Routing Area Working Group
Internet-Draft
Intended status: Informational
Expires: May 3, 2012

A. Atlas, Ed.
R. Kebler
M. Konstantynowicz
Juniper Networks
G. Enyedi
A. Csaszar
Ericsson
R. White
Cisco Systems
M. Shand

October 31, 2011

An Architecture for IP/LDP Fast-Reroute Using Maximally Redundant Trees
draft-atlas-rtgwg-mrt-frr-architecture-01

Abstract

As IP and LDP Fast-Reroute are increasingly deployed, the coverage limitations of Loop-Free Alternates are seen as a problem that requires a straightforward and consistent solution for IP and LDP, for unicast and multicast. This draft describes an architecture based on redundant backup trees where a single failure can cut a point-of-local-repair from the destination only on one of the pair of redundant trees.

One innovative algorithm to compute such topologies is maximally disjoint backup trees. Each router can compute its next-hops for each pair of maximally disjoint trees rooted at each node in the IGP area with computational complexity similar to that required by Dijkstra.

The additional state, address and computation requirements are believed to be significantly less than the Not-Via architecture requires.

Status of this Memo

Table of Contents

1. Introduction

There is still work required to completely provide IP and LDP Fast-Reroute[RFC5714] for unicast and multicast traffic. This draft proposes an architecture to provide 100% coverage.

Loop-free alternates (LFAs)[RFC5286] provide a useful mechanism for link and node protection but getting complete coverage is quite hard. [LFARevisited] defines sufficient conditions to determine if a network provides link-protecting LFAs and also proves that augmenting a network to provide better coverage is NP-hard. [I-D.ietf-rtgwg-lfa-applicability] discusses the applicability of LFA to different topologies with a focus on common PoP architectures.

While Not-Via [I-D.ietf-rtgwg-ipfrr-notvia-addresses] is defined as an architecture, in practice, it has proved too complicated and stateful to spark substantial interest in implementation or deployment. Academic implementations [LightweightNotVia] exist and have found the address management complexity high (but no standardization has been done to reduce this).

A different approach is needed and that is what is described here. It is based on the idea of using disjoint backup topologies as realized by Maximally Redundant Trees (described in [LightweightNotVia]); the general architecture could also apply to future improved redundant tree algorithms.

1.1. Goals for Extending IP Fast-Reroute coverage beyond LFA

Any scheme proposed for extending IPFRR network topology coverage beyond LFA, apart from attaining basic IPFRR properties, should also aim to achieve the following usability goals:

- ensure maximum physically feasible link and node disjointness regardless of topology,
- automatically compute backup next-hops based on the topology information distributed by link-state IGP,
- do not require any signaling in the case of failure and use pre-programmed backup next-hops for forwarding,
- introduce minimal amount of additional addressing and state on routers,
- enable gradual introduction of the new scheme and backward compatibility,

o and do not impose requirements for external computation.

2. Terminology

2-connected: A graph that has no cut-vertices. This is a graph that requires two nodes to be removed before the network is partitioned.

2-connected cluster: A maximal set of nodes that are 2-connected.

2-edge-connected: A network graph where at least two links must be removed to partition the network.

ADAG: Almost Directed Acyclic Graph - a graph that, if all links incoming to the root were removed, would be a DAG.

block: Either a 2-connected cluster, a cut-edge, or an isolated vertex.

cut-link: A link whose removal partitions the network. A cut-link by definition must be connected between two cut-vertices. If there are multiple parallel links, then they are referred to as cut-links in this document if removing the set of parallel links would partition the network.

cut-vertex: A vertex whose removal partitions the network.

DAG: Directed Acyclic Graph - a graph where all links are directed and there are no cycles in it.

GADAG: Generalized ADAG - a graph that is the combination of the ADAGs of all blocks.

Maximally Redundant Trees (MRT): A pair of trees where the path from any node X to the root R along the first tree and the path from the same node X to the root along the second tree share the minimum number of nodes and the minimum number of links. Each such shared node is a cut-vertex. Any shared links are cut-links. Any RT is an MRT but many MRTs are not RTs.

network graph: A graph that reflects the network topology where all links connect exactly two nodes and broadcast links have been transformed into the standard pseudo-node representation.

Redundant Trees (RT): A pair of trees where the path from any node X to the root R along the first tree is node-disjoint with the path from the same node X to the root along the second tree. These can be computed in 2-connected graphs.

3. Maximally Redundant Trees (MRT)

In the last few years, there's been substantial research on how to compute and use redundant trees. Redundant trees are directed spanning trees that provide disjoint paths towards their common root. These redundant trees only exist and provide link protection if the network is 2-edge-connected and node protection if the network is 2-connected. Such connectiveness may not be the case in real networks, either due to architecture or due to a previous failure. The work on maximally redundant trees has added two useful pieces that make them ready for use in a real network.

- o Computable regardless of network topology: The maximally redundant trees are computed so that only the cut-edges or cut-vertices are shared between the multiple trees.

- o Computationally practical algorithm is based on a common network topology database. Algorithm variants can compute in O(e) or O(e + n log n), as given in [I-D.enyedi-rtgwg-mrt-frr-algorithm].

There is, of course, significantly more in the literature related to redundant trees and even fast-reroute, but the formulation of the Maximally Redundant Trees (MRT) algorithm makes it very well suited to use in routers.

A known disadvantage of MRT, and redundant trees in general, is that the trees do not necessarily provide shortest detour paths. The use of the shortest-path-first algorithm in tree-building and including all links in the network as possibilities for one path or another should improve this. Modeling is underway to investigate and compare the MRT alternates to the optimal [I-D.enyedi-rtgwg-mrt-frr-algorithm]. Providing shortest detour paths would require failure-specific detour paths to the destinations, but the state-reduction advantage of MRT lies in the detour being established per destination (root) instead of per destination AND per failure.

The specific algorithm to compute MRTs as well as the logic behind that algorithm and alternative computational approaches are given in detail in [I-D.enyedi-rtgwg-mrt-frr-algorithm]. Those interested are highly recommended to read that document. This document describes how the MRTs can be used and not how to compute them.

The most important thing to understand about MRTs is that for each pair of destination-routed MRTs, there is a path from every node X to the destination D on the Blue MRT that is as disjoint as possible from the path on the Red MRT. The two paths along the two MRTs to a given destination-root of a 2-connected graph are node-disjoint, while in any non-2-connected graph, only the cut-vertices and cut-edges can be contained by both of the paths.

For example, in Figure 1, there is a network graph that is 2-connected in (a) and associated MRTs in (b) and (c). One can consider the paths from B to R; on the Blue MRT, the paths are B->F->D->E->R or B->F->C->E->R. On the Red MRT, the path is B->A->R. These are clearly link and node-disjoint. These MRTs are redundant trees because the paths are disjoint.

```
   [E]---[D]---|           [E]<--[D]<--|                [E]-->[D]---|
    |     |    |            |     ^    |                       |    |
    |     |    |            V     |    |                       V    V
   [R]   [F]  [C]          [R]   [F]  [C]              [R]   [F]  [C]
    |     |    |                  ^    ^                 ^     |    |
    |     |    |                  |    |                 |     V    |
   [A]---[B]---|           [A]-->[B]---|                [A]---[B]<--|

         (a)                     (b)                         (c)
   a 2-connected graph     Blue MRT towards R          Red MRT towards R
```

Figure 1: A 2-connected Network

By contrast, in Figure 2, the network in (a) is not 2-conneted. If F, G or the link F<->G failed, then the network would be partitioned. It is clearly impossible to have two link-disjoint or node-disjoint paths from G, I or J to R. The MRTs given in (b) and (c) offer paths that are as disjoint as possible. For instance, the paths from B to R are the same as in Figure 1 and the path from G to R on the Blue MRT is G->F->D->E->R and on the Red MRT is G->F->B->A->R.

```
                      [E]---[D]---|
                       |     |    |     |----[I]
                       |     |    |     |     |
                      [R]---[C]  [F]---[G]    |
                       |     |    |     |     |
                       |     |    |     |----[J]
                      [A]---[B]---|

                                  (a)
                        a non-2-connected graph

       [E]<--[D]<--|                        [E]-->[D]---|
        |     ^    |          [I]                 |    |          [I]
        V     |    |           ^                  V    V           |
       [R]<--[C]  [F]<--[G]    |            [R]---[C]  [F]<--[G]    |
              ^    ^     |     |             ^     |    |     ^     V
              |    |     |--->[J]            |     V    |     |----[J]
       [A]-->[B]---|                        [A]<--[B]<--|

                   (b)                                    (c)
            Blue MRT towards R                  Red MRT towards R
```

Figure 2: A non-2-connected network

4. Maximally Redundant Trees (MRT) and Fast-Reroute

In normal IGP routing, each router has its shortest-path-tree to all destinations. From the perspective of a particular destination, D, this looks like a reverse SPT (rSPT). To use maximally redundant trees, in addition, each destination D has two MRTs associated with it; by convention these will be called the blue and red MRTs.

MRTs are practical to maintain redundancy even after a single link or node failure. If a pair of MRTs is computed rooted at each destination, all the destinations remain reachable along one of the MRTs in the case of a single link or node failure.

When there is a link or node failure affecting the rSPT, each node will still have at least one path via one of the MRTs to reach the destination D. For example, in Figure 2, C would normally forward traffic to R across the C<->R link. If that C<->R link fails, then C could use either the Blue MRT path C->D->E->R or the Red MRT path C->B->A->R.

As is always the case with fast-reroute technologies, forwarding does not change until a local failure is detected. Packets are forwarded along the shortest path. The appropriate alternate to use is pre-computed. [I-D.enyedi-rtgwg-mrt-frr-algorithm] describes exactly how to determine whether the Blue MRT next-hops or the Red MRT next-hops should be the MRT alternate next-hops for a particular primary next-hop N to a particular destination D.

MRT alternates are always available to use, unless the network has been partitioned. It is a local decision whether to use an MRT alternate, a Loop-Free Alternate or some other type of alternate. When a network needs to use a micro-loop prevention mechanism [RFC5715] such as Ordered FIB[I-D.ietf-rtgwg-ordered-fib] or Farside Tunneling[RFC5715], then the whole IGP area needs to have alternates available so that the micro-loop prevention mechanism, which requires slower network convergence, can take the necessary time without impacting traffic badly.

As described in [RFC5286], when a worse failure than is anticipated happens, using LFAs that are not downstream neighbors can cause micro-looping. An example is given of link-protecting alternates causing a loop on node failure. Even if a worse failure than anticipated happened, the use of MRT alternates will not cause looping. Therefore, while node-protecting LFAs may be prefered, there are advantages to using MRT alternates when such a node-protecting LFA is not a downstream path.

4.1. Multi-homed Prefixes

One advantage of LFAs that is necessary to preserve is the ability to protect multi-homed prefixes against ABR failure. For instance, if a prefix from the backbone is available via both ABR A and ABR B, if A fails, then the traffic should be redirected to B. This can also be done for backups via MRT.

This generalizes to any multi-homed prefix. A multi-homed prefix could be:

- o An out-of-area prefix announced by more than one ABR,
- o An AS-External route announced by 2 or more ASBRs,
- o A prefix with iBGP multipath to different ASBRs,
- o etc.

For each prefix, the two lowest total cost ABRs are selected and a proxy-node is created connected to those two ABRs. If there exist multiple multi-homed prefixes that share the same two best connectivity, then a single proxy-node can be used to represent the set. An example of this is shown in Figure 3.

```
                    2    2                     2    2
                  A----B----C                A----B----C
                2 |         | 2            2 |         | 2
                  |         |                |         |
                [ABR1]    [ABR2]           [ABR1]    [ABR2]
                  |         |                |         |
                 p,10      p,15           10 |---[P]---| 15

                (a) Initial topology         (b)with proxy-node

                A<---B<---C                 A--->B--->C
                |         ^                 ^         |
                V         |                 |         V
              [ABR1]    [ABR2]            [ABR1]    [ABR2]
                |                                     |
                |-->[P]                         [P]<--|

                (c) Blue MRT                (d) Red MRT
```

Figure 3: Prefixes Advertised by Multiple ABRs

The proxy-nodes and associated links are added to the network topology after all real links have been assigned to a direction and before the actual MRTs are computed. Proxy-nodes cannot be transited when computing the MRTs. In addition to computing the pair of MRTs associated with each router destination D in the area, a pair of MRTs can be computed for each such proxy-node to fully protect against ABR failure.

Each ABR or attaching router must remove the MRT marking[see Section 4.2] and then forward the traffic outside of the area (or island of MRT-fast-reroute-supporting routers).

When directing traffic along an MRT towards a multi-homed prefix, if a topology-identifier label[see Section 4.2.1] is not used, then the proxy-node must be named and either additional LDP labels or IP addresses associated with it.

4.2. Unicast Forwarding with MRT Fast-Reroute

With LFA, there is no need to tunnel unicast traffic, whether IP or LDP. The traffic is simply sent to an alternate. The behavior with MRT Fast-Reroute is different depending upon whether IP or LDP unicast traffic is considered.

Logically, one could use the same IP address or LDP FEC and then also use 2 bits to express the topology to use. The topology options are (00) IGP/SPT, (01) blue MRT, (10) red MRT. Unfortunately, there just aren't 2 spare bits available in the IPv4 or IPv6 header. This has different consequences for IP and LDP because LDP can just add a topology label on top or take 2 spare bits from the label space.

Once the MRTs are computed, the two sets of MRTs are seen by the forwarding plane as essentially two additional topologies. The same considerations apply for forwarding along the MRTs as for handling multiple topologies.

4.2.1. LDP Unicast Forwarding - Avoid Tunneling

For LDP, it is very desirable to avoid tunneling because, for at least node protection, tunneling requires knowledge of remote LDP label mappings and thus requires targeted LDP sessions and the associated management complexity. There are two different mechanisms that can be used.

1. Option A - Encode Topology in Labels: In addition to sending a single label for a FEC, a router would provide two additional labels with their associated MRT colors. This is simple, but reduces the label space for other uses. It also increases the memory to store the labels and the communication required by LDP.

2. Option B - Create Topology-Identification Labels: Use the label-stacking ability of MPLS and specify only two additional labels - one for each associated MRT color - by a new FEC type. When sending a packet onto an MTR, first swap the LDP label and then push the topology-identification label for that MTR color. When receiving a packet with a topology-identification label, pop it and use it to guide the next-hop selection in combination with the next label in the stack; then swap the remaining label, if appropriate, and push the topology-identification label for the next-hop. This has minimal usage of additional labels, memory and LDP communication. It does increase the size of packets and the complexity of the required label operations and look-ups. This can use the same mechanisms as are needed for context-aware label spaces.

Note that with LDP unicast forwarding, regardless of whether topology-identification label or encoding topology in label is used, no additional loopbacks per router are required as are required in the IP unicast forwarding case. This is because LDP labels are used on a hop-by-hop basis to identify MRT-blue and MRT-red forwarding trees.

For greatest hardware compatibility, routers should support Option B of encoding the topology in the labels.

4.2.1.1. Protocol Extensions and Considerations: LDP

This captures an initial understanding of what may need to be specified.

1. Specify Topology in Label: When sending a Label Mapping, have the ability to send a Label TLV and multiple Topology-Label TLVs. The Topology-Label TLV would specify MRT and the associated MRT color.

2. Topology-Identification Labels: Define a new FEC type that describes the topology for MRT and the associated MRT color.

4.2.2. IP Unicast Traffic

For IP, there is no currently practical alternative except tunneling. The tunnel egress could be the original destination in the area, the next-next-hop, etc.. If the tunnel egress is the original destination router, then the traffic remains on the redundant tree with sub-optimal routing. If the tunnel egress is the next-next-hop, then protection of multi-homed prefixes and node-failure for ABRs is not available. Selection of the tunnel egress is a router-local decision.

There are three options available for marking IP packets with which MRT it should be forwarded in.

1. Tunnel IP packets via an LDP LSP. This has the advantage that more installed routers can do line-rate encapsulation and decapsulation. Also, no additional IP addresses would need to be allocated or signaled.

    A. Option A - LDP Destination-Topology Label: Use a label that indicates both destination and MRT. This method allows easy tunneling to the next-next-hop as well as to the IGP-area destination. For multi-homed prefixes, this requires that additional labels be advertised for each proxy-node.

    B. Option B - LDP Topology Label: Use a Topology-Identifier label on top of the IP packet. This is very simple and doesn't require additional labels for proxy-nodes. If tunneling to a next-next-hop is desired, then a two-deep label stack can be used with [Topology-ID label, Next-Next-Hop Label].

2. Tunnel IP packets in IP. Each router supporting this option would announce two additional loopback addresses and their associated MRT color. Those addresses are used as destination addresses for MRT-blue and MRT-red IP tunnels respectively. They allow the transit nodes to identify the traffic as being forwarded along either MRT-blue or MRT-red tree topology to reach the tunnel destination. Announcements of these two additional loopback addresses per router with their MRT color requires IGP extensions.

For proxy-nodes associated with one or more multi-homed prefixes, the problem is harder because there is no router associated with the proxy-node, so its loopbacks can't be known or used. In this case, each router attached to the proxy-node could announce two common IP addresses with their associated MRT colors. This would require configuration as well as the previously mentioned IGP extensions. Similarly, in the LDP case, two additional FEC bindings could be announced.

4.2.2.1. Protocol Extensions and Considerations: OSPF and ISIS

This captures an initial understanding of what may need to be specified.

- Capabilities: Does a router support MRT? Does the router do MRT tunneling with LDP or IP or GRE or...?

- Topology Association: A router needs to advertise a loopback and associate it with an MRT whether blue or red. Additional flexibility for future uses would be good.

- Proxy-nodes for Multi-homed Prefixes: We need a way to advertise common addresses with MRT for multi-homed prefixes' proxy-nodes. Currently, those proxy-nodes aren't named or considered.

As with LFA, it is expected that OSPF Virtual Links will not be supported.

4.2.3. Inter-Area and ABR Forwarding Behavior

In regular forwarding, packets destined outside the area arrive at the ABR and the ABR forwards them into the other area because the next-hops from the area with the best route (according to tie-breaking rules) are used by the ABR. The question is then what to do with packets marked with an MRT that are received by the ABR.

The only option that doesn't require forwarding based upon incoming interface is to forward an MRT marked packet in the area with the best route along its associated MRT. If the packet came from that area, this correctly avoids the failure. If the packet came from a different area, at least this gets the packet to the destination even though it is along an MRT rather than the shortest-path.

```
       +----[C]----     --[D]--[E]                --[D]--[E]
       |           \   /         \               /         \
   p--[A] Area 10 [ABR1]  Area 0 [H]--p   +-[ABR1]  Area 0 [H]-+
       |           /   \         /        |      \         /   |
       +----[B]----     --[F]--[G]        |       --[F]--[G]   |
                                          |                    |
                                          | other              |
                                          +----------[p]-------+
                                            area

         (a) Example topology        (b) Proxy node view in Area 0 nodes


                   +----[C]<---       [D]->[E]
                   V           \             \
                +-[A] Area 10 [ABR1]  Area 0 [H]-+
                |  ^           /             /   |
                |  +----[B]<---       [F]->[G]   V
                |                                |
                +------------->[p]<--------------+

                  (c) rSPT towards destination p


             ->[D]->[E]                         -<[D]<-[E]
            /          \                       /         \
       [ABR1]  Area 0 [H]-+             +-[ABR1]         [H]
                      /   |             |      \
               [F]->[G]   V             V       -<[F]<-[G]
                          |             |
                          |             |
                [p]<------+             +--------->[p]

     (d) Blue MRT in Area 0           (e) Red MRT in Area 0


                Figure 4: ABR Forwarding Behavior and MRTs
```

To avoid using an out-of-area MRT, special action can be taken by the penultimate router along the in-local-area MRT immediately before the ABR is reached. The penultimate router can determine that the ABR will forward the packet out of area and, in that case, the penultimate router can remove the MRT marking but still forward the packet along the MRT next-hop to reach the ABR. For instance, in Figure 4, if node H fails, node E has to put traffic towards prefix p onto the red MRT. But since node D knows that ABR1 will use a best from another area, it is safe for D to remove the MRT marking and just send the packet to ABR1 still on the red MRT but unmarked. ABR1 will use the shortest path in Area 10.

In all cases for ISIS and most cases for OSPF, the penultimate router can determine what decision the adjacent ABR will make. The one case where it can't be determined is when two ASBRs are in different non-backbone areas attached to the same ABR, then the ASBR's Area ID may be needed for tie-breaking (prefer the route with the largest OPSF area ID) and the Area ID isn't announced as part of the ASBR link-state advertisement (LSA). In this one case, suboptimal forwarding along the MRT in the other area would happen. If this is a realistic deployment scenario, OSPF extensions could be considered.

4.2.4. Issues with Area Abstraction

MRT fast-reroute provides complete coverage in a area that is 2-connected. Where a failure would partition the network, of course, no alternate can protect against that failure. Similarly, there are ways of connecting multi-homed prefixes that make it impractical to protect them without excessive complexity.

```
  50
 |----[ASBR Y]---[B]---[ABR 2]---[C]      Backbone Area 0:
 |                                |           ABR 1, ABR 2, C, D
 |                                |
 |                                |       Area 20:  A, ASBR X
 |                                |
 p ---[ASBR X]---[A]---[ABR 1]---[D]      Area 10: B, ASBR Y
    5                                  p is a Type 1 AS-external
```

Figure 5: AS external prefixes in different areas

Consider the network in Figure 5 and assume there is a richer connective topology that isn't shown, where the same prefix is announced by ASBR X and ASBR Y which are in different non-backbone areas. If the link from A to ASBR X fails, then an MRT alternate could forward the packet to ABR 1 and ABR 1 could forward it to D, but then D would find the shortest route is back via ABR 1 to Area 20. The only real way to get it from A to ASBR Y is to explicitly tunnel it to ASBR Y.

Tunnelling to the backup ASBR is for future consideration. The previously proposed PHP approach needs to have an exception if BGP policies (e.g. BGP local preference) determines which ASBR to use. Consider the case in Figure 6. If the link between A and ASBR X (the preferred border router) fails, A can put the packets to p onto an MRT alternate, even tunnel it towards ASBR Y. Node B, however, must not remove the MRT marking in this case, as nodes in Area 0, including ASBR Y itself would not know that their preferred ASBR is down.

```
                    Area 20                    BB Area 0
       p ---[ASBR X]-X-[A]---[B]---[ABR 1]---[D]---[ASBR Y]--- p

                    BGP prefers ASBR X for prefix p
```

Figure 6: Failure of path towards ASBR preferred by BGP

The fine details of how to solve multi-area external prefix cases, or identifying certain cases as too unlikely and too complex to protect is for further consideration.

4.2.5. Partial Deployment and Islands of Compatible MRT FRR routers

A natural concern with new functionality is how to have it be useful when it is not deployed across an entire IGP area. In the case of MRT FRR, where it provides alternates when appropriate LFAs aren't available, there are also deployment scenarios where it may make sense to only enable some routers in an area with MRT FRR. A simple example of such a scenario would be a ring of 6 or more routers that is connected via two routers to the rest of the area.

First, a computing router S must determine its local island of compatible MRT fast-reroute routers. A router that has common forwarding mechanisms and common algorithm and is connected to either to S or to another router already determined to be in S's local island can be added to S's local island.

Destinations inside the local island can obviously use MRT alternates. Destinations outside the local island can be treated like a multi-homed prefix with caveats to avoid looping. For LDP labels including both destination and topology, the routers at the borders of the local island need to originate labels for the original FEC and the associated MRT-specific labels. Packets sent to an LDP label marked as blue or red MRT to a destination outside the local island will have the last router in the local island swap the label to one for the destination and forward the packet along the outgoing interface on the MRT towards a router outside the local island that was represented by the proxy-node.

For IP in IP encapsulations, remote destinations may not be advertising additional IP loopback addresses for the MRTs. In that case, a router attached to a proxy-node, which represents destinations outside the local island, must advertise IP addresses associated with that proxy-node. Packets sent to an address associated with a proxy-node will have their outer IP header removed by the router attached to the proxy-node and be forwarded by the router along the outgoing interface on the MRT towards a router outside the local island that was represented by the proxy-node.

4.2.6. Network Convergence and Preparing for the Next Failure

After a failure, MRT detours ensure that packets reach their intended destination while the IGP has not reconverged onto the new topology. As link-state updates reach the routers, the IGP process calculates the new shortest paths. Two things need attention: micro-loop prevention and MRT re-calculation.

4.2.6.1. Micro-forwarding loop prevention and MRTs

As is well known[RFC5715], micro-loops can occur during IGP convergence; such loops can be local to the failure or remote from the failure. Managing micro-loops is an orthogonal issue to having alternates for local repair, such as MRT fast-reroute provides.

There are two possible micro-loop prevention mechanism discussed in [RFC5715]. The first is Ordered FIB [I-D.ietf-rtgwg-ordered-fib]. The second is Farside Tunneling which requires tunnels or an alternate topology to reach routers on the farside of the failure.

Since MRTs provide an alternate topology through which traffic can be sent and which can be manipulated separately from the SPT, it is possible that MRTs could be used to support Farside Tunneling. Details of how to do so are outside of this document.

4.2.6.2. MRT Recalculation

When a failure event happens, traffic is put by the PLRs onto the MRT topologies. After that, each router recomputes its shortest path tree (SPT) and moves traffic over to that. Only after all the PLRs have switched to using their SPTs and traffic has drained from the MRT topologies should each router install the recomputed MRTs into the FIBs.

At each router, therefore, the sequence is as follows:

1. Receive failure notification

2. Recompute SPT

3. Install new SPT

4. Recompute MRTs

5. Wait configured period for all routers to be using their SPTs and traffic to drain from the MRTs.

6. Install new MRTs.

While the recomputed MRTs are not installed in the FIB, protection coverage is lowered. Therefore, it is important to recalculate the MRTs and install them as quickly as possible.

It is for further study whether MRT re-calculation is possible in an incremental fashion, such that the sections of the MRT in use after a failure are not changed.

4.3. Multicast and MRT Fast-Reroute

There are several basic issues with doing Fast-Reroute for multicast traffic, whether the alternates used are LFA or MRT. They are given below:

1. The Point-of-Local-Repair (PLR) does not know the set of next-next-hops in the multicast tree.

2. A potential Merge Point(MP) does not know its previous-previous-hop in the multicast tree.

3. For mLDP, the PLR does not know the appropriate labels to use for the next-next-hops in the multicast tree.

4. The Merge Point (MP) does not know upon what interface to expect backup traffic. For LFAs, this is a particular issue since the LFA selected by a PLR is known only to that PLR.

Additionally, fast-reroute is to protect against a link failure, a node failure, or even local SRLG or general SRLG failures, but the mechanisms for such detection cannot distinguish easily between a link failure and a node failure (much less more complicated failures). In unicast forwarding, the assumption can be made that any failure is a node failure, unless the destination is the next-hop, and traffic is simply forwarded to the final destination avoiding the next-hop. For multicast, the final destination is not useful - what matters is the set of next-hop routers and the set of next-next-hop routers reached via each of the next-hop routers on the relevant multicast tree.

In multicast, it is possible that traffic is required by the next-hop as well as the next-next-hop and beyond. Therefore, whenever a local failure is detected and node protection is configured, it may be necessary to send traffic to both the affected next-hop routers and the set of next-next-hops reached via those next-hop routers.

4.3.1. Traffic Handling

When the PLR detects a failure, it forwards the multicast traffic on the link-protecting alternates. If node-protection is desired, then the traffic is also replicated to the node-protecting alternates.

The PLR sends traffic on the alternates for a configurable time-out. There is no clean way for the next-hop routers and/or next-next-hop routers to indicate that the traffic is no longer needed.

Critically, the potential Merge Point can independently determine whether to accept alternate traffic. If the primary upstream link(s) have failed, then accept and forward alternate traffic. When traffic is received on a new primary upstream link, stop accepting and forwarding alternate traffic.

This MP behavior involves a new action on detecting a local failure. When the local failure is detected, if that was the last primary upstream link, then the associated FIB entry for the alternate traffic is updated from discard to forward.

The final question is can anything be done about traffic missed due to different latencies along new primary and alternate/old primary trees? Any such techniques are outside the scope of this document.

4.3.2. PLR Replication and Tunneled

The disadvantages of tunneling unicast traffic do not fully translate to those for multicast. With MRT fast-reroute, IP unicast traffic is tunneled. With mLDP, with the suggested extensions, along with learning the next-next-hops on the multicast tree, the associated labels can be learned so there is no need for targeted sessions. If multicast traffic weren't tunneled, then multicast state would need to be created ahead of the failure along the alternate paths.

In this approach, the PLR tunnels multicast traffic into the unicast alternates destined to each particular MP. This is simply PLR-replication. For node-protection, the PLR learns of the MPs and their labels via protocol extensions[See Section 4.3.4.1 and Section 4.3.5].

The downside of PLR replication is that the same packets may appear multiple times on a link if they are tunneled to different destinations. The upside is that PLR replication avoids creating any alternate multicast state in the network.

4.3.3. Alternate Trees

To minimize replication of packets, it is possible to create alternate-trees. Each alternate-tree would be for a given PLR and neighbor - the alternate-tree would be failure-specific. It is not possible to merge alternate-trees for different PLRs or for different neighbors. This is shown in Figure 7 where G can't select an acceptable upstream node on the alternate tree that doesn't violate either the need to avoid C (for PLR A) or D (for PLR B).

```
          |-------[S]--------|          Alternate from A must avoid C
          V                  V          Alternate from B ust avoid D
         [A]------[E]-------[B]
          |        |         |
          V        |         V
      |--[C]------[F]-------[D]---|
      |   |                  |    |
      |   |-------[G]--------|    |
      |            |              |
      |            |              |
      |->[R1]-----[H]-------[R2]<-|

         (a) Multicast tree from S
       S->A->C->R1  and  S->B->D->R2
```

Figure 7: Alternate Trees from PLR A and B can't be merged

Backup Joins can be used to create the per-failure-point alternate trees. A Backup Join would indicate the PLR and the node to avoid. Each router that receives the Backup Join would determine which of the Blue MRT or Red MRT could offer an acceptable path and forward the traffic that way.

This method is still under investigation and consideration as its scaling properties are unfortunate.

4.3.3.1. Protocol Extensions

To create alternates from the potential Merge Points to the PLR and provide the MP and PLR with sufficient information, the following protocol extensions are needed.

- o Extend PIM and mLDP to signal Backup Joins: A backup Join can be sent from the MP towards the PLR going hop-by-hop.

- o Extend PIM and mLDP to send Join Confirmations with upstream router information. This provides the MP with information about the PLR for node protection scenarios.

4.3.4. PIM Forwarding

For node-protection, the merge points would be the next-next-hops in the tree. For a PLR to learn them, additional PIM Join Attributes [RFC5384] need to be defined to specify the set of next-hops from which the sending node has received Joins. For link-protection, of course a PLR knows the address of the neighbor.

PIM currently sends its JoinPrune messages periodically (60 seconds by default). Upon a change to the next-next hop list, the router can send a triggered JoinPrune with the updated Join Attribute, or it can wait for the next periodic refresh. It would be a tradeoff of increased control messages against a window of being unprotected.

Once the failure is detected, the PLR will send the traffic encapsulated to the list of downstream MPs. The PLR will send the encapsulated traffic for the duration of the protection-timeout. The protection-timer starts when the PLR detects a local failure. Once the timeout expires, the PLR can then prune upstream if there are no longer any receivers after the failure.

As is done today, the MP will forward traffic received on its normal incoming interface. If that interface fails, the MP will forward traffic if it is received with the correct encapsulation. After the incoming interface changes and new traffic arrives on the new incoming interface, received encapsulated traffic will not be forwarded until the protection-timer expires. This reduces sending of duplicate traffic at the cost of being briefly unprotected after a failure event.

4.3.4.1. Protocol Extensions and Considerations: PIM

This captures an initial understanding of what may need to be specified. This is focusing on PIM Sparse mode.

- o Capabilities: New Hello Option Capabilities to indicate the ability to understand the new Join Attributes.

- o Next-Hops: Need a new Join Attribute[RFC5384] to send the next-hops and the type of acceptable encapsulation to the PLR.

4.3.5. mLDP Forwarding

As in PIM, in mLDP[I-D.ietf-mpls-ldp-p2mp] a mechanism must be added so that the PLR can learn the next-next-hops. The PLR also needs to learn the associated label-bindings. This can be done via a new P2MP Child Data Object. This object would include the primary loopback of an LSR that has provided labels for the FEC to the sending LSR along with the label specified. Multiple P2MP Child Data Objects could be included in a P2MP Label Mapping; only those specified in the most recent P2MP Label Mapping should be stored and used.

This will provide the PLR with the MPs and their associated labels. The MPs will accept traffic received with that label from any interface, so no signaling is required before the alternates are used.

Traffic sent out each alternate will be tunneled with a destination of the MP.

4.4. Live-Live Multicast

In MoFRR [I-D.karan-mofrr], the idea of joining both a primary and a secondary tree is introduced with the requirement that the primary and secondary trees be link and node disjoint. This works well for networks where there are dual-planes, as explained in [I-D.karan-mofrr]. For other networks, it may still be desirable to have two disjoint multicast trees and allow a receiver to join both and make its own decision about what to do.

Using MRTs gives the ability to guarantee that the two trees are as disjoint as possible and to dynamically recompute the two MRTs whenever the topology changes.

Unlike for fast-reroute where the MRTs are rooted at the destination, with Live-Live Multicast, the MRTs would be routed at the multicast group source S. If the multicast source S is in a different area, then it could be represented via a proxy-node. If asymmetric link costs aren't a concern, then the same set of next-hops (previous-hops in this case) could be used as is used for MRT fast-reroute. A new P2MP FEC with Tree Identifier Element would need to be defined; it would include the topology to be used which could be IGP, MRT red, or MRT blue. For PIM, the existing PIM MT-ID Join

Attribute[I-D.ietf-pim-mtid] could be used to specify which MRT to use (blue or red).

For PIM, a different group could be used on the Blue MRT than on the Red MRT. Similarly, a different Opaque-Value could be used in mLDP for the Blue MRT and the Red MRT. Receiving routers would join both the blue MRT group and the red MRT group to receive traffic.

4.4.1. Forwarding Plane

If the two MRTs are not fully disjoint due to a network with a single point of failure, then traffic must self-identify as to which P2MP tree it belongs to. This means there must be a way to distinguish packets on the blue-MRT from the red-MRT. When different multicast groups are used, this is quite straightforward. For PIM, packets on the blue MRT would be destined to the group G-blue and packets on the red MRT would be destined to the group G-red. For mLDP, different labels will have been distributed for the Opaque-Value-blue and for the Opaque-Value-red.

RPF checks would still be enabled by the control plane. The control plane can program differnet forwarding entries on the G-blue incoming interface and on the G-red incoming interface. The other interfaces would still discard both G-blue and G-red traffic.

The receiver would still need to detect failures and handle traffic discarding as is specified in [I-D.karan-mofrr].

5. Acknowledgements

The authors would like to thank Hannes Gredler, Jeff Tantsura, Ted Qian, Kishore Tiruveedhula, Santosh Esale, Nitin Bahadur, Harish Sitaraman and Raveendra Torvi for their suggestions and review.

6. IANA Considerations

This doument includes no request to IANA.

7. Security Considerations

This architecture is not currently believed to introduce new security concerns.

8. References

8.1. Normative References

[I-D.enyedi-rtgwg-mrt-frr-algorithm]
Atlas, A., Envedi, G., and A. Csaszar, "Algorithms for computing Maximally Redundant Trees for IP/LDP Fast-Reroute", draft-enyedi-rtgwg-mrt-frr-algorithm-00 (work in progress), October 2011.

[I-D.ietf-mpls-ldp-p2mp]
Minei, I., Wijnands, I., Kompella, K., and B. Thomas, "Label Distribution Protocol Extensions for Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths", draft-ietf-mpls-ldp-p2mp-15 (work in progress), August 2011.

[I-D.ietf-pim-mtid]
Cai, Y. and H. Ou, "PIM Multi-Topology ID (MT-ID) Join Attribute", draft-ietf-pim-mtid-10 (work in progress), September 2011.

[I-D.karan-mofrr]
Karan, A., Filsfils, C., Farinacci, D., Decraene, B., Leymann, N., and T. Telkamp, "Multicast only Fast Re-Route", draft-karan-mofrr-01 (work in progress), March 2011.

[RFC5286] Atlas, A. and A. Zinin, "Basic Specification for IP Fast Reroute: Loop-Free Alternates", RFC 5286, September 2008.

[RFC5384] Boers, A., Wijnands, I., and E. Rosen, "The Protocol Independent Multicast (PIM) Join Attribute Format", RFC 5384, November 2008.

[RFC5714] Shand, M. and S. Bryant, "IP Fast Reroute Framework", RFC 5714, January 2010.

8.2. Informative References

[I-D.ietf-rtgwg-ipfrr-notvia-addresses]
Shand, M., Bryant, S., and S. Previdi, "IP Fast Reroute Using Not-via Addresses", draft-ietf-rtgwg-ipfrr-notvia-addresses-07 (work in progress), April 2011.

[I-D.ietf-rtgwg-lfa-applicability]
Filsfils, C., Francois, P., Shand, M., Decraene, B., Uttaro, J., Leymann, N., and M. Horneffer, "LFA applicability in SP networks",

draft-ietf-rtgwg-lfa-applicability-03 (work in progress), August 2011.

[I-D.ietf-rtgwg-ordered-fib]
Shand, M., Bryant, S., Previdi, S., and C. Filsfils, "Loop-free convergence using oFIB", draft-ietf-rtgwg-ordered-fib-05 (work in progress), April 2011.

[LFARevisited]
Retvari, G., Tapolcai, J., Enyedi, G., and A. Csaszar, "IP Fast ReRoute: Loop Free Alternates Revisited", Proceedings of IEEE INFOCOM , 2011, <http://opti.tmit.bme.hu/˜tapolcai/papers/retvari2011lfa_infocom.pdf>.

[LightweightNotVia]
Enyedi, G., Retvari, G., Szilagyi, P., and A. Csaszar, "IP Fast ReRoute: Lightweight Not-Via without Additional Addresses", Proceedings of IEEE INFOCOM , 2009, <http://mycite.omikk.bme.hu/doc/71691.pdf>.

[RFC5715] Shand, M. and S. Bryant, "A Framework for Loop-Free Convergence", RFC 5715, January 2010.

Authors' Addresses

Alia Atlas (editor)
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
USA

Email: akatlas@juniper.net

Robert Kebler
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
USA

Email: rkebler@juniper.net

Maciek Konstantynowicz
Juniper Networks

Email: maciek@juniper.net

Gabor Sandor Enyedi
Ericsson
Konyves Kalman krt 11.
Budapest 1097
Hungary

Email: Gabor.Sandor.Enyedi@ericsson.com

Andras Csaszar
Ericsson
Konyves Kalman krt 11
Budapest 1097
Hungary

Email: Andras.Csaszar@ericsson.com

Russ White
Cisco Systems

Email: russwh@cisco.com

Mike Shand

Email: mike@mshand.org.uk

Network Working Group
Internet Draft
Intended status: Standards Track
Expires: April 30, 2012

A. Csaszar (Ed.)
G. Enyedi
J. Tantsura
S. Kini
Ericsson

J. Sucec
S. Das
Telcordia

October 30, 2011

IP Fast Re-Route with Fast Notification

draft-csaszar-ipfrr-fn-02.txt

Status of this Memo

Copyright Notice

Abstract

This document describes a mechanism that provides IP fast reroute (IPFRR) by using a failure notification (FN) to nodes beyond the ones that first detect the failure (i.e. nodes that are directly connected to the failure point). The paths used when IPFRR-FN is active are in most cases identical to those used after Interior Gateway Protocol (IGP) convergence. The proposed mechanism can address all single link, node, and SRLG failures in an area and has been designed to allow traffic recovery traffic to happen quickly (The goal being to keep traffic loss under 50msec). IPFRR-FN can be a supplemental tool to provide FRR when LFA cannot repair a failure case.

