Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 Keller Hall
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 13-028

Toward Shortest Backup Paths for IP Fast ReRoute

Pengkui Luo

September 19, 2013

Toward Shortest Backup Paths for IP Fast ReRoute

Pengkui Luo (pluo@cs.umn.edu)

***Abstract*— The slow convergence under intra-domain link or router failures is a challenge for latency-sensitive applications in the IP networks. A variety of IP Fast Re-Route (IPFRR) schemes have been proposed to address the issue in the literature. Our paper reviews several of these IPFRR schemes: *Loop Free Alternate*, *U-Turn Alternate*, and in particular, *NotVia addresses*. Having observed issues caused by the selection of the end-points of backup tunnels in the *NotVia* scheme, we explored a variety of designs that improve NotVia by changing the end point of the tunnel. In particular, tunneling to the destination yields the shortest backup paths in the survived subgraph. However, it introduces significant increase of the routing table size, shedding light on the unbalanced trade-off between the length of the backup path and the routing table size.**

***Index Terms*—Intra-domain routing, IP Fast Re-Route, NotVia addresses, Backup routes, IP Tunneling.**

I. Introduction

Internet real-time or interactive applications such as VoIP and video conference are very sensitive to packet loss or delay, which occurs when fibers or routers fail. Upon the detection of a failure, routers in the affected area have to signal all their neighbors via IGP to re-compute new primary next-hops for all affected prefixes and update their forwarding plane. Traffic destined to these prefixes will be discarded until these next-hops are installed. This failure reaction procedure may take up to seconds to converge, and is usually unbearable for interactive applications.

In order to reduce the traffic loss and enable fast local reactions to outages, fast reroute (FRR) mechanisms have been designed over the past years. While fast reroute designs for MPLS [12] networks have been standardized for years, only recently research interests have been placed on fast reroute proposals for the prevalent IP networks [15], [17], [4], [3], [16], [14], [8], [11], [13]. The basic idea of IPFRR is to temporarily route traffic via pre-computed backup paths upon the failure detection, until the intra-domain network converges to a stable state with regular routing tables. To illustrative the general behaviors of IPFRR, consider a simple topology in Figure 1 (a), where source node S chooses node A as its *primary next-hop* in its shortest path destined to D. With IP Fast Reroute, S also uses B as its *alternate next-hop*. When link S-A goes down and S detects the failure, S will stop sending traffic to A via the failed link. Instead, S will switch to the alternate next-hop B, until the results of a new shortest path first (*SPF*) [2], [10] algorithm are installed in the routing tables.

The remaining part is organized as follows. In Section II, we review several existing IPFRR schemes (*Loop Free Alternates (LFA)* [4], *U-Turn Alternates* [3] and *NotVia* [16]), in terms of their working mechanisms, advantages and intrinsic limitations. In Section III, we focus on issues of the selected backup tunnels end-points in the NotVia scheme, and explored several potential proposals that change the tunnel end-points, in particular, tunneling to the destination. We show the price paid for achieving the shortest backup paths. We conclude the paper in Section IV.

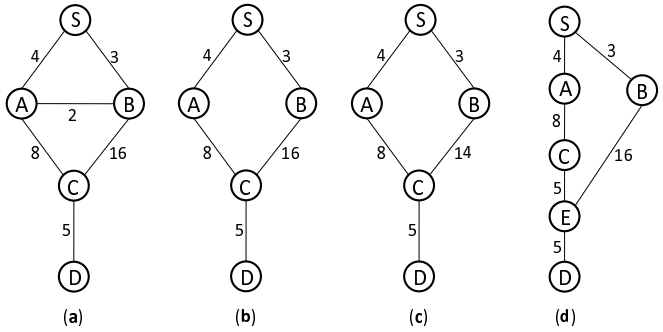

Fig. 1. Four simple topologies used in this paper.

II. IP Fast ReRoute Mechanisms

In this section, we review two classes of IPFRR mechanisms against link or node failures: the *LFA* [4] and *U-Turn* [3] schemes which provide *backup next-hops*, and the class of *NotVia* [16] schemes providing *backup paths*.

