Lightweight IP Fast Reroute with Tunnel-AT

Lingtao Pan, Mingwei Xu, Qing Li
Department of Computer Science, Tsinghua University
{plt, xmw, liqing}@csnet1.cs.tsinghua.edu.cn

Dan Jen
UCLA
jenster@cs.ucla.edu

***Abstract*—The demand for faster failure-recovery in pure intra-domain IP networks has led to the development of several IP Fast ReRoute (IPFRR) mechanisms. However, these mechanisms are all either too computationally expensive, or fail to provide satisfiable protection coverage. For these reasons, IPFRR schemes have yet to see widespread commercial deployment. In this paper, we present a new IPFRR mechanism called *Tunnel-AT* based on incremental SPF algorithm and the concept of *Attaching Tree*. Tunnel-AT outperforms existing IPFRR mechanisms by providing 100% node protection coverage at a computational cost of less than one full SPF calculation.**

I. Introduction

When a link or node failure occurs in an IP network, there is a period of disruption to the delivery of traffic until the network re-converges on a new topology. IP fast reroute (IPFRR) mechanisms are methods used in pure IP networks to provide protection from such disruptions. This is done by having failure-adjacent nodes compute backup routes in advance that allow them to repair failures immediately upon detection of failures. Traffic can be delivered to their correct destinations even before the network has re-converged on a new topology.

Thus far, there are three different IPFRR mechanisms that are documented in RFCs or IETF drafts: LFA in [1], Tunnel in [2], and NotVia in [4]. However, each of these schemes are either too computationally expensive or do not provide complete protection coverage.

In this paper, we present Tunnel-AT. It requires less than one full SPF computation, provides 100% single node protection, and does not require routers to maintain extra addresses. Furthermore, the length of its calculated protection paths are almost always optimal.

II. Tunnel-AT Scheme

Tunnel-AT scheme contains forwarding plane mechanism and routing plane algorithm. The forwarding plane mechanism is similar to Tunnel. But in the routing plane, Tunnel-AT is totally different. The routing algorithm is based on the incremental SPF (iSPF) algorithm introduced in [3] and the concept of *Attaching Tree* (thus comes the name Tunnel-AT).

The iSPF in [3] has two major features different from other ones: 1) In every iteration, the node with the smallest change of distance is selected. 2) In every iteration, a subtree, instead of just one node, is added to form the new shortest path tree.

The research was supported by the National Basic Research Program of China (973) under Grant 2009CB320502, the National High-Tech Research and Development Program of China (863) under Grants 2007AA01Z2A2 and 2009AA01Z251, the National Science & Technology Pillar Program of China under Grant 2008BAH37B03

The second feature enables the algorithm to achieve better performance in terms of computational cost. We use Figure 1 to demonstrate it. Figure 1(a) contains six nodes with labels from A to F. Solid lines are links in the shortest path tree rooted at node A. Dotted lines are links not in the tree. The weight of each link is shown beside the link. The number in a node is the distance from A to that node.

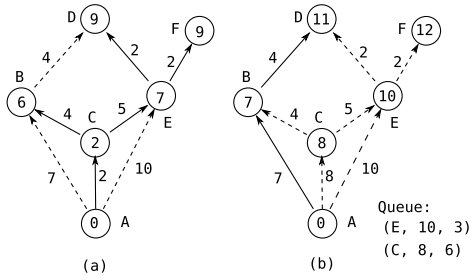

Fig. 1. iSPF

Suppose at some time, the link between A and C changes its weight from 2 to 8. Nodes from B to F are all *affected nodes* and are marked as *floating*. Only the root node A is marked as *attached*. For all the floating nodes, we check whether they have links to the attached nodes, and calculate the new distance and the change (delta) in distance, which gives queue {(B, 7, 1), (E, 10, 3), (C, 8, 6)}, where the third value is the delta. In the next iteration, (B, 7, 1) will be considered first since it has the smallest delta. B will be marked as attached. Since B has an outing edge to D, the new distance and delta of D is calculated, which gives queue {(D, 11, 2), (E, 10, 3), (C, 8, 6)}. Then (D, 11, 2) is selected and D is marked as attached. We show the current result in Figure 1(b).

In the next iteration, node (E, 10, 3) is selected, instead of just mark E as attached, *the original subtree rooted at E is considered together*. Both E and F are marked as attached, and the new distance of F calculated by adding the delta 3 to its original distance, the result is 12. In the next iteration, node C is selected and the new shortest path tree is computed.

Intuitively, the process of the iSPF algorithm can be viewed as *attaching* subtrees back one by one and finally forming the new shortest path tree. We give the subtrees the term *Attaching Tree*s. An attaching tree may be attached to a node of another attaching tree. In this way, they form a *super attaching tree*. The root of the super attaching tree attached to a node that is not affected by the changing of topology. Traffic from the original node to nodes of the super attaching tree will come through the root. Thus, we call the root the traffic incoming node, or simply the *incoming node*.

We use a simple example to clarify the above concepts. Figure 2(a) shows the shortest path tree of node 0, which wants

to protect traffic over link 0→1 in case of node 1 failure. When node 0 detects the failure of link 0→1, it assumes that node 1 failed. The *affected nodes* are node 2, 3, 4, 5, 6, 9, 10, 11 and 12. Figure 2(b) shows the new shortest path tree of node 0 after deleting all the links from(to) node 1. We can identify the root of attaching trees by checking which nodes change their parents. Node 4 and node 11 are the roots of two *super attaching tree*s and thus are *incoming nodes*.