Table of Contents

1. Introduction

Convergence of link-state IGPs, such as OSPF or IS-IS, after a link or node failure is known to be relatively slow. While this may be sufficient for many applications, some network SLAs and applications require faster reaction to network failures.

IGP convergence time is composed mainly of:

1. Failure detection at nodes adjacent to the failure

2. Advertisement of the topology change

3. Calculation of new routes

4. Installing new routes to linecards

Traditional Hello-based failure detection methods of link-state IGPs are relatively slow, hence a new, optimized, Hello protocol has been standardized [BFD] which can reduce failure detection times to the range of 10ms even if no lower layer notices the failure quickly (like loss of signal, etc.).

Even with fast failure detection, reaction times of IGPs may take several seconds, and even with a tuned configuration it may take at least a couple of hundreds of milliseconds.

To decrease fail-over time even further, IPFRR techniques [RFC5714], can be introduced. IPFRR solutions compliant with [RFC5714] are targeting fail-over time reduction of steps 2-4 with the following design principles:

IGP		IPFRR
2. Advertisement of the topology change	==>	No explicit advertisement, only local repair
3. Calculation of new routes	==>	Pre-computation of new routes
4. Installing new routes to linecards	==>	Pre-installation of backup routes

Pre-computing means that the way of bypassing a failed resource is computed before any failure occurs. In order to limit complexity, IPFRR techniques typically prepare for single link, single node and single Shared Risk Link Group (SRLG) failures, which failure types are undoubtedly the most common ones. The pre-calculated backup routes are also downloaded to linecards in preparation for the failure, in this way sparing the lengthy communication between control plane and data plane when a failure happens.

The principle of local rerouting requires forwarding a packet along a detour even if only the immediate neighbors of the failed resource know the failure. IPFRR methods observing the local rerouting principle do not explicitly propagate the failure information. Unfortunately, packets on detours must be handled in a different way than normal packets as otherwise they might get returned to the failed resource. Rephrased, a node not having *any* sort of information about the failure may loop the packet back to the node from where it was rerouted - simply because its default routing/forwarding configuration dictates that. As an example, see the following figure. Assuming a link failure between A and Dst, A needs to drop packets heading to Dst. If node A forwarded packets to Src, and if the latter had absolutely no knowledge of the failure, a loop would be formed between Src and A.

```
          +---+            +---+
          | B |------------| C |
          +---+            +---+
         /                      \
        /                        \
       /                          \
   +---+          +---+  failure   +---+
   |Src|----------| A |-----X------|Dst|
   +---+          +---+            +---+
     =========>==============>=========>
                Primary path
```

Figure 1 Forwarding inconsistency in case of local repair: The path of Src to Dst leads through A

The basic problem that previous IPFRR solutions struggle to solve is, therefore, to provide consistent routing hop-by-hop without explicit signaling of the failure.

To provide protection for all single failure cases in arbitrary topologies, the information about the failure must be given in *some* way to other nodes. That is, IPFRR solutions targeting full failure coverage need to signal the fact and to some extent the identity of the failure within the data packet as no explicit signaling is allowed. Such solutions have turned out to be considerably complex and hard or impossible to implement practically. The Loop Free Alternates (LFA) solution [RFC5286] does not give the failure information in any way to other routers, and so it cannot repair all failure cases such as the one in Figure 1.

As discussed in Section 2. solutions that address full failure coverage and rely on local repair, i.e. carrying some failure information within the data packets, fail to present a practical alternative to LFA. This draft, therefore, suggests that relaxing the local re-routing principle with carefully engineered explicit failure signaling is an effective approach.

The idea of using explicit failure notification for IPFRR has been proposed before for Remote LFA Paths [RLFAP]. RLFAP sends explicit notifications and can limit the radius in which the notification is propagated to enhance scalability. Design, implementation and enhancements for the remote LFAP concept are reported in [Hok2007], [Hok2008] and [Cev2010].

This draft attempts to work out in more detail what kind of failure dissemination mechanism is required to facilitate remote repair efficiently. Requirements for explicit signaling are given in Section 3. This draft does not limit the failure advertisement radius as opposed to RLFAP. As a result, the detour paths remain stable in most cases, since they are identical to those that the IGP will calculation after IGP convergence. Hence, micro-loop will not occur after IGP convergence.

2. Overview of current IPFRR Proposals based on Local Repair

The only practically feasible solution, Loop Free Alternates [RFC5286], offers the simplest resolution of the consistency problem: a node performing fail-over may only use a next-hop as backup if it is guaranteed that it does not send the packets back. These neighbors are called Loop-Free Alternates (LFA). LFAs, however, do not always exist, as shown in Figure 1 above, i.e., node A has no LFAs with respect to Dst. while it is true that tweaking the network configuration may boost LFA failure case coverage considerably [Ret2011], LFAs cannot protect all failure cases in arbitrary network topologies.

The exact way of adding the information to data packets and its usage for forwarding is the most important property that differentiates most existing IPFRR proposals.

Packets can be marked "implicitly", when they are not altered in any way, but some extra information owned by the router helps deciding the correct way of forwarding. Such extra information can be for instance the direction of the packet, e.g., the interface, which the packet arrived through, e.g. as in [FIFR]. Such solutions require what is called interface-based or interface-specific forwarding.

Interface-based forwarding significantly changes the well-established nature of IP's destination-based forwarding principle, where the IP destination address alone describes the next hop. One embodiment would need to download different FIBs for each physical or virtual IP interface - not a very compelling idea. Another embodiment would alter the next-hop selection process by adding the incoming interface id also to the lookup fields, which would impact forwarding performance considerably.

Other solutions mark data packets explicitly. Some proposals suggest using free bits in the IP header [MRC], which unfortunately do not exist in the IPv4 header. Other proposals resort to encapsulating re-routed packets with an additional IP header as in e.g. [NotVia] or [Eny2009]. Encapsulation raises the problem of fragmentation and reassembly, which could be a performance bottleneck, if many packets are sent at MTU size. Another significant problem is the additional management complexity of the encapsulation addresses, which have their own semantics and need to be calculated in a failure specific manner.

3. Requirements of an Explicit Failure Signaling Mechanism

Any signaling mechanism which should be used to advertise failure notifications and so to facilitate extremely quick remote repair should have the following properties.

1. The signaling mechanism should be reliable. The mechanism needs to propagate the failure information to all interested nodes even in a network where a single link or a node is down.

2. The mechanism should be fast in the sense that getting the notification packet to remote nodes through possible multiple hops should not require (considerably) more processing at each hop than plain fast path packet forwarding.

3. The mechanism should involve simple and efficient processing to be feasible for implementation in the dataplane. This goal manifests itself in three ways: Origination of notification should be very easy, e.g. creating a simple IP packet, the payload of which can be filled easily. When receiving the packet, it should be easy to recognize by dataplane linecards so that processing can commence after forwarding. No complex operations should be required in order to extract the information from the packet needed to activate the correct backup routes.

4. The mechanism should be trustable; that is, it should provide means to verify the authenticity of the notifications without significant increase of the processing burden in the dataplane.

5. Duplication of notification packets should be either strictly bounded or handled without significant dataplane processing burden.

These requirements present a trade-off. A proper balance needs to be found that offers good enough authentication and reliability while keeping processing complexity sufficiently low to be feasible for data plane implementation. One such solution is proposed in [fn-transport], which is the assumed notification protocol in the following.

4. Conceptual Operation of IPFRR relying on Fast Notification

This section outlines the operation of an IPFRR mechanism relying on Fast Notification.

4.1. Preparation Phase

Like each IPFRR solution, here it is also required to have means for quick failure detection in place, such as lower layer upcalls or BFD. The FN service needs to be activated and configured. The FN service should be bound to failure detection in such a way that FN can disseminate the information identifying the failure to the area.

Based on the detailed topology database obtained by a link state IGP, the node should calculate alternative paths considering *relevant* link or node failures in the area. Failure specific alternative path computation should typically be executed at lower priority than other routing processing. Note that the calculation can be done "offline", while the network is intact and the CP has few things to do.

Also note the word *relevant* above: a node does not needed to compute all the shortest paths with respect to each possible failure; only those link failures need to be taken into consideration, which are in the shortest path tree starting from the node.

To provide protection for Autonomous System Border Router (ASBR) failures, the node will need information not only from the IGP but also from BGP. This is described in detail in Section 5.3.

After having calculated the failure specific alternative next-hops, only those which represent a change to the primary next-hop, should be pre-installed to the linecards together with the identifier of the failure, which triggers the switch-over. (The resource needs of an example implementation are briefly discussed in Appendix A.)

4.2. Failure Reaction Phase

The main steps to be taken after a failure are the following:

1. Quick dataplane failure detection

2. Send information about failure using FN service right from dataplane.

3. Forward the received notification as defined by the actually used FN protocol such as the one in [fn-transport]

4. After learning about a local or remote failure, identify failure and activate failure specific backups, if needed, directly within dataplane

5. Start forwarding data traffic using the updated FIB

After a node detects the loss of connectivity to another node, it should make a decision whether the failure can be handled locally. If local repair is not possible or not configured, for example because LFA is not configured or there are destinations for which no LFA exists, a failure should trigger the FN service to disseminate the failure description. For instance, if BFD detects a dataplane failure it not only should invoke routines to notify the control plane but it should first trigger FN before notifying the CP.

After receiving the trigger, without any DP-CP communication involved, FN constructs a packet and adds the description of the failure (described in Section 5.1.) to the payload. The contains the information that

o a node X has lost connectivity

o to a node Z

o via a link L.

The proposed encoding of the IPFRR-FN packet is described in Section 5.1.

The packet is then disseminated by the FN service in the routing area. Note the synergy of the relation between BFD and IGP Hellos and between FN and IGP link state advertisements. BFD makes a dataplane optimized implementation of the routing protocol's Hello mechanism, while Fast Notification makes a dataplane optimized implementation of the link state advertisement flooding mechanism of IGPs.

In each hop, the recipient node needs to perform a "punt and forward". That is, the FN packet not only needs to be forwarded to the FN neighbors as the specific FN mechanism dictates, but a replica needs to be detached and, after forwarding, started to be processed by the dataplane card.

4.2.1. Activating Failure Specific Backups

After the forwarding element extracted the contents of the notification packet, it knows that a node X has lost connectivity to a node Z via a link L. The recipient now needs to decide whether the failure was a link or a node failure. Two approaches can be thought of. Both options are based on the property that notifications advance in the network as fast as possible.

In the first option, the router does not immediately make the decision, but instead starts a timer set to fire after a couple of milliseconds. If, the failure was a node failure, the node will receive further notifications saying that another node Y has lost connectivity to node Z through another link M. That is, if node Z is common in the notifications, the recipient can conclude that it is a node failure and already knows which node it is (Z). If link L is common in the notifications, then the recipient can decide for link failure (L). If further inconclusive notifications arrive, then it means multiple failures which case is not in scope for IPFRR, and is left for regular IGP convergence.

After concluding about the exact failure, the data plane element needs to check in its pre-installed IPFRR database whether this particular failure results in any route changes. If yes, the linecard replaces the next-hops impacted by that failure with their failure specific backups which were pre-installed in the preparation phase.

In the second option, the first received notification is handled immediately as a link failure, hence the router may start replacing its next-hops. In many cases this is a good decision. If, however, another notification arrives a couple of milliseconds later that points to a node failure, the router then needs to start replacing its next-hops again. This may cause a route flap but due to the quick dissemination mechanism the routing inconsistency is very short lived and likely takes only a couple of milliseconds.

4.2.2. SRLG Handling

The above conceptual solution is easily extensible to support pre-configured SRLGs. Namely, if the failed link is part of an SRLG, then the disseminated link ID should identify the SRLG itself. As a result, possible notifications describing other link failures of the same SRLG will identify the same resource.

If the control plane knows about SRLGs, it can prepare for failures of these, e.g. by calculating a path that avoids all links in that SRLG. SRLG identifier may have been pre-configured or have been obtained by automated mechanisms such as [RFC4203].

4.3. Example and Timing

TBA

To explain why packet loss is not impacted by big delay links, even if FN has to get far away

4.4. Scoping FN Messages with TTL

In a large routing area it is often the case that a failure (i.e. a topology change) causes next-hop changes only in routers relatively close to the failure. Analysis of certain random topologies and two example ISP topologies revealed that a single link failure event generated routing table changes only in routers not more than 2 hops away from the failure site for the particular topologies under study [Hok2008]. Based on this analysis, it is anticipated that in practice the TTL for failure notification messages can be set to a relatively small radius, perhaps as small as 2 or 3 hops.

A chief benefit of correct TTL scoping is that it reduces the overhead on routers that have no use for the information (i.e. which do not need to re-route). Another benefit (that is particularly important for links with scarce capacity) of proper scoping of failure notification messages is that it helps to constrain the control overhead incurred on network links. Determining a suitable TTL value for each locally originated event and controlling failure notification dissemination, in general, is discussed further in Section 5.7.

5. Operation Details

5.1. Transport of Fast Notification Messages

This draft recommends that out of the several FN delivery options defined in [fn-transport], the flooding transport option is preferred, which ensures that any event can reach each node from any source with any failure present in the network area as long as theoretically possible. Flooding also ensures that FN messages reach each node on the shortest (delay) path, and as a side effect failure notifications always reach *each* node *before* re-routed data packets could reach that node. This means that looping is minimized.

[fn-transport] describes that the dataplane flooding procedure requires routers to perform duplicate checking before forwarding the notifications to other interfaces this way to avoid duplicating notification and increasing overhead superfluously. [fn-transport] describes that duplicate check can be performed by a simple storage queue, where previously received notification packets or their signatures are stored.

IPFRR-FN enables another duplicate check process that is based on the internal state machine. Routers, after receiving a notification but before forwarding it to other peers, check the authenticity of the message, if authentication is used. Now the router may check what is the stored event and what is the event described by the received notification.

Two variables and a bit describe what is the known failure state:

- o Suspected failed node ID (denoted by N)
- o Suspected link/SRLG ID (denoted by S)
- o Bit indicating the type of the failure, i.e. link/SRLG failure or node failure (denoted by T)

Recall that the incoming notification describes that a node X has lost connectivity to a node Z via a link L. Now, the state machine can be described with the following pseudo-code:

```
//current state:
//  N: ID of suspected failed node
//  S: ID of suspected failed link/SRLG
//  T: bit indicating the type of the failure
//     T=0 indicates link/SRLG
//     T=1 indicates node
//
Proc notification_received(Node Originator_X, Node Y, SRLG L) {
    if (N == NULL) {
        // this is a new event, store it and forward it
        N=Y;
        S=L;
        T=0; //which is the default anyway
        Forward_notification;
    }
    else if (S == L AND T == 0) {
        // this is the same link or SRLG as before, need not do
        // anything
        Discard_notification;
    }
    else if (N == Y) {
        // This is a node failure
        if (T == 0) {
            // Just now turned out that it is a node failure
            T=1;
            Forward_notification;
        }
        else {
            // Known before that it is a node failure,
            // no need to forward it
            Discard_notification;
        }
    }
    else {
        // multiple failures
    }
}
```

Figure 2 Pseudo-code of state machine for FN forwarding

5.2. Message Handling and Encoding

A failure identifier is needed that unambiguously describes the failed resource consistently among the nodes in the area. The schemantics of the identifiers are defined by the IGP used to pre-calculate and pre-install the backup forwarding entries, e.g. OSPF or ISIS.

This draft defines a Failure Identification message class. Members of this class represent a routing protocol specific Failure Identification message to be carried with the Fast Notification transport protocol. Each message within the Failure Identification message class shall contain the following fields, the lengths of which are routing protocol specific. The exact values shall be aligned with the WG of the routing protocol:

- o Originator Router ID: the identifier of the router advertising the failure;

- o Neighbour Router ID: the identifier of the neighbour node to which the originator lost connectivity.

- o Link ID: the identifier of the link, through which connectivity was lost to the neighbour. The routing protocol should assign the same Link ID for bidirectional, broadcast or multi access links from each access point, consistently.

- o Sequence Number: [fn-transport] expects the applications of the FN service that require replay attack protection to create and verify a sequence number in FN messages. It is described in Section 6.

Routers forwarding the FN packets should ensure that Failure Identification messages are not lost, e.g. due to congestion. FN packets can be put a high precedence traffic class (e.g. Network Control). If the network environment is known to be lossy, the FN sender should repeat the same notification a couple of times, like a salvo fire.

After the forwarding element processed the FN packet and extracted the Failure Identification message, it should decide what backups need to be activated if at all - as described in Section 4.2.1.

5.2.1. Failure Identification Message for OSPF

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           FN Length           |  FN App Type  | AuType|unused |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Originator Router ID                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Neighbour Router ID                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Link ID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
|                       Sequence Number                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Sequence Number (cont'd)                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

FN Header fields:

FN Length
The length of the Failure Identification message for OSPF is 16 bytes.

FN App Type
The exact values are to be assigned by IANA for the Failure Identification message class. For example, FN App Type values between 0x0008 and 0x000F could represent Failure Identification messages, from which 0x0008 could mean OSPF, 0x0009 could be ISIS.

AuType
IPFRR-FN relies on the authentication options offered the FN transport service. Cryptographic authentication is recommended.

Originator Router ID
If the routing protocol is OSPF, then the value can take the OSPF Router ID of the advertising router.

Neighbour Router ID
The OSPF Router ID of the neighbour router to which connectivity was lost.

Link ID
If the link is a LAN, the Link ID takes the LSAID of its representing Network LSA.
If the link is a point-to-point link, the Link ID can take the minimum or the maximum of the two interface IDs. The requirement is that it is performed consistently.

Sequence Number
This field stores a digest of the LSDB of the routing protocol, as described in Section 6. 5.7.1.

5.2.2. Failure Identification Message for ISIS

TBA.

5.3. Protecting External Prefixes

5.3.1. Failure on the Intra-Area Path Leading to the ASBR

Installing failure specific backup next-hops for each external prefix would be a scalability problem as the number of these prefixes may one or two orders of magnitude higher than intra-area destinations. To avoid this, it is suggested to make use of indirection already offered by most router vendors.

Indirection means that when a packet needs to be forwarded to an external destination, the IP address lookup in the FIB will not return a direct result but a pointer to another FIB entry, i.e. to the FIB entry of the ASBR. In LDP/MPLS this means that all prefixes reachable through the same ASBR constitute the same FEC.

As an example, consider that in an area ASBR1 is the primary BGP route for prefixes P1, P2, P3 and P4 and ASBR2 is the primary route for prefixes P5, P6 and P7. A FIB arrangement for this scenario could be the one shown on the following figure. Prefixes using the same ASBR could be resolved to the same pointer that references to the next-hop leading to the ASBR. Prefixes resolved to the same pointer are said to be part of the same "prefix group" or FEC.

```
      FIB lookup          |          FIB lookup
                          |
ASBR2 ========> NH2       |   ASBR2 ========> NH2 <----+
ASBR1 ========> NH1       |   ASBR1 ========> NH1 <-+  |
                          |                         |  |
P1    ========> NH1       |   P1    ========> Ptr1 -+  |
P2    ========> NH1       |   P2    ========> Ptr1 -+  |
P3    ========> NH1       |   P3    ========> Ptr1 -+  |
P4    ========> NH1       |   P4    ========> Ptr1 -+  |
                          |                            |
P5    ========> NH2       |   P5    ========> Ptr2 ----+
P6    ========> NH2       |   P6    ========> Ptr2 ----+
P7    ========> NH2       |   P7    ========> Ptr2 ----+
                            |
```

Figure 3 FIB without (left) and with (right) indirection

If the next-hop to an ASBR changes, it is enough to update in the FIB the next-hop of the ASBR route. In the above example, this means that if the next-hop of ASBR1 changes, it is enough to update the route entry for ASBR1 and due to indirection through pointer Ptr1 this updates several prefixes.

5.3.2. Protecting ASBR Failures: BGP-FRR

IPFRR-FN can make use of alternative BGP routes advertised in an AS by new extensions of BGP such as [BGPAddPaths], [DiverseBGP] or [BGPBestExt]. Using these extensions, for each destination prefix, a node may learn a "backup" ASBR besides the primary ASBR learnt by normal BGP operation.

5.3.2.1. Primary and Backup ASBR in the Same Area

If the failed ASBR is inside the area, all nodes within that area get notified by FN. Grouping prefixes into FECs, however, needs to be done carefully. Prefixes now constitute a common group (i.e. are resolved to the same pointer) if *both* their primary AND their backup ASBRs are the same. This is due to the fact that even if two prefixes use the ASBR by default, they may use different ASBRs when their common default ASBR fails.

Considering the previous example, let us assume that the backup ASBR of prefixes P1 and P2 is ASBR3 but that the backup ASBR of P3 and P4 is an ASBR2. Let us further assume that P5 also has ASBR3 as its backup ASBR but P6 and P7 have an ASBR 4 as their backup ASBR. The resulting FIB structure is shown in the following figure:

```
      FIB lookup
ASBR4 ========> NH4
ASBR2 ========> NH2
ASBR3 ========> NH3
ASBR1 ========> NH1

P1    ========> Ptr1 -+-> NH1
P2    ========> Ptr1 -+

P3    ========> Ptr2 -+-> NH1
P4    ========> Ptr2 -+

P5    ========> Ptr3 ---> NH2

P6    ========> Ptr4 -+-> NH2
P7    ========> Ptr4 -+
```

Figure 4 Indirect FIB for ASBR protection

If, for example, ASBR1 goes down, this affects prefixes P1 through P4. In order to set the correct backup routes, the container referenced by Ptr1 needs to be updated to NH2 (next-hop of ASBR2) but the location referenced by Ptr2 needs to be updated to NH3 (next-hop of ASBR3). This means that P1 and P2 may constitute the same FEC but P3 and P4 needs to be another FEC so that there backups can be set independently.

Note that the routes towards ASBR2 or ASBR3 may have changed, too. For example, if after the failure ASBR3 would use a new next-hop NH5, then the container referenced by Ptr2 should be updated to NH5. A resulting detour FIB is shown in the following figure.

```
      FIB lookup
ASBR4 ========>    NH4
ASBR2 ========>    NH2
ASBR3 ========>    NH5
ASBR1 ========>     X

P1    ========> Ptr1 -+-> NH2
P2    ========> Ptr1 -+

P3    ========> Ptr2 -+-> NH5
P4    ========> Ptr2 -+

P5    ========> Ptr3 ---> NH2

P6    ========> Ptr4 -+-> NH2
P7    ========> Ptr4 -+
```

Figure 5 Indirect "detour" FIB in case of ASBR1 failure

During pre-calculation, the control plane pre-downloaded the failure identifier of ASBR1 and assigned NH5 as the failure specific backup for routes for ASBR3 and pointer Ptr2 and assigned NH2 as the failure specific backup for the route referenced by Ptr1.

5.3.2.2. Primary and Backup ASBR in Different Areas

By default, the scope of FN messages is limited to a single routing area.

The IPFRR-FN application of FN, may, however, need to redistribute some specific notifications across areas in a limited manner.

If an ASBR1 in Area1 goes down and some prefixes need to use ASBR2 in another Area2, then, besides Area1, routers in Area2 need to know about this failure. Since communication between non-backbone areas is done through the backbone areas, it may also need the information.

Naturally, if ASBR2 resides in the backbone area, then the FN of ASBR1 failure needs to be leaked only to the backbone area.

Leaking is facilitated by area border routers (ABR). During failure preparation phase, the routing engine of an ABR can determine that for an intra-area ASBR the backup ASBR is in a different area to which it is the ABR. Therefore, the routing engine installs such intra-area ASBRs in a "redistribution list" at the dataplane cards.

The ABR, after receiving FN messages, may conclude in its state machine that a node failure happened. If this node failure is in the redistribution list, the ABR will generate an FN with the following data:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               16              |     0x008     | AuType|unused |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         ABR Router ID                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        ASBR Router ID                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              0x0                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Sequence Number                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Sequence Number (cont'd)                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

This message is then distributed to the neighbour area specified in the redistribution list as a regular FN message. A Link ID of 0x0 specifically signals in the neighbour area that this failure is a known node failure of the node specified by the "Neighbour Router ID" field (which was set to the failed ASBR's ID).

ABRs in a non-backbone area need to prepare to redistribute ASBR failure notifications from within their area to the backbone area.

ABRs in the backbone area need to prepare to redistribute an ASBR failure notification from the backbone area to that area where a backup ASBR resides.

Consider the previous example, but now let us assume that the current area is Area0, ASBR2 and ASBR3 reside in Area1 (reachable through ABR1) but ASBR 4 resides in Area2 (reachable through ABR2). The resulting FIBs are shown in the following figures: in case of ASBR2 failure, only Ptr4 needs an update.

```
    FIB lookup
ABR1 ========> NH6
ABR2 ========> NH7

(ASBR4 ========> NH7)  //may or may not be in the FIB
(ASBR2 ========> NH6)  //may or may not be in the FIB
(ASBR3 ========> NH6)  //may or may not be in the FIB
(ASBR1 ========> NH1)  //may or may not be in the FIB

P1   ========> Ptr1 -+-> NH1
P2   ========> Ptr1 -+

P3   ========> Ptr2 -+-> NH1
P4   ========> Ptr2 -+

P5   ========> Ptr3 ---> NH6

P6   ========> Ptr4 -+-> NH6
P7   ========> Ptr4 -+
```

Figure 6 Indirect FIB for inter-area ASBR protection

```
    FIB lookup
ABR1 ========>    NH6
ABR2 ========>    NH7

(ASBR4  =======> NH7)  //may or may not be in the FIB
(ASBR2  =======>  X )  //may or may not be in the FIB
(ASBR3 ========> NH6)  //may or may not be in the FIB
(ASBR1 ========> NH1)  //may or may not be in the FIB

P1   ========> Ptr1 -+-> NH1
P2   ========> Ptr1 -+

P3   ========> Ptr2 -+-> NH1
P4   ========> Ptr2 -+

P5   ========> Ptr3 ---> NH6

P6   ========> Ptr4 -+-> NH7
P7   ========> Ptr4 -+
```

Figure 7 Indirect "detour" FIB for inter-area ASBR protection, ASBR2 failure

5.4. Application to LDP

It is possible for LDP traffic to follow path other than those indicated by the IGP. To do so, it is necessary for LDP to have the appropriate labels available for the alternate so that the appropriate out-segments can be installed in the forwarding plane before the failure occurs.

This means that a Label Switching Router (LSR) running LDP must distribute its labels for the Forwarding Equivalence Classes (FECs) it can provide to all its neighbours, regardless of whether or not they are upstream. Additionally, LDP must be acting in liberal label retention mode so that the labels that correspond to neighbours that aren't currently the primary neighbour are stored. Similarly, LDP should be in downstream unsolicited mode, so that the labels for the FEC are distributed other than along the SPT.

The above criteria are identical to those defined in [RFC5286].

In IP, a received FN message may result in rewriting the next-hop in FIB. If LDP is applied, the label FIB also needs to be updated in accordance with the new IP next-hop; in the LFIB, however, not only the outgoing interface needs to be replaced but also the label that is valid to this non-default next-hop. The latter is available due to liberal label retention and unsolicited downstream mode.

5.5. Bypassing Legacy Nodes

Legacy nodes, while cannot originate fast notifications and cannot process them either, can be assumed to be able to forward the notifications. As [fn-transport] discusses, FN forwarding is based on multicast. It is safe to assume that legacy routers' multicast configuration can be set up statically so as to be able to propagate fast notifications as needed.

When calculating failure specific alternative routes, IPFRR-FN capable nodes must consider legacy nodes as being fixed directed links since legacy nodes do not change packet forwarding in the case of failure. There are situations when an FN-IPFRR capable node can, exceptionally, bypass a non-IPFRR-FN capable node in order to handle a remote failure.

As an example consider the topology depicted in Figure 8, where the link between C and D fails. C cannot locally repair the failure.

```
+---+   +---+   +---+   +---+
| E |---| F |---| G |---| H |
+---+   +---+   +---+   +---+
  |            /          |
  |           /           |
  |          /            |
+---+   +---+   +---+   +---+
| A |---| B |---| C |-X-| D |
+---+   +---+   +---+   +---+
  >========>==============>
     Traffic from A to D
```

Figure 8 Example for bypassing legacy nodes

First, let us assume that each node is IPFRR-FN capable. C would advertise the failure information using FN. Each node learns that the link between C and D fails, as a result of which C changes its forwarding table to send any traffic destined to D via B. B also makes a change, replacing its default next-hop (C) with G. Note that other nodes do not need to modify their forwarding at all.

Now, let us assume that B is a legacy router not supporting IPFRR-FN but it is statically configured to multicast fast notifications as needed. As such, A will receive the notification. A's pre-calculations have been done knowing that B is unable to correct the failure. Node A, therefore, has pre-calculated E as the failure specific next-hop. Traffic entering at A and heading to D can thus be repaired.

5.6. Capability Advertisement

The solution requires nodes to know which other nodes in the area are capable of IPFRR-FN. The most straightforward way to achieve this is to rely on the Router Capability TLVs available both in OSPF [RFC4970] and in IS-IS [RFC4971].

5.7. Constraining the Dissemination Scope of Fast Notification Packets

As discussed earlier in Section 4.4. it is desirable to constrain the dissemination scope of failure notification messages. This section presents three candidate methods for controlling the scope of failure notification: (1) Pre-configure the TTL for FN messages in routers based on best current practices and related studies of available ISP and enterprise network topologies; (2) dynamically calculate the minimum TTL value needed to ensure 100% remote LFAP coverage; and (3) dynamically calculate the set of neighbours for which FN message should given the identity of the link that has failed.

These candidate dissemination options are mechanisms with different levels of optimality and complexity. The intent here is to present some options that will generate further discussion on the tradeoffs between different FN message scoping methods.

5.7.1. Pre-Configured FN TTL Setting

As discussed, earlier in Section 4.4. studies of various network topologies suggest that a fixed TTL setting of 2 hops may be sufficient to ensure failure notification message for typical OSPF area topologies. Therefore, a potentially simple solution for constraining FN message dissemination is for network managers to configure their routers with fixed TTL setting (e.g., TTL=2 hops) for FN messages. This TTL setting can be adjusted by network managers to consider implementation-specific details of the topology such as configuring a larger TTL setting for topologies containing, say, large ring sub-graph structures.

In terms of performance trades, pre-configuring the FN TTL, since it is fixed at configuration time, incurs no computational overhead for the router. On the other hand, it represents a configurable router parameter that network administrators must manage. Furthermore, the fixed, pre-configured FN TTL approach is sub-optimal in terms of constraining the FN dissemination as most single link events will not require FN messages send to up to TTL hops away from the failure site.

5.7.2. Advanced FN Scoping

While the static pre-configured setting of the FN TTL will likely work in practice for a wide range of OSPF area topologies, it has at two least weaknesses: (1) There may be certain topologies for which the TTL setting happens to be insufficient to provide the needed failure coverage; and (2) as discussed above, it tends to result in FN being disseminated to a larger radius than needed to facilitate re-routing.

The solution to these drawbacks is for routers to dynamically compute the FN TTL radius needed for each of the local links it monitors. Doing so addresses the two weakness of a pre-configured TTL setting by computing a custom TTL setting for each of its local links that matches exactly the FN message radius for the given topology. The drawback, of course, is the additional computations. However, given a quasi-static network topology, it is possible this dynamic FN TTL computation is performed infrequently and, therefore, on average incurs relatively small computation overhead.

While a pre-configured TTL eliminates computation overhead at the expense of FN dissemination overhead and dynamic updates of the TTL settings achieve better dissemination efficiency by incurring some computational complexity, directed FN message forwarding attempts to minimize the FN dissemination scope by leveraging additional computation power. Here, rather than computing a FN TTL setting for each local link, a network employing directed forwarding has each router instance R compute the sets of one-hop neighbours to which a FN message must be forwarded for every possible failure event in the routing area. This has the beneficial effect of constraining the FN scope to the direction where there are nodes that require the FN update as opposed to disseminating to the entire TTL hop radius about a failure site. The trade off here, of course, is the additional computation complexity incurred and the maintenance of forwarding state for each possible failure case. Reference [Cev2010] gives an algorithm for finding, for each failure event, the direct neighbours to which the notification should be forwarded.

6. Protection against Replay Attacks

To defend against replay attacks, recipients should be able to ignore a re-sent recording of a previously sent FN packet. This suggests that some sort of sequence number should be included in the FN packet, the verification of which should not need control plane involvement. Since the solution should be simple to implement in the dataplane, maintaining and verifying per-source sequence numbers is not the best option.

We propose, therefore, that messages should be stamped with the digest of the actual routing configuration, i.e., a digest of the link state database of the link state routing protocol. The digest has to be picked carefully, so that if two LSDBs describe the same connectivity information, their digest should be identical as well, and different LSDBs should result in different digest values with high probability.

The conceptual way of handling these digests could be the following:

- o When the LSDB changes, the IGP re-calculates the digest and downloads the new value to the dataplane element(s), in a secure way.

- o When a FN packet is originated, the digest is put into the FN message into the Sequence Number field.

- o Network nodes distribute (forward) the FN packet.

- o When processing, the dataplane element first performs an authentication check of the FN packet, as described in [fn-transport].

- o Finally, before processing the failure notification, the dataplane element should check whether its own known LSDB digest is identical with the one in the message.

If due to a failure event a node disseminates a failure notification with FN, an attacker might capture the whole packet and re-send it later. If it resends the packet after the IGP re-converged on the new topology, the active LSDB digest is different, so the packet can be ignored. If the packet is replayed to a recipient who still has the same LSDB digest, then it means that the original failure notification was already processed but the IGP has not yet finished converging; the IPFRR detour is already active, the replica has no impact.

6.1. Calculating LSDB Digest

We propose to create an LSDB digest that is conceptually similar to [ISISDigest]. The operation is proposed to be the following:

- o Create a hash from each LSA(OSPF)/LSP(ISIS) one by one

- o XOR these hashes together

- o When an LSA/LSP is removed, the new LSDB digest is received by computing the hash of the removed LSA, and then XOR to the existing digest

- o When an LSA/LSP is added, the new LSDB digest is received by computing the hash of the new LSA, and then XOR to the existing digest

7. Security Considerations

The IPFRR application of Fast Notification does not raise further known security consideration in addition to those already present in Fast Notification itself. If an attacker could send false Failure Identification Messages or could hinder the transmission of legal messages, then the network would produce an undesired routing behaviour. These issues should be solved, however, in [fn-transport].

IPFRR-FN relies on the authentication mechanism provided by the Fast Notification transport protocol [fn-transport]. The specification of the FN transport protocol requires applications to protect against replay attacks with application specific sequence numbers. This draft, therefore, describes its own proposed sequence number in Section 5.7.1.

8. IANA Considerations

The Failure Identification message types need to be allocated a value in the FN App Type field.

IPFRR-FN capability needs to be allocated within Router Capability TLVs both for OSPF [RFC4970] and in IS-IS [RFC4971].

9. References

9.1. Normative References

[RFC5286] A. Atlas, A. Zinin, "Basic specification for IP Fast-Reroute: Loop-Free Alternates", Internet Engineering Task Force: RFC 5286, 2008.

[fn-transport] W. Lu, S. Kini, A. Csaszar, G. Enyedi, J. Tantsura, A. Tian, "Transport of Fast Notifications Messages", draft-lu-fn-transport, 2011

[RFC4970] A. Lindem et al., Extensions to OSPF for Advertising Optional Router Capabilities, RFC 4970, 2007

[RFC4971] JP. Vasseur et al., Intermediate System to Intermediate System (IS-IS) Extensions for Advertising Router Information, RFC 4971, 2007

[RFC4203] K. Kompella, Y. Rekhter, " OSPF Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS)", RFC4203, 2005

9.2. Informative References

[BFD] D. Katz, D. Ward, "Bidirectional forwarding detection", RFC 5880, IETF, 2010

[RFC5714] M. Shand, S. Bryant, "IP Fast Reroute Framework", RFC 5714, IETF, 2010.

[Cev2010] S. Sevher, T. Chen, I. Hokelek, J. Kang, V. Kaul, Y.J. Lin, M. Pang, R. Rodoper, S. Samtani, C. Shah, J. Bowcock, G. B. Rucker, J. L. Simbol and A. Staikos, "An Integrated Soft Handoff Approach in IP Fast Reroute in Wireless Mobile Networks", In Proceedings IEEE COMSNETS, 2011.

[Eny2009] Gabor Enyedi, Peter Szilagyi, Gabor Retvari, Andras Csaszar, "IP Fast ReRoute: Lightweight Not-Via without Additional Addresses", IEEE INFOCOM-MiniConference, Rio de Janeiro, Brazil, 2009.

[FIFR] J. Wand, S. Nelakuditi, "IP fast reroute with failure inferencing", In Proceedings of ACM SIGCOMM Workshop on Internet Network Management - The Five-Nines Workshop, 2007.

[Hok2007] I. Hokelek, M. A. Fecko, P. Gurung, S. Samtani, J. Sucec, A. Staikos, J. Bowcock and Z. Zhang, "Seamless Soft Handoff in Wireless Battlefield Networks Using Local and Remote LFAPs", In Proceedings IEEE MILCOM, 2007.

[Hok2008] I. Hokelek, S. Cevher, M. A. Fecko, P. Gurung, S. Samtani, Z. Zhang, A. Staikos and J. Bowcock, "Testbed Implementation of Loop-Free Soft Handoff in Wireless Battlefield Networks", In Proceedings of the 26th Army Science Conference, December 1-4, 2008.

[MRC] T. Cicic, A. F. Hansen, A. Kvalbein, M. Hartmann, R. Martin, M. Menth, S. Gjessing, O. Lysne, "Relaxed multiple routing configurations IP fast reroute for single and correlated failures", IEEE Transactions on Network and Service Management, available online: http://www3.informatik.uni-wuerzburg.de/staff/menth/Publications/papers/Menth08-Sub-4.pdf, September 2010.

[NotVia] S. Bryant, M. Shand, S. Previdi, "IP fast reroute using Not-via addresses", Internet Draft, draft-ietf-rtgwg-ipfrr-notvia-addresses, 2010.

[RLFAP] I. Hokelek, M. Fecko, P. Gurung, S. Samtani, S. Cevher, J. Sucec, "Loop-Free IP Fast Reroute Using Local and Remote LFAPs", Internet Draft, draft-hokelek-rlfap-01 (expired), 2008.

[Ret2011] G. Retvari, J. Tapolcai, G. Enyedi, A. Csaszar, "IP Fast ReRoute: Loop Free Alternates Revisited", to appear at IEEE INFOCOM 2011

[ISISDigest] J. Chiabaut and D. Fedyk. IS-IS Multicast Synchronization Digest. Available online: http://www.ieee802.org/1/files/public/docs2008/aq-fedyk-ISIS-digest-1108-v1.pdf, Nov 2008.