A. Loop Free Alternates and U-Turn Alternates

"Loop Free" Alternates (*LFA*) is considered the simplest scheme for fast failure protection, aiming to eliminate forwarding loops during the routing table convergence. To see how loops would arise, consider a change to Figure 1(a): if the link weight of B-A were changed to 8, then choosing B as the alternate next-hop for S would produce a micro loop between S and B, because the cost of B-A-C-D would be larger than that of B-S-A-C-D. To prevent such loops, Atlas and Zinin [4] proposed two criteria for a neighbor to provide an *LFA*: (1) this neighbor does not belong to the same sub-tree of the reverse shortest path tree or *SPT* rooted at the next-hop of the protected component, and (2) it lies in a downstream path.

To illustrate how this design reacts to link or node failures, let us consider Figure 1(a) again. Node B, which satisfies the first criterion, is installed as node S's pre-computed *LFA* when destined to D. If S detects a failure on link S-A – recall that A is the primary next-hop of S – it then forwards packets to B, and the new transient path will be S-B-A-C-D until the IGP converges. Consider another scenario where node A fails: according to the second criterion, S uses B as its *LFA*, but B

can only use C (as opposed to $\{S, C\}$) as its *LFA*. Therefore, B will not forward packets back to S and thus a micro loop between S and B is avoided.

The existence of LFAs, however, is topology-dependent. In Figure 1(b), for instance, S does not have LFAs since node B does not meet either criteria. To avoid packet drops when S-A link fails, Atlas [3] proposed to allow B to be S's *U-Turn Alternate (UTA)*, via which traffic can be rerouted upon the failure of node A or link S-A, even in the absence of the *LFA* criteria.

The main advantage of *LFA* and *UTA* designs lies in their simplicity, resulting in small routing table sizes, light control overhead and good manageability. They do not impose additional increase on routing table size, because each node only needs to maintain a small number – no more than the node degree – of next-hop alternates, which can be operated by explicitly setting rules. And the pre-computation overhead only lies in the procedure of selecting alternatives, and is limited in the range of each node and its neighbors, no matter for node protection or for link protection.

The main issue of *LFA* and *UTA* is their limited *coverage*, i.e., they cannot protect all nodes and links. As we have demonstrated using Figure 1(b), *LFA* is topology-dependent. A study [6] has shown that over 40% of links and nodes are not protected by *LFA* on real ISP topologies, partially owing to the two "over-killing" criteria. *UTA* slightly relaxes the criteria, covering more components, but it still cannot protect all nodes and links.

B. NotVia Addresses

NotVia [16] is one of the link/node protection techniques providing a full coverage. It works by delivering packets to their destinations "not via" a known failure. The key idea is to explore a path bypassing that component, for each protected "component" that is vulnerable to failures. In Figure 1(a) again, for instance, node A is S's primary next-hop when destining to D. To protect link S-A against failures, a backup path S-B-A that bypasses this protected link is constructed; whereas to protect node A, a path S-B-C "not via" node A is constructed. In this example, the backup path of NotVia approach coincides with that of *LFA*.

Consider using the NotVia approach in Figure 1(b) where S does not have *LFAs*. If node A fails, S then tunnels [5] traffic to A's primary next-hop which is C, along the shortest path (destined to C) which is computed based on the subgraph with the failure node A removed, i.e., S-B-C, and then delivered to D. This is how NotVia handles node failures.

A NotVia backup path is established by a set of special addresses called the *NotVia addresses*, specifically, the link-protection NotVia addresses and the node-protection NotVia addresses. When a node S does not have an *LFA* neighbor to protect all destinations it reaches via the link S-A (e.g., the scenario in Figure 1(b)), a link-protection NotVia address $NV_{S!\rightarrow A}$ is assigned to node A (i.e., the next-hop of the potential failure). Similarly, when node A fails and its upstream node S does not have an *LFA* to reach D, a node-protection NotVia address $NV_{!A\rightarrow C}$ is assigned to node C – the primary next-hop of the potential failure node A – such that all nodes compute shortest paths to reach this address.

Comparing with *LFA* and *UTA*, the biggest advantage of the NotVia approach is its *full* coverage, namely, it can protect *any* node and *any* link in the network. Li et al. [9] shows that if a node is protected by NotVia, no further link protection is needed for the links associated with the node. And [16] recommends always preferring node protection addresses over link protection addresses.

