IP Fast Reroute Using Tunnel-AT

draft-xu-ipfrr-tunnelat-00

Mingwei Xu, Lingtao Pan, Qing Li

Tsinghua University, China

75th IETF Meeting, Stockholm
July 2009

Background

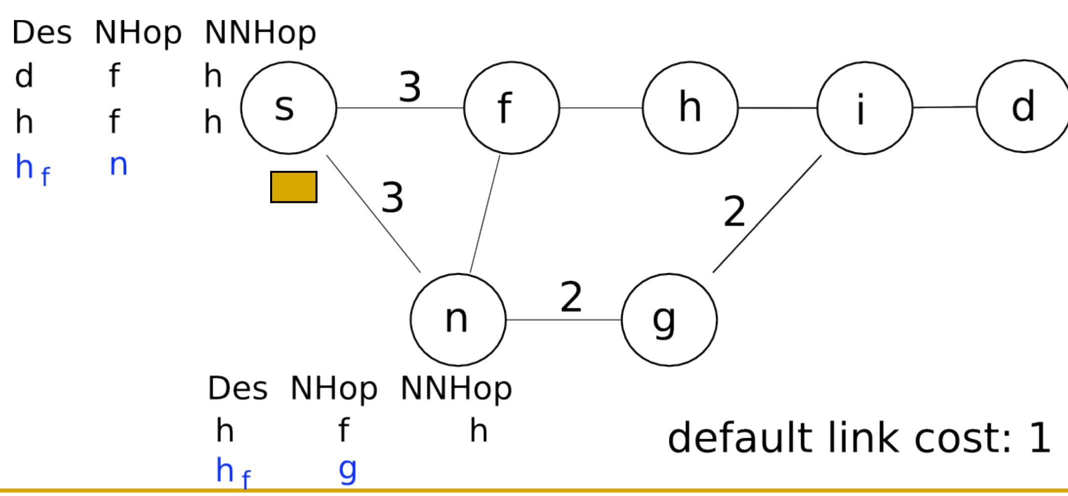
- What is IPFRR
  - When a link or node failure occurs in an IP network, there is a period of disruption to the delivery of traffic until the network re-converges on a new topology.
  - IP fast reroute (IPFRR) mechanisms are methods used in pure IP networks to provide protection from such disruptions.

Background – Main stream mechanisms

- LFA: (in RFC 5286).
  - Send packet to an alternate neighbor.
  - Simple, but can not provide 100% protection coverage.

Background – Main stream mechanisms

- NotVia: (in ietf-rtgwg-ipfrr-notvia-addresses)
  - 100% single node protection coverage
  - But, has to maintain extra NotVia addresses, high management burden
  - Has to reroute packet to next next hop, unnecessary back tracing (see example below)

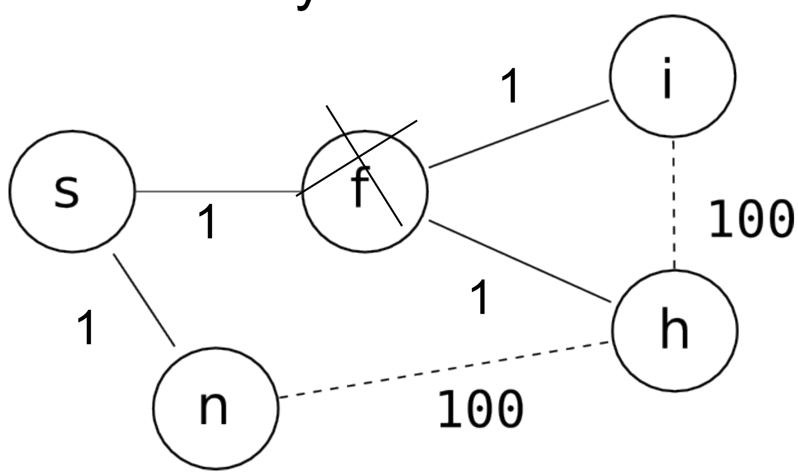


Background – Main stream mechanisms

- Tunnel: (in ietf-bryant-ipfrr-tunnels)
  - Repairing source node S chooses a tunnel end node T that can forward packets to destination D bypassing failed neighbor F.
  - But
    - The original draft does not describe an efficient algorithm to find tunnel end points.
    - Failed to provide 100% single node failure protection