[BGPAddPaths] D. Walton, A. Retana, E. Chen, J. Scudder, "Advertisement of Multiple Paths in BGP", draft-ietf-idr-add-paths, Work in progress

[DiverseBGP] R. Raszuk, et. Al, "Distribution of diverse BGP paths", draft-ietf-grow-diverse-bgp-path-dist, Work in progress

[BGPBestExt] P. Marques, R. Fernando, E. Chen, P. Mohapatra, H. Gredler, "Advertisement of the best external route in BGP", draft-ietf-idr-best-external, Work in progress

[BRITE] Oliver Heckmann et al., "How to use topology generators to create realistic topologies", Technical Report, Dec 2002.

10. Acknowledgments

The authors would like to thank Albert Tian, Wenhu Lu, Acee Lindem and Ibrahim Hokelek for the continuous discussions and comments on the topic, as well as Joel Halpern for his comments and review.

Appendix A. Memory Needs of a Naive Implementation

Practical background might suggest that storing and maintaining backup next-hops for many potential remote failures could overwhelm the resources of router linecards. This section attempts to provide a calculation describing the approximate memory needs in reasonable sized networks with a possible implementation.

A.1. An Example Implementation

Let us suppose that for exterior destinations the forwarding engine is using recursive lookup or indirection in order to improve updating time such as described in Section 5.3. We are also supposing that the concept of "prefix groups" is applied, i.e. there is an internal entity for the prefixes using exactly the same primary and backup ASBRs, and the next hop entry for a prefix among them is pointing to the next hop towards this entity. See e.g. Figure 6.

In the sequel, the term of "area" refers to an extended area, made up by the OSPF or IS-IS area containing the router, with the prefix groups added to the area as virtual nodes. Naturally, a prefix group is connected to the egress routers (ABRs) through which it can be reached. We just need to react to the failure ID of an ASBR for all the prefix groups connected to that ASBR; technically, we must suppose that one of the virtual links of all the affected prefix groups go down.

Here we show a simple naive implementation which can easily be beaten in real routers. This implementation uses an array for all the nodes (including real routers and virtual nodes representing prefix groups) in the area (node array in the sequel), made up by two pointers and a length filed (an integer) per record. One of the pointers points to another array (called alternative array). That second array is basically an enumeration containing the IDs of those failures influencing a shortest path towards that node and an alternative neighbor, which can be used, when such a failure occurs. When a failure is detected, (either locally, or by FN), we can easily find the proper record in all the lists. Moreover, since these arrays can be sorted based on the failure ID, we can even use binary search to find the needed record. The length of this array is stored in the record of the node array pointing to the alternative list.

Now, we only need to know, which records in the FIB should be updated. Therefore there is a second pointer in the node array pointing to that record.

```
          +-------+-------+-------+--   --+-------+
          |  r1   |  r2   |  r3   | ... |   rk  |
          +-------+-------+-------+--   --+-------+
              |       |       |               |
              |       |       |               |
              |       |       |               |
             \|/     \|/     \|/             \|/
              *       *       *               *
          +-------+-------+-------+--   --+-------+
          | fail1 | fail2 | fail3 |       | failk |
          | alt.1 | alt.2 | alt.3 |  ...  | alt.k |
          +-------+-------+-------+--   --+-------+
          | fail4 |       | fail5 |
          | alt.4 |       | alt.5 |
          +-------+       +-------+
          | fail6 |
          | alt.6 |
          +-------+
```

Figure 9 The way of storing alternatives

A.2. Estimation of Memory Requirements.

Now, suppose that there are V nodes in the extended area, the network diameter is D, a neighbor descriptor takes X bytes, a failure ID takes Y bytes and a pointer takes Z bytes. We suppose that lookup for external prefixes are using indirection, so we only need to deal with destinations inside the extended area. In this way, if there is no ECMP, this data structure takes

$$(2*Z+Y)*(V-1) + 2*(X+Y)*D*(V-1)$$

bytes altogether. The first part is the memory consumption of the node array. The memory needed by alternative arrays: any path can contain at most D nodes and D links, each record needs X+Y bytes; there are records for all the other nodes in the area (V-1 nodes). Observe that this is a very rough overestimation, since most of the possible failures influencing the path will not change the next hop.

For computing memory consumption, suppose that neighbor descriptors, failure IDs and pointers take 4 bytes, there are 10000 nodes in the extended area (so both real routers and virtual nodes representing prefix groups are included) and the network diameter is 20 hops. In this case, we get that the node array needs about 120KB, the alternative array needs about 3.2MB, so altogether 3.4MB if there is no ECMP. Observe that the number of external prefixes is not important.

If however, there are paths with equal costs, the size of the alternative array increases. Suppose that there are 10 equal paths between ANY two nodes in the network. This would cause that the alternative list gets 10 times bigger, and now it needs a bit less than 32MB. Observe that the node array still needs only about 160KB, so 32MB is a good overestimation, which is likely acceptable for modern linecards with gigs of DRAM. Moreover, we need to stress here again that this is an extremely rough overestimation, so in reality much less memory will be enough. Furthermore, usually only protecting outer prefixes is needed, so we only need to protect the paths towards the prefix groups, which further decreases both the size of node array and the number of alternative lists.

A.3. Estimation of Failover Time

After a failover was detected either locally or by using FN, the nodes need to change the entries in their FIB. Here we do a rough estimation to show that the previous implementation can do it in at most a few milliseconds.

We are supposing that we have the data structure described in the previous section. When a failure happens we need to decide for each node in the node table whether the shortest path towards that destination was influenced by the failure. We can sort the elements in the alternative list, so now we can use binary search, which needs ceil(log(2D)) memory access (log here has base 2) for worst case. We need one more access to get the node list entry and another to rewrite the FIB.

We suppose DDR3 SDRAM with 64 byte cache line, which means that up to 8 entries of the alternative list can be fetched from the RAM at a time, so the previous formula is modified as we need ceil(log(D/4))+2 transactions. In this way for D=20 and V=10.000 we need (3+2)*10.000=50.000 transactions. If we suppose 10 ECMP paths as previously, D=200 and we need (5+2)*10000=70.000 transactions.

We can do a very conservative estimation by supposing a recent DDR3 SDRAM module which can do 5MT/s with completely random access, so doing 50.000 or 70.000 transaction takes 10ms or 14ms. Keep in mind that we assumed that there is only one memory controller, we always got the result of the search with the last read, and all the alternative lists were full. Moreover, internal system latencies (e.g. multiple memory requests) were overestimated seriously, since a DDR3 SDRAM can reach even 6 times this speed with random access.

Appendix B. Impact Scope of Fast Notification

The memory and fail-over time calculations presented in Appendix A are based on worst-case estimation. They assume that basically in a network with diameter equal to 20 hops, each failure has a route changing consequence on all routers in the full diameter.

This section provides experimental results on real-world topologies, showing that already 100% failure coverage can be achieved within a 2-hop radius around the failure.

We performed the coverage analysis of the fast reroute mechanism presented here on realistic topologies, which were generated by the BRITE topology generator in bottom-up mode [BRITE]. The coverage percentage is defined here as the percentage of the number of useable backup paths for protecting the primary paths which are failed because of link failures to the number of all failed primary paths.

The realistic topologies include AT&T and DFN using pre-determined BRITE parameter values from [BRITE] and various random topologies with different number of nodes and varying network connectivity. For example, the number of nodes for AT&T and DFN are 154 and 30, respectively, while the number of nodes for other random topologies is varied from 20 to 100. The BRITE parameters which are used in our topology generation process are summarized in Figure 10 (see [BRITE] for the details of each parameter). In summary, m represents the average number of edges per node and is set to either 2 or 3. A uniform bandwidth distribution in the range 100-1024 Mbps is selected and the link cost is obtained deterministically from the link bandwidth (i.e., inversely proportional to the link bandwidth as used by many vendors). Since the values for p(add) and beta determine the number of edges in the generated topologies, their values are varied to obtain network topologies with varying connectivity (e.g., sparse and dense).

	Bottom up
Grouping Model	Random pick
Model	GLP
Node Placement	Random
Growth Type	Incremental
Preferential Connectivity	On
BW Distribution	Uniform
Minimum BW	100
Maximum BW	1024
m	2-3
Number of Nodes (N)	20,30,50,100,154
p(add)	0.01,0.05,0.10,0.42
beta	0.01,0.05,0.15,0.62

Figure 10 BRITE topology generator parameters

The coverage percentage of our fast reroute method is reported for different network topologies (e.g., different number of nodes and varying network connectivity) using neighborhood depths of 0, 1, and 2. (i.e., X=0, 1, and 2). For a particular failure, backup routes protecting the failed primary paths are calculated only by those nodes which are within the selected radious of this failure. Note that these nodes are determined by the parameter X as follows: For X=0, two nodes which are directly connected to the failed link, for X=1, two nodes which are directly connected to the failed link and also neighboring nodes which are adjacent to one of the outgoing links of these two nodes, and so on.

The coverage percentage for a certain topology is computed by the following formula: Coverage Percentage = N_backupsexist*100/N_fpp where N_backupsexist is the number of source-destination pairs whose primary paths are failed because of link failures and have backup paths for protecting these failed paths, and N_fpp is the number of source-destination pairs whose primary paths are failed because of link failures. The source-destination pairs, in which source and destination nodes do not have any physical connectivity after a failure, are excluded from N_fpp. Note that the coverage percentage includes a network-wide result which is calculated by averaging all coverage results obtained by individually failing all edges for a certain network topology.

Figure 11 shows the coverage percentage results for random topologies with different number of nodes (N) and network connectivity, and Figure 12 shows these results for AT&T and DFN topologies. In these figures, E_mean represents the average number of edges per node for a certain topology. Note that the average number of edges per node is determined by the parameters m, p(add), and beta. We observed that E_mean increases when p(add) and beta values increase. For each topology, coverage analysis is repeated for 10 topologies generated randomly by using the same BRITE parameters. E_mean and coverage percentage are obtained by averaging the results of these ten experiments.

Case	N	E_mean	X=0	X=1	X=2
p(add)=0.01	20	3.64	82.39	98.85	100.0
beta=0.01	50	3.86	82.10	98.69	100.0
	100	3.98	83.21	98.04	100.0
p(add)=0.05	20	3.70	85.60	99.14	100.0
beta=0.05	50	4.01	84.17	99.09	100.0
	100	4.08	83.35	98.01	100.0
p(add)=0.1	20	5.52	93.24	100.0	100.0
beta=0.15	50	6.21	91.46	99.87	100.0
	100	6.39	91.17	99.86	100.0

Figure 11 Coverage percentage results for random topologies

Case	N	E_mean	X=0	X=1	X=2
p(add)=0.42	154 (AT&T)	6.88	91.04	99.81	100.0
beta=0.62	30 (DFN)	8.32	93.76	100.0	100.0

Figure 12 Coverage percentage results for AT&T and DFN topologies

There are two main observations from these results:

1. As the neighborhood depth (X) increases the coverage percentage increases and the complete coverage is obtained using a low neighborhood depth value (i.e., X=2). This result is significant since failure notification message needs to be sent only to nodes which are two-hop away from the point of failure for the complete

coverage. This result supports that our method provides fast convergence by introducing minimal signaling overhead within only the two-hop neighborhood.

2. The topologies with higher connectivity (i.e., higher E_mean values) have better coverage compared to the topologies with lower connectivity (i.e., lower E_mean values). This is an intuitive result since the number of possible alternate hops in dense network topologies is higher than the number of possible alternate hops in sparse topologies. This phenomenon increases the likelihood of finding backup paths, and therefore the coverage percentage.

Authors' Addresses

Andras Csaszar
Ericsson
Irinyi J utca 4-10, Budapest, Hungary, 1117
Email: Andras.Csaszar@ericsson.com

Gabor Sandor Enyedi
Ericsson
Irinyi J utca 4-10, Budapest, Hungary, 1117
Email: Gabor.Sandor.Enyedi@ericsson.com

Jeff Tantsura
Ericsson
300 Holger Way, San Jose, CA 95134
Email: jeff.tantsura@ericsson.com

Sriganesh Kini
Ericsson
300 Holger Way, San Jose, CA 95134
Email: sriganesh.kini@ericsson.com

John Sucec
Telcordia Technologies
One Telcordia Drive, Piscataway, NJ 08854
Email: sucecj@telcordia.com

Subir Das
Telcordia Technologies
One Telcordia Drive, Piscataway, NJ 08854
Email: sdas2@telcordia.com

Network Working Group A. Csaszar (Ed.)
Internet Draft G. Enyedi
Intended status: Standards Track J. Tantsura
Expires: December 6, 2012 S. Kini
Ericsson

J. Sucec
S. Das
Telcordia

June 6, 2012

IP Fast Re-Route with Fast Notification
draft-csaszar-ipfrr-fn-03.txt

Abstract

This document describes the benefits and main applications of sending explicit fast notification (FN) packets to routers in an area. FN packets are generated and processed in the dataplane, and a single FN service can substitute existing OAM methods for remote failure detection, such as a full mesh of multi-hop BFD session. The FN service, therefore, decreases network overhead considerable. The main application is fast reroute in pure IP and in IP/LDP-MPLS networks called IPFRR-FN. The detour paths used when IPFRR-FN is active are in most cases identical to those used after Interior Gateway Protocol (IGP) convergence. The proposed mechanism can address all single link, node, and SRLG failures in an area; moreover it is an efficient solution to protect against BGP ASBR failures as well as VPN PE router failures. IPFRR-FN can be a supplemental tool to provide FRR when LFA cannot repair a failure case, while it can be a replacement of existing ASBR/PE protection mechanisms by overcoming their scalability and complexity issues.

Table of Contents

1. Introduction

Convergence of link-state IGPs, such as OSPF or IS-IS, after a link or node failure is known to be relatively slow. While this may be sufficient for many applications, some network SLAs and applications require faster reaction to network failures.

IGP convergence time is composed mainly of:

1. Failure detection at nodes adjacent to the failure

2. Advertisement of the topology change

3. Calculation of new routes

4. Installing new routes to linecards

Traditional Hello-based failure detection methods of link-state IGPs are relatively slow, hence a new, optimized, Hello protocol has been standardized [BFD] which can reduce failure detection times to the range of 10ms even if no lower layer notices the failure quickly (like loss of signal, etc.).

Even with fast failure detection, reaction times of IGPs may take several seconds, and even with a tuned configuration it may take at least a couple of hundreds of milliseconds.

To decrease fail-over time even further, IPFRR techniques [RFC5714], can be introduced. IPFRR solutions compliant with [RFC5714] are targeting fail-over time reduction of steps 2-4 with the following design principles:

IGP		IPFRR
2. Advertisement of the topology change	==>	No explicit advertisement, only local repair
3. Calculation of new routes	==>	Pre-computation of new routes
4. Installing new routes to linecards	==>	Pre-installation of backup routes

Pre-computing means that the way of bypassing a failed resource is computed before any failure occurs. In order to limit complexity, IPFRR techniques typically prepare for single link, single node and single Shared Risk Link Group (SRLG) failures, which failure types are undoubtedly the most common ones. The pre-calculated backup routes are also downloaded to linecards in preparation for the failure, in this way sparing the lengthy communication between control plane and data plane when a failure happens.

The principle of local rerouting requires forwarding a packet along a detour even if only the immediate neighbors of the failed resource know the failure. IPFRR methods observing the local rerouting principle do not explicitly propagate the failure information. Unfortunately, packets on detours must be handled in a different way than normal packets as otherwise they might get returned to the failed resource. Rephrased, a node not having *any* sort of information about the failure may loop the packet back to the node from where it was rerouted - simply because its default routing/forwarding configuration dictates that. As an example, see the following figure. Assuming a link failure between A and Dst, A needs to drop packets heading to Dst. If node A forwarded packets to Src, and if the latter had absolutely no knowledge of the failure, a loop would be formed between Src and A.

```
            +---+              +---+
            | B |--------------| C |
            +---+              +---+
           /                        \
          /                          \
         /                            \
    +---+             +---+  failure   +---+
    |Src|-------------| A |-----X------|Dst|
    +---+             +---+            +---+
      =========>==============>=========>
                  Primary path
```

Figure 1 Forwarding inconsistency in case of local repair: The path of Src to Dst leads through A

The basic problem that previous IPFRR solutions struggle to solve is, therefore, to provide consistent routing hop-by-hop without explicit signaling of the failure.

To provide protection for all single failure cases in arbitrary topologies, the information about the failure must be given in *some* way to other nodes. That is, IPFRR solutions targeting full failure coverage need to signal the fact and to some extent the identity of the failure within the data packet as no explicit signaling is allowed. Such solutions have turned out to be considerably complex and hard or impossible to implement practically. The Loop Free Alternates (LFA) solution [RFC5286] does not give the failure information in any way to other routers, and so it cannot repair all failure cases such as the one in Figure 1.

As discussed in Section 2. solutions that address full failure coverage and rely on local repair, i.e. carrying some failure information within the data packets, present an overly complex and therefore often inpractical alternative to LFA. This draft, therefore, suggests that relaxing the local re-routing principle with carefully engineered explicit failure signaling is an effective approach.

The idea of using explicit failure notification for IPFRR has been proposed before for Remote LFA Paths [RLFAP]. RLFAP sends explicit notifications and can limit the radius in which the notification is propagated to enhance scalability. Design, implementation and enhancements for the remote LFAP concept are reported in [Hok2007], [Hok2008] and [Cev2010].

This draft attempts to work out in more detail what kind of failure dissemination mechanism is required to facilitate remote repair efficiently. Requirements for explicit signaling are given in Section 3. This draft does not limit the failure advertisement radius as opposed to RLFAP. As a result, the detour paths remain stable in most cases, since they are identical to those that the IGP will calculation after IGP convergence. Hence, micro-loop will not occur after IGP convergence.

A key contribution of this memo is to recognize that a Fast Notification service is not only an enabler for a new IPFRR approach but it is also a replacement for various OAM remote connectivity verification procedures such as multi-hop BFD. These previous methods posed considerable overhead to the network: (i) management of many OAM sessions; (ii) careful configuration of connectivity verification packet interval so that no false alarm is given for network internal failures which are handled by other mechanisms; and (iii) packet processing overhead, since connectivity verification packets have to be transmitted continuously through the network in a mesh, even in fault-free conditions.

2. Overview of current IPFRR Proposals based on Local Repair

The only practically feasible solution, Loop Free Alternates [RFC5286], offers the simplest resolution of the hop-by-hop routing consistency problem: a node performing fail-over may only use a next-hop as backup if it is guaranteed that it does not send the packets back. These neighbors are called Loop-Free Alternates (LFA). LFAs, however, do not always exist, as shown in Figure 1 above, i.e., node A has no LFAs with respect to Dst. while it is true that tweaking the network configuration may boost LFA failure case coverage considerably [Ret2011], LFAs cannot protect all failure cases in arbitrary network topologies.

The exact way of adding extra information to data packets and its usage for forwarding is the most important property that differentiates most existing IPFRR proposals.

Packets can be marked "implicitly", when they are not altered in any way, but some extra information owned by the router helps deciding the correct way of forwarding. Such extra information can be for instance the direction of the packet, e.g., the incoming interface, e.g. as in [FIFR]. Such solutions require what is called interface-based or interface-specific forwarding.

Interface-based forwarding significantly changes the well-established nature of IP's destination-based forwarding principle, where the IP destination address alone describes the next hop. One embodiment would need to download different FIBs for each physical or virtual IP interface - not a very compelling idea. Another embodiment would alter the next-hop selection process by adding the incoming interface id also to the lookup fields, which would impact forwarding performance considerably.

Other solutions mark data packets explicitly. Some proposals suggest using free bits in the IP header [MRC], which unfortunately do not exist in the IPv4 header. Other proposals resort to encapsulating re-routed packets with an additional IP header as in e.g. [NotVia], [Eny2009] or [MRT-ARCH]. Encapsulation raises the problem of fragmentation and reassembly, which could be a performance bottleneck, if many packets are sent at MTU size. Another significant problem is the additional management complexity of the encapsulation addresses, which have their own semantics and require cumbersome routing calculations, see e.g. [MRT-ALG]. Encapsulation in the IP header translates to label stacking in LDP-MPLS. The above mentioned mechanisms either encode the active topology ID in a label on the stack or encode the failure point in a label, and also require an increasing mesh of targeted LDP sessions to acquire a valid label at the detour endpoint, which is another level of complexity.

3. Requirements of an Explicit Failure Signaling Mechanism

All local repair mechanisms touched above try to avoid explicit notification of the failure via signaling, and instead try to hack some failure-related information into data packets. This is mainly due to relatively low signaling performance of legacy hardware. Failure notification, therefore, should fulfill the following properties to be practically feasible:

1. The signaling mechanism should be reliable. The mechanism needs to propagate the failure information to all interested nodes even in a network where a single link or a node is down.

2. The mechanism should be fast in the sense that getting the notification packet to remote nodes through possible multiple hops should not require (considerably) more processing at each hop than plain fast path packet forwarding.

3. The mechanism should involve simple and efficient processing to be feasible for implementation in the dataplane. This goal manifests itself in three ways:

    a. Origination of notification should be very easy, e.g. creating a simple IP packet, the payload of which can be filled easily.

b. When receiving the packet, it should be easy to recognize by dataplane linecards so that processing can commence after forwarding.

c. No complex operations should be required in order to extract the information from the packet needed to activate the correct backup routes.

4. The mechanism should be trustable; that is, it should provide means to verify the authenticity of the notifications without significant increase of the processing burden in the dataplane.

5. Duplication of notification packets should be either strictly bounded or handled without significant dataplane processing burden.

These requirements present a trade-off. A proper balance needs to be found that offers good enough authentication and reliability while keeping processing complexity sufficiently low to be feasible for data plane implementation. One such solution is proposed in [fn-transport], which is the assumed notification protocol in the following.

4. Conceptual Operation of IPFRR relying on Fast Notification

This section outlines the operation of an IPFRR mechanism relying on Fast Notification.

4.1. Preparation Phase

As any other IPFRR solution, IPFRR-FN also requires quick failure detection mechanisms in place, such as lower layer upcalls or BFD. The FN service needs to be activated and configured so that FN disseminates the information identifying the failure to the area once triggered by a local failure detection method.

Based on the detailed topology database obtained by a link state IGP, the node should pre-calculate alternative paths considering *relevant* link or node failures in the area. Failure specific alternative path computation should typically be executed at lower priority than other routing processing. Note that the calculation can be done "offline", while the network is intact and the CP has few things to do.

Also note the word *relevant* above: a node does not needed to compute all the shortest paths with respect to each possible failure;

only those link failures need to be taken into consideration, which are in the shortest path tree starting from the node.

To provide protection for Autonomous System Border Router (ASBR) failures, the node will need information not only from the IGP but also from BGP. This is described in detail in Section 5.3.

After calculating the failure specific alternative next-hops, only those which represent a change to the primary next-hop, should be pre-installed to the linecards together with the identifier of the failure, which triggers the switch-over. In order to preserve scalability, external prefixes are handled through FIB indirection available in most routers already. Due to indirection, backup routes need to be installed only for egress routers. (The resource needs of an example implementation are briefly discussed in Appendix A.)

4.2. Failure Reaction Phase

The main steps to be taken after a failure are the following:

1. Quick dataplane failure detection

2. Send information about failure using FN service right from dataplane.

3. Forward the received notification as defined by the actually used FN protocol such as the one in [fn-transport]

4. After learning about a local or remote failure, extract failure identifier and activate failure specific backups, if needed, directly within dataplane

5. Start forwarding data traffic using the updated FIB

After a node detects the loss of connectivity to another node, it should make a decision whether the failure can be handled locally. If local repair is not possible or not configured, for example because LFA is not configured or there are destinations for which no LFA exists, a failure should trigger the FN service to disseminate the failure description. For instance, if BFD detects a dataplane failure it not only should invoke routines to notify the control plane but it should first trigger FN before notifying the CP.

After receiving the trigger, without any DP-CP communication involved, FN constructs a packet and adds the description of the failure (described in Section 5.1.) to the payload. The notification describes that

- a node X has lost connectivity
- to a node Z
- via a link L.

The proposed encoding of the IPFRR-FN packet is described in Section 5.1.

The packet is then disseminated by the FN service in the routing area. Note the synergy of the relation between BFD and IGP Hellos and between FN and IGP link state advertisements. BFD makes a dataplane optimized implementation of the routing protocol's Hello mechanism, while Fast Notification makes a dataplane optimized implementation of the link state advertisement flooding mechanism of IGPs.

In each hop, the recipient node needs to perform a "punt and forward". That is, the FN packet not only needs to be forwarded to the FN neighbors as the specific FN mechanism dictates, but a replica needs to be detached and, after forwarding, started to be processed by the dataplane card.

4.2.1. Activating Failure Specific Backups

After the forwarding element extracted the contents of the notification packet, it knows that a node X has lost connectivity to a node Z via a link L. The recipient now needs to decide whether the failure was a link or a node failure. Two approaches can be thought of. Both options are based on the property that notifications advance in the network as fast as possible.

In the first option, the router does not immediately make the decision, but instead starts a timer set to fire after a couple of milliseconds. If, the failure was a node failure, the node will receive further notifications saying that another node Y has lost connectivity to node Z through another link M. That is, if node Z is common in multiple notifications, the recipient can conclude that it is a node failure and already knows which node it is (Z). If link L is common, then the recipient can decide for link failure (L). If further inconclusive notifications arrive, then it means multiple failures which case is not in scope for IPFRR, and is left for regular IGP convergence.

After concluding about the exact failure, the data plane element needs to check in its pre-installed IPFRR database whether this particular failure results in any route changes. If yes, the linecard replaces the next-hops impacted by that failure with their failure specific backups which were pre-installed in the preparation phase.

In the second option, the first received notification is handled immediately as a link failure, hence the router may start replacing its next-hops. In many cases this is a good decision, as it has been shown before that most network failures are link failures. If, however, another notification arrives a couple of milliseconds later that points to a node failure, the router then needs to start replacing its next-hops again. This may cause a route flap but due to the quick dissemination mechanism the routing inconsistency is very short lived and likely takes only a couple of milliseconds.

4.2.2. SRLG Handling

The above conceptual solution is easily extensible to support pre-configured SRLGs. Namely, if the failed link is part of an SRLG, then the disseminated link ID should identify the SRLG itself. As a result, possible notifications describing other link failures of the same SRLG will identify the same resource.

If the control plane knows about SRLGs, it can prepare for failures of these, e.g. by calculating a path that avoids all links in that SRLG. SRLG identifier may have been pre-configured or have been obtained by automated mechanisms such as [RFC4203].

4.3. Example and Timing

The main message of this section is that big delay links do not represent a problem for IPFRR-FN. The FN message of course propagates on long-haul links slower but the same delay is incurred by normal data packets as well. Packet loss only takes place as long as a node forwards traffic to an incorrect or inconsistent next-hop. This may happen in two cases:

First, as long as the failure is not detected, the node adjacent to the failure only has the failed next-hop installed.

Secondly, when a node (A) selects a new next-hop (B) after detecting the failure locally or by receiving an FN, the question is if the routing in the new next-hop (B) is consistent by the time the first data packets get from A to B. The following timeline depicts the situation:

```
   Legend: X : period with packet loss
           FN forwarding delay: |--|

         |--|--------|
   A:----1XX2XXXXXXXX3--------------------------------------------------
            |----|   |----|
   B:------------4--5-----6XX7------------------------------------------
                 |--|--------|

   (a) Link delay is |----| FIB update delay is |--------|



         |--|--------|
   A:----1XX2XXXXXXXX3--------------------------------------------------
            |---------------|
                     |---------------|
   B:-----------------------4--5-----6XX7-------------------------------
                            |--|--------|

   (b) Link delay is |---------------| FIB update delay is |--------|
```

Figure 2 Timing of FN and data packet forwarding

As can be seen above, the outage time is only influenced by the FN forwarding delay and the FIB update time. The link delay is not a factor. Node A forwards the first re-routed packets from time instance 3 to node B. These reach node B at time instance 6. Node B is doing incorrect/inconsistent forwarding when it tries to forward those packets back to A which have already been put onto a detour by A. This is the interval between time instances 6 and 7.

4.4. Scoping FN Messages with TTL

In a large routing area it is often the case that a failure (i.e. a topology change) causes next-hop changes only in routers relatively close to the failure. Analysis of certain random topologies and two example ISP topologies revealed that a single link failure event generated routing table changes only in routers not more than 2 hops away from the failure site for the particular topologies under study [Hok2008]. Based on this analysis, it is anticipated that in practice the TTL for failure notification messages can be set to a relatively small radius, perhaps as small as 2 or 3 hops.

A chief benefit of TTL scoping is that it reduces the overhead on routers that have no use for the information (i.e. which do not need to re-route). Another benefit (that is particularly important for links with scarce capacity) is that it helps to constrain the control overhead incurred on network links. Determining a suitable TTL value for each locally originated event and controlling failure notification dissemination, in general, is discussed further in Section 5.8.

5. Operation Details

5.1. Transport of Fast Notification Messages

This draft recommends that out of the several FN delivery options defined in [fn-transport], the flooding transport option is preferred, which ensures that any event can reach each node from any source with any failure present in the network area as long as theoretically possible. Flooding also ensures that FN messages reach each node on the shortest (delay) path, and as a side effect failure notifications always reach *each* node *before* re-routed data packets could reach that node. This means that looping is minimized.

[fn-transport] describes that the dataplane flooding procedure requires routers to perform duplicate checking before forwarding the notifications to other interfaces to avoid duplicating notifications. [fn-transport] describes that duplicate check can be performed by a simple storage queue, where previously received notification packets or their signatures are stored.

IPFRR-FN enables another duplicate check process that is based on the internal state machine. Routers, after receiving a notification but before forwarding it to other peers, check the authenticity of the message, if authentication is used. Now the router may check what is the stored event and what is the event described by the received notification.

Two variables and a bit describe what is the known failure state:

- o Suspected failed node ID (denoted by N)

- o Suspected link/SRLG ID (denoted by S)

- o Bit indicating the type of the failure, i.e. link/SRLG failure or node failure (denoted by T)

Recall that the incoming notification describes that a node X has lost connectivity to a node Z via a link L. Now, the state machine can be described with the following pseudo-code:

```
//current state:
//  N: ID of suspected failed node
//  S: ID of suspected failed link/SRLG
//  T: bit indicating the type of the failure
//     T=0 indicates link/SRLG
//     T=1 indicates node
//
Proc notification_received(Node Originator_X, Node Y, SRLG L) {
    if (N == NULL) {
        // this is a new event, store it and forward it
        N=Y;
        S=L;
        T=0; //which is the default anyway
        Forward_notification;
    }
    else if (S == L AND T == 0) {
        // this is the same link or SRLG as before, need not do
        // anything
        Discard_notification;
    }
    else if (N == Y) {
        // This is a node failure
        if (T == 0) {
            // Just now turned out that it is a node failure
            T=1;
            Forward_notification;
        }
        else {
            // Known before that it is a node failure,
            // no need to forward it
            Discard_notification;
        }
    }
    else {
        // multiple failures
    }
}
```

Figure 3 Pseudo-code of state machine for FN forwarding

5.2. Message Handling and Encoding

A failure identifier is needed that unambiguously describes the failed resource consistently among the nodes in the area. The schemantics of the identifiers are defined by the IGP used to pre-calculate and pre-install the backup forwarding entries, e.g. OSPF or ISIS.

This draft defines a Failure Identification message class. Members of this class represent a routing protocol specific Failure Identification message to be carried with the Fast Notification transport protocol. Each message within the Failure Identification message class shall contain the following fields, the lengths of which are routing protocol specific. The exact values shall be aligned with the WG of the routing protocol:

- o Originator Router ID: the identifier of the router advertising the failure;
- o Neighbour Router ID: the identifier of the neighbour node to which the originator lost connectivity.
- o Link ID: the identifier of the link, through which connectivity was lost to the neighbour. The routing protocol should assign the same Link ID for bidirectional, broadcast or multi access links from each access point, consistently.
- o Sequence Number: [fn-transport] expects the applications of the FN service that require replay attack protection to create and verify a sequence number in FN messages. It is described in Section 6.

Routers forwarding the FN packets should ensure that Failure Identification messages are not lost, e.g. due to congestion. FN packets can be put a high precedence traffic class (e.g. Network Control class). If the network environment is known to be lossy, the FN sender should repeat the same notification a couple of times, like a salvo fire.

After the forwarding element processed the FN packet and extracted the Failure Identification message, it should decide what backups need to be activated if at all - as described in Section 4.2.1.

5.2.1. Failure Identification Message for OSPF

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           FN Length           |  FN App Type  | AuType|unused |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Originator Router ID                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Neighbour Router ID                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Link ID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
 |                      Sequence Number                              |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                      Sequence Number (cont'd)                     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

FN Header fields:

FN Length
The length of the Failure Identification message for OSPF is 16 bytes.

FN App Type
The exact values are to be assigned by IANA for the Failure Identification message class. For example, FN App Type values between 0x0008 and 0x000F could represent Failure Identification messages, from which 0x0008 could mean OSPF, 0x0009 could be ISIS.

AuType
IPFRR-FN relies on the authentication options offered the FN transport service. Cryptographic authentication is recommended.

Originator Router ID
If the routing protocol is OSPF, then the value can take the OSPF Router ID of the advertising router.

Neighbour Router ID
The OSPF Router ID of the neighbour router to which connectivity was lost.

Link ID
If the link is a LAN, the Link ID takes the LSAID of its representing Network LSA.
If the link is a point-to-point link, the Link ID can take the minimum or the maximum of the two interface IDs. The requirement is that it is performed consistently.

Sequence Number
This field stores a digest of the LSDB of the routing protocol, as described in Section 6. 5.8.1.

5.2.2. Failure Identification Message for ISIS

TBA.

5.3. Protecting External Prefixes

5.3.1. Failure on the Intra-Area Path Leading to the ASBR

Installing failure specific backup next-hops for each external prefix would be a scalability problem as the number of these prefixes may be one or two orders of magnitude higher than intra-area destinations. To avoid this, it is suggested to make use of indirection already offered by router vendors.

Indirection means that when a packet needs to be forwarded to an external destination, the IP address lookup in the FIB will not return a direct result but a pointer to another FIB entry, i.e. to the FIB entry of the ASBR. In LDP/MPLS this means that all prefixes reachable through the same ASBR constitute the same FEC.

As an example, consider that in an area ASBR1 is the primary BGP route for prefixes P1, P2, P3 and P4 and ASBR2 is the primary route for prefixes P5, P6 and P7. A FIB arrangement for this scenario could be the one shown on the following figure. Prefixes using the same ASBR could be resolved to the same pointer that references to the next-hop leading to the ASBR. Prefixes resolved to the same pointer are said to be part of the same "prefix group" or FEC.

```
      FIB lookup         |         FIB lookup
                         |
ASBR2 ========> NH2      |    ASBR2 ========> NH2 <----+
ASBR1 ========> NH1      |    ASBR1 ========> NH1 <-+  |
                         |                          |  |
P1    ========> NH1      |    P1    ========> Ptr1 -+  |
P2    ========> NH1      |    P2    ========> Ptr1 -+  |
P3    ========> NH1      |    P3    ========> Ptr1 -+  |
P4    ========> NH1      |    P4    ========> Ptr1 -+  |
                         |                             |
P5    ========> NH2      |    P5    ========> Ptr2 ----+
P6    ========> NH2      |    P6    ========> Ptr2 ----+
P7    ========> NH2      |    P7    ========> Ptr2 ----+
                         |
```

Figure 4 FIB without (left) and with (right) indirection

If the next-hop to an ASBR changes, it is enough to update in the FIB the next-hop of the ASBR route. In the above example, this means that if the next-hop of ASBR1 changes, it is enough to update the route entry for ASBR1 and due to indirection through pointer Ptr1 this updates several prefixes at the same time.

5.3.2. Protecting ASBR Failures: BGP-FRR

IPFRR-FN can make use of alternative BGP routes advertised in an AS by new extensions of BGP such as [BGPAddPaths], [DiverseBGP] or [BGPBestExt]. Using these extensions, for each destination prefix, a node may learn a "backup" ASBR besides the primary ASBR learnt by normal BGP operation.

5.3.2.1. Primary and Backup ASBR in the Same Area

If the failed ASBR is inside the area, all nodes within that area get notified by FN. Grouping prefixes into FECs, however, needs to be done carefully. Prefixes now constitute a common group (i.e. are resolved to the same pointer) if *both* their primary AND their backup ASBRs are the same. This is due to the fact that even if two prefixes use the ASBR by default, they may use different ASBRs when their common default ASBR fails.

Considering the previous example, let us assume that the backup ASBR of prefixes P1 and P2 is ASBR3 but that the backup ASBR of P3 and P4 is an ASBR2. Let us further assume that P5 also has ASBR3 as its backup ASBR but P6 and P7 have an ASBR 4 as their backup ASBR. The resulting FIB structure is shown in the following figure:

```
      FIB lookup
ASBR4 ========> NH4
ASBR2 ========> NH2
ASBR3 ========> NH3
ASBR1 ========> NH1

P1    ========> Ptr1 -+-> NH1
P2    ========> Ptr1 -+

P3    ========> Ptr2 -+-> NH1
P4    ========> Ptr2 -+

P5    ========> Ptr3 ---> NH2

P6    ========> Ptr4 -+-> NH2
P7    ========> Ptr4 -+
```

Figure 5 Indirect FIB for ASBR protection

If, for example, ASBR1 goes down, this affects prefixes P1 through P4. In order to set the correct backup routes, the container referenced by Ptr1 needs to be updated to NH2 (next-hop of ASBR2) but the location referenced by Ptr2 needs to be updated to NH3 (next-hop of ASBR3). This means that P1 and P2 may constitute the same FEC but P3 and P4 needs to be another FEC so that there backups can be set independently.

Note that the routes towards ASBR2 or ASBR3 may have changed, too. For example, if after the failure ASBR3 would use a new next-hop NH5, then the container referenced by Ptr2 should be updated to NH5. A resulting detour FIB is shown in the following figure.

```
      FIB lookup
ASBR4 ========>    NH4
ASBR2 ========>    NH2
ASBR3 ========>    NH5
ASBR1 ========>     X

P1    ========> Ptr1 -+-> NH2
P2    ========> Ptr1 -+

P3    ========> Ptr2 -+-> NH5
P4    ========> Ptr2 -+

P5    ========> Ptr3 ---> NH2

P6    ========> Ptr4 -+-> NH2
P7    ========> Ptr4 -+
```

Figure 6 Indirect "detour" FIB in case of ASBR1 failure

During pre-calculation, the control plane pre-downloaded the failure identifier of ASBR1 and assigned NH5 as the failure specific backup for routes for ASBR3 and pointer Ptr2 and assigned NH2 as the failure specific backup for the route referenced by Ptr1.

5.3.2.2. Primary and Backup ASBR in Different Areas

By default, the scope of FN messages is limited to a single routing area.

The IPFRR-FN application of FN, may, however, need to redistribute some specific notifications across areas in a limited manner.

If an ASBR1 in Area1 goes down and some prefixes need to use ASBR2 in another Area2, then, besides Area1, routers in Area2 need to know about this failure. Since communication between non-backbone areas is done through the backbone areas, it may also need the information.

Naturally, if ASBR2 resides in the backbone area, then the FN of ASBR1 failure needs to be leaked only to the backbone area.