However, the increase of routing table sizes in NotVia design is considerably high. For an intra-domain network that consists of M links, the total number of node-protection NotVia addresses to be installed on each node would be $O(M)$, which is not scalable.

Manageability-wise, for each NotVia address, *all* nodes – except for the protected node or its primary next-hop – need to run an SPF algorithm on the subgraph with the protected component removed. This control overhead is considerable. Although this pre-computation is done off-line ahead of time, a simple topological change would re-trigger this expensive process, making routers to send out link-state announcements to inform others to compute new NotVia addresses or to avoid computing unnecessary NotVia addresses. The messages incurred would delay the protection path restoration time [9], and make the management more challenging. Therefore, a mixture of NotVia and *LFA*/*UTA* is often recommended [15], [16], [7], [9], and NotVia alone is only used in scenarios where *LFAs* or *UTAs* do not exist.

In the past years, various attempts have been made to improve the efficiency and manageability of NotVia. Li et al. [9] proposed a *NotVia Aggregation* approach, aiming to reduce the number of unnecessary NotVia addresses. Their key idea is that, if next-hops for the NotVia addresses of a node are the same as the next hops for the regular IP addresses of the node, then those unnecessary NotVia addresses can be *aggregated* into the routing entries for regular IP addresses. For example, consider Figure 1(c) which is constructed by changing the weight of the B-C link in Figure 1(b) to 14. Since path B-C is shorter than path B-S-A-C, the shortest path from B to C is the link B-C itself, which does not traverse link A-C or node A. Therefore, node B does not need to store the link-protection NotVia address $NV_{A!\rightarrow C}$ or node-protection NotVia address $NV_{!A\rightarrow C}$, which shares the same next-hop (i.e. node C) as the next-hop destining to C in B's regular IP forwarding table. In other words, the two NotVia addresses above can be aggregated into the regular entry C at B's forwarding table. The approach helps reduce the number of NotVia addresses in a large network.

III. Toward Shortest Backup Paths

A. The Near-Tunnel Issue

One of the fundamental issues of the NotVia scheme is that the end-point of a backup tunnel is the next-hop of the potential failure, causing redundant traversals of re-routed packets in many scenarios. For example in Figure 1(b), if link S-A fails, S tunnels [5] packets to A, along the shortest path computed based on the subgraph with link S-A removed, i.e., S-B-C-A. Node A then decapsulates and forwards the packets to its primary next-hop C, and finally delivers packets to D. So the backup path of this re-route turns out S-B-C-A-C-D. We see that the link A-C has been unnecessarily traversed twice. Various attempts can be made to solve this issue.

One may observe that node S does not need to care whether link S-A fails or link A-C fails or even node A fails. Instead, if S treats them all as if node A fails, the reroute path will be S-B-C-D without any repeated links. Therefore, a modified design that one may propose would be to eliminate all link-protection NotVia addresses and solely use node-protection addresses instead. But we argue that it is not an effective solution. Let us consider Figure 1(d): even if we used solely node-protection NotVia addresses, and node S treats the failure of link S-A as that of node A, the re-routed path would still have to traverse along path S-B-E-C-E-D with the repeated segment of link C-E. Therefore, eliminating link-protection NotVia addresses does not prevent repeated traversals. The fundamental issue here is still the choice of the tunnel end-point, that is, the next-hop of the protected component.

B. The Choice of the Tunnel End Point

Another remedy that one may propose by changing the end-point of the NotVia backup tunnel is as follows. Consider a shortest path tree or *SPT* rooted at destination node D, in which there is a subtree rooted at the protected node A. Suppose the source node S resides on this subtree; obviously the shortest path from S to D should traverse via A. In order to protect node A or its associated links on the SPT against failures, S picks – assuming it manages to find one – a node B as the end-point of the NotVia backup tunnel. A qualified node B satisfies: (1) it is not on A's subtree, and (2) the shortest path from S to B does not traverse via the protected node A. Hence a backup path S-...-B-...-D is established against the node failure of A.

