switched Path (LSP) is preserved because the outgoing label is always derived from the received label within the BGP3107 NLRI. The index in the BGP-Prefix-SID is only used as a hint on how to allocate the local label (the incoming label) but never for the outgoing label.

Incoming label or IP destination	outgoing label	Outgoing Interface
12345	16011	10

Figure 7: Node7 Forwarding Table

Incoming label or IP destination	outgoing label	Outgoing Interface
16011	12345	7

Figure 8: Node4 Forwarding Table

The BGP-Prefix-SID can thus be deployed incrementally one node at a time.

When deployed together with a homogeneous SRGB (same SRGB across the fabric), the operator incrementally enjoys the global prefix segment benefits as the deployment progresses through the fabric.

4.3. iBGP Labeled Unicast (draft-ietf-mpls-rfc3107bis)

The same exact design as eBGP3107 is used with the following modifications:

All nodes use the same AS number.

Each node peers with its neighbors via an internal BGP session (iBGP) with extensions defined in [I-D.ietf-mpls-rfc3107bis] (named "iBGP3107" throughout this document) and with the BGP-Prefix-SID attribute extension defined in this document.

Each node acts as a route-reflector for each of its neighbors and with the next-hop-self option. Next-hop-self is a well known operational feature which consists of rewriting the next-hop of a BGP update prior to send it to the neighbor. Usually, it's a common practice to apply next-hop-self behavior towards iBGP peers for eBGP learned routes. In the case outlined in this section it is proposed to use the next-hop-self mechanism also to iBGP learned routes.

```
                                  Cluster-1
                               +-----------+
                               |  Tier-1   |
                               |  +-----+  |
                               |  |NODE |  |
                               |  |  5  |  |
                    Cluster-2  |  +-----+  |  Cluster-3
                   +---------+ |           | +---------+
                   | Tier-2  | |           | |  Tier-2 |
                   | +-----+ | |  +-----+  | | +-----+ |
                   | |NODE | | |  |NODE |  | | |NODE | |
                   | |  3  | | |  |  6  |  | | |  9  | |
                   | +-----+ | |  +-----+  | | +-----+ |
                   |         | |           | |         |
                   |         | |           | |         |
                   | +-----+ | |  +-----+  | | +-----+ |
                   | |NODE | | |  |NODE |  | | |NODE | |
                   | |  4  | | |  |  7  |  | | |  10 | |
                   | +-----+ | |  +-----+  | | +-----+ |
                   +---------+ |           | +---------+
                               |           |
                               |  +-----+  |
                               |  |NODE |  |
             Tier-3            |  |  8  |  |         Tier-3
         +-----+ +-----+       |  +-----+  |      +-----+ +-----+
         |NODE | |NODE |       +-----------+      |NODE | |NODE |
         |  1  | |  2  |                          | 11  | |  12 |
         +-----+ +-----+                          +-----+ +-----+
```

Figure 9: iBGP Sessions with Reflection and Next-Hop-Self

For simple and efficient route propagation filtering and as illustrated in Figure 9:

Node5, Node6, Node7 and Node8 use the same Cluster ID (Cluster-1)

Node3 and Node4 use the same Cluster ID (Cluster-2)

Node9 and Node10 use the same Cluster ID (Cluster-3)

The control-plane behavior is mostly the same as described in the previous section: the only difference is that the eBGP3107 path propagation is simply replaced by an iBGP3107 path reflection with next-hop changed to self.

The data-plane tables are exactly the same.

5. Applying Segment Routing in the DC with IPv6 dataplane

The design described in [RFC7938] is reused with one single modification. It is highlighted using the example of the reachability to Node11 via spine node Node5.

Node5 originates 2001:DB8::5/128 with the attached BGP-Prefix-SID for IPv6 packets destined to segment 2001:DB8::5 ([I-D.ietf-idr-bgp-prefix-sid]).

Node11 originates 2001:DB8::11/128 with the attached BGP-Prefix-SID advertising the support of the SRH for IPv6 packets destined to segment 2001:DB8::11.

The control-plane and data-plane processing of all the other nodes in the fabric is unchanged. Specifically, the routes to 2001:DB8::5 and 2001:DB8::11 are installed in the FIB along the eBGP best-path to Node5 (spine node) and Node11 (ToR node) respectively.

An application on HostA which needs to send traffic to HostZ via only Node5 (spine node) can do so by sending IPv6 packets with a Segment Routing header (SRH, [I-D.ietf-6man-segment-routing-header]). The destination address and active segment is set to 2001:DB8::5. The next and last segment is set to 2001:DB8::11.