Leaking is facilitated by area border routers (ABR). During failure preparation phase, the routing engine of an ABR can determine that for an intra-area ASBR the backup ASBR is in a different area to which it is the ABR. Therefore, the routing engine installs such intra-area ASBRs in an "FN redistribution list" at the dataplane cards.

The ABR, after receiving FN messages, may conclude in its state machine that a node failure happened. If this node failure is in the redistribution list, the ABR will generate an FN with the following data:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               16              |     0x008     | AuType|unused |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         ABR Router ID                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        ASBR Router ID                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               0x0                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Sequence Number                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Sequence Number (cont'd)                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

This message is then distributed to the neighbour area specified in the redistribution list as a regular FN message. A Link ID of 0x0 specifically signals in the neighbour area that this failure is a known node failure of the node specified by the "Neighbour Router ID" field (which was set to the failed ASBR's ID).

ABRs in a non-backbone area need to prepare to redistribute ASBR failure notifications from within their area to the backbone area.

ABRs in the backbone area need to prepare to redistribute an ASBR failure notification from the backbone area to that area where a backup ASBR resides.

Consider the previous example, but now let us assume that the current area is Area0, ASBR2 and ASBR3 reside in Area1 (reachable through ABR1) but ASBR 4 resides in Area2 (reachable through ABR2). The resulting FIBs are shown in the following figures: in case of ASBR2 failure, only Ptr4 needs an update.

```
    FIB lookup
ABR1 ========> NH6
ABR2 ========> NH7

(ASBR4 ========> NH7)  //may or may not be in the FIB
(ASBR2 ========> NH6)  //may or may not be in the FIB
(ASBR3 ========> NH6)  //may or may not be in the FIB
(ASBR1 ========> NH1)  //may or may not be in the FIB

P1   ========> Ptr1 -+-> NH1
P2   ========> Ptr1 -+

P3   ========> Ptr2 -+-> NH1
P4   ========> Ptr2 -+

P5   ========> Ptr3 ---> NH6

P6   ========> Ptr4 -+-> NH6
P7   ========> Ptr4 -+
```

Figure 7 Indirect FIB for inter-area ASBR protection

```
        FIB lookup
   ABR1 ========>    NH6
   ABR2 ========>    NH7

   (ASBR4  =======> NH7)  //may or may not be in the FIB
   (ASBR2  =======>  X )  //may or may not be in the FIB
   (ASBR3 ========> NH6)  //may or may not be in the FIB
   (ASBR1 ========> NH1)  //may or may not be in the FIB

   P1   ========> Ptr1 -+-> NH1
   P2   ========> Ptr1 -+

   P3   ========> Ptr2 -+-> NH1
   P4   ========> Ptr2 -+

   P5   ========> Ptr3 ---> NH6

   P6   ========> Ptr4 -+-> NH7
   P7   ========> Ptr4 -+
```

Figure 8 Indirect "detour" FIB for inter-area ASBR protection, ASBR2 failure

5.4. Application to LDP

It is possible for LDP traffic to follow paths other than those indicated by the IGP. To do so, it is necessary for LDP to have the appropriate labels available for the alternate so that the appropriate out-segments can be installed in the forwarding plane before the failure occurs.

This means that a Label Switching Router (LSR) running LDP must distribute its labels for the Forwarding Equivalence Classes (FECs) it can provide to all its neighbours, regardless of whether or not they are upstream. Additionally, LDP must be acting in liberal label retention mode so that the labels that correspond to neighbours that aren’t currently the primary neighbour are stored. Similarly, LDP should be in downstream unsolicited mode, so that the labels for the FEC are distributed other than along the SPT.

The above criteria are identical to those defined in [RFC5286].

In IP, a received FN message may result in rewriting the next-hop in the FIB. If LDP is applied, the label FIB also needs to be updated in accordance with the new next-hop; in the LFIB, however, not only the outgoing interface needs to be replaced but also the label that is valid to this non-default next-hop. The latter is available due to liberal label retention and unsolicited downstream mode.

5.5. Application to VPN PE Protection

Protecting against (egress) PE router failures in VPN scenarios is conceptually similar to protecting against ASBR failures for Internet traffic. The difference is that in case of ASBR protection core routers are normally aware of external prefixes using iBGP, while in VPN cases P routers can only route inside the domain. In case of VPNs, tunnels running between ingress PE and egress PE decrease the burden for P routers. The task here is to redirect traffic to a backup egress PE.

Egress PE protection effectively calls out for an explicit failure notification, yet existing proposals try to avoid it.

[I-D.bashandy-bgp-edge-node-frr] proposes that the P routers adjacent to the primary PE maintain the necessary routing state and perform the tunnel decaps/re-encaps to the backup PE, thereby proposing considerable complexity for P routers.

[I-D.ietf-pwe3-redundancy] describes a mechanism for pseudowire redundancy, where PE routers need to run multi-hop BFD sessions to detect the loss of a primary egress PE. This leads to a potentially full mesh of multihop BFD session, which is a tremendous complexity. In addition, in some cases the egress PE of the secondary PW might need to explicitly set the PW state from standby to active.

FN provides the needed mechanism to actively inform all nodes including PE routers that a failure happened, and also identifies that a node failure happened. Furthermore, since both the ingress PE and the secondary egress PE are informed, all information is available for a proper switch-over. This is without a full mesh of BFD sessions running all the time between PE routers.

5.6. Bypassing Legacy Nodes

Legacy nodes, while cannot originate fast notifications and cannot process them either, can be assumed to be able to forward the notifications. As [fn-transport] discusses, FN forwarding is based on multicast. It is safe to assume that legacy routers' multicast configuration can be set up statically so as to be able to propagate fast notifications as needed.

When calculating failure specific alternative routes, IPFRR-FN capable nodes must consider legacy nodes as being fixed directed links since legacy nodes do not change packet forwarding in the case of failure. There are situations when an FN-IPFRR capable node can, exceptionally, bypass a non-IPFRR-FN capable node in order to handle a remote failure.

As an example consider the topology depicted in Figure 9, where the link between C and D fails. C cannot locally repair the failure.

```
 +---+   +---+   +---+   +---+
 | E |---| F |---| G |---| H |
 +---+   +---+   +---+   +---+
   |           /           |
   |          /            |
   |         /             |
 +---+   +---+   +---+   +---+
 | A |---| B |---| C |-X-| D |
 +---+   +---+   +---+   +---+
   >========>==============>
      Traffic from A to D
```

Figure 9 Example for bypassing legacy nodes

First, let us assume that each node is IPFRR-FN capable. C would advertise the failure information using FN. Each node learns that the link between C and D fails, as a result of which C changes its forwarding table to send any traffic destined to D via B. B also makes a change, replacing its default next-hop (C) with G. Note that other nodes do not need to modify their forwarding at all.

Now, let us assume that B is a legacy router not supporting IPFRR-FN but it is statically configured to multicast fast notifications as needed. As such, A will receive the notification. A's pre-calculations have been done knowing that B is unable to correct the failure. Node A, therefore, has pre-calculated E as the failure specific next-hop. Traffic entering at A and heading to D can thus be repaired.

5.7. Capability Advertisement

The solution requires nodes to know which other nodes in the area are capable of IPFRR-FN. The most straightforward way to achieve this is to rely on the Router Capability TLVs available both in OSPF [RFC4970] and in IS-IS [RFC4971].

5.8. Constraining the Dissemination Scope of Fast Notification Packets

As discussed earlier in Section 4.4. it is desirable to constrain the dissemination scope of failure notification messages. This section presents three candidate methods for controlling the scope of failure notification: (1) Pre-configure the TTL for FN messages in routers based on best current practices and related studies of available ISP and enterprise network topologies; (2) dynamically calculate the minimum TTL value needed to ensure 100% remote LFAP coverage; and (3) dynamically calculate the set of neighbours for which FN message should given the identity of the link that has failed.

These candidate dissemination options are mechanisms with different levels of optimality and complexity. The intent here is to present some options that will generate further discussion on the tradeoffs between different FN message scoping methods.

5.8.1. Pre-Configured FN TTL Setting

As discussed, earlier in Section 4.4. studies of various network topologies suggest that a fixed TTL setting of 2 hops may be sufficient to ensure failure notification message for typical OSPF area topologies. Therefore, a potentially simple solution for constraining FN message dissemination is for network managers to configure their routers with fixed TTL setting (e.g., TTL=2 hops) for FN messages. This TTL setting can be adjusted by network managers to consider implementation-specific details of the topology such as configuring a larger TTL setting for topologies containing, say, large ring sub-graph structures.

In terms of performance trades, pre-configuring the FN TTL, since it is fixed at configuration time, incurs no computational overhead for the router. On the other hand, it represents a configurable router parameter that network administrators must manage. Furthermore, the fixed, pre-configured FN TTL approach is sub-optimal in terms of constraining the FN dissemination as most single link events will not require FN messages send to up to TTL hops away from the failure site.

5.8.2. Advanced FN Scoping

While the static pre-configured setting of the FN TTL will likely work in practice for a wide range of OSPF area topologies, it has at two least weaknesses: (1) There may be certain topologies for which the TTL setting happens to be insufficient to provide the needed failure coverage; and (2) as discussed above, it tends to result in

FN being disseminated to a larger radius than needed to facilitate re-routing.

The solution to these drawbacks is for routers to dynamically compute the FN TTL radius needed for each of the local links it monitors. Doing so addresses the two weakness of a pre-configured TTL setting by computing a custom TTL setting for each of its local links that matches exactly the FN message radius for the given topology. The drawback, of course, is the additional computations. However, given a quasi-static network topology, it is possible this dynamic FN TTL computation is performed infrequently and, therefore, on average incurs relatively small computation overhead.

While a pre-configured TTL eliminates computation overhead at the expense of FN dissemination overhead and dynamic updates of the TTL settings achieve better dissemination efficiency by incurring some computational complexity, directed FN message forwarding attempts to minimize the FN dissemination scope by leveraging additional computation power. Here, rather than computing a FN TTL setting for each local link, a network employing directed forwarding has each router instance R compute the sets of one-hop neighbours to which a FN message must be forwarded for every possible failure event in the routing area. This has the beneficial effect of constraining the FN scope to the direction where there are nodes that require the FN update as opposed to disseminating to the entire TTL hop radius about a failure site. The trade off here, of course, is the additional computation complexity incurred and the maintenance of forwarding state for each possible failure case. Reference [Cev2010] gives an algorithm for finding, for each failure event, the direct neighbours to which the notification should be forwarded.

6. Protection against Replay Attacks

To defend against replay attacks, recipients should be able to ignore a re-sent recording of a previously sent FN packet. This suggests that some sort of sequence number should be included in the FN packet, the verification of which should not need control plane involvement. Since the solution should be simple to implement in the dataplane, maintaining and verifying per-source sequence numbers is not the best option.

We propose, therefore, that messages should be stamped with the digest of the actual routing configuration, i.e., a digest of the link state database of the link state routing protocol. The digest has to be picked carefully, so that if two LSDBs describe the same connectivity information, their digest should be identical as well, and different LSDBs should result in different digest values with high probability.

The conceptual way of handling these digests could be the following:

- o When the LSDB changes, the IGP re-calculates the digest and downloads the new value to the dataplane element(s), in a secure way.

- o When a FN packet is originated, the digest is put into the FN message into the Sequence Number field.

- o Network nodes distribute (forward) the FN packet.

- o When processing, the dataplane element first performs an authentication check of the FN packet, as described in [fn-transport].

- o Finally, before processing the failure notification, the dataplane element should check whether its own known LSDB digest is identical with the one in the message.

If due to a failure event a node disseminates a failure notification with FN, an attacker might capture the whole packet and re-send it later. If it resends the packet after the IGP re-converged on the new topology, the active LSDB digest is different, so the packet can be ignored. If the packet is replayed to a recipient who still has the same LSDB digest, then it means that the original failure notification was already processed but the IGP has not yet finished converging; the IPFRR detour is already active, the replica has no impact.

6.1. Calculating LSDB Digest

We propose to create an LSDB digest that is conceptually similar to [ISISDigest]. The operation is proposed to be the following:

- o Create a hash from each LSA(OSPF)/LSP(ISIS) one by one

- o XOR these hashes together

- o When an LSA/LSP is removed, the new LSDB digest is received by computing the hash of the removed LSA, and then XOR to the existing digest

- o When an LSA/LSP is added, the new LSDB digest is received by computing the hash of the new LSA, and then XOR to the existing digest

7. Security Considerations

The IPFRR application of Fast Notification does not raise further known security consideration in addition to those already present in Fast Notification itself. If an attacker could send false Failure Identification Messages or could hinder the transmission of legal messages, then the network would produce an undesired routing behaviour. These issues should be solved, however, in [fn-transport].

IPFRR-FN relies on the authentication mechanism provided by the Fast Notification transport protocol [fn-transport]. The specification of the FN transport protocol requires applications to protect against replay attacks with application specific sequence numbers. This draft, therefore, describes its own proposed sequence number in Section 5.8.1.

8. IANA Considerations

The Failure Identification message types need to be allocated a value in the FN App Type field.

IPFRR-FN capability needs to be allocated within Router Capability TLVs both for OSPF [RFC4970] and in IS-IS [RFC4971].

9. References

9.1. Normative References

[RFC5286]
A. Atlas, A. Zinin, "Basic specification for IP Fast-Reroute: Loop-Free Alternates", Internet Engineering Task Force: RFC 5286, 2008.

[fn-transport]
W. Lu, S. Kini, A. Csaszar, G. Enyedi, J. Tantsura, A. Tian, "Transport of Fast Notifications Messages", draft-lu-fn-transport, 2011

[RFC4970]
A. Lindem et al., Extensions to OSPF for Advertising Optional Router Capabilities, RFC 4970, 2007

[RFC4971]
JP. Vasseur et al., Intermediate System to Intermediate System (IS-IS) Extensions for Advertising Router Information, RFC 4971, 2007

[RFC4203]
K. Kompella, Y. Rekhter, " OSPF Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS)", RFC4203, 2005

9.2. Informative References

[BFD]
D. Katz, D. Ward, "Bidirectional forwarding detection", RFC 5880, IETF, 2010

[RFC5714]
M. Shand, S. Bryant, "IP Fast Reroute Framework", RFC 5714, IETF, 2010.

[Cev2010]
S. Sevher, T. Chen, I. Hokelek, J. Kang, V. Kaul, Y.J. Lin, M. Pang, R. Rodoper, S. Samtani, C. Shah, J. Bowcock, G. B. Rucker, J. L. Simbol and A. Staikos, "An Integrated Soft Handoff Approach in IP Fast Reroute in Wireless Mobile Networks", In Proceedings IEEE COMSNETS, 2011.

[Eny2009]
Gabor Enyedi, Peter Szilagyi, Gabor Retvari, Andras Csaszar, "IP Fast ReRoute: Lightweight Not-Via without Additional Addresses", IEEE INFOCOM-MiniConference, Rio de Janeiro, Brazil, 2009.

[FIFR]
J. Wand, S. Nelakuditi, "IP fast reroute with failure inferencing", In Proceedings of ACM SIGCOMM Workshop on Internet Network Management - The Five-Nines Workshop, 2007.

[Hok2007]
I. Hokelek, M. A. Fecko, P. Gurung, S. Samtani, J. Sucec, A. Staikos, J. Bowcock and Z. Zhang, "Seamless Soft Handoff in Wireless Battlefield Networks Using Local and Remote LFAPs", In Proceedings IEEE MILCOM, 2007.

[Hok2008]
I. Hokelek, S. Cevher, M. A. Fecko, P. Gurung, S. Samtani, Z. Zhang, A. Staikos and J. Bowcock, "Testbed Implementation of Loop-Free Soft Handoff in Wireless Battlefield Networks", In Proceedings of the 26th Army Science Conference, December 1-4, 2008.

[MRC]
T. Cicic, A. F. Hansen, A. Kvalbein, M. Hartmann, R. Martin, M. Menth, S. Gjessing, O. Lysne, "Relaxed multiple routing configurations IP fast reroute for single and correlated failures", IEEE Transactions on Network and Service Management, available online: http://www3.informatik.uni-wuerzburg.de/staff/menth/Publications/papers/Menth08-Sub-4.pdf, September 2010.

[NotVia]
S. Bryant, M. Shand, S. Previdi, "IP fast reroute using Not-via addresses", Internet Draft, draft-ietf-rtgwg-ipfrr-notvia-addresses, 2010.

[RLFAP]
I. Hokelek, M. Fecko, P. Gurung, S. Samtani, S. Cevher, J. Sucec, "Loop-Free IP Fast Reroute Using Local and Remote LFAPs", Internet Draft, draft-hokelek-rlfap-01 (expired), 2008.

[Ret2011]
G. Retvari, J. Tapolcai, G. Enyedi, A. Csaszar, "IP Fast ReRoute: Loop Free Alternates Revisited", to appear at IEEE INFOCOM 2011

[ISISDigest]
J. Chiabaut and D. Fedyk. IS-IS Multicast Synchronization Digest. Available online: http://www.ieee802.org/1/files/public/docs2008/aq-fedyk-ISIS-digest-1108-v1.pdf, Nov 2008.

[BGPAddPaths]
D. Walton, A. Retana, E. Chen, J. Scudder, "Advertisement of Multiple Paths in BGP", draft-ietf-idr-add-paths, Work in progress

[DiverseBGP]
R. Raszuk, et. Al, "Distribution of diverse BGP paths", draft-ietf-grow-diverse-bgp-path-dist, Work in progress

[BGPBestExt]
P. Marques, R. Fernando, E. Chen, P. Mohapatra, H. Gredler, "Advertisement of the best external route in BGP", draft-ietf-idr-best-external, Work in progress

[BRITE]
Oliver Heckmann et al., "How to use topology generators to create realistic topologies", Technical Report, Dec 2002.

[MRT-ARCH]
A. Atlas et al., "An Architecture for IP/LDP Fast-Reroute Using Maximally Redundant Trees", Internet Draft, draft-ietf-rtgwg-mrt-frr-architecture-01, 2012

[MRT-ALG]
A. Atlas, G. Enyedi, A. Csaszar, "Algorithms for computing Maximally Redundant Trees for IP/LDP Fast-Reroute", Internet Draft, draft-enyedi-rtgwg-mrt-frr-algorithm-01, 2012

[I-D.ietf-pwe3-redundancy]
P. Muley, M. Aissaoui, M. Bocci, "Pseudowire Redundancy", draft-ietf-pwe3-redundancy (Work in progress!), May 2012

[I-D.bashandy-bgp-edge-node-frr]
A. Bashandy, B. Pithawala, K. Patel, "Scalable BGP FRR Protection against Edge Node Failure", draft-bashandy-bgp-edge-node-frr (Work in progress!), March 2012

10. Acknowledgments

The authors would like to thank Albert Tian, Wenhu Lu, Acee Lindem and Ibrahim Hokelek for the continuous discussions and comments on the topic, as well as Joel Halpern for his comments and review.

Appendix A. Memory Needs of a Naive Implementation

Practical background might suggest that storing and maintaining backup next-hops for many potential remote failures could overwhelm the resources of router linecards. This section attempts to provide a calculation describing the approximate memory needs in reasonable sized networks with a possible implementation.

A.1. An Example Implementation

Let us suppose that for exterior destinations the forwarding engine is using recursive lookup or indirection in order to improve updating time such as described in Section 5.3. We are also supposing that the concept of "prefix groups" is applied, i.e. there is an internal entity for the prefixes using exactly the same primary and backup ASBRs, and the next hop entry for a prefix among them is pointing to the next hop towards this entity. See e.g. Figure 7.

In the sequel, the term of "area" refers to an extended area, made up by the OSPF or IS-IS area containing the router, with the prefix groups added to the area as virtual nodes. Naturally, a prefix group is connected to the egress routers (ABRs) through which it can be reached. We just need to react to the failure ID of an ASBR for all the prefix groups connected to that ASBR; technically, we must suppose that one of the virtual links of all the affected prefix groups go down.

Here we show a simple naive implementation which can easily be beaten in real routers. This implementation uses an array for all the nodes (including real routers and virtual nodes representing prefix groups) in the area (node array in the sequel), made up by two pointers and a length filed (an integer) per record. One of the pointers points to another array (called alternative array). That second array is basically an enumeration containing the IDs of those failures influencing a shortest path towards that node and an alternative neighbor, which can be used, when such a failure occurs. When a failure is detected, (either locally, or by FN), we can easily find the proper record in all the lists. Moreover, since these arrays can be sorted based on the failure ID, we can even use binary search to find the needed record. The length of this array is stored in the record of the node array pointing to the alternative list.

Now, we only need to know, which records in the FIB should be updated. Therefore there is a second pointer in the node array pointing to that record.

```
          +-------+-------+-------+--   --+-------+
          |  r1   |  r2   |  r3   |  ...  |  rk   |
          +-------+-------+-------+--   --+-------+
              |       |       |               |
              |       |       |               |
              |       |       |               |
             \|/     \|/     \|/             \|/
              *       *       *               *
          +-------+-------+-------+--   --+-------+
          | fail1 | fail2 | fail3 |       | failk |
          | alt.1 | alt.2 | alt.3 |  ...  | alt.k |
          +-------+-------+-------+--   --+-------+
          | fail4 |       | fail5 |
          | alt.4 |       | alt.5 |
          +-------+       +-------+
          | fail6 |
          | alt.6 |
          +-------+
```

Figure 10The way of storing alternatives

A.2. Estimation of Memory Requirements.

Now, suppose that there are V nodes in the extended area, the network diameter is D, a neighbor descriptor takes X bytes, a failure ID takes Y bytes and a pointer takes Z bytes. We suppose that lookup for external prefixes are using indirection, so we only need to deal with destinations inside the extended area. In this way, if there is no ECMP, this data structure takes

(2*Z+Y)*(V-1) + 2*(X+Y)*D*(V-1)

bytes altogether. The first part is the memory consumption of the node array. The memory needed by alternative arrays: any path can contain at most D nodes and D links, each record needs X+Y bytes; there are records for all the other nodes in the area (V-1 nodes). Observe that this is a very rough overestimation, since most of the possible failures influencing the path will not change the next hop.

For computing memory consumption, suppose that neighbor descriptors, failure IDs and pointers take 4 bytes, there are 10000 nodes in the extended area (so both real routers and virtual nodes representing prefix groups are included) and the network diameter is 20 hops. In this case, we get that the node array needs about 120KB, the alternative array needs about 3.2MB, so altogether 3.4MB if there is no ECMP. Observe that the number of external prefixes is not important.

If however, there are paths with equal costs, the size of the alternative array increases. Suppose that there are 10 equal paths between ANY two nodes in the network. This would cause that the alternative list gets 10 times bigger, and now it needs a bit less than 32MB. Observe that the node array still needs only about 160KB, so 32MB is a good overestimation, which is likely acceptable for modern linecards with gigs of DRAM. Moreover, we need to stress here again that this is an extremely rough overestimation, so in reality much less memory will be enough. Furthermore, usually only protecting outer prefixes is needed, so we only need to protect the paths towards the prefix groups, which further decreases both the size of node array and the number of alternative lists.

A.3. Estimation of Failover Time

After a failover was detected either locally or by using FN, the nodes need to change the entries in their FIB. Here we do a rough estimation to show that the previous implementation can do it in at most a few milliseconds.

We are supposing that we have the data structure described in the previous section. When a failure happens we need to decide for each node in the node table whether the shortest path towards that destination was influenced by the failure. We can sort the elements in the alternative list, so now we can use binary search, which needs ceil(log(2D)) memory access (log here has base 2) for worst case. We need one more access to get the node list entry and another to rewrite the FIB.

We suppose DDR3 SDRAM with 64 byte cache line, which means that up to 8 entries of the alternative list can be fetched from the RAM at a time, so the previous formula is modified as we need ceil(log(D/4))+2 transactions. In this way for D=20 and V=10.000 we need (3+2)*10.000=50.000 transactions. If we suppose 10 ECMP paths as previously, D=200 and we need (5+2)*10000=70.000 transactions.

We can do a very conservative estimation by supposing a recent DDR3 SDRAM module which can do 5MT/s with completely random access, so doing 50.000 or 70.000 transaction takes 10ms or 14ms. Keep in mind that we assumed that there is only one memory controller, we always got the result of the search with the last read, and all the alternative lists were full. Moreover, internal system latencies (e.g. multiple memory requests) were overestimated seriously, since a DDR3 SDRAM can reach even 6 times this speed with random access.

Appendix B. Impact Scope of Fast Notification

The memory and fail-over time calculations presented in Appendix A are based on worst-case estimation. They assume that basically in a network with diameter equal to 20 hops, each failure has a route changing consequence on all routers in the full diameter.

This section provides experimental results on real-world topologies, showing that already 100% failure coverage can be achieved within a 2-hop radius around the failure.

We performed the coverage analysis of the fast reroute mechanism presented here on realistic topologies, which were generated by the BRITE topology generator in bottom-up mode [BRITE]. The coverage percentage is defined here as the percentage of the number of useable backup paths for protecting the primary paths which are failed because of link failures to the number of all failed primary paths.

The realistic topologies include AT&T and DFN using pre-determined BRITE parameter values from [BRITE] and various random topologies with different number of nodes and varying network connectivity. For example, the number of nodes for AT&T and DFN are 154 and 30, respectively, while the number of nodes for other random topologies is varied from 20 to 100. The BRITE parameters which are used in our topology generation process are summarized in Figure 11 (see [BRITE] for the details of each parameter). In summary, m represents the average number of edges per node and is set to either 2 or 3. A uniform bandwidth distribution in the range 100-1024 Mbps is selected and the link cost is obtained deterministically from the link bandwidth (i.e., inversely proportional to the link bandwidth as used by many vendors). Since the values for p(add) and beta determine the number of edges in the generated topologies, their values are varied to obtain network topologies with varying connectivity (e.g., sparse and dense).

	Bottom up
Grouping Model	Random pick
Model	GLP
Node Placement	Random
Growth Type	Incremental
Preferential Connectivity	On
BW Distribution	Uniform
Minimum BW	100
Maximum BW	1024
m	2-3
Number of Nodes (N)	20,30,50,100,154
p(add)	0.01,0.05,0.10,0.42
beta	0.01,0.05,0.15,0.62

Figure 11 BRITE topology generator parameters

The coverage percentage of our fast reroute method is reported for different network topologies (e.g., different number of nodes and varying network connectivity) using neighborhood depths of 0, 1, and 2. (i.e., X=0, 1, and 2). For a particular failure, backup routes protecting the failed primary paths are calculated only by those nodes which are within the selected radious of this failure. Note that these nodes are determined by the parameter X as follows: For X=0, two nodes which are directly connected to the failed link, for X=1, two nodes which are directly connected to the failed link and also neighboring nodes which are adjacent to one of the outgoing links of these two nodes, and so on.

The coverage percentage for a certain topology is computed by the following formula: Coverage Percentage = N_backupsexist*100/N_fpp where N_backupsexist is the number of source-destination pairs whose primary paths are failed because of link failures and have backup paths for protecting these failed paths, and N_fpp is the number of source-destination pairs whose primary paths are failed because of link failures. The source-destination pairs, in which source and destination nodes do not have any physical connectivity after a failure, are excluded from N_fpp. Note that the coverage percentage includes a network-wide result which is calculated by averaging all coverage results obtained by individually failing all edges for a certain network topology.

Figure 12 shows the coverage percentage results for random topologies with different number of nodes (N) and network connectivity, and Figure 13 shows these results for AT&T and DFN topologies. In these figures, E_mean represents the average number of edges per node for a certain topology. Note that the average number of edges per node is determined by the parameters m, p(add), and beta. We observed that E_mean increases when p(add) and beta values increase. For each topology, coverage analysis is repeated for 10 topologies generated randomly by using the same BRITE parameters. E_mean and coverage percentage are obtained by averaging the results of these ten experiments.

Case	N	E_mean	X=0	X=1	X=2
p(add)=0.01	20	3.64	82.39	98.85	100.0
beta=0.01	50	3.86	82.10	98.69	100.0
	100	3.98	83.21	98.04	100.0
p(add)=0.05	20	3.70	85.60	99.14	100.0
beta=0.05	50	4.01	84.17	99.09	100.0
	100	4.08	83.35	98.01	100.0
p(add)=0.1	20	5.52	93.24	100.0	100.0
beta=0.15	50	6.21	91.46	99.87	100.0
	100	6.39	91.17	99.86	100.0

Figure 12 Coverage percentage results for random topologies

Case	N	E_mean	X=0	X=1	X=2
p(add)=0.42	154 (AT&T)	6.88	91.04	99.81	100.0
beta=0.62	30 (DFN)	8.32	93.76	100.0	100.0

Figure 13 Coverage percentage results for AT&T and DFN topologies

There are two main observations from these results:

1. As the neighborhood depth (X) increases the coverage percentage increases and the complete coverage is obtained using a low neighborhood depth value (i.e., X=2). This result is significant since failure notification message needs to be sent only to nodes which are two-hop away from the point of failure for the complete

coverage. This result supports that our method provides fast convergence by introducing minimal signaling overhead within only the two-hop neighborhood.

2. The topologies with higher connectivity (i.e., higher E_mean values) have better coverage compared to the topologies with lower connectivity (i.e., lower E_mean values). This is an intuitive result since the number of possible alternate hops in dense network topologies is higher than the number of possible alternate hops in sparse topologies. This phenomenon increases the likelihood of finding backup paths, and therefore the coverage percentage.

Authors' Addresses

Andras Csaszar
Ericsson
Irinyi J utca 4-10, Budapest, Hungary, 1117
Email: Andras.Csaszar@ericsson.com

Gabor Sandor Enyedi
Ericsson
Irinyi J utca 4-10, Budapest, Hungary, 1117
Email: Gabor.Sandor.Enyedi@ericsson.com

Jeff Tantsura
Ericsson
300 Holger Way, San Jose, CA 95134
Email: jeff.tantsura@ericsson.com

Sriganesh Kini
Ericsson
300 Holger Way, San Jose, CA 95134
Email: sriganesh.kini@ericsson.com

John Sucec
Telcordia Technologies
One Telcordia Drive, Piscataway, NJ 08854
Email: sucecj@telcordia.com

Subir Das
Telcordia Technologies
One Telcordia Drive, Piscataway, NJ 08854
Email: sdas2@telcordia.com

Routing Area Working Group A. Atlas
Internet-Draft Juniper Networks
Intended status: Informational G. Enyedi
Expires: April 26, 2012 A. Csaszar
Ericsson
October 24, 2011

Algorithms for computing Maximally Redundant Trees for IP/LDP Fast-Reroute
draft-enyedi-rtgwg-mrt-frr-algorithm-00

Abstract

A complete solution for IP and LDP Fast-Reroute using Maximally Redundant Trees is presented in [I-D.atlas-rtgwg-mrt-frr-architecture]. This document describes an algorithm that can be used to compute the necessary Maximally Redundant Trees and the associated next-hops.

Status of this Memo

Copyright Notice

1. Introduction

MRT Fast-Reroute requires that packets can be forwarded not only on the shortest-path tree, but also on two Maximally Redundant Trees (MRTs), referred to as the Blue MRT and the Red MRT. A router which experiences a local failure must also have pre-determined which alternate to use. This document describes how to compute these three things and the algorithm design decisions and rationale. The algorithms are based on those presented in [MRTLinear] and expanded in [EnyediThesis].

Just as packets routed on a hop-by-hop basis require that each router compute a shortest-path tree which is consistent, it is necessary for each router to compute the Blue MRT and Red MRT in a consistent fashion. This is the motivation for the detail in this document.

As now, a router's FIB will contain primary next-hops for the current shortest-path tree for forwarding traffic. In addition, a router's FIB will contain primary next-hops for the Blue MRT for forwarding received traffic on the Blue MRT and primary next-hops for the Red MRT for forwarding received traffic on the Red MRT.

What alternate next-hops a point-of-local-repair (PLR) selects need not be consistent - but loops must be prevented. To reduce congestion, it is possible for multiple alternate next-hops to be selected; in the context of MRT alternates, each of those alternate next-hops would be equal-cost paths.

This document provides an algorithm for selecting an appropriate MRT alternate for consideration. Other alternates, e.g. LFAs that are downstream paths, may be prefered when available and that decision-making is not captured in this document.

```
   [E]---[D]---|           [E]<--[D]<--|                [E]-->[D]
    |     |    |            |     ^    |                       |
    |     |    |            V     |    |                       V
   [R]   [F]  [C]          [R]   [F]  [C]               [R]   [F]  [C]
    |     |    |                  ^                      ^     |    |
    |     |    |                  |                      |     V    |
   [A]---[B]---|           [A]-->[B]                    [A]---[B]<--|

         (a)                     (b)                         (c)
   a 2-connected graph     Blue MRT towards R          Red MRT towards R
```

Figure 1

Algorithms for computing MRTs can handle arbitrary network topologies where the whole network graph is not 2-connected, as in Figure 2, as well as the easier case where the network graph is 2-connected (Figure 1). Each MRT is a spanning tree. The pair of MRTs provide two paths from every node X to the root of the MRTs. Those paths share the minimum number of nodes and the minimum number of links. Each such shared node is a cut-vertex. Any shared links are cut-links.

```
                         [E]---[D]---|     |---[J]
                          |     |    |     |    |
                          |     |    |     |    |
                         [R]   [F]  [C]---[G]   |
                          |     |    |     |    |
                          |     |    |     |    |
                         [A]---[B]---|     |---[H]

                        (a)  a graph that isn't 2-connected

          [E]<--[D]<--|     |---[J]        [E]-->[D]             [J]
           |     ^    |     |    ^               |               |
           V     |    |     V    |               V               |
          [R]   [F]  [C]<--[G]   |         [R]   [F]  [C]<--[G]   |
                 ^               |          ^     |    |     ^    |
                 |               |          |     V    |     |    V
          [A]-->[B]             [H]        [A]---[B]<--|     |---[H]

           (b)  Blue MRT towards R         (c)  Red MRT towards R
```

Figure 2

2. Terminology and Definitions

Redundant Trees (RT): A pair of trees where the path from any node X to the root R along the first tree is node-disjoint with the path from the same node X to the root along the second tree. These can be computed in 2-connected graphs.

Maximally Redundant Trees (MRT): A pair of trees where the path from any node X to the root R along the first tree and the path from the same node X to the root along the second tree share the minimum number of nodes and the minimum number of links. Each such shared node is a cut-vertex. Any shared links are cut-links. Any RT is an MRT but many MRTs are not RTs.

network graph: A graph that reflects the network topology where all links connect exactly two nodes and broadcast links have been transformed into the standard pseudo-node representation.

cut-vertex: A vertex whose removal partitions the network.

cut-link: A link whose removal partitions the network. A cut-link by definition must be connected between two cut-vertices. If there are multiple parallel links, then they are referred to as cut-links in this document if removing the set of parallel links would partition the network.

2-connected: A graph that has no cut-vertices. This is a graph that requires two nodes to be removed before the network is partitioned.

spanning tree: A tree containing links that connects all nodes in the network graph.

back-edge: In the context of a spanning tree computed via a depth-first search, a back-edge is a link that connects a descendant of a node x with an ancestor of x.

DAG: Directed Acyclic Graph - a graph where all links are directed and there are no cycles in it.

ADAG: Almost Directed Acyclic Graph - a graph that, if all links incoming to the root were removed, would be a DAG.

2-connected cluster: A maximal set of nodes that are 2-connected. In a network graph with at least one cut-vertex, there will be multiple 2-connected clusters.

block: Either a 2-connected cluster, a cut-edge, or an isolated vertex.

GADAG: Generalized ADAG - a graph that is the combination of the ADAGs of all blocks.

DFS: Depth-First Search

DFS ancestor: A node n is a DFS ancestor of x if n is on the DFS-tree path from the DFS root to x.

DFS descendant: A node n is a DFS descendant of x if x is on the DFS-tree path from the DFS root to n.

ear: A path along not-yet-included-in-the-GADAG nodes that starts at a node that is already-included-in-the-GADAG and that ends at a node that is already-included-in-the-GADAG. The starting and ending nodes may be the same node if it is a cut-vertex.

X >> Y or Y << X: Indicates the relationship between X and Y in a partial order, such as found in a GADAG. X >> Y means that X is higher in the partial order than Y. Y << X means that Y is lower in the partial order than X.

X > Y or Y < X: Indicates the relationship between X and Y in the total order, such as found via a topological sort. X > Y means that X is higher in the total order than Y. Y < X means that Y is lower in the total order than X.

3. Algorithm Key Concepts

There are five key concepts that are critical for understanding the algorithms for computing MRTs. The first is the idea of partially ordering the nodes in a network graph with regard to each other and to the GADAG root. The second is the idea of finding an ear of nodes and adding them in the correct direction. The third is the idea of a Low-Point value and how it can be used to identify cut-vertices and to find a second path towards the root. The fourth is the idea that a non-2-connected graph is made up of blocks, where a block is a 2-connected cluster, a cut-edge or an isolated node. The fifth is the idea of a local-root for each node; this is used to compute ADAGs in each block.

3.1. Partial Ordering for Disjoint Paths

Given any two nodes X and Y in a graph, a particular total order means that either X < Y or X > Y in that total order. An example would be a graph where the nodes are ranked based upon their IP loopback addresses. In a partial order, there may be some nodes for which it can't be determined whether X << Y or X >> Y. A partial order can be captured in a directed graph, as shown in Figure 3. In a graphical representation, a link directed from X to Y indicates that X is a neighbor of Y in the network graph and X << Y.

```
      [A]<---[R]    [E]       R << A << B << C << D << E
       |             ^        R << A << B << F << G << H << D << E
       |             |
       V             |        Unspecified Relationships:
      [B]--->[C]--->[D]             C and F
       |             ^              C and G
       |             |              C and H
       V             |
      [F]--->[G]--->[H]
```

Figure 3: Directed Graph showing a Partial Order

To compute MRTs, it is very useful to have the root of the MRTs be at the very bottom and the very top of the partial ordering. This means that from any node X, one can pick nodes higher in the order until the root is reached. Similarly, from any node X, one can pick nodes lower in the order until the root is reached. For instance, in Figure 4, from G the higher nodes picked can be traced by following the directed links and are H, D, E and R. Similarly, from G the lower nodes picked can be traced by reversing the directed links and are F, B, A, and R. A graph that represents this modified partial order is no longer a DAG; it is termed an Almost DAG (ADAG) because if the links directed to the root were removed, it would be a DAG.

```
   [A]<---[R]<---[E]      R << A << B << C << R
    |      ^      ^       R << A << B << C << D << E << R
    |      |      |       R << A << B << F << G << H << D << E << R
    V      |      |
   [B]--->[C]--->[D]      Unspecified Relationships:
    |             ^              C and F
    |             |              C and G
    V             |              C and H
   [F]--->[G]--->[H]
```

Figure 4: ADAG showing a Partial Order with R lowest and highest

Most importantly, if a node Y >> X, then Y can only appear on the increasing path from X to the root and never on the decreasing path. Similarly, if a node Z << X, then Z can only appear on the decreasing path from X to the root and never on the inceasing path.

Additionally, when following the increasing paths, it is possible to pick multiple higher nodes and still have the certainty that those paths will be disjoint from the decreasing paths. E.g. in the previous example node B has multiple possibilities to forward packets along an increasing path: it can either forward packets to C or F.