However, the problem of this design lies in two aspects: (1) Many potentially feasible tunnel end-points in A's subtree are discarded, and thus S may not manage to find a feasible end-point to tunnel encapsulated traffic to, and (2) Even if S manages to find a feasible end-point outside the subtree rooted at A, the backup path spliced by the two shortest path (i.e., the shortest path S-...-B, and the shortest path B-...-D) may *not necessarily* be the shortest path that bypasses the failure, that is, there may exist another *shorter* path not traversing node A. Since the stitched backup path may not be the shortest one, the necessity of finding two segments of shortest paths in the first place is questionable, provided that the *SPF* computation is costly.

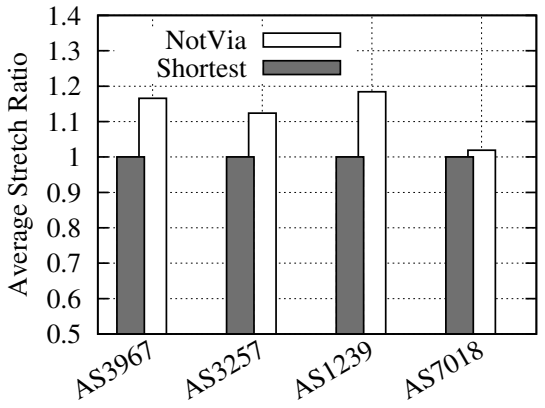

Fig. 2. A comparison between the shortest backup path and NotVia in terms of the stretch ratio on real ISP topologies.

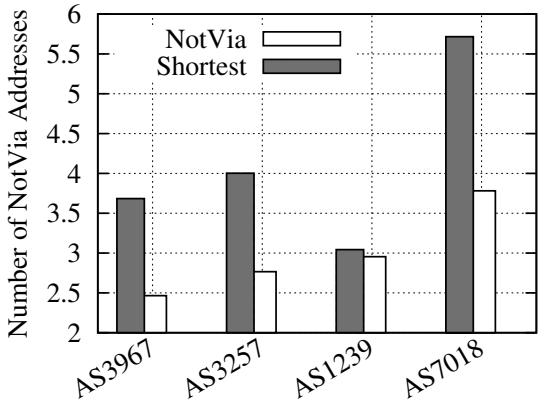

Fig. 3. Comparison of the number of NotVia addresses on real ISP topologies in terms of the used addresses; notice that the y-axis is in $\log_{10}$-scale.

C. Tunneling to the Destination

In theory, the "optimal backup path" is the shortest path to the *destination*[1] in the subgraph after removing the failures (i.e., protected links or nodes). In other words, it is the shortest path after IGP re-converges from failures. The costs (or rather, lengths) of backup paths in various IPFRR schemes can only approach this shortest path. The ratio of the length of actual backup path length relative to that of the shortest path is called the *stretch ratio* of the path, which is always no less than one.

In order to approach the ideal stretch ration of 1, one may come up with a proposal based on a modification on the *NotVia* scheme: changing the end-point of the backup path straightly to the *destination*, from the *next-hop* of the failure as in NotVia. Consequently, the NotVia address is attached to the destination, instead of the failure's next-hop. For any node in the network, a NotVia address corresponds to an SPT rooted at that node, with a protected link/node removed. Suppose E is the egress router for a specific destination prefixes. To protect a component C for this destination, a NotVia address $NV_{E,C}$ is assigned to E and propagated to all other routers via the link

[1]If packets are to be transit to other ISPs, the *destination* here refers to the egress router that is responsible for the specified set of destination prefixes.

state exchange, such that they can create this entry or its prefix in their routing tables, and figure out their next-hops based on their self-computed SPTs rooted at node E with component C removed.

This design completely eliminates the problem of repeated traversals, since it achieves the same path as the shortest path after the convergence. In this sense, it is "optimal." However, it pays the price of much larger routing table: if we wanted to protect any single-node failures in a network that comprises N nodes and M edges, the total number of NotVia addresses under this design would be $O(N^2)$, which is larger than $O(M)$ for the original NotVia design.

To show this, we conduct simulations on some inferred ISP topologies (AS numbers: 3967, 3257, 1239, 7018) as well as inferred link weights provided by the Rocketfuel [1] project. Figure 2 shows the comparison of the stretch ratio, namely, the length of the protection path relative to that of the shortest path after re-convergence. Figure 3 shows the comparison of the number, on a log10 base, of NotVia addresses on four real ISP topologies. We can see that this tunneling-to-destination design requires significantly more NotVia addresses (thus more SPT computation) than the original NotVia design. This price paid for achieving the "optimal" path seems too high to be tolerable.