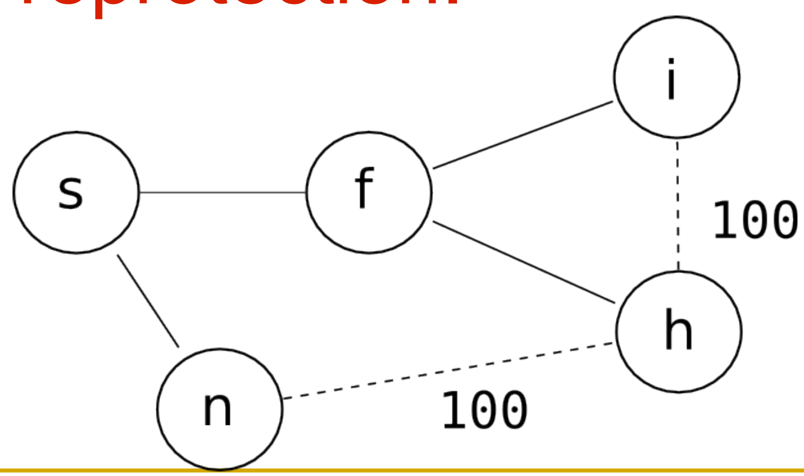
Background – More on Tunnel

- Why tunnel can't provide 100% single node failure protection coverage.
  - Node n can't be used as tunnel end point for s to reach h since the cost of link n-h is high.
    - Directed forwarding can be used to solve it. (Tell n to forward packet to h directly)
  - But even with directly forwarding, node i can't be reached by s.



Background -- Reprotection

- But actually s can send packets with destination i to n, and telling n to directly forward them to h. Since h also notices the failure of f, it will reroute packets to i.
- We call this reprotection.