3.2. Finding an Ear and the Correct Direction

For simplicity, the basic idea of creating a GADAG by adding ears is described assuming that the network graph is a single 2-connected cluster so that an ADAG is sufficient. Generalizing to multiple blocks is done by considering the block-roots instead of the GADAG root - and the actual algorithms given in Section 4.3 and Section 4.4.

In order to understand the basic idea of finding an ADAG, first suppose that we have already a partial ADAG, which doesn't contain all the nodes in the block yet, and we want to extend it to cover all the nodes. Suppose that we find a path from a node X to Y such that X and Y are already contained by our partial ADAG, but all the remaining nodes along the path are not added to the ADAG yet. We refer to such a path as an ear.

Recall that our ADAG is closely related to a partial order, more precisely, if we remove root R, the remaining DAG describes a partial order of the nodes. If we suppose that neither X nor Y is the root, we may be able to compare them. If one of them is definitely lesser with respect to our partial order (say X<<Y), we can add the new path to the ADAG in a direction from X to Y. As an example consider Figure 5

```
            E---D---|              E<--D---|           E<--D<--|
            |   |   |              |   ^   |           |   ^   |
            |   |   |              V   |   |           V   |   |
            R   F   C              R   F   C           R   F   C
            |   |   |              |   ^   |           |   ^   ^
            |   |   |              V   |   |           V   |   |
            A---B---|              A-->B---|           A-->B---|

               (a)                    (b)                 (c)
```

(a) A 2-connected graph
(b) Partial ADAG (C is not included)
(c) Resulting ADAG after adding path (or ear) B-C-D

Figure 5

In this partial ADAG, node C is not yet included. However, we can find path B-C-D, where both endpoints are contained by this partial ADAG (we say those nodes are *ready* in the sequel), and the remaining node (node C) is not contained yet. If we remove R, the remaining DAG defines a partial order, and with respect to this partial order we can say that B<<D, so we can add the path to the ADAG in the direction from B to D (arcs B->C and C->D are added). If B were strictly greater than D, we would add the same path in reverse direction.

If in the partial order where an ear's two ends are X and Y, X << Y, then there must already be a directed path from X to Y already in the ADAG. The ear must be added in a direction such that it doesn't create a cycle; therefore the ear must go from X to Y.

In the case, when X and Y are not ordered with each other, we can select either direction for the ear. We have no restriction since neither of the directions can result in a cycle. In the corner case when one of the endpoints of an ear, say X, is the root (recall that the two endpoints must be different), we could use both directions again for the ear because the root can be considered both as smaller and as greater than Y. However, we strictly pick that direction in which the root is greater than Y. The logic for this decision is explained in Section 4.6

A partial ADAG is started by finding a cycle from the root R back to itself. This can be done by selecting a non-ready neighbor N of R and then finding a path from N to R that doesn't use any links between R and N. The direction of the cycle can be assigned either way since it is starting the ordering.

Once a partial ADAG is already present, we can always add ears to it:

just select a non-ready neighbor N of a ready node Q, such that Q is not the root, find a path from N to the root in the graph with Q removed. This path is an ear where the first node of the ear is Q, the next is N, then the path until the first ready node the path reached (that second ready node is the other endpoint of the path). Since the graph is 2-connected, there must be a path from N to R without Q.

It is always possible to select a non-ready neighbor N of a ready node Q so that Q is not the root R. Because the network is 2-connected, N must be connected to two different nodes and only one can be R. Because the initial cycle has already been added to the ADAG, there are ready nodes that are not R. Since the graph is 2-connected, while there are non-ready nodes, there must be a non-ready neighbor N of a ready node that is not R.

```
 Generic_Find_Ears_ADAG(root)
    Create an empty ADAG.  Add root to the ADAG.
    Mark root as IN_GADAG.
    Select the shortest cycle containing root.
    Add the shortest cycle to the ADAG.
    Mark cycle's nodes as IN_GADAG.
    Add cycle's non-root nodes to process_list.
    while there exists connected nodes in graph that are not IN_GADAG
       Select a new ear.  Let its endpoints be X and Y.
       if (X is root) or (Y<<X)
          add the ear towards X to the ADAG
       else // (a) Y is root or (b) Y>>X or (c)X, Y not ordered
          Add the ear towards Y to the ADAG
```

Figure 6: Generic Algorithm to find ears and their direction in 2-connected graph

Algorithm Figure 6 merely requires that a cycle or ear be selected without specifying how. Regardless of the way of selecting the path, we will get an ADAG. The method used for finding and selecting the ears is important; shorter ears result in shorter paths along the MRTs. There are two options being considered. The Low-Point Inheritance option is described in Section 4.3. The SPF-based option is described in Section 4.4.

As an example, consider Figure 5 again. First, we select the shortest cycle containing R, which can be R-A-B-F-D-E (uniform link costs were assumed), so we get to the situation depicted in Figure 5 (b). Finally, we find a node next to a ready node; that must be node C and assume we reached it from ready node B. We search a path from C to R without B in the original graph. The first ready node along this is node D, so the open ear is B-C-D. Since B<<D, we add arc

B->C and C->D to the ADAG. Since all the nodes are ready, we stop at this point.

3.3. Low-Point Values and Their Uses

A basic way of computing a spanning tree on a network graph is to run a depth-first-search, such as given in Figure 7. This tree has the important property that if there is a link (x, n), then either n is a DFS ancestor of x or n is a DFS descendant of x. In other words, either n is on the path from the root to x or x is on the path from the root to n.

```
global_variable: dfs_number

DFS_Visit(node x, node parent)
   D(x) = dfs_number
   dfs_number += 1
   x.dfs_parent = parent
   for each link (x, w)
     if D(w) is not set
       DFS_Visit(w, x)

Run_DFS(node root)
   dfs_number = 0
   DFS_Visit(root, NONE)
```

Figure 7: Basic Depth-First Search algorithm

Given a node x, one can compute the minimal DFS number of the neighbours of x, i.e. min(D(w) if (x,w) is a link). This gives the highest attachment point neighbouring x. What is interesting, though, is what is the highest attachment point from x and x's descendants. This is what is determined by computing the Low-Point value, as given in Algorithm Figure 9 and illustrated on a graph in Figure 8.

```
            [E]---|    [J]-------[I]   [P]---[O]
             |    |     |         |     |     |
             |    |     |         |     |     |
            [R]  [D]---[C]--[F]  [H]---[K]   [N]
             |          |    |    |     |     |
             |          |    |    |     |     |
            [A]--------[B]  [G]---|    [L]---[M]

               (a) a non-2-connected graph

             [E]----|    [J]---------[I]    [P]------[O]
            (5, )   |  (10, )       (9, ) (16,  ) (15,  )
              |     |     |           |      |        |
              |     |     |           |      |        |
             [R]   [D]---[C]---[F]   [H]----[K]      [N]
            (0, ) (4, ) (3, ) (6, ) (8, ) (11, )  (14, )
              |           |     |     |      |        |
              |           |     |     |      |        |
             [A]---------[B]   [G]----|     [L]------[M]
            (1, )       (2, ) (7, )       (12,  )  (13,  )

               (b) with DFS values assigned   (D(x), L(x))

             [E]----|    [J]---------[I]    [P]------[O]
            (5,0)   |  (10,3)       (9,3) (16,11) (15,11)
              |     |     |           |      |        |
              |     |     |           |      |        |
             [R]   [D]---[C]---[F]   [H]----[K]      [N]
            (0, ) (4,0) (3,0) (6,3) (8,3) (11,11) (14,11)
              |           |     |     |      |        |
              |           |     |     |      |        |
             [A]---------[B]   [G]----|     [L]------[M]
            (1,0)       (2,0) (7,3)       (12,11)  (13,11)

                (c) with low-point values assigned (D(x), L(x))
```

Figure 8

```
global_variable: dfs_number

Lowpoint_Visit(node x, node parent, interface p_to_x)
   D(x) = dfs_number
   L(x) = D(x)
   dfs_number += 1
   x.dfs_parent = parent
   x.dfs_parent_intf = p_to_x
   x.lowpoint_parent = NONE
   for each interface intf of x:
     if D(intf.remote_node) is not set
       Lowpoint_Visit(intf.remote_node, x, intf)
       if L(intf.remote_node) < L(x)
          L(x) = L(intf.remote_node)
          x.lowpoint_parent = intf.remote_node
          x.lowpoint_parent_intf = intf
     else if intf.remote_node is not parent
       if D(intf.remote_node) < L(x)
         L(x) = D(intf.remote)
         x.lowpoint_parent = intf.remote_node
         x.lowpoint_parent_intf = intf

Run_Lowpoint(node root)
   dfs_number = 0
   Lowpoint_Visit(root, NONE, NONE)
```

Figure 9: Computing Low-Point value

From the low-point value and lowpoint parent, there are two very useful things which motivate our computation.

First, if there is a child c of x such that L(c) >= D(x), then there are no paths in the network graph that go from c or its descendants to an ancestor of x - and therefore x is a cut-vertex. This is useful because it allows identification of the cut-vertices and thus the blocks. As seen in Figure 8, even if L(x) < D(x), there may be a block that contains both the root and a DFS-child of a node while other DFS-children might be in different blocks. In this example, C's child D is in the same block as R while F is not.

Second, by repeatedly following the path given by lowpoint_parent, there is a path from x back to an ancestor of x that does not use the link [x, x.dfs_parent] in either direction. The full path need not be taken, but this gives a way of finding an initial cycle and then ears.

3.4. Blocks in a Graph

A key idea for the MRT algorithm is that any non-2-connected graph is made up by blocks (e.g. 2-connected clusters, cut-links, and/or isolated nodes). To compute GADAGs and thus MRTs, computation is done in each block to compute ADAGs or Redundant Trees and then those ADAGs or Redundant Trees are combined into a GADAG or MRT.

```
              [E]---|     [J]-------[I]   [P]---[O]
               |    |      |         |     |     |
               |    |      |         |     |     |
              [R]  [D]---[C]--[F]  [H]---[K]   [N]
               |           |    |    |     |     |
               |           |    |    |     |     |
              [A]--------[B]  [G]---|    [L]---[M]

              (a)  A graph with four blocks that are:
                   3 2-connected clusters and a cut-link


              [E]<--|     [J]<------[I]   [P]<--[O]
               |    |      |         ^     |     ^
               V    |      V         |     V     |
              [R]  [D]<--[C]  [F]  [H]<---[K]  [N]
                           ^    |    ^           ^
                           |    V    |           |
              [A]------->[B]  [G]---|     [L]-->[M]

                          (b) Blue MRT


              [E]---|     [J]-------->[I]    [P]-->[O]
                    |                  |            |
                    V                  V            V
              [R]  [D]-->[C]<---[F]  [H]<---[K]   [N]
               ^           |      ^    |      ^     |
               |           V      |    |      |     V
              [A]<-------[B]    [G]<--|     [L]<--[M]

                          (c) Red MRT


                              Figure 10
```

Consider the example depicted in Figure 10 (a). In this figure, a special graph is presented, showing us all the ways 2-connected clusters can be connected. It has four blocks: block 1 contains R, A, B, C, D, E, block 2 contains C, F, G, H, I, J, block 3 contains K, L, M, N, O, P, and block 4 is a cut-edge containing H and K. As can be observed, the first two blocks have one common node (node C) and blocks 2 and 3 do not have any common node, but they are connected through a cut-edge that is block 4. No two blocks can have more than one common node, since two blocks with at least 2 common nodes would qualify as a single 2-connected cluster.

Moreover, observe that if we want to get from one block to another, we must use a cut-vertex (the cut-vertices in this graph are C, H, K), regardless of the path selected, so we can say that all the paths from block 3 along the MRTs rooted at R will cross K first. This observation means that if we want to find a pair of MRTs rooted at R, then we need to build up a pair of RTs in block 3 with K as a root. Similarly, we need to find another one in block 2 with C as a root, and finally, we need the last one in block 1 with R as a root. When all the trees are selected, we can simply combine them; when a block is a cut-edge (as in block 4), that cut-edge is added in the same direction to both of the trees. The resulting trees are depicted in Figure 10 (b) and (c).

Similarly, to create a GADAG it is sufficient to compute ADAGs in each block and connect them.

It is necessary, therefore, to identify the cut-vertices, the blocks and identify the appropriate local-root to use for each block.

3.5. Determining Local-Root

Each node in a network graph has a local-root, which is the cut-vertex (or root) in the same block that is closest to the root. The local-root is used to determine whether two nodes share a common block.

```
Compute_Localroot(node x, node localroot)
    x.localroot = localroot
    for each DFS child c
        if L(c) < D(x)   //x is not a cut-vertex
            Compute_Localroot(c, x.localroot)
        else
            mark x as cut-vertex
            Compute_Localroot(c, x)

Compute_Localroot(root, root)
```

Figure 11: A method for computing local-roots

There are two different ways of computing the local-root for each node. The stand-alone method is given in Figure 11 and better illustrates the concept. It is used in the second option for computing a GADAG using SPFs. The other method is used in the first option for computing a GADAG using Low-Point inheritance and the essence of it is given in Figure 12.

```
            Get the current node, s.
            Compute an ear from s to a child c
               and then via lowpoint inheritance, e.g.
                  ( n = c
                    while n is not ready:
                       n = n.lowpoint_parent
                    e = n
                   )
                  to a ready node e.
            if s is e
               s is a cut-vertex
               x.localroot = s
            else
               for each node x in the ear that is not s or e
                   x.localroot = s.localroot
```

Figure 12: Ear-based method for computing local-roots

Once the local-roots are known, two nodes X and Y are in a common block if and only if one of the following three conditions apply.

- o Y's local-root is X's local-root : They are in the same block and neither is the cut-vertex closest to the root.
- o Y's local-root is X: X is the cut-vertex closest to the root for Y's block
- o Y is X's local-root: Y is the cut-vertex closest to the root for X's block

4. Algorithm Sections

This algorithm computes one GADAG that is then used by a router to determine its blue MRT and red MRT next-hops to all destinations. Finally, based upon that information, alternates are selected for each next-hop to each destination. The different parts of this algorithm are described below. These work on a network graph after, for instance, its interfaces are ordered as per Figure 13.

1. Select the root to use for the GADAG. [See Section 4.1.]

2. Initialize all interfaces to UNDIRECTED. [See Section 4.2.]

3. Compute the DFS value,e.g. D(x), and lowpoint value, L(x). [See Figure 9.]

4. Construct the GADAG. [See Section 4.3 for Option 1 using Lowpoint Inheritance and Section 4.4 for Option 2 using SPFs.]

5. Assign directions to all interfaces that are still UNDIRECTED. [See Section 4.5.]

6. From the computing router x, compute the next-hops for the blue MRT and red MRT. [See Section 4.6.]

7. Identify alternates for each next-hop to each destination by determining which one of the blue MRT and the red MRT the computing router x should select. [See Section 4.7.]

To ensure consistency in computation, it is necessary that all routers order interfaces identically. This is necessary for the DFS, where the selection order of the interfaces to explore results in different trees, and for computing the GADAG, where the selection order of the interfaces to use to form ears can result in different GADAGs. The recommended ordering between two interfaces from the same router x is given in Figure 13.

```
Interface_Compare(interface a, interface b)
  if a.metric < b.metric
     return A_LESS_THAN_B
  if b.metric < a.metric
     return B_LESS_THAN_A
  if a.neighbor.loopback_addr < b.neighbor.loopback_addr
     return A_LESS_THAN_B
  if b.neighbor.loopback_addr < a.neighbor.loopback_addr
     return B_LESS_THAN_A
  // Same metric to same node, so the order doesn’t matter anymore.
  // To have a unique, consistent total order,
  // tie-break based on ifindex.
  if a.ifindex < b.ifindex
     return A_LESS_THAN_B
  return B_LESS_THAN_A
```

Figure 13: Rules for ranking multiple interfaces. Order is from low to high.

4.1. Root Selection

The precise mechanism by which routers advertise a priority for the GADAG root is not described in this document. Nor is the algorithm for selecting routers based upon priority described in this document.

A network may be partitioned or there may be islands of routers that support MRT fast-reroute. Therefore, the root selected for use in a GADAG must be consistent only across each connected island of MRT fast-reroute support. Before beginning computation, the network graph is reduced to contain only the set of routers that support a compatible MRT fast-reroute.

The selection of a GADAG root is done among only those routers in the same MRT fast-reroute island as the computing router x. Additionally, only routers that are not marked as unusable or overloaded (e.g. ISIS overload or [RFC3137]) are eligible for selection as root.

4.2. Initialization

Before running the algorithm, there is the standard type of initialization to be done, such as clearing any computed DFS-values, lowpoint-values, DFS-parents, lowpoint-parents, any MRT-computed next-hops, and flags associated with algorithm.

It is assumed that a regular SPF computation has been run so that the primary next-hops from the computing router to each destination are known. This is required for determining alternates at the last step.

Initially, all interfaces must be initialized to UNDIRECTED. Whether they are OUTGOING, INCOMING or both is determined when the GADAG is constructed and augmented.

It is possible that some links and nodes will be marked as unusable, whether because of configuration, overload, or due to a transient cause such as [RFC3137]. In the algorithm description, it is assumed that such links and nodes will not be explored or used and no more disussion is given of this restriction.

4.3. Option 1: Computing GADAG using lowpoint inheritance

The basic idea of this is to find ears from a node x that is already in the GADAG (known as IN_GADAG). There are two methods to find ears; both are required. The first is by going to a not IN_GADAG DFS-child and then following the chain of low-point parents until an IN_GADAG node is found. The second is by going to a not IN_GADAG neighbor and then following the chain of DFS parents until an

IN_GADAG node is found. As an ear is found, the associated interfaces are marked based on the direction taken. The nodes in the ear are marked as IN_GADAG. In the algorithm, first the ears via DFS-children are found and then the ears via DFS-neighbors are found.

By adding both types of ears when an IN_GADAG node is processed, all ears that connect to that node are found. The order in which the IN_GADAG nodes is processed is, of course, key to the algorithm. The order is a stack of ears so the most recent ear is found at the top of the stack. Of course, the stack stores nodes and not ears, so an ordered list of nodes, from the first node in the ear to the last node in the ear, is created as the ear is explored and then that list is pushed onto the stack.

Each ear represents a partial order (see Figure 4) and processing the nodes in order along each ear ensures that all ears connecting to a node are found before a node higher in the partial order has its ears explored. This means that the direction of the links in the ear is always from the node x being processed towards the other end of the ear. Additionally, by using a stack of ears, this means that any unprocessed nodes in previous ears can only be ordered higher than nodes in the ears below it on the stack.

In this algorithm that depends upon Low-Point inheritance, it is necessary that every node have a low-point parent that is not itself. If a node is a cut-vertex, that will not yet be the case. Therefore, any nodes without a low-point parent will have their low-point parent set to their DFS parent and their low-point value set to the DFS-value of their parent. This assignment also properly allows an ear to a cut-vertex to start and end at the same node.

Finally, the algorithm simultaneously computes each node's local-root, as described in Figure 12. The local-root can be inherited from the node x being processed to the nodes in the ear unless the child of x is a cut-vertex in which case the rest of the nodes in the ear are in a different block than x and have the child of x as their local-root.

```
Construct_GADAG_via_Lowpoint(topology, root)
  root.IN_GADAG = true
  Initialize Stack to empty
  push root onto Stack
  while (Stack is not empty)
     x = pop(Stack)
     foreach interface intf of x
        if ((intf.remote_node.IN_GADAG == false) and
            (intf.remote_node.dfs_parent is x))
            Construct_Ear(x, Stack, intf, CHILD)
     foreach interface intf of x
        if ((intf.remote_node.IN_GADAG == false) and
            (intf.remote_node.dfs_parent is not x))
            Construct_Ear(x, Stack, intf, NEIGHBOR)

Construct_Ear(x, Stack, intf, type)
   ear_list = empty
   cur_node = intf.remote_node
   cur_intf = intf

   while cur_node.IN_GADAG is false
      cur_intf.UNDIRECTED = false
      cur_intf.OUTGOING = true
      cur_intf.remote_intf.UNDIRECTED = false
      cur_intf.remote_intf.INCOMING = true
      cur_node.IN_GADAG = true
      add_to_list_end(ear_list, cur_node)

      if type is CHILD
         cur_intf = cur_node.lowpoint_parent_intf
      else type must be NEIGHBOR
         cur_intf = cur_node.dfs_parent_intf
      cur_node = cur_intf.remote_node

   if (type is CHILD) and (cur_node is x)
      localroot = x
   else
      localroot = x.localroot
   while ear_list is not empty
      y = remove_end_item_from_list(ear_list)
      y.localroot = localroot
      push(Stack, y)
```

Figure 14: Low-point Inheritance GADAG algorithm

4.4. Option 2: Computing GADAG using SPFs

The basic idea in this option is to use slightly-modified SPF computations to find ADAGs in each block. In each block, an SPF computation is first done to find a cycle from the local root and then SPF computations find ears until there are no more interfaces to be explored. The used result from the SPF computation is the path of interfaces indicated by following the previous hops from the mininized IN_GADAG node back to the SPF root.

To do this, first all cut-vertices must be identified and local-roots assigned as specified in Figure 11

The slight modifications to the SPF are as follows. The root of the block is referred to as the block-root; it is either the GADAG root or a cut-vertex.

a. The SPF is rooted at a neighbor x of an IN_GADAG node y. All links between y and x are marked as TEMP_UNUSABLE. They should not be used during the SPF computation.

b. If y is not the block-root, then it is marked TEMP_UNUSABLE. It should not be used during the SPF computation. This prevents ears from starting and ending at the same node and avoids cycles; the exception is because cycles to/from the block-root are acceptable and expected.

c. Do not explore links to nodes whose local-root is not the block-root. This keeps the SPF confined to the particular block.

d. Terminate when the first IN_GADAG node z is minimized.

e. Respect the existing directions (e.g. INCOMING, OUTGOING, UNDIRECTED) already specified for each interface.

```
Mod_SPF(spf_root, block_root)
   Initialize spf_heap to empty
   Initialize nodes' spf_metric to infinity
   spf_root.spf_metric = 0
   insert(spf_heap, spf_root)
   found_in_gadag = false
   while (spf_heap is not empty) and (found_in_gadag is false)
       min_node = remove_lowest(spf_heap)
       if min_node.IN_GADAG is true
          found_in_gadag = true
       else
          foreach interface intf of min_node
             if ((intf.OUTGOING or intf.UNDIRECTED) and
                 (intf.remote_node.localroot is block_root) and
                 (intf.remote_node is not TEMP_UNUSABLE))
                path_metric = min_node.spf_metric + intf.metric
                if path_metric < intf.remote_node.spf_metric
                   intf.remote_node.spf_metric = path_metric
                   intf.remote_node.spf_prev_intf = intf
                   insert_or_update(spf_heap, intf.remote_node)
   return min_node

SPF_for_Ear(spf_root, block_root, ear_list, cut_vertex_list)
   end_ear = Mod_SPF(spf_root, block_root)
   y = end_ear.spf_prev_hop
   while y.local_node is not spf_root
     add_to_list_start(cut_vertex_list, y)
     if y.local_node is a cut-vertex
        add_to_list_end(cut_vertex_list, y.local_node)
     y = y.local_node.spf_prev_intf
```

Figure 15: Modified SPF for GADAG computation

In Figure 15, while the path is determined, any non-end node in the path that is a cut-vertex is added to the list of cut-vertices. This ensures that there is a path from the GADAG root to that cut-vertex before adding it to the list of nodes. All such cut-vertices will be treated as the root of a block and the ADAG in that block will be computed.

Assume that an ear is found by going from y to x and then running an SPF that terminates by minimizing z (e.g. y<->x...q<->z). Now it is necessary to determine the direction of the ear; if y << z, then the path should be y->x...q->z but if y >> z, then the path should be y<-x...q<-z. In Section 4.3, the same problem was handled by finding all ears that started at a node before looking at ears starting at nodes higher in the partial order. In this algorithm, using that approach could mean that new ears aren't added in order of their total cost since all ears connected to a node would need to be found before additional nodes could be found.

The alternative is to track the order relationship of each node with respect to every other node. This can be accomplished by maintaining two sets of nodes at each node. The first set, Higher_Nodes, contains all nodes that are known to be ordered above the node. The second set, Lower_Nodes, contains all nodes that are known to be ordered below the node. This is the approach used in this algorithm.

```
Set_Ear_Direction(ear_list, end_a, end_b, block_root)
  // Default of A_TO_B for the following cases:
  //  (a) end_a and end_b are the same (root)
  // or (b) end_a is in end_b's Lower Nodes
  // or (c) end_a and end_b were unordered with respect to each
  //        other
  direction = A_TO_B
  if (end_a is block_root) and (end_a is not end_b)
     direction = B_TO_A
  else if end_a is in end_b.Higher_Nodes
     direction = B_TO_A
  if direction is B_TO_A
     foreach interface i in ear_list
         i.UNDIRECTED = false
         i.INCOMING = true
         i.remote_intf.UNDIRECTED = false
         i.remote_intf.OUTGOING = true
  else
     foreach interface i in ear_list
         i.UNDIRECTED = false
         i.OUTGOING = true
         i.remote_intf.UNDIRECTED = false
         i.remote_intf.INCOMING = true
   if end_a is end_b
      return
   // Next, update all nodes' Lower_Nodes and Higher_Nodes
   if (end_a is in end_b.Higher_Nodes)
      foreach node x where x.localroot is block_root
          if end_a is in x.Lower_Nodes
             foreach interface i in ear_list
                add i.remote_node to x.Lower_Nodes
          if end_b is in x.Higher_Nodes
             foreach interface i in ear_list
                add i.local_node to x.Higher_Nodes
    else
      foreach node x where x.localroot is block_root
          if end_b is in x.Lower_Nodes
             foreach interface i in ear_list
                add i.local_node to x.Lower_Nodes
          if end_a is in x.Higher_Nodes
             foreach interface i in ear_list
                add i.remote_node to x.Higher_Nodes
```

Figure 16: Algorithm to assign links of an ear direction

A goal of the algorithm is to find the shortest cycles and ears. An ear is started by going to a neighbor x of an IN_GADAG node y. The path from x to an IN_GADAG node is minimal, since it is computed via

SPF. Since a shortest path is made of shortest paths, to find the shortest ears requires reaching from the set of IN_GADAG nodes to the closest node that isn't IN_GADAG. Therefore, an ordered tree is maintained of interfaces that could be explored from the IN_GADAG nodes. The interfaces are ordered by their characteristics of metric, local loopback address, remote loopback address, and ifindex, as in the algorithm previously described in Figure 13.

Finally, cut-edges are a special case because there is no point in doing an SPF on a block of 2 nodes. The algorithm identifies cut-edges simply as links where both ends of the link are cut-vertices. Cut-edges can simply be added to the GADAG with both OUTGOING and INCOMING specified on their interfaces.

```
Construct_GADAG_via_SPF(topology, root)
   Compute_Localroot(root, root)
   if root has multiple DFS-children
      mark root as a cut-vertex
   Initialize cut_vertex_list to empty
   Initialize ordered_intfs_tree to empty
   add_to_list_end(cut_vertex_list, root)
   while cut_vertex_list is not empty
      v = remove_start_item_from_list(cut_vertex_list)
      foreach interface intf of v
         if intf.remote_node is a cut-vertex
            // Special case for cut-edges
            intf.UNDIRECTED = false
            intf.remote_intf.UNDIRECTED = false
            intf.OUTGOING = true
            intf.INCOMING = true
            intf.remote_intf.OUTGOING = true
            intf.remote_intf.INCOMING = true
         else if intf.remote_node.localroot is v
            insert(ordered_intfs_tree, intf)
      v.IN_GADAG = true
      while ordered_intfs_trees is not empty
        cand_intf = remove_lowest(ordered_intfs_tree)
        if cand_intf.remote_node.IN_GADAG is false
           Mark all interfaces between cand_intf.remote_node
              and cand_intf.local_node as TEMP_UNUSABLE
           if cand_intf.local_node is not v
              Mark cand_intf.local_node as TEMP_UNUSABLE
           Initialize ear_list to empty
           ear_end = SPF_for_Ear(cand_intf.remote_node, v, ear_list,
                     cut_vertex_list)
           add_to_list_start(ear_list, cand_intf)
           Set_Ear_Direction(ear_list, cand_intf.remote, ear_end, v)
           Clear TEMP_UNUSABLE from all interfaces between
              cand_intf.remote_node and cand_intf.local_node
           Clear TEMP_UNUSABLE from cand_intf.local_node
```

Figure 17: SPF-based GADAG algorithm

4.5. Augmenting the GADAG by directing all links

The GADAG, whether constructed via Low-Point Inheritance or with SPFs, at this point could be used to find MRTs but the topology does not include all links in the network graph. That has two impacts. First, there might be shorter paths that respect the GADAG partial ordering and so the alternate paths would not be as short as possible. Second, there may be additional paths between a router x and the root that are not included in the GADAG. Including those provides potentially more bandwidth to traffic flowing on the alternates and may reduce congestion compared to just using the GADAG as currently constructed.

The goal is thus to assign direction to every remaining link marked as UNDIRECTED to improve the paths and number of paths found when the MRTs are computed.

To do this, we need to establish a total order that respects the partial order described by the GADAG. This can be done using Kahn's topological sort[Kahn_1962_topo_sort] which essentially assigns a number to a node x only after all nodes before it (e.g. with a link incoming to x) have had their numbers assigned. The only issue with the topological sort is that it works on DAGs and not ADAGs or GADAGs.

To convert a GADAG to a DAG, it is necessary to remove all links that point to a root of block from within that block. That provides the necessary conversion to a DAG and then a topological sort can be done. Finally, all UNDIRECTED links are assigned a direction based upon the partial ordering. Any UNDIRECTED links that connect to a root of a block from within that block are assigned a direction INCOMING to that root. The exact details of this whole process are captured in Figure 18

```
Set_Block_Root_Incoming_Links(topo, root, mark_or_clear)
   foreach node x in topo
      if node x is a cut-vertex or root
         foreach interface i of x
            if (i.remote_node.localroot is x)
               if i.UNDIRECTED
                  i.INCOMING = true
                  i.remote_intf.OUTGOING = true
                  i.UNDIRECTED = false
                  i.remote_intf.UNDIRECTED = false
               if i.INCOMING
                  if mark_or_clear is mark
                     i.TEMP_UNUSABLE = true
                     i.remote_intf.TEMP_UNUSABLE = true

Run_Topological_Sort(topo, root)
   foreach node x
     set x.unvisited to the count of x's incoming interfaces
        that aren't marked TEMP_UNUSABLE
   Initialize working_list to empty
   Initialize topo_order_list to empty
```

```
          add_to_list_end(working_list, root)
          while working_list is not empty
             y = remove_start_item_from_list(working_list)
             add_to_list_end(topo_order_list, y)
             foreach interface i of y
                 if (i.OUTGOING) and (not i.TEMP_UNUSABLE)
                    i.remote_node.unvisited -= 1
                    if i.remote_node.unvisited is 0
                        add_to_list_end(working_list, i.remote_node)
           next_topo_order = 1
           while topo_order_list is not empty
               y = remove_start_item_from_list(topo_order_list)
               y.topo_order = next_topo_order
               next_topo_order += 1

       Add_Undirected_Links(topo, root)
           Set_Block_Root_Incoming_Links(topo, root, MARK)
           Run_Topological_Sort(topo, root)
           Set_Block_Root_Incoming_Links(topo, root, CLEAR)
           foreach node x in topo
             foreach interface i of x
                if i.UNDIRECTED
                  if x.topo_order < i.remote_node.topo_order
                     i.OUTGOING = true
                     i.UNDIRECTED = false
                     i.remote_intf.INCOMING = true
                     i.remote_intf.UNDIRECTED = false
                  else
                     i.INCOMING = true
                     i.UNDIRECTED = false
                     i.remote_intf.OUTGOING = true
                     i.remote_intf.UNDIRECTED = false

       Add_Undirected_Links(topo, root)
```

Figure 18: Assigning direction to UNDIRECTED links

4.6. Compute MRT next-hops

As was discussed in Section 3.1, once a ADAG is found, it is straightforward to find the next-hops from any node X to the ADAG root. However, in this algorithm, we want to reuse the common GADAG and find not only one pair of redundant trees with it, but a pair rooted at each node. This is ideal, since it is faster and it results packet forwarding easier to trace and/or debug. The method for doing that is based on two basic ideas. First, if two nodes X and Y are ordered with respect to each other in the partial order, then the same SPF and reverse-SPF can be used to find the increasing and decreasing paths. Second, if two nodes X and Y aren't ordered with respect to each other in the partial order, then intermediary nodes can be used to create the paths by increasing/decreasing to the intermediary and then decreasing/increasing to reach Y.

As usual, the two basic ideas will be discussed assuming the network is two-connected. The generalization to multiple blocks is discussed in Section 4.6.4. The full algorithm is given in Section 4.6.5.

4.6.1. MRT next-hops to all nodes partially ordered with respect to the computing node

To find two node-disjoint paths from the computing router X to any node Y, depends upon whether Y >> X or Y << X. As shown in Figure 19, if Y >> X, then there is an increasing path that goes from X to Y without crossing R; this contains nodes in the interval [X,Y]. There is also a decreasing path that decreases towards R and then decreases from R to Y; this contains nodes in the interval [X,R-small] or [R-great,Y]. The two paths cannot have common nodes other than X and Y.

```
      [Y]<---(Cloud 2)<--- [X]
       |                    ^
       |                    |
       V                    |
    (Cloud 3)--->[R]--->(Cloud 1)

  Blue MRT path: X->Cloud 2->Y
  Red MRT path: X->Cloud 1->R->Cloud 3->Y
```

Figure 19: Y >> X

Similar logic applies if Y << X, as shown in Figure 20. In this case, the increasing path from X increases to R and then increases from R to Y to use nodes in the intervals [X,R-great] and [R-small, Y]. The decreasing path from X reaches Y without crossing R and uses nodes in the interval [Y,X].

```
      [X]<---(Cloud 2)<--- [Y]
       |                    ^
       |                    |
       V                    |
    (Cloud 3)--->[R]--->(Cloud 1)

  Blue MRT path: X->Cloud 3->R->Cloud 1->Y
  Red MRT path: X->Cloud 2->Y
```

Figure 20: Y << X

4.6.2. MRT next-hops to all nodes not partially ordered with respect to the computing node

When X and Y are not ordered, the first path should increase until we get to a node G, where G >> Y. At G, we need to decrease to Y. The other path should be just the opposite: we must decrease until we get to a node H, where H << Y, and then increase. Since R is smaller and greater than Y, such G and H must exist. It is also easy to see that these two paths must be node disjoint: the first path contains nodes in interval [X,G] and [Y,G], while the second path contains nodes in interval [H,X] and [H,Y]. This is illustrated in Figure 21. It is necessary to decrease and then increase for the Blue MRT and increase and then decrease for the Red MRT; if one simply increased for one and decreased for the other, then both paths would go through the root R.

```
 (Cloud 6)<---[Y]<---(Cloud 5)<------------|
   |                                       |
   |                                       |
   V                                       |
  [G]--->(Cloud 4)--->[R]--->(Cloud 1)--->[H]
   ^                                       |
   |                                       |
   |                                       |
  (Cloud 3)<---[X]<---(Cloud 2)<-----------|

Blue MRT path: decrease to H and increase to Y
     X->Cloud 2->H->Cloud 5->Y
Red MRT path:  increase to G and decrease to Y
     X->Cloud 3->G->Cloud 6->Y
```

Figure 21: X and Y unordered

This gives disjoint paths as long as G and H are not the same node. Since G >> Y and H << Y, if G and H could be the same node, that would have to be the root R. This is not possible because there is only one out-going interface from the root R which is created when the initial cycle is found. Recall from Figure 6 that whenever an ear was found to have an end that was the root R, the ear was directed towards R so that the associated interface on R is incoming and not outgoing. Therefore, there must be exactly one node M which is the smallest one after R, so the Blue MRT path will never reach R; it will turn at M and increase to Y.

4.6.3. Computing Redundant Tree next-hops in a 2-connected Graph

The basic ideas for computing RT next-hops in a 2-connected graph were given in Section 4.6.1 and Section 4.6.2. Given these two ideas, how can we find the trees?

If some node X only wants to find the next-hops (which is usually the case for IP networks), it is enough to find which nodes are greater and less than X, and which are not ordered; this can be done by running an SPF and a reverse-SPF rooted at X and not exploring any links from the ADAG root. (Other traversal algorithms could safely be used instead where one traversal takes the links in their given directions and the other reverses the links' directions.)

An SPF rooted at X and not exploring links from the root will find the increasing next-hops to all Y >> X. Those increasing next-hops are X's next-hops on the Blue MRT to reach Y. A reverse-SPF rooted at X and not exploring links from the root will find the decreasing next-hops to all Z << X. Those decreasing next-hops are X's next-hops on the Red MRT to reach Z. Since the root R is both greater than and less than X, after this SPF and reverse-SPF, X's next-hops on the Blue MRT and on the Red MRT to reach R are known. For every node Y >> X, X's next-hops on the Red MRT to reach Y are set to those on the Red MRT to reach R. For every node Z << X, X's next-hops on the Blue MRT to reach Z are set to those on the Blue MRT to reach R.

For those nodes, which were not reached, we have the next-hops as well. The increasing Blue MRT next-hop for a node, which is not ordered, is the next-hop along the decreasing Red MRT towards R and the decreasing Red MRT next-hop is the next-hop along the increasing Blue MRT towards R. Naturally, since R is ordered with respect to all the nodes, there will always be an increasing and a decreasing path towards it. This algorithm does not provide the specific path taken but only the appropriate next-hops to use. The identity of G and H is not determined.

The final case to considered is when the root R computes its own next-hops. Since the root R is << all other nodes, running an SPF rooted at R will reach all other nodes; the Blue MRT next-hops are those found with this SPF. Similarly, since the root R is >> all other nodes, running a reverse-SPF rooted at R will reach all other nodes; the Red MRT next-hops are those found with this reverse-SPF.

```
              E---D---|              E<--D<--|
              |   |   |              |   ^   |
              |   |   |              V   |   |
              R   F   C              R   F   C
              |   |   |              |   ^   ^
              |   |   |              V   |   |
              A---B---|              A-->B---|

                 (a)                    (b)
          A 2-connected graph    A spanning ADAG rooted at R
```

Figure 22

As an example consider the situation depicted in Figure 22. There node C runs an SPF and a reverse-SPF The SPF reaches D, E and R and the reverse SPF reaches B, A and R. So we immediately get that e.g. towards E the increasing next-hop is D (it was reached though D), and the decreasing next-hop is B (since R was reached though B). Since both D and B, A and R will compute the next hops similarly, the packets will reach E.

We have the next-hops towards F as well: since F is not ordered with respect to C, the increasing next-hop is the decreasing one towards R (which is B) and the decreasing next-hop is the increasing one towards R (which is D). Since B is ordered with F, it will find a real increasing next-hop, so packet forwarded to B will get to F on path C-B-F. Similarly, D will have a real decreasing next-hop, and packet will use path C-D-F.

4.6.4. Generalizing for graph that isn't 2-connected

If a graph isn't 2-connected, then the basic approach given in Section 4.6.3 needs some extensions to determine the appropriate MRT next-hops to use for destinations outside the computing router X's blocks. In order to find a pair of maximally redundant trees in that graph we need to find a pair of RTs in each of the blocks (the root of these trees will be discussed later), and combine them.