Another issue of this design is that it lacks a mechanism to locate the failure. In the original NotVia design, the next-hop is responsible for broadcasting upon the failure detection; all other nodes in the network can figure out where the failure is through the link state exchange. In this tunneling-to-destination design, however, locating the failure is challenging for other nodes, which may be several hops away from the failure.

IV. Conclusion

We have reviewed three promising IPFRR schemes - *Loop Free Alternates (LFA)* [4], *U-Turn Alternates* [3] and *NotVia* [16], in terms of their working mechanisms and especially their intrinsic limitations. Having observed the issues caused by the selection of the end-points of backup tunnels in the NotVia scheme, we explored a varieties of proposals that attempt to further improve the original NotVia by changing the end point of the tunnel, in particular, tunneling to the destination, which yields the shortest backup paths in the survived subgraph. However, it significantly increases the routing table size, shedding light on the unbalanced trade-off between the length of the backup path and the routing table size.

In this paper we did not mention how to locate failures, which is inherently crucial because our generalized design relies on computing SPTs on survived subgraph. Due to the distributed nature of IP networks, precisely locating failures is challenging, and further propagating this information would take relatively long time, which is less applicable for IP *Fast* Re-Route. Some mechanisms [13], [11], [8], [18] have been proposed to roughly *infer* the location of failures, but they are orthogonal to our work.

References

[1] Rocketfuel: An ISP Topology Mapping Engine, http://www.cs.washington.edu/research/networking/rocketfuel/.

[2] ISO/IEC 10589:2002: Intermediate System to Intermediate System Intra-domain Routeing Information Exchange Protocol for Use in Conjunction with the Protocol for Providing the Connectionless-mode Network Service (ISO 8473). November 2002.

[3] A. Atlas. U-turn Alternates for IP/LDP Fast-Reroute. *IETF Internet draft version 03*, February 2006.

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

[5] S. Bryant, C. Filsfils, S. Previdi, and M. Shand. IP Fast Reroute using Tunnels. *IETF Internet draft version 03*, November 2007.

[6] P. Francois and O. Bonaventure. An Evaluation of IP-based Fast Reroute Techniques. In *CoNEXT '05: Proceedings of the 2005 ACM conference on Emerging network experiment and technology*, pages 244–245, New York, NY, USA, 2005. ACM.

[7] M. Gjoka, V. Ram, and X. Yang. Evaluation of IP Fast Reroute Proposals. In *COMSWARE 2007. 2nd International Conference on*, pages 1–8. IEEE, January 2007.

[8] S. Lee, Y. Yu, S. Nelakuditi, Z.-L. Zhang, and C.-N. Chuah. Proactive vs Reactive Approaches to Failure Resilient Routing. In *IEEE INFOCOM*, volume 1, page 186, 2004.

[9] A. Li, P. Francois, and X. Yang. On Improving the Efficiency and Manageability of NotVia. In *Proc. ACM CoNEXT*, pages 1–12, New York, NY, USA, 2007. ACM.

[10] J. Moy. OSPF version 2. *IETF RFC 2328*.

[11] S. Nelakuditi, S. Lee, Y. Yu, Z.-L. Zhang, and C.-N. Chuah. Fast Local Rerouting for Handling Transient Link Failures. *IEEE/ACM Transaction on Networking*, 15(2):359–372, 2007.

[12] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. In *IETF RFC 3031*, 2001.

[13] S. N. Sanghwan, S. Nelakuditi, S. Lee, Y. Yu, and Z.-L. Zhang. Failure Insensitive Routing for Ensuring Service Availability. In *IWQoS*, 2003.

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

[15] M. Shand and S. Bryant. IP Fast Reroute Framework. *IETF RFC 5714*, January 2010.

[16] M. Shand, S. Bryant, and S. Previdi. IP Fast Reroute using Not-Via Addresses. *IETF Internet draft version 05*, March 2010.

[17] A. J. Tian, N. Shen, and R. Networks. Fast Reroute using Alternative Shortest Paths. *IETF Internet Draft version 01*, July 2004.

[18] J. Wang and S. Nelakuditi. IP Fast Reroute with Failure Inferencing. *ACM INM*, 2007.