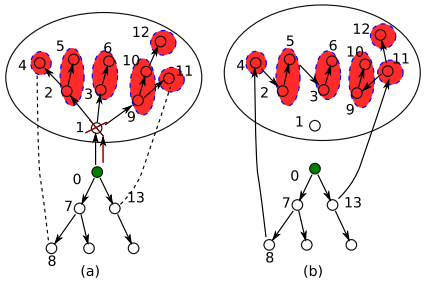

Fig. 2. Tunnel-AT Example. The original shortest path tree of node 0 is on the left. The new shortest path tree after the failure of node 1 is on the right. The six attaching trees are circled out. The two edges denoted by dotted lines are edges with big weight that are not part of the original tree.

Tunnel-AT routing algorithm is based on the following proposition:

Proposition 1. *All the nodes in an super attaching tree can be protected in the same way by rerouting the packets to the* incoming node.

The algorithm is separated into two stages. In the first stage, given the shortest path tree of a protection node s and a failed neighbor f, the algorithm calculates a new shortest path tree by adjusting the original one and finds the incoming node for every affected node. In the second stage, protection routing table entries for affected nodes are calculated.

In the forwarding layer, every destination is associated with a list of routes of the form $(nhop, mark)$. In the above example, node 0 has two routes for destination 2, $(1, ())$ and $(7, (T_8, F_4))$. If a packet of destination 2 comes, the two routes are returned by table looking up. If node 1 is not detected as down, the first route is used, and the packet will be sent to node 1. Otherwise, the second action will be used. The next hop is 7 and the mark is (T_8, F_4), which means adding a new header with destination 8 and marking the packet so that node 8 will forward it directly to node 4.

III. Simulation Results

To verify the correctness and study the performance of Tunnel-AT, we have written a stand alone simulator as well as implemented it in SSFNET. We have also proposed an IETF draft [5], and currently we are implementing Tunnel-AT in Quagga on Linux platform. We use the data set generated by the Rocketfuel project which includes six backbone topologies with inferred weights and PoP structures.

Our simulation confirms that Tunnel-AT can provide 100% single node protection coverage in a bi-connected network, outperforming the original Tunnel mechanism. We also generate random link failures to study the performance of Tunnel-AT at multi-link failures. The result shows that Tunnel-AT can maintain a high protection coverage even when six random picked links failed (Figure 3).

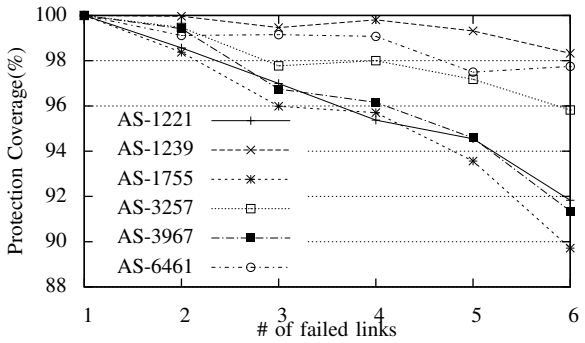

Fig. 3. Protection Coverage for multi-link failures

We define the relative protection path length from s to d as the protection path length divided by the new topology's shortest path length. Table I compares the average relative protection path length of Tunnel-AT and schemes that reroute packets to next-next-hop (NNHop) such as NotVia. The relative length of Tunnel-AT is very close to optimal.

TABLE I
RELATIVE PROTECTION PATH LENGTH (IN PERCENTAGE) COMPARED TO THE SHORTEST PATH LENGTH OF THE NEW TOPOLOGY AFTER FAILURE

	AS1221	AS1239	AS1755	AS3257	AS3967	AS6441
TunnelAT	100.0	100.02	100.29	100.03	100.05	100.03
NNHop	108.59	109.03	113.29	112.73	116.37	108.59

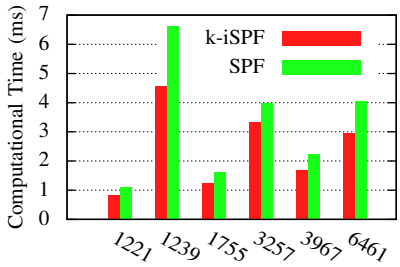

Fig. 4. Computational complexity for one router (network wide average).

Every node has to perform a full SPF to construct its normal routing table entries, and k times (# of neighbors) iSPF to construct protection routing table entries. Figure 4 shows the actual computational time in our simulation (implemented in SSFNET) running on a Linux machine with 2.0GHz CPU. The result confirms that the computational cost of Tunnel-AT is less than that of one full SPF.

References

[1] A. Alia, Z. Alex, T. Raveendra, C. Gagan, M. Christian, I. Brent, and F. Don, "Basic specification for IP fast-reroute: Loop-free alternates," RFC 5286, 2008.

[2] S. Bryant, C. Filsfils, S. Previdi, and M. Shands, "IP fast reroute using tunnels," Internet draft, draft-bryant-ipfrr-tunnels-03, 2008.

[3] P. Narváez, K.-Y. Siu, and H.-Y. Tzeng, "New dynamic SPT algorithm based on a ball-and-string model," *IEEE/ACM Trans. Netw.*, vol. 9, no. 6, pp. 706–718, 2001.

[4] M. Shand, S. Bryant, and S. Previdi, "IP fast reroute using not-via addresses," Internet draft, draft-ietf-rtgwg-ipfrr-notvia-addresses-02, 2008.

[5] M. Xu, L. Pan, and Q. Li, "IP fast reroute using tunnel-at," draft-xu-ipfrr-tunnelat-00, 2009.