When computing the MRT next-hops from a router X, there are three basic differences:

1. Only nodes in a common block with X should be explored in the SPF and reverse-SPF.

2. Instead of using the GADAG root, X's local-root should be used. This has the following implications:

A. The links from X's local-root should not be explored.

B. If a node is explored in the increasing SPF so Y >> X, then X's Red MRT next-hops to reach Y uses X's Red MRT next-hops to reach X's local-root and if Z <<, then X's Blue MRT next-hops to reach Z uses X's Blue MRT next-hops to reach X's local-root.

C. If a node W in a common block with X was not reached in the SPF or reverse-SPF, then W is unordered with respect to X. X's Blue MRT next-hops to W are X's decreasing aka Red MRT next-hops to X's local-root. X's Red MRT next-hops to W are X's increasing aka Blue MRT next-hops to X's local-root.

3. For nodes in different blocks, the next-hops must be inherited via the relevant cut-vertex.

These are all captured in the detailed algorithm given in Section 4.6.5.

4.6.5. Complete Algorithm to Compute MRT Next-Hops

The complete algorithm to compute MRT Next-Hops for a particular router X is given in Figure 23. In addition to computing the Blue MRT next-hops and Red MRT next-hops used by X to reach each node Y, the algorithm also stores an "order_proxy", which is the proper cut-vertex to reach Y if it is outside the block, and which is used later in deciding whether the Blue MRT or the Red MRT can provide an acceptable alternate for a particular primary next-hop.

```
In_Common_Block(x, y)
  if ((x.localroot is y.localroot) or (x is y.localroot) or
      (y is x.localroot))
     return true
  return false

Store_Results(y, direction, spf_root)
   if direction is FORWARD
      y.higher = true
      y.blue_next_hops = y.next_hops
   if direction is REVERSE
      y.lower = true
      y.red_next_hops = y.next_hops

SPF_No_Traverse_Root(spf_root, block_root, direction)
   Initialize spf_heap to empty
   Initialize nodes' spf_metric to infinity and next_hops to empty
   spf_root.spf_metric = 0
```

```
       insert(spf_heap, spf_root)
       while (spf_heap is not empty)
           min_node = remove_lowest(spf_heap)
           Store_Results(min_node, direction, spf_root)
           if min_node is not block_root
              foreach interface intf of min_node
                 if (((direction is FORWARD) and intf.OUTGOING) or
                     ((direction is REVERSE) and intf.INCOMING)  and
                     In_Common_Block(spf_root, intf.remote_node))
                    if direction is FORWARD
                       path_metric = min_node.spf_metric + intf.metric
                    else
                       path_metric = min_node.spf_metric +
                                     intf.remote_intf.metric
                    if path_metric < intf.remote_node.spf_metric
                       intf.remote_node.spf_metric = path_metric
                       if min_node is spf_root
                         intf.remote_node.next_hops = make_list(intf)
                       else
                         intf.remote_node.next_hops = min_node.next_hops
                       insert_or_update(spf_heap, intf.remote_node)
                    else if path_metric is intf.remote_node.spf_metric
                       if min_node is spf_root
                          add_to_list(intf.remote_node.next_hops, intf)
                       else
                          add_list_to_list(intf.remote_node.next_hops,
                                           min_node.next_hops)

    SetEdge(y)
      if y.blue_next_hops is empty and y.red_next_hops is empty
         SetEdge(y.localroot)
         y.blue_next_hops = y.localroot.blue_next_hops
         y.red_next_hops = y.localroot.red_next_hops
         y.order_proxy = y.localroot.order_proxy

    Compute_MRT_NextHops(x, root)
       foreach node y
         y.higher = y.lower = false
         clear y.red_next_hops and y.blue_next_hops
         y.order_proxy = y
       SPF_No_Traverse_Root(x, x.localroot, FORWARD)
       SPF_No_Traverse_Root(x, x.localroot, REVERSE)

       // red and blue next-hops are stored to x.localroot as different
       // paths are found via the SPF and reverse-SPF.
       // Similarly any nodes whose local-root is x will have their
       // red_next_hops and blue_next_hops already set.
```

```
       // Handle nodes in the same block that aren't the local-root
       foreach node y
         if (y is not x) and (y.localroot is x.localroot)
            if y.higher
               y.red_next_hops = x.localroot.red_next_hops
            else if y.lower
               y.blue_next_hops = x.localroot.blue_next_hops
            else
               y.blue_next_hops = x.localroot.red_next_hops
               y.red_next_hops = x.localroot.blue_next_hops

       // Inherit next-hops and order_proxies to other components
       if x is not root
          root.blue_next_hops = x.localroot.blue_next_hops
          root.red_next_hops = x.localroot.red_next_hops
          root.order_proxy = x.localroot
       foreach node y
          if (y is not root) and (y is not x)
            SetEdge(y)

    Copute_RT_NextHops(x, root)
```

Figure 23

4.7. Identify MRT alternates

At this point, a computing router S knows its Blue MRT next-hops and Red MRT next-hops for each destination. The primary next-hops along the SPT are also known. It remains to determine for each primary next-hop to a destination D, which of the MRTs avoids the primary next-hop node F. This computation depends upon data set in Compute_MRT_NextHops such as each node y's y.blue_next_hops, y.red_next_hops, y.order_proxy, y.higher, y.lower and topo_orders. Recall that any router knows only which are the nodes greater and lesser than itself, but it cannot decide the relation between any two given nodes easily; that is why we need topological ordering.

For each primary next-hop node F to each destination D, S can call Select_Alternates(S, D, F) to determine whether to use the Blue MRT next-hops as the alternate next-hop(s) for that primary next-hop or to use the Red MRT next-hops. The algorithm is given in Figure 24 and discussed afterwards.

```
Select_Alternates(S, D, F)
   if D.order_proxy is not D
      D_lower = D.order_proxy.lower
      D_higher = D.order_proxy.higher
      D_topo_order = D.order_proxy.topo_order
   else
      D_lower = D.lower
      D_higher = D.higher
      D_topo_order = D.topo_order

   if D_higher
      if F.higher
         if F.topo_order < D_topo_order
            return USE_RED
         else
            return USE_BLUE
      else if F.lower
         return USE_BLUE
      else
        // F and S are neighbors so either F << S or F >> S
   else if D_lower
      if F.higher
         return USE_RED
      else if F.lower
         if F.topo_order < D_topo_order
            return USE_RED
         else
            return USE_BLUE
      else
         // F and S are neighbors so either F << S or F >> S
   else // D and S not ordered
      if F.lower
         return USE_BLUE
      else if F.upper
         return USE_RED
      else
         // F and S are neighbors so either F << S or F >> S
```

Figure 24

If either D>>S>>F or D<<S<<F holds true, the situation is simple: in the first case we should choose the increasing Blue next-hop, in the second case, the decreasing Red next-hop is the right choice.

However, when both D and F are greater than S the situation is not so simple, there can be three possibilities: (i) F>>D (ii) F<<D or (iii) F and D are not ordered. In the first case, we should choose the path towards D along the Blue tree. In contrast, in case (ii) the

Red path towards the root and then to D would be the solution. Finally, in case (iii) both paths would be acceptable. However, observe that if e.g. F.topo_order>D.topo_order, either case (i) or case (iii) holds true, which means that selecting the Blue next-hop is safe. Similarly, if F.topo_order<D.topo_order, we should select the Red next-hop. The situation is almost the same if both F and D are less than S.

Recall that we have added each link to the GADAG in some direction, so that is imposible that S and F are not ordered. But it is possible that S and D are not ordered, so we need to deal with this case as well. If F<<S, we can use the Red next-hop, because that path is first increasing until a node definitely greater than D is reached, than decreasing; this path must avoid using F. Similarly, if F>>S, we should use the Blue next-hop.

As an example consider the ADAG depicted in Figure 25 and first suppose that G is the source, D is the destination and H is the failed next-hop. Since D>>G, we need to compare H.topo_order and D.topo_order. Since D.topo_order>H.topo_order D must be not smaller than H, so we should select the decreasing path towards the root. If, however, the destination were instead J, we must find that H.topo_order>J.topo_order, so we must choose the increasing Blue next-hop to J, which is I. In the case, when instead the destination is C, we find that we need first decrease to avoid using H, so the Blue, first decreasing then increasing, path is selected.

```
[E]<-[D]<-[H]<-[J]
 |    ^    ^    |
 V    |    |    |
[R]  [C]  [G]->[I]
 |    ^    ^    ^
 V    |    |    |
[A]->[B]->[F]---|

      (a)
a 2-connected graph
```

Figure 25

5. Algorithm Alternatives and Evaluation

This description of the algorithm assumes a particular approach that is believed to be a reasonable compromise between complexity and computation. There are two options given for constructing the GADAG as both are reasonable and promising.

SPF-based GADAG Compute the common GADAG using Option 2 of SPF-based inheritance. This considers metrics when constructing the GADAG, which is important for path length and operational control. It has higher computational complexity than the Low-Point Inheritance GADAG.

Low-Point Inheritance GADAG Compute the common GADAG using Option 1 of Low-Point Inheritance. This ignores metrics when constructing the GADAG, but its computational complexity is O(links) which is attractive. It is possible that augmenting the GADAG by assigning directions to all links in the network graph and adding them to the GADAG will make the difference between this and the SPF-based GADAG minimal.

In addition, it is possible to calculate Destination-Rooted GADAG, where for each destination, a GADAG rooted at that destination is computed. The GADAG can be computed using either Low-Point Inheritance or SPF-based. Then a router would need to compute the blue MRT and red MRT next-hops to that destination. Building GADAGs per destination is computationally more expensive, but may give somewhat shorter alternate paths. It may be useful for live-live multicast along MRTs.

5.1. Algorithm Evaluation

When evaluating different algorithms and methods for IP Fast Reroute [RFC5714], there are three critical points to consider.

- o Coverage: For every Point of Local Repair (PLR) and local failure, is there an alternate to reach every destination? Those destinations include not only routers in the IGP area, but also prefixes outside the IGP area.

- o Alternate Length: What is the length of the alternate path offered compared to the optimal alternate route in the network? This is computed as the total length of the alternate path divided by the length of an optimal alternate path. The optimal alternate path is computed by removing the failed node and running an SPF to find the shortest path from the PLR to the destination.

- o Alternate Bandwidth: What percentage of the traffic sent to the failed point can be sent on the alternates? This is computed as the sum of the bandwidths along the alternate paths divided by the bandwidth of the primary paths that go through the failure point.

Simulation and modeling to evalute the MRT algorithms is underway. The algorithms being compared are:

- o SPF-based GADAG
- o Low-Point Inheritance GADAG
- o Destination-Rooted SPF-based GADAG
- o Destination-Rooted Low-Point Inheritance GADAG
- o Not-Via to Next-Next Hop[I-D.ietf-rtgwg-ipfrr-notvia-addresses]
- o Loop-Free Alternates[RFC5286]
- o Remote LFAs[I-D.shand-remote-lfa]

6. IANA Considerations

This doument includes no request to IANA.

7. Security Considerations

This architecture is not currently believed to introduce new security concerns.

8. References

8.1. Normative References

[I-D.atlas-rtgwg-mrt-frr-architecture]
Atlas, A., Konstantynowicz, M., Envedi, G., Csaszar, A., White, R., and M. Shand, "An Architecture for IP/LDP Fast-Reroute Using Maximally Redundant Trees", draft-atlas-rtgwg-mrt-frr-architecture-00 (work in progress), July 2011.

8.2. Informative References

[EnyediThesis]
Enyedi, G., "Novel Algorithms for IP Fast Reroute", Department of Telecommunications and Media Informatics, Budapest University of Technology and Economics Ph.D. Thesis, February 2011, <http://timon.tmit.bme.hu/theses/thesis_book.pdf>.

[I-D.ietf-rtgwg-ipfrr-notvia-addresses]
Shand, M., Bryant, S., and S. Previdi, "IP Fast Reroute

Using Not-via Addresses",
draft-ietf-rtgwg-ipfrr-notvia-addresses-07 (work in
progress), April 2011.

[I-D.ietf-rtgwg-lfa-applicability]
Filsfils, C., Francois, P., Shand, M., Decraene, B.,
Uttaro, J., Leymann, N., and M. Horneffer, "LFA
applicability in SP networks",
draft-ietf-rtgwg-lfa-applicability-03 (work in progress),
August 2011.

[I-D.shand-remote-lfa]
Bryant, S., Filsfils, C., Shand, M., and N. So, "Remote
LFA FRR", draft-shand-remote-lfa-00 (work in progress),
October 2011.

[Kahn_1962_topo_sort]
Kahn, A., "Topological sorting of large networks",
Communications of the ACM, Volume 5, Issue 11 , Nov 1962,
<http://dl.acm.org/citation.cfm?doid=368996.369025>.

[LFARevisited]
Retvari, G., Tapolcai, J., Enyedi, G., and A. Csaszar, "IP
Fast ReRoute: Loop Free Alternates Revisited", Proceedings
of IEEE INFOCOM , 2011, <http://opti.tmit.bme.hu/
~tapolcai/papers/retvari2011lfa_infocom.pdf>.

[LightweightNotVia]
Enyedi, G., Retvari, G., Szilagyi, P., and A. Csaszar, "IP
Fast ReRoute: Lightweight Not-Via without Additional
Addresses", Proceedings of IEEE INFOCOM , 2009,
<http://mycite.omikk.bme.hu/doc/71691.pdf>.

[MRTLinear]
Enyedi, G., Retvari, G., and A. Csaszar, "On Finding
Maximally Redundant Trees in Strictly Linear Time", IEEE
Symposium on Computers and Comunications (ISCC) , 2009,
<http://opti.tmit.bme.hu/~enyedi/ipfrr/
distMaxRedTree.pdf>.

[RFC3137] Retana, A., Nguyen, L., White, R., Zinin, A., and D.
McPherson, "OSPF Stub Router Advertisement", RFC 3137,
June 2001.

[RFC5286] Atlas, A. and A. Zinin, "Basic Specification for IP Fast
Reroute: Loop-Free Alternates", RFC 5286, September 2008.

[RFC5714] Shand, M. and S. Bryant, "IP Fast Reroute Framework", RFC 5714, January 2010.

Authors' Addresses

Alia Atlas
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
USA

Email: akatlas@juniper.net

Gabor Sandor Enyedi
Ericsson
Konyves Kalman krt 11
Budapest 1097
Hungary

Email: Gabor.Sandor.Enyedi@ericsson.com

Andras Csaszar
Ericsson
Konyves Kalman krt 11
Budapest 1097
Hungary

Email: Andras.Csaszar@ericsson.com

RTGWG C. Villamizar, Ed.
Internet-Draft OCCNC, LLC
Intended status: Informational D. McDysan, Ed.
Expires: August 10, 2014 Verizon
S. Ning
Tata Communications
A. Malis
Huawei
L. Yong
Huawei USA
February 06, 2014

Requirements for Advanced Multipath in MPLS Networks
draft-ietf-rtgwg-cl-requirement-16

Abstract

This document provides a set of requirements for Advanced Multipath in MPLS Networks.

Advanced Multipath is a formalization of multipath techniques currently in use in IP and MPLS networks and a set of extensions to existing multipath techniques.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

There is often a need to provide large aggregates of bandwidth that are best provided using parallel links between routers or carrying traffic over multiple MPLS Label Switched Paths (LSPs). In core networks there is often no alternative since the aggregate capacities of core networks today far exceed the capacity of a single physical link or single packet processing element.

The presence of parallel links, with each link potentially comprised of multiple layers has resulted in additional requirements. Certain services may benefit from being restricted to a subset of the component links or a specific component link, where component link characteristics, such as latency, differ. Certain services require that an LSP be treated as atomic and avoid reordering. Other services will continue to require only that reordering not occur within a flow as is current practice.

Numerous forms of multipath exist today including MPLS Link Bundling [RFC4201], Ethernet Link Aggregation [IEEE-802.1AX], and various forms of Equal Cost Multipath (ECMP) such as for OSPF ECMP, IS-IS ECMP, and BGP ECMP. Refer to the Appendices in [I-D.ietf-rtgwg-cl-use-cases] for a description of existing techniques and a set of references.

The purpose of this document is to clearly enumerate a set of requirements related to the protocols and mechanisms that provide MPLS based Advanced Multipath. The intent is to first provide a set of functional requirements, in Section 3, that are as independent as possible of protocol specifications . A set of general protocol requirements are defined in Section 4. A set of network management requirements are defined 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].

Any statement which requires the solution to support some new functionality through use of [RFC2119] keywords should be interpreted as follows. The implementation either MUST or SHOULD support the new functionality depending on the use of either MUST or SHOULD in the requirements statement. The implementation SHOULD in most or all cases allow any new functionality to be individually enabled or disabled through configuration. A service provider or other deployment MAY enable or disable any feature in their network, subject to implementation limitations on sets of features which can be disabled.

2. Definitions

Multipath
: The term multipath includes all techniques in which

1. Traffic can take more than one path from one node to a destination.

2. Individual packets take one path only. Packets are not subdivided and reassembled at the receiving end.

3. Packets are not resequenced at the receiving end.

4. The paths may be:

    a. parallel links between two nodes, or

b. specific paths across a network to a destination node, or

c. links or paths to an intermediate node used to reach a common destination.

The paths need not have equal capacity. The paths may or may not have equal cost in a routing protocol.

Advanced Multipath
: Advanced Multipath is a formalization of multipath techniques that meets the requirements defined in this document. A key capability of Advanced Multipath is the support of non-homogeneous component links.

Advanced Multipath Group (AMG)
: An Advanced Multipath Group (AMG) is a collection of component links where Advanced Multipath techniques are applied.

Composite Link
: The term Composite Link had been a registered trademark of Avici Systems, but was abandoned in 2007. The term composite link is now defined by the ITU-T in [ITU-T.G.800]. The ITU-T definition includes multipath as defined here, plus inverse multiplexing which is explicitly excluded from the definition of multipath.

Inverse Multiplexing
: Inverse multiplexing is another method of sending traffic over multiple links. Inverse multiplexing either transmits whole packets and resequences the packets at the receiving end or subdivides packets and reassembles the packets at the receiving end. Inverse multiplexing requires that all packets be handled by a common egress packet processing element and is therefore not useful for very high bandwidth applications.

Component Link
: The ITU-T definition of composite link in [ITU-T.G.800] and the IETF definition of link bundling in [RFC4201] both refer to an individual link in the composite link or link bundle as a component link. The term component link is applicable to all forms of multipath. The IEEE uses the term member rather than component link in Ethernet Link Aggregation [IEEE-802.1AX].

Client Layer
: A client layer is the layer immediately above a server layer.

Server Layer
: A server layer is the layer immediately below a client layer.

Higher Layers
Relative to a particular layer, a client layer and any layer above that is considered a higher layer. Upper layer is synonymous with higher layer.

Lower Layers
Relative to a particular layer, a server layer and any layer below that is considered a lower layer.

Client LSP
A client LSP is an LSP which has been set up over one or more lower layers. In the context of this discussion, one type of client LSP is a LSP which has been set up over an AMG.

Flow
A sequence of packets that should be transferred in order on one component link of a multipath.

Flow Identification
The label stack and other information that uniquely identifies a flow. Other information in flow identification may include an IP header, pseudowire (PW) control word, Ethernet MAC address, etc. Note that a client LSP may contain one or more Flows or a client LSP may be equivalent to a Flow. Flow identification is used to locally select a component link, or a path through the network toward the destination.

Load Balance
Load split, load balance, or load distribution refers to subdividing traffic over a set of component links such that load is fairly evenly distributed over the set of component links and certain packet ordering requirements are met. Some existing techniques better achieve these objectives than others.

Performance Objective
Numerical values for performance measures, principally availability, latency, and delay variation. Performance objectives may be related to Service Level Agreements (SLA) as defined in RFC2475 or may be strictly internal. Performance objectives may span links, edge-to-edge, or end-to-end. Performance objectives may span one provider or may span multiple providers.

A Component Link may be a point-to-point physical link (where a "physical link" includes one or more link layer plus a physical layer) or a logical link that preserves ordering in the steady state. A component link may have transient out of order events, but such events must not exceed the network's Performance Objectives. For example, a component link may be comprised of any supportable combination of link layers over a physical layer or over logical sub-layers, including those providing physical layer emulation, or over MPLS server layer LSP.

The ingress and egress of a multipath may be midpoint LSRs with respect to a given client LSP. A midpoint LSR does not participate in the signaling of any clients of the client LSP. Therefore, in general, multipath endpoints cannot determine requirements of clients of a client LSP through participation in the signaling of the clients of the client LSP.

This document makes no statement on whether Advanced Multipath is itself a layer or whether an instance of AMG is itself a layer. This is to avoid engaging in long and pointless discussions about what consistitutes a proper layer.

The term Advanced Multipath is intended to be used within the context of this document and the related documents, [I-D.ietf-rtgwg-cl-use-cases] and [I-D.ietf-rtgwg-cl-framework] and any other related document. Other advanced multipath techniques may in the future arise. If the capabilities defined in this document become commonplace, they would no longer be considered "advanced". Use of the term "advanced multipath" outside this document, if referring to the term as defined here, should indicate Advanced Multipath as defined by this document, citing the current document name. If using another definition of "advanced multipath", documents may optionally clarify that they are not using the term "advanced multipath" as defined by this document if clarification is deemed helpful.

3. Functional Requirements

The Functional Requirements in this section are grouped in subsections starting with the highest priority.

3.1. Availability, Stability and Transient Response

Limiting the period of unavailability in response to failures or transient events is extremely important as well as maintaining stability.

FR#1 The transient period between some service disrupting event and the convergence of the routing and/or signaling protocols MUST occur within a time frame specified by Performance Objective values.

FR#2 An AMG MAY be announced in conjunction with detailed parameters about its component links, such as bandwidth and latency. The AMG SHALL behave as a single IGP adjacency.

FR#3 The solution SHALL provide a means to summarize some routing advertisements regarding the characteristics of an AMG such that the updated protocol mechanisms maintain convergence times within the timeframe needed to meet or not significantly exceed existing Performance Objective for convergence on the same network or convergence on a network with a similar topology.

FR#4 The solution SHALL ensure that restoration operations happen within the timeframe needed to meet existing Performance Objective for restoration time on the same network or restoration time on a network with a similar topology.

FR#5 The solution shall provide a mechanism to select a set of paths for an LSP across a network in such a way that flows within the LSP are distributed across the set of paths while meeting all of the other requirements stated above. The solution SHOULD work in a manner similar to existing multipath techniques except as necessary to accommodate Advanced Multipath requirements.

FR#6 If extensions to existing protocols are specified and/or new protocols are defined, then the solution SHOULD provide a means for a network operator to migrate an existing deployment in a minimally disruptive manner.

FR#7 Any load balancing solutions MUST NOT oscillate. Some change in path MAY occur. The solution MUST ensure that path stability and traffic reordering continue to meet Performance Objective on the same network or on a network with a similar topology. Since oscillation may cause reordering, there MUST be means to control the frequency of changing the component link over which a flow is placed.

FR#8 Management and diagnostic protocols MUST be able to operate over AMGs.

Existing scaling techniques used in MPLS networks apply to MPLS networks which support Advanced Multipath. Scalability and stability are covered in more detail in [I-D.ietf-rtgwg-cl-framework].

3.2. Component Links Provided by Lower Layer Networks

A component link may be supported by a lower layer network. For example, the lower layer may be a circuit switched network or another MPLS network (e.g., MPLS-TP)). The lower layer network may change the latency (and/or other performance parameters) seen by the client layer. Currently, there is no protocol for the lower layer network to inform the higher layer network of a change in a performance parameter. Communication of the latency performance parameter is a very important requirement. Communication of other performance parameters (e.g., delay variation) is desirable.

FR#9 The solution SHALL specify a protocol means to allow a server layer network to communicate latency to the client layer network.

FR#10 The precision of latency reporting SHOULD be configurable. A reasonable default SHOULD be provided. Implementations SHOULD support precision of at least 10% of the one way latencies for latency of 1 msec or more.

The intent is to measure the predominant latency in uncongested service provider networks, where geographic delay dominates and is on the order of milliseconds or more. The argument for including queuing delay is that it reflects the delay experienced by applications. The argument against including queuing delay is that if used in routing decisions it can result in routing instability. This tradeoff is discussed in detail in [I-D.ietf-rtgwg-cl-framework].

3.3. Component Links with Different Characteristics

As one means to provide high availability, network operators deploy a topology in the MPLS network using lower layer networks that have a certain degree of diversity at the lower layer(s). Many techniques have been developed to balance the distribution of flows across component links that connect the same pair of nodes or ultimately lead to a common destination.

FR#11 In requirements that follow in this document the word "indicate" is used where information may be provided by either the combination of link state IGP advertisement and MPLS LSP signaling or via management plane protocols. In later documents providing framework and protocol definitions both signaling and management plane mechanisms MUST be defined.

FR#12 The solution SHALL provide a means for the client layer to indicate a requirement that a client LSP will traverse a component link with the minimum latency value. This will provide a means by which minimum latency Performance Objectives of flows within the client LSP can be supported.

FR#13 The solution SHALL provide a means for the client layer to indicate a requirement that a client LSP will traverse a component link with a maximum acceptable latency value as specified by protocol. This will provide a means by which bounded latency Performance Objectives of flows within the client LSP can be supported.

FR#14 The solution SHALL provide a means for the client layer to indicate a requirement that a client LSP will traverse a component link with a maximum acceptable delay variation value as specified by protocol.

The above set of requirements apply to component links with different characteristics regardless as to whether those component links are provided by parallel physical links between nodes or provided by sets of paths across a network provided by server layer LSP.

Allowing multipath to contain component links with different characteristics can improve the overall load balance and can be accomplished while still accommodating the more strict requirements of a subset of client LSP.

3.4. Considerations for Bidirectional Client LSP

Some client LSP MAY require a path bound to a specific set of component links. This case is most likely to occur in bidirectional client LSP where time synchronization protocols such as Precision Time Protocol (PTP) or Network Time Protocol (NTP) are carried, or in any other case where symmetric delay is highly desirable. There may be other uses of this capability.

Other client LSP may only require that the LSP path serve the same set of nodes in both directions. This is necessary if protocols are carried which make use of the reverse direction of the LSP as a back channel in cases such OAM protocols using IPv4 Time to Live (TTL) or IPv4 Hop Limit to monitor or diagnose the underlying path. There may be other uses of this capability.

FR#15 The solution SHALL provide a means for the client layer to indicate a requirement that a client LSP be bound to a particular component link within an AMG. If this option is not exercised, then a client LSP that is carried over an AMG may be bound to any component link or set of component links matching all other signaled requirements, and different directions of a bidirectional client LSP can be bound to different component links.

FR#16 The solution MUST support a means for the client layer to indicate a requirement that for a specific co-routed bidirectional client LSP both directions of the co-routed bidirectional client LSP MUST be bound to the same set of nodes.

FR#17 A client LSP which is bound to a specific component link SHOULD NOT exceed the capacity of a single component link. This is inherent in the assumption that a network SHOULD NOT operate in a congested state if congestion is avoidable.

For some large bidirectional client LSP it may not be necessary (or possible due to the client LSP capacity) to bind the LSP to a common set of component links but may be necessary or desirable to constrain the path taken by the LSP to the same set of nodes in both directions. Without an entirely new and highly dynamic protocol, it is not feasible to constrain such an bidirectional client LSP to take multiple paths and coordinate load balance on each side to keep both directions of flows within such an LSP on common paths.

3.5. Multipath Load Balancing Dynamics

Multipath load balancing attempts to keep traffic levels on all component links below congestion levels if possible and preferably well balanced. Load balancing is minimally disruptive (see discussion below this section’s list of requirements). The sensitivity to these minimal disruptions of traffic flows within specific client LSP needs to be considered.

FR#18 The solution SHALL provide a means for the client layer to indicate a requirement that a specific client LSP MUST NOT be split across multiple component links.

FR#19 The solution SHALL provide a means local to a node that automatically distributes flows across the component links in the AMG such that Performance Objectives are met as described in prior requirements in Section 3.3.

FR#20 The solution SHALL measure traffic flows or groups of traffic flows and dynamically select the component link on which to place this traffic in order to balance the load so that no component link in the AMG between a pair of nodes is overloaded.

FR#21 When a traffic flow is moved from one component link to another in the same AMG between a set of nodes, it MUST be done so in a minimally disruptive manner.

FR#22 Load balancing MAY be used during sustained low traffic periods to reduce the number of active component links for the purpose of power reduction.

FR#23 The solution SHALL provide a means for the client layer to indicate a requirement that a specific client LSP contains traffic whose frequency of component link change due to load balancing needs to be bounded by a specific value. The solution MUST provide a means to bound the frequency of component link change due to load balancing for subsets of traffic flow on AMGs.

FR#24 The solution SHALL provide a means to distribute traffic flows from a single client LSP across multiple component links to handle at least the case where the traffic carried in an client LSP exceeds that of any component link in the AMG.

FR#25 The solution SHOULD support the use case where an AMG itself is a component link for a higher order AMG. For example, an AMG comprised of MPLS-TP bi-directional tunnels viewed as logical links could then be used as a component link in yet another AMG that connects MPLS routers.

FR#26 If the total demand offered by traffic flows exceeds the capacity of the AMG, the solution SHOULD define a means to cause some client LSP to move to an alternate set of paths that are not congested. These "preempted LSP" may not be restored if there is no uncongested path in the network.

A minimally disruptive change implies that as little disruption as is practical occurs. Such a change can be achieved with zero packet loss. A delay discontinuity may occur, which is considered to be a minimally disruptive event for most services if this type of event is sufficiently rare. A delay discontinuity is an example of a minimally disruptive behavior corresponding to current techniques.

A delay discontinuity is an isolated event which may greatly exceed the normal delay variation (jitter). A delay discontinuity has the following effect. When a flow is moved from a current link to a target link with lower latency, reordering can occur. When a flow is moved from a current link to a target link with a higher latency, a time gap can occur. Some flows (e.g., timing distribution, PW circuit emulation) are quite sensitive to these effects. A delay discontinuity can also cause a jitter buffer underrun or overrun affecting user experience in real time voice services (causing an audible click). These sensitivities may be specified in a Performance Objective.

As with any load balancing change, a change initiated for the purpose of power reduction may be minimally disruptive. Typically the disruption is limited to a change in delay characteristics and the potential for a very brief period with traffic reordering. The network operator when configuring a network for power reduction should weigh the benefit of power reduction against the disadvantage of a minimal disruption.

4. General Requirements for Protocol Solutions

This section defines requirements for protocol specification used to meet the functional requirements specified in Section 3.

GR#1 The solution SHOULD extend existing protocols wherever possible, developing a new protocol only where doing so adds a significant set of capabilities.

GR#2 A solution SHOULD extend LDP capabilities to meet functional requirements. This MUST be accomplished without defining LDP Traffic Engineering (TE) methods as decided in [RFC3468]).

GR#3 Coexistence of LDP and RSVP-TE signaled LSPs MUST be supported on an AMG. Function requirements SHOULD, where possible, be accommodated in a manner that supports LDP signaled LSP, RSVP signaled LSP, and LSP set up using management plane mechanisms.

GR#4 When the nodes connected via an AMG are in the same routing domain, the solution MAY define extensions to the IGP.

GR#5 When the nodes are connected via an AMG are in different MPLS network topologies, the solution SHALL NOT rely on extensions to the IGP.

GR#6 The solution SHOULD support AMG IGP advertisement that results in convergence time better than that of advertising the individual component links. The solution SHALL be designed so that it represents the range of capabilities of the individual component links such that functional requirements are met, and also minimizes the frequency of advertisement updates which may cause IGP convergence to occur.

Examples of advertisement update triggering events to be considered include: client LSP establishment/release, changes in component link characteristics (e.g., latency, up/down state), and/or bandwidth utilization.

GR#7 When a worst case failure scenario occurs, the number of RSVP-TE client LSPs to be resignaled will cause a period of unavailability as perceived by users. The resignaling time of the solution MUST support protocol mechanisms meeting existing provider Performance Objective for the duration of unavailability without significantly relaxing those existing Performance Objectives for the same network or for networks with similar topology. For example, the processing load due to IGP readvertisement MUST NOT increase significantly and the resignaling time of the solution MUST NOT increase significantly as compared with current methods.

5. Management Requirements

MR#1 Management Plane MUST support polling of the status and configuration of an AMG and its individual component links and support notification of status change.

MR#2 Management Plane MUST be able to activate or de-activate any component link in an AMG in order to facilitate operation maintenance tasks. The routers at each end of an AMG MUST redistribute traffic to move traffic from a de-activated link to other component links based on the traffic flow TE criteria.

MR#3 Management Plane MUST be able to configure a client LSP over an AMG and be able to select a component link for the client LSP.

MR#4 Management Plane MUST be able to trace which component link a client LSP is assigned to and monitor individual component link and AMG performance.

MR#5 Management Plane MUST be able to verify connectivity over each individual component link within an AMG.

MR#6 Component link fault notification MUST be sent to the management plane.

MR#7 AMG fault notification MUST be sent to the management plane and MUST be distributed via link state message in the IGP.

MR#8 Management Plane SHOULD provide the means for an operator to initiate an optimization process.

MR#9 An operator initiated optimization MUST be performed in a minimally disruptive manner as described in Section 3.5.

6. Acknowledgements

Frederic Jounay of France Telecom and Yuji Kamite of NTT Communications Corporation co-authored a version of this document.

A rewrite of this document occurred after the IETF77 meeting. Dimitri Papadimitriou, Lou Berger, Tony Li, the former WG chairs John Scuder and Alex Zinin, the current WG chair Alia Atlas, and others provided valuable guidance prior to and at the IETF77 RTGWG meeting.

Tony Li and John Drake have made numerous valuable comments on the RTGWG mailing list that are reflected in versions following the IETF77 meeting.

Iftekhar Hussain and Kireeti Kompella made comments on the RTGWG mailing list after IETF82 that identified a new requirement. Iftekhar Hussain made numerous valuable comments on the RTGWG mailing list that resulted in improvements to document clarity.

In the interest of full disclosure of affiliation and in the interest of acknowledging sponsorship, past affiliations of authors are noted. Much of the work done by Ning So occurred while Ning was at Verizon. Much of the work done by Curtis Villamizar occurred while at Infinera. Much of the work done by Andy Malis occurred while Andy was at Verizon.

Tom Yu and Francis Dupont provided the SecDir and GenArt reviews respectively. Both reviews provided useful comments. The current wording of the security section is based on suggested wording from Tom Yu. Lou Berger provided the RtgDir review which resulted in the document being renamed and substantial clarification of terminology and document wording, particularly in the Abstract, Introduction, and Definitions sections.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

The security considerations for MPLS/GMPLS and for MPLS-TP are documented in [RFC5920] and [RFC6941]. This document does not impact the security of MPLS, GMPLS, or MPLS-TP.

The additional information that this document requires does not provide significant additional value to an attacker beyond the information already typically available from attacking a routing or signaling protocol. If the requirements of this document are met by extending an existing routing or signaling protocol, the security considerations of the protocol being extended apply. If the requirements of this document are met by specifying a new protocol, the security considerations of that new protocol should include an evaluation of what level of protection is required by the additional information specified in this document, such as data origin authentication.

9. References

9.1. Normative References

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

9.2. Informative References

[I-D.ietf-rtgwg-cl-framework]
Ning, S., McDysan, D., Osborne, E., Yong, L., and C. Villamizar, "Advanced Multipath Framework in MPLS", draft-ietf-rtgwg-cl-framework-04 (work in progress), July 2013.

[I-D.ietf-rtgwg-cl-use-cases]
Ning, S., Malis, A., McDysan, D., Yong, L., and C. Villamizar, "Advanced Multipath Use Cases and Design Considerations", draft-ietf-rtgwg-cl-use-cases-05 (work in progress), November 2013.

[IEEE-802.1AX]
IEEE Standards Association, "IEEE Std 802.1AX-2008 IEEE Standard for Local and Metropolitan Area Networks - Link Aggregation", 2006, <http://standards.ieee.org/getieee802/download/802.1AX-2008.pdf>.

[ITU-T.G.800]
ITU-T, "Unified functional architecture of transport networks", 2007, <http://www.itu.int/rec/T-REC-G/recommendation.asp?parent=T-REC-G.800>.

[RFC3468] Andersson, L. and G. Swallow, "The Multiprotocol Label Switching (MPLS) Working Group decision on MPLS signaling protocols", RFC 3468, February 2003.

[RFC4201] Kompella, K., Rekhter, Y., and L. Berger, "Link Bundling in MPLS Traffic Engineering (TE)", RFC 4201, October 2005.

[RFC5920] Fang, L., "Security Framework for MPLS and GMPLS Networks", RFC 5920, July 2010.

[RFC6941] Fang, L., Niven-Jenkins, B., Mansfield, S., and R. Graveman, "MPLS Transport Profile (MPLS-TP) Security Framework", RFC 6941, April 2013.

Authors' Addresses

Curtis Villamizar (editor)
OCCNC, LLC

Email: curtis@occnc.com

Dave McDysan (editor)
Verizon
22001 Loudoun County PKWY
Ashburn, VA 20147
USA

Email: dave.mcdysan@verizon.com

So Ning
Tata Communications

Email: ning.so@tatacommunications.com

Andrew Malis
Huawei Technologies

Email: agmalis@gmail.com

Lucy Yong
Huawei USA
5340 Legacy Dr.
Plano, TX 75025
USA

Phone: +1 469-277-5837
Email: lucy.yong@huawei.com

S. Ning
Tata Communications
D. McDysan
Verizon
E. Osborne
Cisco
L. Yong
Huawei USA
C. Villamizar
Outer Cape Cod Network
Consulting
June 29, 2012

Composite Link Framework in Multi Protocol Label Switching (MPLS)
draft-so-yong-rtgwg-cl-framework-06

Abstract

This document specifies a framework for support of composite link in MPLS networks. A composite link consists of a group of homogenous or non-homogenous links that have the same forward adjacency and can be considered as a single TE link or an IP link in routing. A composite link relies on its component links to carry the traffic over the composite link. Applicability is described for a single pair of MPLS-capable nodes, a sequence of MPLS-capable nodes, or a set of layer networks connecting MPLS-capable nodes.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

Composite Link functional requirements are specified in [I-D.ietf-rtgwg-cl-requirement]. Composite Link use cases are described in [I-D.symmvo-rtgwg-cl-use-cases]. This document specifies a framework to meet these requirements.

Classic multipath, including Ethernet Link Aggregation has been widely used in today's MPLS networks [RFC4385][RFC4928]. Classic multipath using non-Ethernet links are often advertised using MPLS Link bundling. A link bundle [RFC4201] bundles a group of homogeneous links as a TE link to make IGP-TE information exchange and RSVP-TE signaling more scalable. A composite link allows bundling non-homogenous links together as a single logical link. The motivations for using a composite link are descried in [I-D.ietf-rtgwg-cl-requirement] and [I-D.symmvo-rtgwg-cl-use-cases].

This document describes a composite link framework in the context of MPLS networks using an IGP-TE and RSVP-TE MPLS control plane with GMPLS extensions [RFC3209][RFC3630][RFC3945][RFC5305].

A composite link is a single logical link in MPLS network that contains multiple parallel component links between two MPLS LSR. Unlike a link bundle [RFC4201], the component links in a composite link can have different properties such as cost or capacity.