The application must only use IPv6 addresses that have been advertised as capable for SRv6 segment processing (e.g. for which the BGP prefix segment capability has been advertised). How applications learn this (e.g.: centralized controller and orchestration) is outside the scope of this document.

6. Communicating path information to the host

There are two general methods for communicating path information to the end-hosts: "proactive" and "reactive", aka "push" and "pull" models. There are multiple ways to implement either of these methods. Here, it is noted that one way could be using a centralized controller: the controller either tells the hosts of the prefix-to-path mappings beforehand and updates them as needed (network event driven push), or responds to the hosts making request for a path to specific destination (host event driven pull). It is also possible to use a hybrid model, i.e., pushing some state from the controller in response to particular network events, while the host pulls other state on demand.

It is also noted, that when disseminating network-related data to the end-hosts a trade-off is made to balance the amount of information Vs. the level of visibility in the network state. This applies both

to push and pull models. In the extreme case, the host would request path information on every flow, and keep no local state at all. On the other end of the spectrum, information for every prefix in the network along with available paths could be pushed and continuously updated on all hosts.

7. Addressing the open problems

This section demonstrates how the problems described above (in section 3) could be solved using the segment routing concept. It is worth noting that segment routing signaling and data-plane are only parts of the solution. Additional enhancements, e.g., such as the centralized controller mentioned previously, and host networking stack support are required to implement the proposed solutions. Also the applicability of the solutions described below are not restricted to the data-center alone, the same could be re-used in context of other domains as well.

7.1. Per-packet and flowlet switching

A flowlet is defined as a burst of packets from the same flow followed by an idle interval. [KANDULA04] developed a scheme that uses flowlets to split traffic across multiple parallel paths in order to optimize traffic load sharing.

With the ability to choose paths on the host, one may go from per-flow load-sharing in the network to per-packet or per-flowlet. The host may select different segment routing instructions either per packet, or per flowlet, and route them over different paths. This allows for solving the "elephant flow" problem in the data-center and avoiding link imbalances.

Note that traditional ECMP routing could be easily simulated with on-host path selection, using method proposed in [GREENBERG09]. The hosts would randomly pick a Tier-2 or Tier-1 device to "bounce" the packet off of, depending on whether the destination is under the same Tier-2 nodes, or has to be reached across Tier-1. The host would use a hash function that operates on per-flow invariants, to simulate per-flow load-sharing in the network.

Using Figure 1 as reference, let us illustrate this concept assuming that HostA has an elephant flow to HostZ called Flow-f.

Normally, a flow is hashed on to a single path. Let's assume HostA sends its packets associated with Flow-f with top label 16011 (the label for the remote ToR, Node11, where HostZ is connected) and Node1 would hash all the packets of Flow-F via the same next-hop (e.g. Node3). Similarly, let's assume that leaf Node3 would hash all the

packets of Flow-F via the same next-hop (e.g.: spine node Node5). This normal operation would restrict the elephant flow on a small subset of the ECMP paths to HostZ and potentially create imbalance and congestion in the fabric.

Leveraging the flowlet proposal, assuming HostA is made aware of 4 disjoint paths via intermediate segment 16005, 16006, 16007 and 16008 (the BGP prefix SID's of the 4 spine nodes) and also made aware of the prefix segment of the remote ToR connected to the destination (16011), then the application can break the elephant flow F into flowlets F1, F2, F3, F4 and associate each flowlet with one of the following 4 label stacks: {16005, 16011}, {16006, 16011}, {16007, 16011} and {16008, 16011}. This would spread the load of the elephant flow through all the ECMP paths available in the fabric and re-balance the load.

7.2. Performance-aware routing

Knowing the path associated with flows/packets, the end host may deduce certain characteristics of the path on its own, and additionally use the information supplied with path information pushed from the controller or received via pull request. The host may further share its path observations with the centralized agent, so that the latter may keep up-to-date network health map to assist other hosts with this information.

For example, an application A.1 at HostA may pin a TCP flow destined to HostZ via Spine node Node5 using label stack {16005, 16011}. The application A.1 may collect information on packet loss, deduced from TCP retransmissions and other signals (e.g. RTT increases). A.1 may additionally publish this information to a centralized agent, e.g. after a flow completes, or periodically for longer lived flows. Next, using both local and/or global performance data, application A.1 as well as other applications sharing the same resources in the DC fabric may pick up the best path for the new flow, or update an existing path (e.g.: when informed of congestion on an existing path).