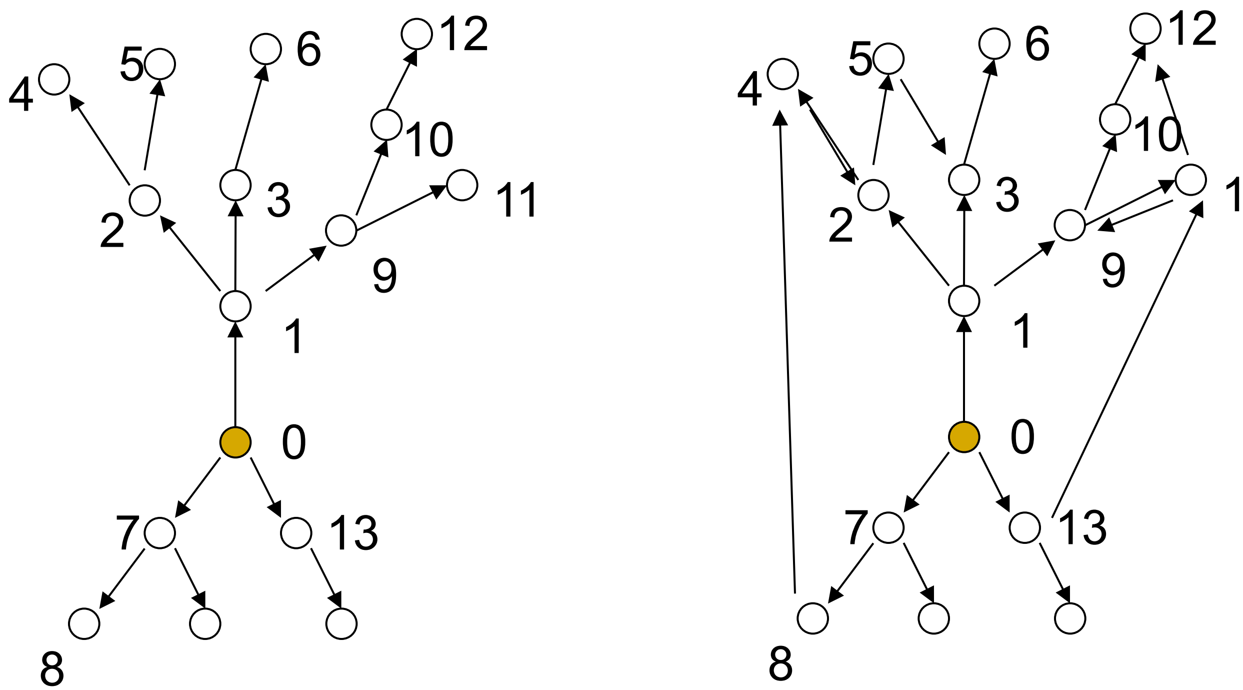


Tunnel-AT

- Our mechanism Tunnel-AT is an improved Tunnel.
  - By exploring reprotection, we achieve 100% protection for single node failure on a bi-connected topology.
  - Inspired by iSPF, we propose an efficient algorithm to find tunnel end points.
  - The length of the backup paths computed under Tunnel-AT are always exactly or very close to the length of the shortest working path to the destination.

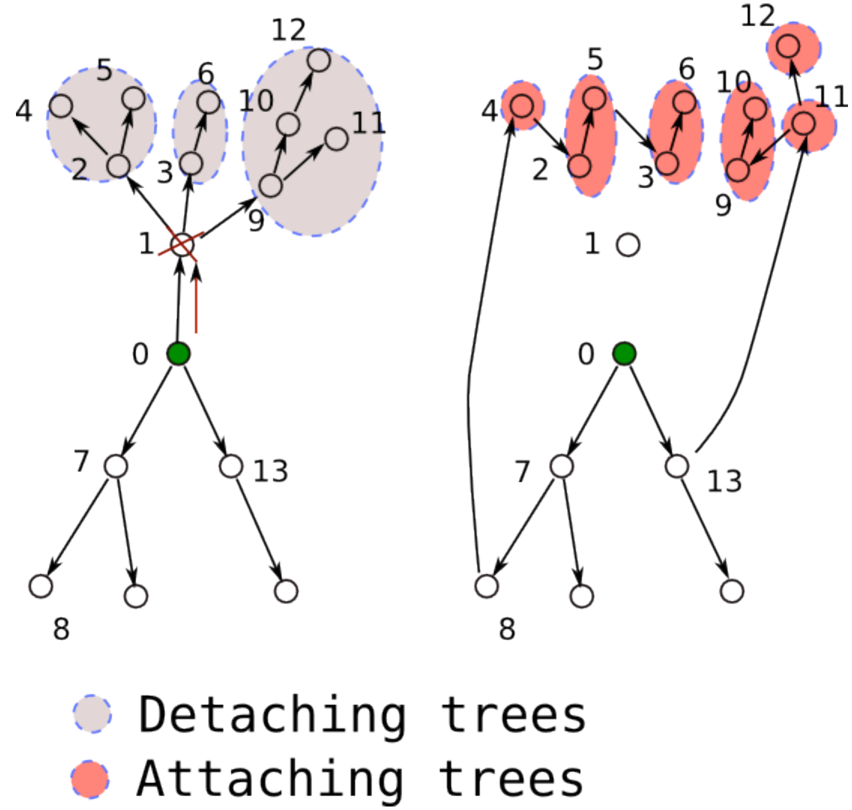
Our Muse -- iSPF

- The iSPF works by reattach subtrees back:



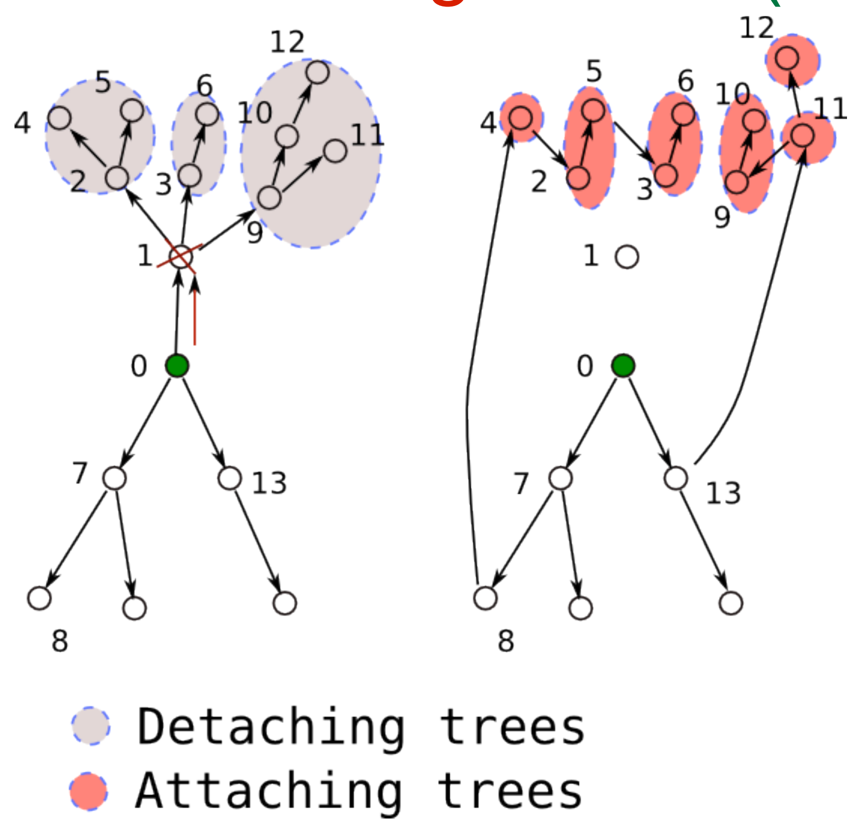
Some definitions

- Detaching trees: Subtrees of the original shortest path tree of S rooted at one of F's neighbor other than S.
- Affected nodes: nodes of detaching trees.
- Attaching trees: maximal common subtrees of the original and new shortest path tree formed by affected nodes. Or the subtrees reattached in the process of iSPF.



Incoming nodes

- We call affected nodes whose parents are not affected nodes incoming nodes. (node 4 and 11)



Tunnel-AT algorithm

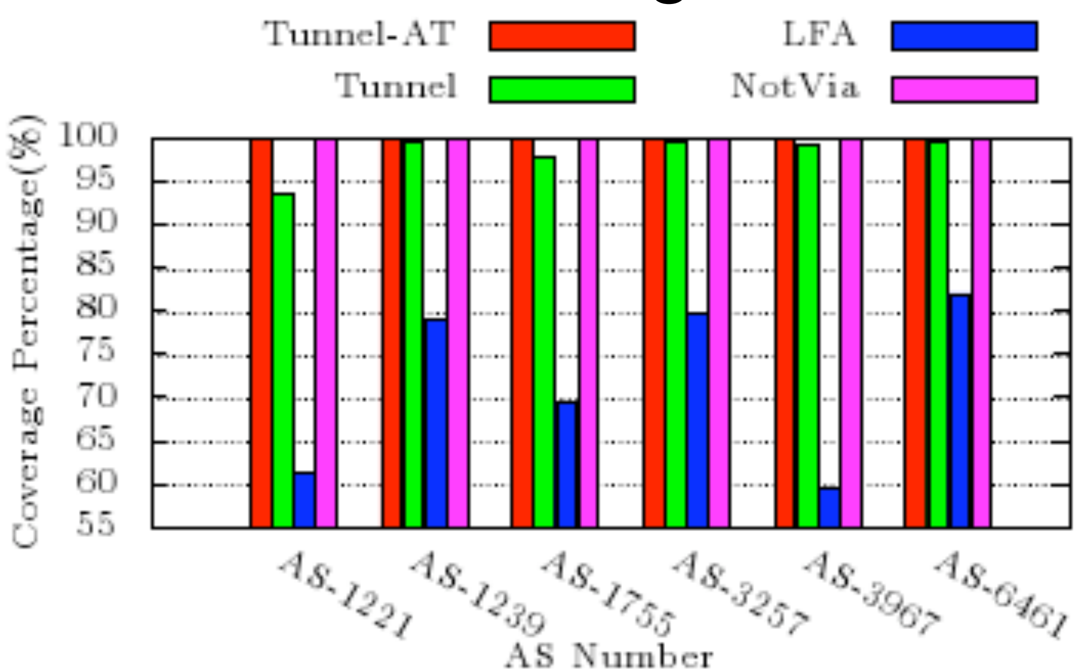
- Step 1: Record the corresponding incoming node of each affected nodes during the process of iSPF.
- Step 2: For each affected destination d, determine the mark needed to reroute packets to d's incoming node.
  - Stage 1: Let P be the incoming node's parent. Decide if we should encapsulate packets with P's header (For example, if P is S's neighbor, new header is not needed.)
  - Stage 2: Determine if Directed Forwarding is needed to ensure P send packets to the incoming node.