Specific protocol solutions are outside the scope of this document, however a framework for the extension of existing protocols is provided. Backwards compatibility is best achieved by extending existing protocols where practical rather than inventing new protocols. The focus is on examining where existing protocol mechanisms fall short with respect to [I-D.ietf-rtgwg-cl-requirement] and on extensions that will be required to accommodate functionality that is called for in [I-D.ietf-rtgwg-cl-requirement].

1.1. Architecture Summary

Networks aggregate information, both in the control plane and in the data plane, as a means to achieve scalability. A tradeoff exists between the needs of scalability and the needs to identify differing path and link characteristics and differing requirements among flows contained within further aggregated traffic flows. These tradeoffs are discussed in detail in Section 3.

Some aspects of Composite Link requirements present challenges for which multiple solutions may exist. In Section 4 various challenges and potential approaches are discussed.

A subset of the functionality called for in [I-D.ietf-rtgwg-cl-requirement] is available through MPLS Link Bundling [RFC4201]. Link bundling and other existing standards applicable to Composite Link are covered in Section 5.

The most straightforward means of supporting Composite Link requirements is to extend MPLS protocols and protocol semantics and in particular to extend link bundling. Extensions which have already been proposed in other documents which are applicable to Composite Link are discussed in Section 6.

Goals of most new protocol work within IETF is to reuse existing protocol encapsulations and mechanisms where they meet requirements and extend existing mechanisms such that additional complexity is minimized while meeting requirements and such that backwards compatibility is preserved to the extent it is practical to do so. These goals are considered in proposing a framework for further protocol extensions and mechanisms in Section 7.

1.2. Conventions used in this document

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].

1.2.1. Terminology

Terminology defined in [I-D.ietf-rtgwg-cl-requirement] is used in this document.

The abbreviation IGP-TE is used as a shorthand indicating either OSPF-TE [RFC3630] or ISIS-TE [RFC5305].

2. Composite Link Key Characteristics

[I-D.ietf-rtgwg-cl-requirement] defines external behavior of Composite Links. The overall framework approach involves extending existing protocols in a backwards compatible manner and reusing ongoing work elsewhere in IETF where applicable, defining new protocols or semantics only where necessary. Given the requirements, and this approach of extending MPLS, Composite Link key characteristics can be described in greater detail than given requirements alone.

2.1. Flow Identification

Traffic mapping to component links is a data plane operation. Control over how the mapping is done may be directly dictated or constrained by the control plane or by the management plane. When unconstrained by the control plane or management plane, distribution of traffic is entirely a local matter. Regardless of constraints or lack or constraints, the traffic distribution is required to keep packets belonging to individual flows in sequence and meet QoS criteria specified per LSP by either signaling or management [RFC2475][RFC3260]. A key objective of the traffic distribution is to not overload any component link, and be able to perform local recovery when one of component link fails.

The network operator may have other objectives such as placing a bidirectional flow or LSP on the same component link in both direction, load balance over component links, composite link energy saving, and etc. These new requirements are described in [I-D.ietf-rtgwg-cl-requirement].

Examples of means to identify a flow may in principle include:

1. an LSP identified by an MPLS label,

2. a sub-LSP [I-D.kompella-mpls-rsvp-ecmp] identified by an MPLS label,

3. a pseudowire (PW) [RFC3985] identified by an MPLS PW label,

4. a flow or group of flows within a pseudowire (PW) [RFC6391] identified by an MPLS flow label,

5. a flow or flow group in an LSP [I-D.ietf-mpls-entropy-label] identified by an MPLS entropy label,

6. all traffic between a pair of IP hosts, identified by an IP source and destination pair,

7. a specific connection between a pair of IP hosts, identified by an IP source and destination pair, protocol, and protocol port pair,

8. a layer-2 conversation within a pseudowire (PW), where the identification is PW payload type specific, such as Ethernet MAC addresses and VLAN tags within an Ethernet PW (RFC4448).

Although in principle a layer-2 conversation within a pseudowire (PW), may be identified by PW payload type specific information, in practice this is impractical at LSP midpoints when PW are carried. The PW ingress may provide equivalent information in a PW flow label [RFC6391]. Therefore, in practice, item #8 above is covered by [RFC6391] and may be dropped from the list.

An LSR must at least be capable of identifying flows based on MPLS labels. Most MPLS LSP do not require that traffic carried by the LSP are carried in order. MPLS-TP is a recent exception. If it is assumed that no LSP require strict packet ordering of the LSP itself (only of flows within the LSP), then the entire label stack can be used as flow identification. If some LSP may require strict packet ordering but those LSP cannot be distinguished from others, then only the top label can be used as a flow identifier. If only the top label is used (for example, as specified by [RFC4201] when the "all-ones" component described in [RFC4201] is not used), then there may not be adequate flow granularity to accomplish well balanced traffic distribution and it will not be possible to carry LSP that are larger than any individual component link.

The number of flows can be extremely large. This may be the case when the entire label stack is used and is always the case when IP addresses are used in provider networks carrying Internet traffic. Current practice for native IP load balancing at the time of writing were documented in [RFC2991], [RFC2992]. These practices as described, make use of IP addresses. The common practices were extended to include the MPLS label stack and the common practice of looking at IP addresses within the MPLS payload. These extended practices are described in [RFC4385] and [RFC4928] due to their impact on pseudowires without a PWE3 Control Word. Additional detail on current multipath practices can be found in the appendices of [I-D.symmvo-rtgwg-cl-use-cases].

Using only the top label supports too coarse a traffic balance. Using the full label stack or IP addresses as flow identification provides a sufficiently fine traffic balance, but is capable of identifying such a high number of distinct flows, that a technique of grouping flows, such as hashing on the flow identification criteria, becomes essential to reduce the stored state, and is an essential scaling technique. Other means of grouping flows may be possible.

In summary:

1. Load balancing using only the MPLS label stack provides too coarse a granularity of load balance.

2. Tracking every flow is not scalable due to the extremely large number of flows in provider networks.

3. Existing techniques, IP source and destination hash in particular, have proven in over two decades of experience to be an excellent way of identifying groups of flows.

4. If a better way to identify groups of flows is discovered, then that method can be used.

5. IP address hashing is not required, but use of this technique is strongly encouraged given the technique's long history of successful deployment.

2.2. Composite Link in Control Plane

A composite Link is advertised as a single logical interface between two connected routers, which forms forwarding adjacency (FA) between the routers. The FA is advertised as a TE-link in a link state IGP, using either OSPF-TE or ISIS-TE. The IGP-TE advertised interface parameters for the composite link can be preconfigured by the network operator or be derived from its component links. Composite link advertisement requirements are specified in [I-D.ietf-rtgwg-cl-requirement].

In IGP-TE, a composite link is advertised as a single TE link between two connected routers. This is similar to a link bundle [RFC4201]. Link bundle applies to a set of homogenous component links. Composite link allows homogenous and non-homogenous component links. Due to the similarity, and for backwards compatibility, extending link bundling is viewed as both simple and as the best approach.

In order for a route computation engine to calculate a proper path for a LSP, it is necessary for composite link to advertise the summarized available bandwidth as well as the maximum bandwidth that can be made available for single flow (or single LSP where no finer flow identification is available). If a composite link contains some non-homogeneous component links, the composite link also should advertise the summarized bandwidth and the maximum bandwidth for single flow per each homogeneous component link group.

Both LDP [RFC5036] and RSVP-TE [RFC3209] can be used to signal a LSP over a composite link. LDP cannot be extended to support traffic engineering capabilities [RFC3468].

When an LSP is signaled using RSVP-TE, the LSP MUST be placed on the component link that meets the LSP criteria indicated in the signaling message.

When an LSP is signaled using LDP, the LSP MUST be placed on the component link that meets the LSP criteria, if such a component link is available. LDP does not support traffic engineering capabilities, imposing restrictions on LDP use of Composite Link. See Section 4.2.5 for further details.

A composite link may contain non-homogeneous component links. The route computing engine may select one group of component links for a LSP. The routing protocol MUST make this grouping available in the TE-LSDB. The route computation used in RSVP-TE MUST be extended to include only the capacity of groups within a composite link which meet LSP criteria. The signaling protocol MUST be able to indicate either the criteria, or which groups may be used. A composite link MUST place the LSP on a component link or group which meets or exceeds the LSP criteria.

Composite link capacity is aggregated capacity. LSP capacity MAY be larger than individual component link capacity. Any aggregated LSP can determine a bounds on the largest microflow that could be carried and this constraint can be handled as follows.

1. If no information is available through signaling, management plane, or configuration, the largest microflow is bound by one of the following:

   A. the largest single LSP if most traffic is RSVP-TE signaled and further aggregated,

   B. the largest pseudowire if most traffic is carrying pseudowire payloads that are aggregated within RSVP-TE LSP,

   C. or the largest source and sink interface if a large amount of IP or LDP traffic is contained within the aggregate.

   If a very large amount of traffic being aggregated is IP or LDP, then the largest microflow is bound by the largest component link on which IP traffic can arrive. For example, if an LSR is acting as an LER and IP and LDP traffic is arrving on 10 Gb/s edge interfaces, then no microflow larger than 10 Gb/s will be present on the RSVP-TE LSP that aggregate traffic across the core, even if the core interfaces are 100 Gb/s interfaces.

2. The prior conditions provide a bound on the largest microflow when no signaling extensions indicate a bounds. If an LSP is aggregating smaller LSP for which the largest expected microflow carried by the smaller LSP is signaled, then the largest microflow expected in the containing LSP (the aggregate) is the maximum of the largest expected microflow for any contained LSP. For example, RSVP-TE LSP may be large but aggregate traffic for which the source or sink are all 1 Gb/s or smaller interfaces

(such as in mobile applications in which cell sites backhauls are no larger than 1 Gb/s). If this information is carried in the LSP originated at the cell sites, then further aggregates across a core may make use of this information.

3. The IGP must provide the bounds on the largest microflow that a composite link can accommodate, which is the maximum capacity on a component link that can be made available by moving other traffic. This information is needed by the ingress LER for path determination.

4. A means to signal an LSP whose capacity is larger than individual component link capacity is needed [I-D.ietf-rtgwg-cl-requirement] and also signal the largest microflow expected to be contained in the LSP. If a bounds on the largest microflow is not signaled there is no means to determine if an LSP which is larger than any component link can be subdivided into flows and therefore should be accepted by admission control.

When a bidirectional LSP request is signaled over a composite link, if the request indicates that the LSP must be placed on the same component link, the routers of the composite link MUST place the LSP traffic in both directions on a same component link. This is particularly challenging for aggregated capacity which makes use of the label stack for traffic distribution. The two requirements are mutually exclusive for any one LSP. No one LSP may be both larger than any individual component link and require symmetrical paths for every flow. Both requirements can be accommodated by the same composite link for different LSP, with any one LSP requiring no more than one of these two features.

Individual component link may fail independently. Upon component link failure, a composite link MUST support a minimally disruptive local repair, preempting any LSP which can no longer be supported. Available capacity in other component links MUST be used to carry impacted traffic. The available bandwidth after failure MUST be advertised immediately to avoid looped crankback.

When a composite link is not able to transport all flows, it preempts some flows based upon local management configuration and informs the control plane on these preempted flows. The composite link MUST support soft preemption [RFC5712]. This action ensures the remaining traffic is transported properly. FR#10 requires that the traffic be restored. FR#12 requires that any change be minimally disruptive. These two requirements are interpreted to include preemption among the types of changes that must be minimally disruptive.

2.3. Composite Link in Data Plane

The data plane must first identify groups of flows. Flow identification is covered in Section 2.1. Having identified groups of flows the groups must be placed on individual component links. This second step is called traffic distribution or traffic placement. The two steps together are known as traffic balancing or load balancing.

Traffic distribution may be determined by or constrained by control plane or management plane. Traffic distribution may be changed due to component link status change, subject to constraints imposed by either the management plane or control plane. The distribution function is local to the routers in which a composite link belongs to and is not specified here.

When performing traffic placement, a composite link does not differentiate multicast traffic vs. unicast traffic.

In order to maintain scalability, existing data plane forwarding retains state associated with the top label only. The use of flow group identification is in a second step in the forwarding process. Data plane forwarding makes use of the top label to select a composite link, or a group of components within a composite link or for the case where an LSP is pinned (see [RFC4201]), a specific component link. For those LSP for which the LSP selects only the composite link or a group of components within a composite link, the load balancing makes use of the flow group identification.

The most common traffic placement techniques uses the a flow group identification as an index into a table. The table provides an indirection. The number of bits of hash is constrained to keep table size small. While this is not the best technique, it is the most common. Better techniques exist but they are outside the scope of this document and some are considered proprietary.

Requirements to limit frequency of load balancing can be adhered to by keeping track of when a flow group was last moved and imposing a minimum period before that flow group can be moved again. This is straightforward for a table approach. For other approaches it may be less straightforward but is acheivable.

3. Architecture Tradeoffs

Scalability and stability are critical considerations in protocol design where protocols may be used in a large network such as today's service provider networks. Composite Link is applicable to networks which are large enough to require that traffic be split over multiple paths. Scalability is a major consideration for networks that reach a capacity large enough to require Composite Link.

Some of the requirements of Composite Link could potentially have a negative impact on scalability. For example, Composite Link requires additional information to be carried in situations where component links differ in some significant way.

3.1. Scalability Motivations

In the interest of scalability information is aggregated in situations where information about a large amount of network capacity or a large amount of network demand provides is adequate to meet requirements. Routing information is aggregated to reduce the amount of information exchange related to routing and to simplify route computation (see Section 3.2).

In an MPLS network large routing changes can occur when a single fault occurs. For example, a single fault may impact a very large number of LSP traversing a given link. As new LSP are signaled to avoid the fault, resources are consumed elsewhere, and routing protocol announcements must flood the resource changes. If protection is in place, there is less urgency to converging quickly. If multiple faults occur that are not covered by shared risk groups (SRG), then some protection may fail, adding urgency to converging quickly even where protection was deployed.

Reducing the amount of information allows the exchange of information during a large routing change to be accomplished more quickly and simplifies route computation. Simplifying route computation improves convergence time after very significant network faults which cannot be handled by preprovisioned or precomputed protection mechanisms. Aggregating smaller LSP into larger LSP is a means to reduce path computation load and reduce RSVP-TE signaling (see Section 3.3).

Neglecting scaling issues can result in performance issues, such as slow convergence. Neglecting scaling in some cases can result in networks which perform so poorly as to become unstable.

3.2. Reducing Routing Information and Exchange

Link bundling at the very least provides a means of aggregating control plane information. Even where the all-ones component link supported by link bundling is not used, the amount of control information is reduced by the average number of component links in a bundle.

Fully deaggregating link bundle information would negate this benefit. If there is a need to deaggregate, such as to distinguish between groups of links within specified ranges of delay, then no more deaggregation than is necessary should be done.

For example, in supporting the requirement for heterogeneous component links, it makes little sense to fully deaggregate link bundles when adding support for groups of component links with common attributes within a link bundle can maintain most of the benefit of aggregation while adequately supporting the requirement to support heterogeneous component links.

Routing information exchange is also reduced by making sensible choices regarding the amount of change to link parameters that require link readvertisement. For example, if delay measurements include queuing delay, then a much more coarse granularity of delay measurement would be called for than if the delay does not include queuing and is dominated by geographic delay (speed of light delay).

3.3. Reducing Signaling Load

Aggregating traffic into very large hierarchical LSP in the core very substantially reduces the number of LSP that need to be signaled and the number of path computations any given LSR will be required to perform when a major network fault occurs.

In the extreme, applying MPLS to a very large network without hierarchy could exceed the 20 bit label space. For example, in a network with 4,000 nodes, with 2,000 on either side of a cutset, would have 4,000,000 LSP crossing the cutset. Even in a degree four cutset, an uneven distribution of LSP across the cutset, or the loss of one link would result in a need to exceed the size of the label space. Among provider networks, 4,000 access nodes is not at all large.

In less extreme cases, having each node terminate hundreds of LSP to achieve a full mesh creates a very large computational load. The time complexity of one CSPF computation is order(N log N), where L is proportional to N, and N and L are the number of nodes and number of links, respectively. If each node must perform order(N) computations when a fault occurs, then the computational load increases as order(N^2 log N) as the number of nodes increases. In practice at the time of writing, this imposes a limit of a few hundred nodes in a full mesh of MPLS LSP before the computational load is sufficient to result in unacceptable convergence times.

Two solutions are applied to reduce the amount of RSVP-TE signaling. Both involve subdividing the MPLS domain into a core and a set of regions.

3.3.1. Reducing Signaling Load using LDP

LDP can be used for edge-to-edge LSP, using RSVP-TE to carry the LDP intra-core traffic and also optionally also using RSVP-TE to carry the LDP intra-region traffic within each region. LDP does not support traffic engineering, but does support multipoint-to-point (MPTP) LSP, which require less signaling than edge-to-edge RSVP-TE point-to-point (PTP) LSP. A drawback of this approach is the inability to use RSVP-TE protection (FRR or GMPLS protection) against failure of the border LSR sitting at a core/region boundary.

3.3.2. Reducing Signaling Load using Hierarchy

When the number of nodes grows too large, the amount of RSVP-TE signaling can be reduced using the MPLS PSC hierarchy [RFC4206]. A core within the hierarchy can divide the topology into M regions of on average N/M nodes. Within a region the computational load is reduced by more than M^2. Within the core, the computational load generally becomes quite small since M is usually a fairly small number (a few tens of regions) and each region is generally attached to the core in typically only two or three places on average.

Using hierarchy improves scaling but has two consequences. First, hierarchy effectively forces the use of platform label space. When a containing LSP is rerouted, the labels assigned to the contained LSP cannot be changed but may arrive on a different interface. Second, hierarchy results in much larger LSP. These LSP today are larger than any single component link and therefore force the use of the all-ones component in link bundles.

3.3.3. Using Both LDP and RSVP-TE Hierarchy

It is also possible to use both LDP and RSVP-TE hierarchy. MPLS networks with a very large number of nodes may benefit from the use of both LDP and RSVP-TE hierarchy. The two techniques are certainly not mutually exclusive.

3.4. Reducing Forwarding State

Both LDP and MPLS hierarchy have the benefit of reducing the amount of forwarding state. Using the example from Section 3.3, and using MPLS hierarchy, the worst case generally occurs at borders with the core.

For example, consider a network with approximately 1,000 nodes divided into 10 regions. At the edges, each node requires 1,000 LSP to other edge nodes. The edge nodes also require 100 intra-region LSP. Within the core, if the core has only 3 attachments to each region the core LSR have less than 100 intra-core LSP. At the border cutset between the core and a given region, in this example there are 100 edge nodes with inter-region LSP crossing that cutset, destined to 900 other edge nodes. That yields forwarding state for on the order of 90,000 LSP at the border cutset. These same routers need only reroute well under 200 LSP when a multiple fault occurs, as long as only links are affected and a border LSR does not go down.

In the core, the forwarding state is greatly reduced. If inter-region LSP have different characteristics, it makes sense to make use of aggregates with different characteristics. Rather than exchange information about every inter-region LSP within the intra-core LSP it makes more sense to use multiple intra-core LSP between pairs of core nodes, each aggregating sets of inter-region LSP with common characteristics or common requirements.

3.5. Avoiding Route Oscillation

Networks can become unstable when a feedback loop exists such that moving traffic to a link causes a metric such as delay to increase, which then causes traffic to move elsewhere. For example, the original ARPANET routing used a delay based cost metric and proved prone to route oscillations [DBP].

Delay may be used as a constraint in routing for high priority traffic, where the movement of traffic cannot impact the delay. The safest way to measure delay is to make measurements based on traffic which is prioritized such that it is queued ahead of the traffic which will be affected. This is a reasonable measure of delay for high priority traffic for which constraints have been set which allow this type of traffic to consume only a fraction of link capacities with the remaining capacity available to lower priority traffic.

Any measurement of jitter (delay variation) that is used in route decision is likely to cause oscillation. Jitter that is caused by queuing effects and cannot be measured using a very high priority measurement traffic flow.

It may be possible to find links with constrained queuing delay or jitter using a theoretical maximum or a probability based bound on queuing delay or jitter at a given priority based on the types and amounts of traffic accepted and combining that theoretical limit with a measured delay at very high priority.

Instability can occur due to poor performance and interaction with protocol timers. In this way a computational scaling problem can become a stability problem when a network becomes sufficiently large. For this reason, [I-D.ietf-rtgwg-cl-requirement] has a number of requirements focusing on minimally impacting scalability.

4. New Challenges

New technical challenges are posed by [I-D.ietf-rtgwg-cl-requirement] in both the control plane and data plane.

Among the more difficult challenges are the following.

1. requirements related delay or jitter (see Section 4.1.1),

2. the combination of ingress control over LSP placement and retaining an ability to move traffic as demands dictate can pose challenges and such requirements can even be conflicting (see target="sect.local-control" />),

3. path symmetry requires extensions and is particularly challenging for very large LSP (see Section 4.1.3),

4. accommodating a very wide range of requirements among contained LSP can lead to inefficiency if the most stringent requirements are reflected in aggregates, or reduce scalability if a large number of aggregates are used to provide a too fine a reflection of the requirements in the contained LSP (see Section 4.1.4),

5. backwards compatibility is somewhat limited due to the need to accommodate legacy multipath interfaces which provide too little information regarding their configured default behavior, and legacy LSP which provide too little information regarding their requirements (see Section 4.1.5),

6. data plane challenges include those of accommodating very large LSP, large microflows, traffic ordering constraints imposed by a subsent of LSP, and accounting for IP and LDP traffic (see Section 4.2).

4.1. Control Plane Challenges

Some of the control plane requirements are particularly challenging. Handling large flows which aggregate smaller flows must be accomplished with minimal impact on scalability. Potentially conflicting are requirements for jitter and requirements for stability. Potentially conflicting are the requirements for ingress control of a large number of parameters, and the requirements for local control needed to achieve traffic balance across a composite link. These challenges and potential solutions are discussed in the following sections.

4.1.1. Delay and Jitter Sensitive Routing

Delay and jitter sensitive routing are called for in [I-D.ietf-rtgwg-cl-requirement] in requirements FR#2, FR#7, FR#8, FR#9, FR#15, FR#16, FR#17, FR#18. Requirement FR#17 is particularly problematic, calling for constraints on jitter.

A tradeoff exists between scaling benefits of aggregating information, and potential benefits of using a finer granularity in delay reporting. To maintain the scaling benefit, measured link delay for any given composite link SHOULD be aggregated into a small number of delay ranges. IGP-TE extensions MUST be provided which advertise the available capacities for each of the selected ranges.

For path selection of delay sensitive LSP, the ingress SHOULD bias link metrics based on available capacity and select a low cost path which meets LSP total path delay criteria. To communicate the requirements of an LSP, the ERO MUST be extended to indicate the per link constraints. To communicate the type of resource used, the RRO SHOULD be extended to carry an identification of the group that is used to carry the LSP at each link bundle hop.

4.1.2. Local Control of Traffic Distribution

Many requirements in [I-D.ietf-rtgwg-cl-requirement] suggest that a node immediately adjacent to a component link should have a high degree of control over how traffic is distributed, as long as network performance objectives are met. Particularly relevant are FR#18 and FR#19.

The requirements to allow local control are potentially in conflict with requirement FR#21 which gives full control of component link select to the LSP ingress. While supporting this capability is mandatory, use of this feature is optional per LSP.

A given network deployment will have to consider this pair of conflicting requirements and make appropriate use of local control of traffic placement and ingress control of traffic placement to best meet network requirements.

4.1.3. Path Symmetry Requirements

Requirement FR#21 in [I-D.ietf-rtgwg-cl-requirement] includes a provision to bind both directions of a bidirectional LSP to the same component. This is easily achieved if the LSP is directly signaled across a composite link. This is not as easily achieved if a set of LSP with this requirement are signaled over a large hierarchical LSP which is in turn carried over a composite link. The basis for load distribution in such as case is the label stack. The labels in either direction are completely independent.

This could be accommodated if the ingress, egress, and all midpoints of the hierarchical LSP make use of an entropy label in the distribution, and use only that entropy label. A solution for this problem may add complexity with very little benefit. There is little or no true benefit of using symmetrical paths rather than component links of identical characteristics.

Traffic symmetry and large LSP capacity are a second pair of conflicting requirements. Any given LSP can meet one of these two requirements but not both. A given network deployment will have to make appropriate use of each of these features to best meet network requirements.

4.1.4. Requirements for Contained LSP

[I-D.ietf-rtgwg-cl-requirement] calls for new LSP constraints. These constraints include frequency of load balancing rearrangement, delay and jitter, packet ordering constraints, and path symmetry.

When LSP are contained within hierarchical LSP, there is no signaling available at midpoint LSR which identifies the contained LSP let alone providing the set of requirements unique to each contained LSP. Defining extensions to provide this information would severely impact scalability and defeat the purpose of aggregating control information and forwarding information into hierarchical LSP. For the same scalability reasons, not aggregating at all is not a viable option for large networks where scalability and stability problems may occur as a result.

As pointed out in Section 4.1.3, the benefits of supporting symmetric paths among LSP contained within hierarchical LSP may not be sufficient to justify the complexity of supporting this capability.

A scalable solution which accommodates multiple sets of LSP between given pairs of LSR is to provide multiple hierarchical LSP for each given pair of LSR, each hierarchical LSP aggregating LSP with common requirements and a common pair of endpoints. This is a network design technique available to the network operator rather than a protocol extension. This technique can accommodate multiple sets of delay and jitter parameters, multiple sets of frequency of load balancing parameters, multiple sets of packet ordering constraints, etc.

4.1.5. Retaining Backwards Compatibility

Backwards compatibility and support for incremental deployment requires considering the impact of legacy LSR in the role of LSP ingress, and considering the impact of legacy LSR advertising ordinary links, advertising Ethernet LAG as ordinary links, and advertising link bundles.

Legacy LSR in the role of LSP ingress cannot signal requirements which are not supported by their control plane software. The additional capabilities supported by other LSR has no impact on these LSR. These LSR however, being unaware of extensions, may try to make use of scarce resources which support specific requirements such as low delay. To a limited extent it may be possible for a network operator to avoid this issue using existing mechanisms such as link administrative attributes and attribute affinities [RFC3209].

Legacy LSR advertising ordinary links will not advertise attributes needed by some LSP. For example, there is no way to determine the delay or jitter characteristics of such a link. Legacy LSR advertising Ethernet LAG pose additional problems. There is no way to determine that packet ordering constraints would be violated for LSP with strict packet ordering constraints, or that frequency of load balancing rearrangement constraints might be violated.

Legacy LSR advertising link bundles have no way to advertise the configured default behavior of the link bundle. Some link bundles may be configured to place each LSP on a single component link and therefore may not be able to accommodate an LSP which requires bandwidth in excess of the size of a component link. Some link bundles may be configured to spread all LSP over the all-ones component. For LSR using the all-ones component link, there is no documented procedure for correctly setting the "Maximum LSP Bandwidth". There is currently no way to indicate the largest microflow that could be supported by a link bundle using the all-ones component link.

Having received the RRO, it is possible for an ingress to look for the all-ones component to identify such link bundles after having signaled at least one LSP. Whether any LSR collects this information on legacy LSR and makes use of it to set defaults, is an implementation choice.

4.2. Data Plane Challenges

Flow identification is briefly discussed in Section 2.1. Traffic distribution is briefly discussed in Section 2.3. This section discusses issues specific to particular requirements specified in

[I-D.ietf-rtgwg-cl-requirement].

4.2.1. Very Large LSP

Very large LSP may exceed the capacity of any single component of a composite link. In some cases contained LSP may exceed the capacity of any single component. These LSP may the use of the equivalent of the all-ones component of a link bundle, or may use a subset of components which meet the LSP requirements.

Very large LSP can be accommodated as long as they can be subdivided (see Section 4.2.2). A very large LSP cannot have a requirement for symetric paths unless complex protocol extensions are proposed (see Section 2.2 and Section 4.1.3).

4.2.2. Very Large Microflows

Within a very large LSP there may be very large microflows. A very large microflow is a very large flows which cannot be further subdivided. Flows which cannot be subdivided must be no larger that the capacity of any single component.

Current signaling provides no way to specify the largest microflow that a can be supported on a given link bundle in routing advertisements. Extensions which address this are discussed in Section 6.4. Absent extensions of this type, traffic containing microflows that are too large for a given composite link may be present. There is no data plane solution for this problem that would not require reordering traffic at the composite link egress.

Some techniques are susceptible to statistical collisions where an algorithm to distribute traffic is unable to disambiguate traffic among two or more very large microflow where their sum is in excess of the capacity of any single component. Hash based algorithms which use too small a hash space are particularly susceptible and require a change in hash seed in the event that this were to occur. A change in hash seed is highly disruptive, causing traffic reordering among all traffic flows over which the hash function is applied.

4.2.3. Traffic Ordering Constraints

Some LSP have strict traffic ordering constraints. Most notable among these are MPLS-TP LSP. In the absence of aggregation into hierarchical LSP, those LSP with strict traffic ordering constraints can be placed on individual component links if there is a means of identifying which LSP have such a constraint. If LSP with strict traffic ordering constraints are aggregated in hierarchical LSP, the hierarchical LSP capacity may exceed the capacity of any single component link. In such a case the load balancing for the containing may be constrained to look only at the top label and the first contained label. This and related issues are discussed further in Section 6.4.

4.2.4. Accounting for IP and LDP Traffic

Networks which carry RSVP-TE signaled MPLS traffic generally carry low volumes of native IP traffic, often only carrying control traffic as native IP. There is no architectural guarantee of this, it is just how network operators have made use of the protocols.

[I-D.ietf-rtgwg-cl-requirement] requires that native IP and native LDP be accommodated. In some networks, a subset of services may be carried as native IP or carried as native LDP. Today this may be accommodated by the network operator estimating the contribution of IP and LDP and configuring a lower set of available bandwidth figures on the RSVP-TE advertisements.

The only improvement that Composite Link can offer is that of measuring the IP and LDP traffic levels and automatically reducing the available bandwidth figures on the RSVP-TE advertisements. The measurements would have to be significantly filtered. This is similar to a feature in existing LSR, commonly known as "autobandwidth" with a key difference. In the "autobandwidth" feature, the bandwidth request of an RSVP-TE signaled LSP is adjusted in response to traffic measurements. In this case the IP or LDP traffic measurements are used to reduce the link bandwidth directly, without first encapsulating in an RSVP-TE LSP.

This may be a subtle and perhaps even a meaningless distinction if Composite Link is used to form a Sub-Path Maintenance Element (SPME). A SPME is in practice essentially an unsignaled single hop LSP with PHP enabled [RFC5921]. A Composite Link SPME looks very much like classic multipath, where there is no signaling, only management plane configuration creating the multipath entity (of which Ethernet Link Aggregation is a subset).

4.2.5. IP and LDP Limitations

IP does not offer traffic engineering. LDP cannot be extended to offer traffic engineering [RFC3468]. Therefore there is no traffic engineered fallback to an alternate path for IP and LDP traffic if resources are not adequate for the IP and/or LDP traffic alone on a given link in the primary path. The only option for IP and LDP would be to declare the link down. Declaring a link down due to resource exhaustion would reduce traffic to zero and eliminate the resource exhaustion. This would cause oscillations and is therefore not a viable solution.

Congestion caused by IP or LDP traffic loads is a pathologic case that can occur if IP and/or LDP are carried natively and there is a high volume of IP or LDP traffic. This situation can be avoided by carrying IP and LDP within RSVP-TE LSP.

It is also not possible to route LDP traffic differently for different FEC. LDP traffic engineering is specifically disallowed by [RFC3468]. It may be possible to support multi-topology IGP extensions to accommodate more than one set of criteria. If so, the additional IGP could be bound to the forwarding criteria, and the LDP FEC bound to a specific IGP instance, inheriting the forwarding criteria. Alternately, one IGP instance can be used and the LDP SPF can make use of the constraints, such as delay and jitter, for a given LDP FEC. [Note: WG needs to discuss this and decide first whether to solve this at all and then if so, how.]

5. Existing Mechanisms

In MPLS the one mechanisms which support explicit signaling of multiple parallel links is Link Bundling [RFC4201]. The set of techniques known as "classis multipath" support no explicit signaling, except in two cases. In Ethernet Link Aggregation the Link Aggregation Control Protocol (LACP) coordinates the addition or removal of members from an Ethernet Link Aggregation Group (LAG). The use of the "all-ones" component of a link bundle indicates use of classis multipath, however the ability to determine if a link bundle makes use of classis multipath is not yet supported.

5.1. Link Bundling

Link bundling supports advertisement of a set of homogenous links as a single route advertisement. Link bundling supports placement of an LSP on any single component link, or supports placement of an LSP on the all-ones component link. Not all link bundling implementations support the all-ones component link. There is no way for an ingress LSR to tell which potential midpoint LSR support this feature and use it by default and which do not. Based on [RFC4201] it is unclear how to advertise a link bundle for which the all-ones component link is available and used by default. Common practice is to violate the specification and set the Maximum LSP Bandwidth to the Available Bandwidth. There is no means to determine the largest microflow that could be supported by a link bundle that is using the all-ones component link.

[RFC6107] extends the procedures for hierarchical LSP but also extends link bundles. An LSP can be explicitly signaled to indicate that it is an LSP to be used as a component of a link bundle. Prior to that the common practice was to simply not advertise the component link LSP into the IGP, since only the ingress and egress of the link bundle needed to be aware of their existence, which they would be aware of due to the RSVP-TE signaling used in setting up the component LSP.

While link bundling can be the basis for composite links, a significant number of small extension needs to be added.

1. To support link bundles of heterogeneous links, a means of advertising the capacity available within a group of homogeneous needs to be provided.

2. Attributes need to be defined to support the following parameters for the link bundle or for a group of homogeneous links.

   A. delay range

   B. jitter (delay variation) range

   C. group metric

   D. all-ones component capable

   E. capable of dynamically balancing load

   F. largest supportable microflow

   G. abilities to support strict packet ordering requirements within contained LSP

3. For each of the prior extended attributes, the constraint based routing path selection needs to be extended to reflect new constraints based on the extended attributes.

4. For each of the prior extended attributes, LSP admission control needs to be extended to reflect new constraints based on the extended attributes.

5. Dynamic load balance must be provided for flows within a given set of links with common attributes such that NPO are not violated including frequency of load balance adjustment for any given flow.

5.2. Classic Multipath

Classic multipath is defined in [I-D.symmvo-rtgwg-cl-use-cases].

Classic multipath refers to the most common current practice in implementation and deployment of multipath. The most common current practice makes use of a hash on the MPLS label stack and if IPv4 or IPv6 are indicated under the label stack, makes use of the IP source and destination addresses [RFC4385] [RFC4928].

Classic multipath provides a highly scalable means of load balancing. Adaptive multipath has proven value in assuring an even loading on component link and an ability to adapt to change in offerred load that occurs over periods of hundreds of milliseconds or more. Classic multipath scalability is due to the ability to effectively work with an extremely large number of flows (IP host pairs) using relatively little resources (a data structure accessed using a hash result as a key or using ranges of hash results).

Classic multipath meets a small subset of Composite Link requirements. Due to scalability of the approach, classic multipath seems to be an excellent candidate for extension to meet the full set of Composite Link forwarding requirements.

Additional detail can be found in [I-D.symmvo-rtgwg-cl-use-cases].

6. Mechanisms Proposed in Other Documents

A number of documents which at the time of writing are works in progress address parts of the requirements of Composite Link, or assist in making some of the goals achievable.

6.1. Loss and Delay Measurement

Procedures for measuring loss and delay are provided in [RFC6374]. These are OAM based measurements. This work could be the basis of delay measurements and delay variation measurement used for metrics called for in [I-D.ietf-rtgwg-cl-requirement].

Currently there are two additional Internet-Drafts that address delay and delay variation metrics.

draft-wang-ccamp-latency-te-metric
: [I-D.wang-ccamp-latency-te-metric] is designed specifically to meet this requirement. OSPF-TE and ISIS-TE extensions are defined to indicate link delay and delay variance. The RSVP-TE ERO is extended to include service level requirements. A latency

accumulation object is defined to provide a means of verification of the service level requirements. This draft is intended to proceed in the CCAMP WG. It is currently and individual submission. The 03 version of this draft expired in September 2012.

draft-giacalone-ospf-te-express-path
This document proposes to extend OSPF-TE only. Extensions support delay, delay variance, loss, residual bandwidth, and available bandwidth. No extensions to RSVP-TE are proposed. This draft is intended to proceed in the CCAMP WG. It is currently and individual submission. The 02 version will expire in March 2012.

A possible course of action may be to combine these two drafts. The delay variance, loss, residual bandwidth, and available bandwidth extensions are particular prone to network instability. The question as to whether queuing delay and delay variation should be considered, and if so for which diffserv Per-Hop Service Class (PSC) is not addressed.

Note to co-authors: The ccamp-latency-te-metric draft refers to [I-D.ietf-rtgwg-cl-requirement] and is well matched to those requirements, including stability. The ospf-te-express-path draft refers to the "Alto Protocol" (draft-ietf-alto-protocol) and therefore may not be intended for RSVP-TE use. The authors of the two drafts may be able to resolve this. It may be best to drop ospf-te-express-path from this framework document.

6.2. Link Bundle Extensions

A set of link bundling extensions are defined in [I-D.ietf-mpls-explicit-resource-control-bundle]. This document provides extensions to the ERO and RRO to explicitly control the labels and resources within a bundle used by an LSP.

The extensions in this document could be further extended to support indicating a group of component links in the ERO or RRO, where the group is given an interface identification like the bundle itself. The extensions could also be further extended to support specification of the all-ones component link in the ERO or RRO.

[I-D.ietf-mpls-explicit-resource-control-bundle] does not provide a means to advertise the link bundle components. It is not certain how the ingress LSR would determine the set of link bundle component links available for a given link bundle.

[I-D.ospf-cc-stlv] provides a baseline draft for extending link bundling to advertise components. A new component TVL (C-TLV) is proposed, which must reference a Composite Link Link TLV. [I-D.ospf-cc-stlv] is intended for the OSPF WG and submitted for the "Experimental" track. The 00 version expired in February 2012.

6.3. Fat PW and Entropy Labels

Two documents provide a means to add entropy for the purpose of improving load balance. MPLS encapsulation can bury information that is needed to identify microflows. These two documents allow a pseudowire ingress and LSP ingress respectively to add a label solely for the purpose of providing a finer granularity of microflow groups.

[RFC6391] allows pseudowires which carry a large volume of traffic, where microflows can be identified to be load balanced across multiple members of an Ethernet LAG or an MPLS link bundle. This is accomplished by adding a flow label below the pseudowire label in the MPLS label stack. For this to be effective the link bundle load balance must make use of the label stack up to and including this flow label.

[I-D.ietf-mpls-entropy-label] provides a means for a LER to put an additional label known as an entropy label on the MPLS label stack. As defined, only the LER can add the entropy label.

Core LSR acting as LER for aggregated LSP can add entropy labels based on deep packet inspection and place an entropy label indicator (ELI) and entropy label (EL) just below the label being acted on. This would be helpful in situations where the label stack depth to which load distribution can operate is limited by implementation or is limited for other reasons such as carrying both MPLS-TP and MPLS with entropy labels within the same hierarchical LSP.

6.4. Multipath Extensions

The multipath extensions drafts address one aspect of Composite Link. These drafts deal with the issue of accommodating LSP which have strict packet ordering constraints in a network containing multipath. MPLS-TP has become the one important instance of LSP with strict packet ordering constraints and has driven this work.