One particularly interesting instance of performance-aware routing is dynamic fault-avoidance. If some links or devices in the network start discarding packets due to a fault, the end-hosts could probe and detect the path(s) that are affected and hence steer the affected flows away from the problem spot. Similar logic applies to failure cases where packets get completely black-holed, e.g., when a link goes down and the failure is detected by the host while probing the path.

For example, an application A.1 informed about 5 paths to Z {16005, 16011}, {16006, 16011}, {16007, 16011}, {16008, 16011} and {16011} might use the last one by default (for simplicity). When performance is degrading, A.1 might then start to pin TCP flows to each of the 4 other paths (each via a distinct spine) and monitor the performance. It would then detect the faulty path and assign a negative preference to the faulty path to avoid further flows using it. Gradually, over time, it may re-assign flows on the faulty path to eventually detect the resolution of the trouble and start reusing the path.

By leveraging Segment Routing, one avoids issues associated with oblivious ECMP hashing. For example, if in the topology depicted on Figure 1 a link between spine node Node5 and leaf node Node9 fails, HostA may exclude the segment corresponding to Node5 from the prefix matching the servers under Tier-2 devices Node9. In the push path discovery model, the affected path mappings may be explicitly pushed to all the servers for the duration of the failure. The new mapping would instruct them to avoid the particular Tier-1 node until the link has recovered. Alternatively, in pull path, the centralized controller may start steering new flows immediately after it discovers the issue. Until then, the existing flows may recover using local detection of the path issues.

7.3. Deterministic network probing

Active probing is a well-known technique for monitoring network elements' health, constituting of sending continuous packet streams simulating network traffic to the hosts in the data-center. Segment routing makes possible to prescribe the exact paths that each probe or series of probes would be taking toward their destination. This allows for fast correlation and detection of failed paths, by processing information from multiple actively probing agents. This complements the data collected from the hosts routing stacks as described in Section 7.2.

For example, imagine a probe agent sending packets to all machines in the data-center. For every host, it may send packets over each of the possible paths, knowing exactly which links and devices these packets will be crossing. Correlating results for multiple destinations with the topological data, it may automatically isolate possible problem to a link or device in the network.

8. Additional Benefits

8.1. MPLS Dataplane with operational simplicity

As required by [RFC7938], no new signaling protocol is introduced. The BGP-Prefix-SID is a lightweight extension to BGP Labeled Unicast (RFC3107 [RFC3107]). It applies either to eBGP or iBGP based designs.

Specifically, LDP and RSVP-TE are not used. These protocols would drastically impact the operational complexity of the Data Center and would not scale. This is in line with the requirements expressed in [RFC7938].

Provided the same SRGB is configured on all nodes, all nodes use the same MPLS label for a given IP prefix. This is simpler from an operation standpoint, as discussed in Section 9

8.2. Minimizing the FIB table

The designer may decide to switch all the traffic at Tier-1 and Tier-2's based on MPLS, hence drastically decreasing the IP table size at these nodes.

This is easily accomplished by encapsulating the traffic either directly at the host or the source ToR node by pushing the BGP-Prefix-SID of the destination ToR for intra-DC traffic, or the BGP-Prefix-SID for the the border node for inter-DC or DC-to-outside-world traffic.

8.3. Egress Peer Engineering

It is straightforward to combine the design illustrated in this document with the Egress Peer Engineering (EPE) use-case described in [I-D.ietf-spring-segment-routing-central-epe].

In such case, the operator is able to engineer its outbound traffic on a per host-flow basis, without incurring any additional state at intermediate points in the DC fabric.

For example, the controller only needs to inject a per-flow state on the HostA to force it to send its traffic destined to a specific Internet destination D via a selected border node (say Node12 in Figure 1 instead of another border node, Node11) and a specific egress peer of Node12 (say peer AS 9999 of local PeerNode segment 9999 at Node12 instead of any other peer which provides a path to the destination D). Any packet matching this state at host A would be encapsulated with SR segment list (label stack) {16012, 9999}. 16012 would steer the flow through the DC fabric, leveraging any ECMP, along the best path to border node Node12. Once the flow gets to

border node Node12, the active segment is 9999 (because of PHP on the upstream neighbor of Node12). This EPE PeerNode segment forces border node Node12 to forward the packet to peer AS 9999, without any IP lookup at the border node. There is no per-flow state for this engineered flow in the DC fabric. A benefit of segment routing is the per-flow state is only required at the source.

As well as allowing full traffic engineering control such a design also offers FIB table minimization benefits as the Internet-scale FIB at border node Node12 is not required if all FIB lookups are avoided there by using EPE.

8.4. Anycast

The design presented in this document preserves the availability and load-balancing properties of the base design presented in [I-D.ietf-spring-segment-routing].

For example, one could assign an anycast loopback 192.0.2.20/32 and associate segment index 20 to it on the border Node11 and Node12 (in addition to their node-specific loopbacks). Doing so, the EPE controller could express a default "go-to-the-Internet via any border node" policy as segment list {16020}. Indeed, from any host in the DC fabric or from any ToR node, 16020 steers the packet towards the border Node11 or Node12 leveraging ECMP where available along the best paths to these nodes.

9. Preferred SRGB Allocation

In the MPLS case, it is recommend to use same SRGBs at each node.

Different SRGBs in each node likely increase the complexity of the solution both from an operational viewpoint and from a controller viewpoint.

From an operation viewpoint, it is much simpler to have the same global label at every node for the same destination (the MPLS troubleshooting is then similar to the IPv6 troubleshooting where this global property is a given).

From a controller viewpoint, this allows us to construct simple policies applicable across the fabric.

Let us consider two applications A and B respectively connected to Node1 and Node2 (ToR nodes). A has two flows FA1 and FA2 destined to Z. B has two flows FB1 and FB2 destined to Z. The controller wants FA1 and FB1 to be load-shared across the fabric while FA2 and FB2 must be respectively steered via Node5 and Node8.

Assuming a consistent unique SRGB across the fabric as described in the document, the controller can simply do it by instructing A and B to use {16011} respectively for FA1 and FB1 and by instructing A and B to use {16005 16011} and {16008 16011} respectively for FA2 and FB2.

Let us assume a design where the SRGB is different at every node and where the SRGB of each node is advertised using the Originator SRGB TLV of the BGP-Prefix-SID as defined in [I-D.ietf-idr-bgp-prefix-sid]: SRGB of Node K starts at value K*1000 and the SRGB length is 1000 (e.g. Node1's SRGB is [1000, 1999], Node2's SRGB is [2000, 2999], ...).

In this case, not only the controller would need to collect and store all of these different SRGB's (e.g., through the Originator SRGB TLV of the BGP-Prefix-SID), furthermore it would need to adapt the policy for each host. Indeed, the controller would instruct A to use {1011} for FA1 while it would have to instruct B to use {2011} for FB1 (while with the same SRGB, both policies are the same {16011}).

Even worse, the controller would instruct A to use {1005, 5011} for FA1 while it would instruct B to use {2011, 8011} for FB1 (while with the same SRGB, the second segment is the same across both policies: 16011). When combining segments to create a policy, one need to carefully update the label of each segment. This is obviously more error-prone, more complex and more difficult to troubleshoot.

10. IANA Considerations

This document does not make any IANA request.

11. Manageability Considerations

The design and deployment guidelines described in this document are based on the network design described in [RFC7938].

The deployment model assumed in this document is based on a single domain where the interconnected DCs are part of the same administrative domain (which, of course, is split into different autonomous systems). The operator has full control of the whole domain and the usual operational and management mechanisms and procedures are used in order to prevent any information related to internal prefixes and topology to be leaked outside the domain.

As recommended in [I-D.ietf-spring-segment-routing], the same SRGB should be allocated in all nodes in order to facilitate the design, deployment and operations of the domain.

When EPE ([I-D.ietf-spring-segment-routing-central-epe]) is used (as explained in Section 8.3, the same operational model is assumed. EPE information is originated and propagated throughout the domain towards an internal server and unless explicitly configured by the operator, no EPE information is leaked outside the domain boundaries.

12. Security Considerations

This document proposes to apply Segment Routing to a well known scalability requirement expressed in [RFC7938] using the BGP-Prefix-SID as defined in [I-D.ietf-idr-bgp-prefix-sid].

It has to be noted, as described in Section 11 that the design illustrated in [RFC7938] and in this document, refer to a deployment model where all nodes are under the same administration. In this context, it is assumed that the operator doesn't want to leak outside of the domain any information related to internal prefixes and topology. The internal information includes prefix-sid and EPE information. In order to prevent such leaking, the standard BGP mechanisms (filters) are applied on the boundary of the domain.