[I-D.villamizar-mpls-tp-multipath] outlines requirements and gives a number of options for dealing with the apparent incompatibility of MPLS-TP and multipath. A preferred option is described.

[I-D.villamizar-mpls-tp-multipath-te-extn] provides protocol extensions needed to implement the preferred option described in [I-D.villamizar-mpls-tp-multipath].

Other issues pertaining to multipath are also addressed. Means to advertise the largest microflow supportable are defined. Means to indicate the largest expected microflow within an LSP are defined. Issues related to hierarchy are addressed.

7. Required Protocol Extensions and Mechanisms

Prior sections have reviewed key characteristics, architecture tradeoffs, new challenges, existing mechanisms, and relevant mechanisms proposed in existing new documents.

This section first summarizes and groups requirements. A set of documents coverage groupings are proposed with existing works-in-progress noted where applicable. The set of extensions are then grouped by protocol affected as a convenience to implementors.

7.1. Brief Review of Requirements

The following list provides a categorization of requirements specified in [I-D.ietf-rtgwg-cl-requirement] along with a short phrase indication what topic the requirement covers.

routing information aggregation
: FR#1 (routing summarization), FR#20 (composite link may be a component of another composite link)

restoration speed
: FR#2 (restoration speed meeting NPO), FR#12 (minimally disruptive load rebalance), DR#6 (fast convergence), DR#7 (fast worst case failure convergence)

load distribution, stability, minimal disruption
: FR#3 (automatic load distribution), FR#5 (must not oscillate), FR#11 (dynamic placement of flows), FR#12 (minimally disruptive load rebalance), FR#13 (bounded rearrangement frequency), FR#18 (flow placement must satisfy NPO), FR#19 (flow identification finer than per top level LSP), MR#6 (operator initiated flow rebalance)

backward compatibility and migration
: FR#4 (smooth incremental deployment), FR#6 (management and diagnostics must continue to function), DR#1 (extend existing protocols), DR#2 (extend LDP, no LDP TE)

delay and delay variation
FR#7 (expose lower layer measured delay), FR#8 (precision of latency reporting), FR#9 (limit latency on per LSP basis), FR#15 (minimum delay path), FR#16 (bounded delay path), FR#17 (bounded jitter path)

admission control, preemption, traffic engineering
FR#10 (admission control, preemption), FR#14 (packet ordering), FR#21 (ingress specification of path), FR#22 (path symmetry), DR#3 (IP and LDP traffic), MR#3 (management specification of path)

single vs multiple domain
DR#4 (IGP extensions allowed within single domain), DR#5 (IGP extensions disallowed in multiple domain case)

general network management
MR#1 (polling, configuration, and notification), MR#2 (activation and de-activation)

path determination, connectivity verification
MR#4 (path trace), MR#5 (connectivity verification)

The above list is not intended as a substitute for [I-D.ietf-rtgwg-cl-requirement], but rather as a concise grouping and reminder or requirements to serve as a means of more easily determining requirements coverage of a set of protocol documents.

7.2. Required Document Coverage

The primary areas where additional protocol extensions and mechanisms are required include the topics described in the following subsections.

There are candidate documents for a subset of the topics below. This grouping of topics does not require that each topic be addressed by a separate document. In some cases, a document may cover multiple topics, or a specific topic may be addressed as applicable in multiple documents.

7.2.1. Component Link Grouping

An extension to link bundling is needed to specify a group of components with common attributes. This can be a TLV defined within the link bundle that carries the same encapsulations as the link bundle. Two interface indices would be needed for each group.

a. An index is needed that if included in an ERO would indicate the need to place the LSP on any one component within the group.

b. A second index is needed that if included in an ERO would indicate the need to balance flows within the LSP across all components of the group. This is equivalent to the "all-ones" component for the entire bundle.

[I-D.ospf-cc-stlv] can be extended to include multipath treatment capabilities. An ISIS solution is also needed. An extension of RSVP-TE signaling is needed to indicate multipath treatment preferences.

If a component group is allowed to support all of the parameters of a link bundle, then a group TE metric would be accommodated. This can be supported with the component TLV (C-TLV) defined in [I-D.ospf-cc-stlv].

The primary focus of this document, among the sets of requirements listed in Section 7.1 is the "routing information aggregation" set of requirements. The "restoration speed", "backward compatibility and migration", and "general network management" requirements must also be considered.

7.2.2. Delay and Jitter Extensions

A extension is needed in the IGP-TE advertisement to support delay and delay variation for links, link bundles, and forwarding adjacencies. Whatever mechanism is described must take precautions that insure that route oscillations cannot occur. [I-D.wang-ccamp-latency-te-metric] may be a good starting point.

The primary focus of this document, among the sets of requirements listed in Section 7.1 is the "delay and delay variation" set of requirements. The "restoration speed", "backward compatibility and migration", and "general network management" requirements must also be considered.

7.2.3. Path Selection and Admission Control

Path selection and admission control changes must be documented in each document that proposes a protocol extension that advertises a new capability or parameter that must be supported by changes in path selection and admission control.

The primary focus of this document, among the sets of requirements listed in Section 7.1 are the "load distribution, stability, minimal disruption" and "admission control, preemption, traffic engineering" sets of requirements. The "restoration speed" and "path determination, connectivity verification" requirements must also be considered. The "backward compatibility and migration", and "general network management" requirements must also be considered.

7.2.4. Dynamic Multipath Balance

FR#11 explicitly calls for dynamic load balancing similar to existing adaptive multipath. In implementations where flow identification uses a coarse granularity, the adjustments would have to be equally coarse, in the worst case moving entire LSP. The impact of flow identification granularity and potential adaptive multipath approaches may need to be documented in greater detail than provided here.

The primary focus of this document, among the sets of requirements listed in Section 7.1 are the "restoration speed" and the "load distribution, stability, minimal disruption" sets of requirements. The "path determination, connectivity verification" requirements must also be considered. The "backward compatibility and migration", and "general network management" requirements must also be considered.

7.2.5. Frequency of Load Balance

IGP-TE and RSVP-TE extensions are needed to support frequency of load balancing rearrangement called for in FR#13, and FR#15-FR#17. Constraints are not defined in RSVP-TE, but could be modeled after administrative attribute affinities in RFC3209 and elsewhere.

The primary focus of this document, among the sets of requirements listed in Section 7.1 is the "load distribution, stability, minimal disruption" set of requirements. The "path determination, connectivity verification" must also be considered. The "backward compatibility and migration" and "general network management" requirements must also be considered.

7.2.6. Inter-Layer Communication

Lower layer to upper layer communication called for in FR#7 and FR#20. This is addressed for a subset of parameters related to packet ordering in [I-D.villamizar-mpls-tp-multipath] where layers are MPLS. Remaining parameters, specifically delay and delay variation, need to be addressed. Passing information from a lower non-MPLS layer to an MPLS layer needs to be addressed, though this may largely be generic advice encouraging a coupling of MPLS to lower layer management plane or control plane interfaces. This topic can be addressed in each document proposing a protocol extension, where applicable.

The primary focus of this document, among the sets of requirements listed in Section 7.1 is the "restoration speed" set of requirements. The "backward compatibility and migration" and "general network management" requirements must also be considered.

7.2.7. Packet Ordering Requirements

A document is needed to define extensions supporting various packet ordering requirements, ranging from requirements to preservce microflow ordering only, to requirements to preservce full LSP ordering (as in MPLS-TP). This is covered by [I-D.villamizar-mpls-tp-multipath] and [I-D.villamizar-mpls-tp-multipath-te-extn].

The primary focus of this document, among the sets of requirements listed in Section 7.1 are the "admission control, preemption, traffic engineering" and the "path determination, connectivity verification" sets of requirements. The "backward compatibility and migration" and "general network management" requirements must also be considered.

7.2.8. Minimally Disruption Load Balance

The behavior of hash methods used in classic multipath needs to be described in terms of FR#12 which calls for minimally disruptive load adjustments. For example, reseeding the hash violates FR#12. Using modulo operations is significantly disruptive if a link comes or goes down, as pointed out in [RFC2992]. In addition, backwards compatibility with older hardware needs to be accommodated.

The primary focus of this document, among the sets of requirements listed in Section 7.1 is the "load distribution, stability, minimal disruption" set of requirements.

7.2.9. Path Symmetry

Protocol extensions are needed to support dynamic load balance as called for to meet FR#22 (path symmetry) and to meet FR#11 (dynamic placement of flows). Currently path symmetry can only be supported in link bundling if the path is pinned. When a flow is moved both ingress and egress must make the move as close to simultaneously as possible to satisfy FR#22 and FR#12 (minimally disruptive load rebalance). If a group of flows are identified using a hash, then the hash must be identical on the pair of LSR at the endpoint, using the same hash seed and with one side swapping source and destination. If the label stack is used, then either the entire label stack must be a special case flow identification, since the set of labels in either direction are not correlated, or the two LSR must conspire to use the same flow identifier. For example, using a common entropy label value, and using only the entropy label in the flow identification would satisfy this requirement.

The primary focus of this document, among the sets of requirements listed in Section 7.1 are the "load distribution, stability, minimal disruption" and the "admission control, preemption, traffic engineering" sets of requirements. The "backward compatibility and migration" and "general network management" requirements must also be considered. Path symetry simplifies support for the "path determination, connectivity verification" set of requirements, but with significant complexity added elsewhere.

7.2.10. Performance, Scalability, and Stability

A separate document providing analysis of performance, scalability, and stability impacts of changes may be needed. The topic of traffic adjustment oscillation must also be covered. If sufficient coverage is provided in each document covering a protocol extension, a separate document would not be needed.

The primary focus of this document, among the sets of requirements listed in Section 7.1 is the "restoration speed" set of requirements. This is not a simple topic and not a topic that is well served by scattering it over multiple documents, therefore it may be best to put this in a separate document and put citations in documents called for in Section 7.2.1, Section 7.2.2, Section 7.2.3, Section 7.2.9, Section 7.2.11, Section 7.2.12, Section 7.2.13, and Section 7.2.14. Citation may also be helpful in Section 7.2.4, and Section 7.2.5.

7.2.11. IP and LDP Traffic

A document is needed to define the use of measurements native IP and native LDP traffic levels to reduce link advertised bandwidth amounts.

The primary focus of this document, among the sets of requirements listed in Section 7.1 are the "load distribution, stability, minimal disruption" and the "admission control, preemption, traffic engineering" set of requirements. The "path determination, connectivity verification" must also be considered. The "backward compatibility and migration" and "general network management" requirements must also be considered.

7.2.12. LDP Extensions

Extending LDP is called for in DR#2. LDP can be extended to couple FEC admission control to local resource availability without providing LDP traffic engineering capability. Other LDP extensions such as signaling a bound on microflow size and LDP LSP requirements would provide useful information without providing LDP traffic engineering capability.

The primary focus of this document, among the sets of requirements listed in Section 7.1 is the "admission control, preemption, traffic engineering" set of requirements. The "backward compatibility and migration" and "general network management" requirements must also be considered.

7.2.13. Pseudowire Extensions

PW extensions such as signaling a bound on microflow size and PW requirements would provide useful information.

The primary focus of this document, among the sets of requirements listed in Section 7.1 is the "admission control, preemption, traffic engineering" set of requirements. The "backward compatibility and migration" and "general network management" requirements must also be considered.

7.2.14. Multi-Domain Composite Link

DR#5 calls for Composite Link to span multiple network topologies. Component LSP may already span multiple network topologies, though most often in practice these are LDP signaled. Component LSP which are RSVP-TE signaled may also span multiple network topologies using at least three existing methods (per domain [RFC5152], BRPC [RFC5441], PCE [RFC4655]). When such component links are combined in a Composite Link, the Composite Link spans multiple network topologies. It is not clear in which document this needs to be described or whether this description in the framework is sufficient. The authors and/or the WG may need to discuss this. DR#5 mandates that IGP-TE extension cannot be used. This would disallow the use of [RFC5316] or [RFC5392] in conjunction with [RFC5151].

The primary focus of this document, among the sets of requirements listed in Section 7.1 are "single vs multiple domain" and "admission control, preemption, traffic engineering". The "routing information aggregation" and "load distribution, stability, minimal disruption" requirements need attention due to their use of the IGP in single domain Composite Link. Other requirements such as "delay and delay variation", can more easily be accomodated by carrying metrics within BGP. The "path determination, connectivity verification" requirements need attention due to requirements to restrict disclosure of topology information across domains in multi-domain deployments. The "backward compatibility and migration" and "general network management" requirements must also be considered.

7.3. Open Issues Regarding Requirements

Note to co-authors: This section needs to be reduced to an empty section and then removed.

The following topics in the requirements document are not addressed. Since they are explicitly mentioned in the requirements document some mention of how they are supported is needed, even if to say nother needed to be done. If we conclude any particular topic is irrelevant, maybe the topic should be removed from the requirement document. At that point we could add the management requirements that have come up and were missed.

1. L3VPN RFC 4364, RFC 4797,L2VPN RFC 4664, VPWS, VPLS RFC 4761, RFC 4762 and VPMS VPMS Framework (draft-ietf-l2vpn-vpms-frmwk-requirements). It is not clear what additional Composite Link requirements these references imply, if any. If no additional requirements are implied, then these references are considered to be informational only.

2. Migration may not be adequately covered in Section 4.1.5. It might also be necessary to say more here on performance, scalability, and stability as it related to migration. Comments on this from co-authors or the WG?

3. We may need a performance section in this document to specifically address #DR6 (fast convergence), and #DR7 (fast worst case failure convergence), though we do already have scalability discussion. The performance section would have to say "no worse than before, except were there was no alternative to make it very slightly worse" (in a bit more detail than that). It would also have to better define the nature of the performance criteria.

7.4. Framework Requirement Coverage by Protocol

As an aid to implementors, this section summarizes requirement coverage listed in Section 7.2 by protocol or LSR functionality affected.

Some documentation may be purely informational, proposing no changes and proposing usage at most. This includes Section 7.2.3, Section 7.2.8, Section 7.2.10, and Section 7.2.14.

Section 7.2.9 may require a new protocol.

7.4.1. OSPF-TE and ISIS-TE Protocol Extensions

Many of the changes listed in Section 7.2 require IGP-TE changes, though most are small extensions to provide additional information. This set includes Section 7.2.1, Section 7.2.2, Section 7.2.5, Section 7.2.6, and Section 7.2.7. An adjustment to existing advertised parameters is suggested in Section 7.2.11.

7.4.2. PW Protocol Extensions

The only suggestion of pseudowire (PW) extensions is in Section 7.2.13.

7.4.3. LDP Protocol Extensions

Potential LDP extensions are described in Section 7.2.12.

7.4.4. RSVP-TE Protocol Extensions

RSVP-TE protocol extensions are called for in Section 7.2.1, Section 7.2.5, Section 7.2.7, and Section 7.2.9.

7.4.5. RSVP-TE Path Selection Changes

Section 7.2.3 calls for path selection to be addressed in individual documents that require change. These changes would include those proposed in Section 7.2.1, Section 7.2.2, Section 7.2.5, and Section 7.2.7.

7.4.6. RSVP-TE Admission Control and Preemption

When a change is needed to path selection, a corresponding change is needed in admission control. The same set of sections applies: Section 7.2.1, Section 7.2.2, Section 7.2.5, and Section 7.2.7. Some resource changes such as a link delay change might trigger preemption. The rules of preemption remain unchanged, still based on holding priority.

7.4.7. Flow Identification and Traffic Balance

The following describe either the state of the art in flow identification and traffic balance or propose changes: Section 7.2.4, Section 7.2.5, Section 7.2.7, and Section 7.2.8.

8. Security Considerations

The security considerations for MPLS/GMPLS and for MPLS-TP are documented in [RFC5920] and [I-D.ietf-mpls-tp-security-framework].

The types protocol extensions proposed in this framework document provide additional information about links, forwarding adjacencies, and LSP requirements. The protocol semantics changes described in this framework document propose additional LSP constraints applied at path computation time and at LSP admission at midpoints LSR. The additional information and constraints provide no additional security considerations beyond the security considerations already documented in [RFC5920] and [I-D.ietf-mpls-tp-security-framework].

9. Acknowledgments

Authors would like to thank Adrian Farrel, Fred Jounay, Yuji Kamite for his extensive comments and suggestions regarding early versions of this document, Ron Bonica, Nabil Bitar, Eric Gray, Lou Berger, and Kireeti Kompella for their reviews of early versions and great suggestions.

Authors would like to thank Iftekhar Hussain for review and suggestions regarding recent versions of this document.

In the interest of full disclosure of affiliation and in the interest of acknowledging sponsorship, past affiliations of authors are noted. Much of the work done by Ning So occurred while Ning was at Verizon. Much of the work done by Curtis Villamizar occurred while at Infinera. Infinera continues to sponsor this work on a consulting basis.

10. References

10.1. Normative References

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

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

[RFC3630] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", RFC 3630, September 2003.

[RFC4201] Kompella, K., Rekhter, Y., and L. Berger, "Link Bundling in MPLS Traffic Engineering (TE)", RFC 4201, October 2005.

[RFC4206] Kompella, K. and Y. Rekhter, "Label Switched Paths (LSP) Hierarchy with Generalized Multi-Protocol Label Switching (GMPLS) Traffic Engineering (TE)", RFC 4206, October 2005.

[RFC5036] Andersson, L., Minei, I., and B. Thomas, "LDP Specification", RFC 5036, October 2007.

[RFC5305] Li, T. and H. Smit, "IS-IS Extensions for Traffic Engineering", RFC 5305, October 2008.

[RFC5712] Meyer, M. and JP. Vasseur, "MPLS Traffic Engineering Soft Preemption", RFC 5712, January 2010.

[RFC6107] Shiomoto, K. and A. Farrel, "Procedures for Dynamically Signaled Hierarchical Label Switched Paths", RFC 6107, February 2011.

[RFC6374] Frost, D. and S. Bryant, "Packet Loss and Delay Measurement for MPLS Networks", RFC 6374, September 2011.

[RFC6391] Bryant, S., Filsfils, C., Drafz, U., Kompella, V., Regan, J., and S. Amante, "Flow-Aware Transport of Pseudowires over an MPLS Packet Switched Network", RFC 6391, November 2011.

10.2. Informative References

[DBP] Bertsekas, D., "Dynamic Behavior of Shortest Path Routing Algorithms for Communication Networks", IEEE Trans. Auto. Control 1982.

[I-D.ietf-mpls-entropy-label] Drake, J., Kompella, K., Yong, L., Amante, S., and W. Henderickx, "The Use of Entropy Labels in MPLS Forwarding", draft-ietf-mpls-entropy-label-01 (work in progress), October 2011.

[I-D.ietf-mpls-explicit-resource-control-bundle] Zamfir, A., Ali, Z., and P. Dimitri, "Component Link Recording and Resource Control for TE Links", draft-ietf-mpls-explicit-resource-control-bundle-10 (work in progress), April 2011.

[I-D.ietf-mpls-tp-security-framework] Niven-Jenkins, B., Fang, L., Graveman, R., and S. Mansfield, "MPLS-TP Security Framework", draft-ietf-mpls-tp-security-framework-02 (work in progress), October 2011.

[I-D.ietf-rtgwg-cl-requirement]
Malis, A., Villamizar, C., McDysan, D., Yong, L., and N. So, "Requirements for MPLS Over a Composite Link", draft-ietf-rtgwg-cl-requirement-05 (work in progress), January 2012.

[I-D.kompella-mpls-rsvp-ecmp]
Kompella, K., "Multi-path Label Switched Paths Signaled Using RSVP-TE", draft-kompella-mpls-rsvp-ecmp-01 (work in progress), October 2011.

[I-D.ospf-cc-stlv]
Osborne, E., "Component and Composite Link Membership in OSPF", draft-ospf-cc-stlv-00 (work in progress), August 2011.

[I-D.symmvo-rtgwg-cl-use-cases]
Malis, A., Villamizar, C., McDysan, D., Yong, L., and N. So, "Composite Link USe Cases and Design Considerations", draft-symmvo-rtgwg-cl-use-cases-00 (work in progress), February 2012.

[I-D.villamizar-mpls-tp-multipath]
Villamizar, C., "Use of Multipath with MPLS-TP and MPLS", draft-villamizar-mpls-tp-multipath-01 (work in progress), March 2011.

[I-D.villamizar-mpls-tp-multipath-te-extn]
Villamizar, C., "Multipath Extensions for MPLS Traffic Engineering", draft-villamizar-mpls-tp-multipath-te-extn-00 (work in progress), July 2011.

[I-D.wang-ccamp-latency-te-metric]
Fu, X., Betts, M., Wang, Q., McDysan, D., and A. Malis, "GMPLS extensions to communicate latency as a traffic engineering performance metric", draft-wang-ccamp-latency-te-metric-03 (work in progress), March 2011.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

[RFC2991] Thaler, D. and C. Hopps, "Multipath Issues in Unicast and Multicast Next-Hop Selection", RFC 2991, November 2000.

[RFC2992] Hopps, C., "Analysis of an Equal-Cost Multi-Path

Algorithm", RFC 2992, November 2000.

[RFC3260] Grossman, D., "New Terminology and Clarifications for Diffserv", RFC 3260, April 2002.

[RFC3468] Andersson, L. and G. Swallow, "The Multiprotocol Label Switching (MPLS) Working Group decision on MPLS signaling protocols", RFC 3468, February 2003.

[RFC3945] Mannie, E., "Generalized Multi-Protocol Label Switching (GMPLS) Architecture", RFC 3945, October 2004.

[RFC3985] Bryant, S. and P. Pate, "Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture", RFC 3985, March 2005.

[RFC4385] Bryant, S., Swallow, G., Martini, L., and D. McPherson, "Pseudowire Emulation Edge-to-Edge (PWE3) Control Word for Use over an MPLS PSN", RFC 4385, February 2006.

[RFC4655] Farrel, A., Vasseur, J., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, August 2006.

[RFC4928] Swallow, G., Bryant, S., and L. Andersson, "Avoiding Equal Cost Multipath Treatment in MPLS Networks", BCP 128, RFC 4928, June 2007.

[RFC5151] Farrel, A., Ayyangar, A., and JP. Vasseur, "Inter-Domain MPLS and GMPLS Traffic Engineering -- Resource Reservation Protocol-Traffic Engineering (RSVP-TE) Extensions", RFC 5151, February 2008.

[RFC5152] Vasseur, JP., Ayyangar, A., and R. Zhang, "A Per-Domain Path Computation Method for Establishing Inter-Domain Traffic Engineering (TE) Label Switched Paths (LSPs)", RFC 5152, February 2008.

[RFC5316] Chen, M., Zhang, R., and X. Duan, "ISIS Extensions in Support of Inter-Autonomous System (AS) MPLS and GMPLS Traffic Engineering", RFC 5316, December 2008.

[RFC5392] Chen, M., Zhang, R., and X. Duan, "OSPF Extensions in Support of Inter-Autonomous System (AS) MPLS and GMPLS Traffic Engineering", RFC 5392, January 2009.

[RFC5441] Vasseur, JP., Zhang, R., Bitar, N., and JL. Le Roux, "A Backward-Recursive PCE-Based Computation (BRPC) Procedure to Compute Shortest Constrained Inter-Domain Traffic Engineering Label Switched Paths", RFC 5441, April 2009.

[RFC5920] Fang, L., "Security Framework for MPLS and GMPLS Networks", RFC 5920, July 2010.

[RFC5921] Bocci, M., Bryant, S., Frost, D., Levrau, L., and L. Berger, "A Framework for MPLS in Transport Networks", RFC 5921, July 2010.

Authors' Addresses

So Ning
Tata Communications

Email: ning.so@tatacommunications.com

Dave McDysan
Verizon
22001 Loudoun County PKWY
Ashburn, VA 20147

Email: dave.mcdysan@verizon.com

Eric Osborne
Cisco

Email: eosborne@cisco.com

Lucy Yong
Huawei USA
5340 Legacy Dr.
Plano, TX 75025

Phone: +1 469-277-5837
Email: lucy.yong@huawei.com

Curtis Villamizar
Outer Cape Cod Network Consulting

Email: curtis@occnc.com

Network Working Group Q. Zhao
Internet-Draft E. Chen
Intended status: Standards Track Huawei Technology
Expires: April 26, 2012 October 24, 2011

Protection Mechanisms for Label Distribution Protocol P2MP/MP2MP Label Switched Paths
draft-zhao-mpls-mldp-protections-00.txt

Abstract

Service providers continue to deploy real-time multicast applications using Multicast LDP (mLDP) across MPLS networks. There is a clear need to protect these real-time applications and to provide the shortest switching times in the event of failure. This document outlines the requirements, describes the protection mechanisms available, and where neccessary proposes extensions to facilitate mLDP P2MP and MP2MP LSP protection within an MPLS network.

Table of Contents

1. Terminology

For a clear narrative, this section gives a general conceptional overview of the terms.

- PLR: The node where the traffic is logically redirected onto the preset backup path is called Point of Local Repair.
- MP: The node where the backup path merges with the primary path is called Merge Point.
- FD: The node that detects the failure on primary path, and then triggers the action(s) for traffic protection is called Failure Detector. Either traffic sender or receiver can be the FD, depending on which protection mode are deployed. More details are described in later sections of this document.
- SP: The node where the traffic is physically switched/duplicated onto the backup path is called Switchover Point. In multicast cases, PLR and SP can be two different nodes. More details are described in later sections of this document.

2. Requirement 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].

3. Introduction

In order to meet user demands, operators and service providers continue to deploy multicast applications using mLDP across MPLS networks. In certain scenarios, traditional IGP-mLDP convergence mechanisms fail to meet protection switching times required to minimise, or negate entirely, application interruptions for real-time applications, including stock trading, on-line games, and multimedia teleconferencing.

Current best practice for protecting services, and higher applications includes the pre-computation and establishment of a backup path, this can decrease the convergence time efficiently. Once a failure has been detected on the primary path, the traffic should be transmitted across the back-up path.

However, two major challenges exist with the aforementioned solution. The first is how to build an absolutely disjointed backup path for each node in a multicast tree; the second is how to balance between convergence time and resource consumption.

This document provides several ways to setup the backup path for mLDP LSP, including local protection, territorial protection, and end-to-end protection. The goal is to build a reliable umbrella to against traffic black hole. How to detect failure is outside the scope of this document.

More and more users are apt to deploy multicast applications on MPLS mLDP network. In some scenarios, traditional IGP-mLDP convergence is hard to meet the requirements of those real-time applications, such as stock business, on-line game, and multimedia teleconference.

The industry has reached a consensus that setting up a backup path previously can decrease the convergence time efficiently. No matter how the above-mentioned backup path was established, once the failure is detected, the traffic should be transmitted at that path as soon as possible. Even so, there are still two major challenges left for us, one is how to build an absolutely disjointed backup path for each node in a multicast tree; the other is how to balance between convergence time and resource consumption.

It is getting urgent to find the ideal protection mechanism(s) to improve the convergence time, and at the meantime minimize the side-effects, such as bandwidth wastage.

For a primary LDP P2MP/MP2MP LSP, there are several ways to set up its backup path. It can use RSVP-TE P2P tunnel as a logical out-going interface, consequently utilize the mature high availability technologies of RSVP-TE. Or, it can make use of LDP P2P backup LSP as a packet encapsulation, so that the complex configuration of P2P RSVP-TE can be skipped. Or, it can build its own P2MP/MP2MP backup LSP according to IGP's loop-free alternative route, thus avoid double label stack. Other than these, it can also build a totally disjointed LSP in another topology, accordingly take advantage of the real end-to-end protection.

When the backup path is present, there are two options for packet forwarding and switchover. If the traffic sender feeds the stream on both paths, and the traffic receiver drops packet on backup path, the switchover will be very quick once the failure is detected, because the whole switchover action is a local behavior on traffic receiver. The disadvantage of this manner is that traffic will be duplicated on both paths, and consume double bandwidth. Contrastively, if the traffic sender feeds stream only on the primary path, the resource wastage can be waived. Cooperation is needed in this manner, so there will be some protocol extensions. But if the performance can be equal or better than the previous option, it is reasonable to choose the second one.

This document describes several methods to setup and switch paths for options to setup the backup LDP P2MP/MP2MP LSP. mLDP LSPs, including local protection, territorial protection, and end-to-end protection. The goal is to identify strengths, weaknesses and gaps, in order to build a reliable set of tools to shield against traffic black holes that would severely impact real-time applications, in the event of primary path failure.

3.1. Requirements

A number of requirements have been identified that allow the optimal set of mechanisms to developed. These currently include:

- o Computation of a disjointed (link and node) backup path within the multicast tree;

- o Minimisation of protection convergence time;

- o Optimisation of bandwitdth usage.

3.2. Scope

The method to detect failure is outside the scope of this document. Also this document does not provide any authorization mechanism for controlling the set of LSRs that may attempt to join a mLDP protection session.

4. Local protection using P2P LSP

By encapsulating mLDP packets within an P2P TE tunnel or P2P LDP backup LSP, the LDP P2MP/MP2MP LSP can be protected by the P2P protection mechanisms. However, this protection mechanism is not capable of detecting, and recovering, if the failure occurs on the destination node of the P2P backup LSP. Thus, this section provides a unified method to protect both node and link with P2P backup LSP.

```
                             +------------+ Point of Local Repair/
                             |     R1     | Switchover Point
                             +------------+ (Upstream LSR)
                                /       \
                               /         \
                            10/           \20
                             /             \
                            /               \
                           /                 \
                     +----------+        +-----------+
                     |    R2    |        |     R3    |
                     +----------+        +-----------+
                       |       \                  |
                       |        \                 |
                       |         \                |
                     10|        10\             20|
                       |           \              |
                       |            \             |
                       |             \            |
                       |              \           |
                       |               \          |
                       |                \         |
                     +-----------+  10  +-----------+ Merge Point
                     |    R4     |------|    R5     | (Downstream LSR)
                     +-----------+      +-----------+
```

mLDP Local Protection Example

Figure 1

In Figure 1 (mLDP Local Protection Example) above, the preferential path from R1 to R4/R5 is through R2, and the secondary path is through R3. In this case, the mLDP LSP will be established according to the IGP preferential path as R1--R2--R4/R5.

It is the responsibility of R2 to inform R1 of its downstream LSRs (in this example R4 and R5) and the respective labels (L4 and L5). Once the link between R1 and R2 fails, or R2 node fails, R1 will duplicate the traffic to R4 and R5, with inner label as L4/L5, and outer label as the P2P backup LSP R1--R3--R5--R4 and R1--R3--R5.

Finally, the previous forwarding states will be removed after R4 and R5 finish their Make-Before-Break (MBB) procedure.

4.1. Signaling procedures for local protection

Continuing to use Figure 1 (mLDP Local Protection Example), R2 sends a notification message to R1 to inform the node that R2 has two downstream nodes, R4 and R5 with forwarding labels L4 and L5 respectively.

When R1 sees R2 node going down, it takes mLDP packets as it would send them to R4 and R5 through R2 and sends them into the two P2P backup tunnels:

- o P2P tunnel R1--R3--R5--R4, using inner label L4.

- o P2P tunnel R1--R3--R5, using inner label L5.

So that R4/R5 will receive same packets as from the interface between R2 and R4/R5, just from different interface.

At the same time, R1 sends notifications with MBB request status code to R4 and R5. So that after R4 and R5 are done with MBB, they will send the notifications to R3 with MBB done status code. And then R3 will remove the old forwarding state which is being protected by the P2P backup tunnels.

4.2. Protocol extensions for local protection

A new type of LDP MP Status Value Element is introduced, for notifying downstream LSRs and respective labels. It is encoded as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|mLDP P2P Type=2|      Length                   |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Downstream Element 1                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Downstream Element N                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

mLDP P2P Encapsulation Status Code

Figure 2

The Downstream Element is encoded as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Downstream Label    |                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Downstream LSR-ID                           |
+                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Downstream Element in mLDP P2P Encapsulation Status Code

Figure 3

5. Territorial protection using mLDP LSP

Making use of IGP-FRR results, LDP can build the backup mLDP LSP for territorial protection. Note that in some scenarios, such as the following example, Failure Detector and Point of Local Repair, Switchover Point and Merge Point can be different nodes.

```
                               +------------+ Point of Local Repair
                               |     R1     | (Upstream LSR)
                               +------------+
                                   /       \
                                  /         \
                                 /           &
                                /             &
                               /               \
           Switchover Point \/_                 \  Failure Detector
                   +----------+            +------------+
                   |    R2    |            |     R3     |
                   +----------+            +------------+
                       /     \                  /
                      /       \                /
                     /         \              /
                    /           \            /
                   /             \          /
                 \/_              \        /
           +----------+          +------------+  Merge Point
           |    R4    |          |     R5     | (Downstream LSR)
           +----------+          +------------+
```

mLDP Territorial Protection Example

Figure 4

In Figure 4 (mLDP Territorial Protection Example), normally R1 feeds traffic to R4 through R2, and feeds traffic to R5 through R3. Once the link between R1 and R3 fails, R1 will be the logical Point of Local Repair node, which feeds the traffic to R5 through backup path on R2. Because R2 is already receiving traffic, so that R1 does not need to take any action. It is responsibility of R2 to duplicate the traffic to R5, as a Switchover Point. In this case, as the Failure Detector, R3 will need to send out the notification to R2, in order to trigger the switchover procedure.

5.1. Signaling Procedures for Territorial Protection

Merge Point (R5) determines the primary and secondary paths according to the IGP-FRR results. Then it sends out label mapping message including an LDP MP Status TLV that carries a FRR Status Code to indicate the primary path and secondary path. At the same time, it triggers a reverse path for failure notification by sending out label request message with an LDP MP Status TLV. The reverse path is uniquely identified by root address, opaque value, and MP address.

When failure is detected by Failure Detector (R3), it will send out the failure notification, then traffic will switch to the secondary path.

When Merge Point (R5) sees the next hop to Root changed, it will advertise the new mapping, and the traffic will re-converge to the new primary path.

5.2. Protocol extensions for Territorial Protection

A new type of LDP MP Status Value Element is introduced, for setting up secondary mLDP LSP. It is encoded as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|mLDP FRR Type=3|      Length                   | Status code   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                     MP Node Address                           ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

mLDP FRR Status Code

Figure 5

mLDP FRR Type: Type 3 (to be assigned by IANA)

Length: If the Address Family is IPv4, the Address Length MUST be 5; if the A
ddress
Family is IPv6, the Address Length MUST be 17.

Status code: 1 = Primary path for traffic forwarding (used in Label Mapping m
essage)
2 = Secondary path for traffic forwarding (used in Label Mapping
message)
3 = Reverse path for failure notification (used in Label Request
message)
4 = Failure notification (used in Notification message)

MP Node Address: A host address encoded according to the Address Family of th
is LSP.

mLDP Bandwidth Reservation Status Code Parameters

Figure 6

6. End-to-end protection using LDP Multiple Topology

[I-D.ietf-mpls-ldp-multi-topology] provides a mechanism to setup disjointed LSPs within different topologies. So that applications can use these redundant LSPs for end-to-end protection.

```
                             +------------+ Point of Local Repair/
                             |    Root    | Switchover Point
                             +------------+
                                 /       *
                                /         *
                               /           *
                              /             *
                             /               *
                            /                 *
                   +-----------+           +------------+
                   |    R1     |           |     R2     |
                   +-----------+           +------------+
                        |   \                   *  *
                        |    \                 *   *
                        |     \               *    *
                        |      \             *     *
                        |       _ _ _ * * * *      *
                        |           *  \           *
                        |          *    \          *
                        |         *      \         *
                        |        *        \        *
                        |       *          \       *
                     +-----------+       +-----------+
                     |  Leaf 1   |       | Leaf 2    |
                     +-----------+       +-----------+
```

mLDP End-to-end Protection Example

Figure 7

In Figure 7 (mLDP End-to-end Protection Example), there are two separated topologies from Root node to Leaf 1 and Leaf 2. For the same FEC element, the Leaf node can trigger mLDP LSPs in each topology. Root node can setup 1:1 or 1+1 end-to-end protection, using these two mLDP LSPs.

6.1. Signaling Procedures for End-to-end Protection

Using Figure 7 (mLDP Local Protection Example), Leaf 1 and Leaf 2 may trigger mLDP LSPs in different topologies, sending label mapping messages with same FEC element, different MT-ID and different label. When the Root node receives the label mapping messages from different topologies, it will set up two mLDP LSPs for application as end-to-end protection. Failure detection for the primary mLDP LSP is outside the scope of this document. But either Root node or Leaf node can be the Failure Detector.

6.2. Protocol extensions for End-to-end Protection

The protocol extensions required to build mLDP LSPs in different topologies is defined in [I-D.ietf-mpls-ldp-multi-topology].

7. Acknowledgements

We would like to thank authors of draft-ietf-mpls-mp-ldp-reqs and the authors of draft-ietf-mpls-ldp-multi-topology from which some text of this document has been inspired.

8. IANA Considerations

This memo includes the following requests to IANA:

- o mLDP P2P Encapsulation type for LDP MP Status Value Element.
- o mLDP FRR type for LDP MP Status Value Element.

9. Manageability Considerations

[Editors Note - This section requires further discussion]

9.1. Control of Function and Policy

9.2. Information and Data Models

9.3. Liveness Detection and Monitoring

9.4. Verifying Correct Operation

9.5. Requirements on Other Protocols and Functional Component

9.6. Impact on Network Operation

9.7. Policy Control

10. Security Considerations

The same security considerations apply as for the base LDP specification, as described in [RFC5036]. The protocol extensions specified in this document do not provide any authorization mechanism for controlling the set of LSRs that may attempt to join a mLDP protection session. If such authorization is desirable, additional mechanisms, outside the scope of this document, are needed.

Note that authorization policies should be implemented and/or configure at all the nodes involved .

11. References

11.1. Normative References

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

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, January 2001.

[RFC5036] Andersson, L., Minei, I., and B. Thomas, "LDP Specification", RFC 5036, October 2007.

[RFC5561] Thomas, B., Raza, K., Aggarwal, S., Aggarwal, R., and JL. Le Roux, "LDP Capabilities", RFC 5561, July 2009.

[RFC6348] Le Roux, JL. and T. Morin, "Requirements for Point-to-Multipoint Extensions to the Label Distribution Protocol", RFC 6348, September 2011.

[I-D.ietf-mpls-ldp-p2mp] Minei, I., Wijnands, I., Kompella, K., and B. Thomas, "Label Distribution Protocol Extensions for Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths", draft-ietf-mpls-ldp-p2mp-15 (work in progress), August 2011.

[I-D.ietf-mpls-ldp-multi-topology] Zhao, Q., Fang, L., Zhou, C., Li, L., So, N., and R. Torvi, "LDP Extension for Multi Topology Support", draft-ietf-mpls-ldp-multi-topology-00 (work in progress), October 2011.

11.2. Informative References

[RFC3468] Andersson, L. and G. Swallow, "The Multiprotocol Label Switching (MPLS) Working Group decision on MPLS signaling protocols", RFC 3468, February 2003.

Authors' Addresses

Quintin Zhao
Huawei Technology
125 Nagog Technology Park
Acton, MA 01719
US

Email: quintin.zhao@huawei.com

Emily Chen
Huawei Technology
2330 Central Expressway
Santa Clara, CA 95050
US

Email: emily.chenying@huawei.com