Therefore, the solution proposed in this document does not introduce any additional security concerns from what expressed in [RFC7938] and [I-D.ietf-idr-bgp-prefix-sid]. It is assumed that the security and confidentiality of the prefix and topology information is preserved by outbound filters at each peering point of the domain as described in Section 11.

13. Acknowledgements

The authors would like to thank Benjamin Black, Arjun Sreekantiah, Keyur Patel, Acee Lindem and Anoop Ghanwani for their comments and review of this document.

14. Contributors

Gaya Nagarajan
Facebook
US

Email: gaya@fb.com

Dmitry Afanasiev
Yandex
RU

Email: fl0w@yandex-team.ru

Tim Laberge
Cisco
US

Email: tlaberge@cisco.com

Edet Nkposong
Salesforce.com Inc.
US

Email: enkposong@salesforce.com

Mohan Nanduri
Microsoft
US

Email: mnanduri@microsoft.com

James Uttaro
ATT
US

Email: ju1738@att.com

Saikat Ray
Unaffiliated
US

Email: raysaikat@gmail.com

15. References

15.1. Normative References

[I-D.ietf-idr-bgp-prefix-sid]
Previdi, S., Filsfils, C., Lindem, A., Sreekantiah, A., and H. Gredler, "Segment Routing Prefix SID extensions for BGP", draft-ietf-idr-bgp-prefix-sid-07 (work in progress), October 2017.

[I-D.ietf-mpls-rfc3107bis]
Rosen, E., "Using BGP to Bind MPLS Labels to Address Prefixes", draft-ietf-mpls-rfc3107bis-04 (work in progress), August 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.

[I-D.ietf-spring-segment-routing-central-epe]
Filsfils, C., Previdi, S., Aries, E., and D. Afanasiev, "Segment Routing Centralized BGP Egress Peer Engineering", draft-ietf-spring-segment-routing-central-epe-06 (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, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3107] Rekhter, Y. and E. Rosen, "Carrying Label Information in BGP-4", RFC 3107, DOI 10.17487/RFC3107, May 2001, <https://www.rfc-editor.org/info/rfc3107>.

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <https://www.rfc-editor.org/info/rfc4271>.

[RFC7311] Mohapatra, P., Fernando, R., Rosen, E., and J. Uttaro, "The Accumulated IGP Metric Attribute for BGP", RFC 7311, DOI 10.17487/RFC7311, August 2014, <https://www.rfc-editor.org/info/rfc7311>.

[RFC7938] Lapukhov, P., Premji, A., and J. Mitchell, Ed., "Use of BGP for Routing in Large-Scale Data Centers", RFC 7938, DOI 10.17487/RFC7938, August 2016, <https://www.rfc-editor.org/info/rfc7938>.

15.2. Informative References

[GREENBERG09]
Greenberg, A., Hamilton, J., Jain, N., Kadula, S., Kim, C., Lahiri, P., Maltz, D., Patel, P., and S. Sengupta, "VL2: A Scalable and Flexible Data Center Network", 2009.

[I-D.ietf-6man-segment-routing-header]
Previdi, S., Filsfils, C., Raza, K., Leddy, J., Field, B., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Matsushima, S., Leung, I., Linkova, J., Aries, E., Kosugi, T., Vyncke, E., Lebrun, D., Steinberg, D., and R. Raszuk, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-07 (work in progress), July 2017.

[I-D.ietf-mpls-seamless-mpls]
Leymann, N., Decraene, B., Filsfils, C., Konstantynowicz, M., and D. Steinberg, "Seamless MPLS Architecture", draft-ietf-mpls-seamless-mpls-07 (work in progress), June 2014.

[KANDULA04]
Sinha, S., Kandula, S., and D. Katabi, "Harnessing TCP's Burstiness with Flowlet Switching", 2004.

[RFC6793] Vohra, Q. and E. Chen, "BGP Support for Four-Octet Autonomous System (AS) Number Space", RFC 6793, DOI 10.17487/RFC6793, December 2012, <https://www.rfc-editor.org/info/rfc6793>.

Authors' Addresses

Clarence Filsfils (editor)
Cisco Systems, Inc.
Brussels
BE

Email: cfilsfil@cisco.com

Stefano Previdi (editor)
Cisco Systems, Inc.
Italy

Email: stefano@previdi.net

Jon Mitchell
Unaffiliated

Email: jrmitche@puck.nether.net

Ebben Aries
Juniper Networks
1133 Innovation Way
Sunnyvale CA 94089
US

Email: exa@juniper.net

Petr Lapukhov
Facebook
US

Email: petr@fb.com