Complexity Analysis

- Based on the original shortest path tree T calculated by the normal link state routing protocol, every node has to perform k times incremental shortest path tree (iSPT) (k is the number of neighbors) to construct backup routes for all destinations.
- Theoretically, k * iSPT < One Full SPT

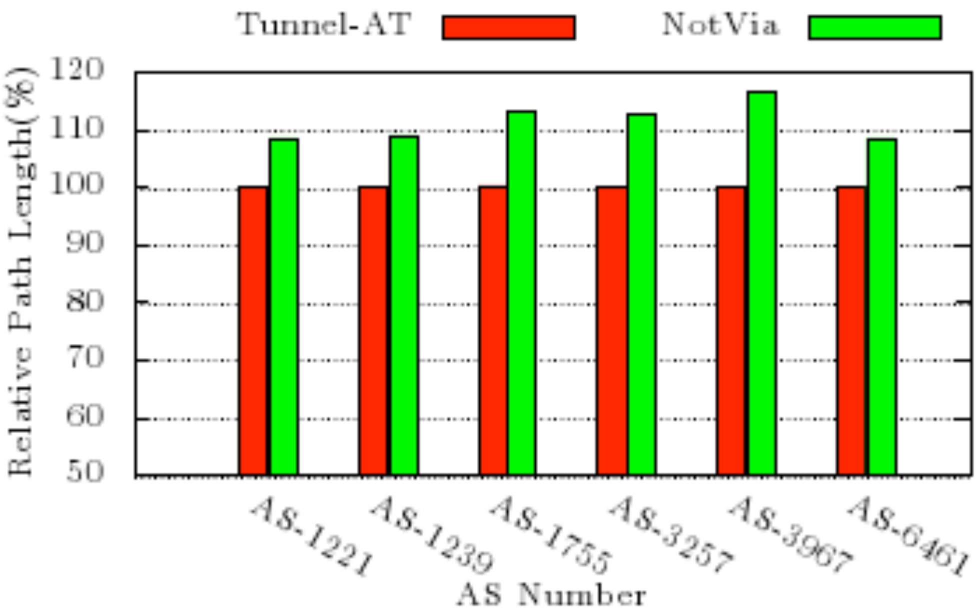
Evaluation

- Dataset
  - 6 ASes from Rocketfuel topology database.
  - Extract bi-connected component.
- 100% Protection coverage:



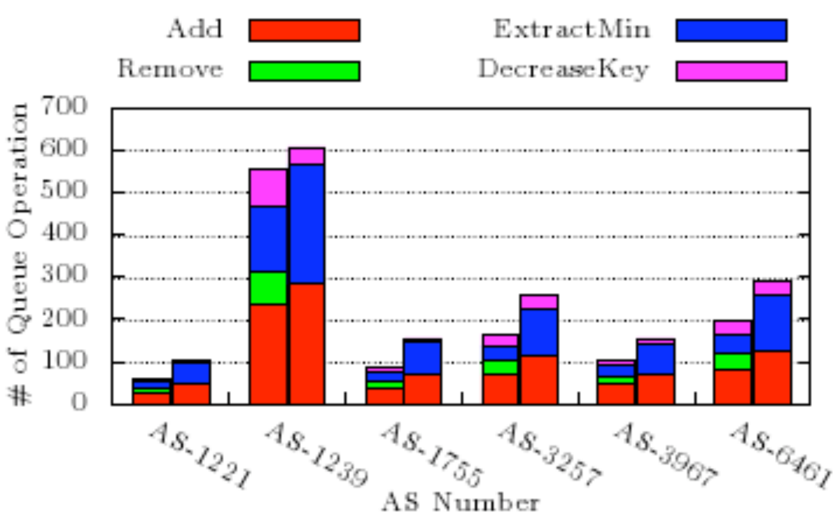
Evaluation

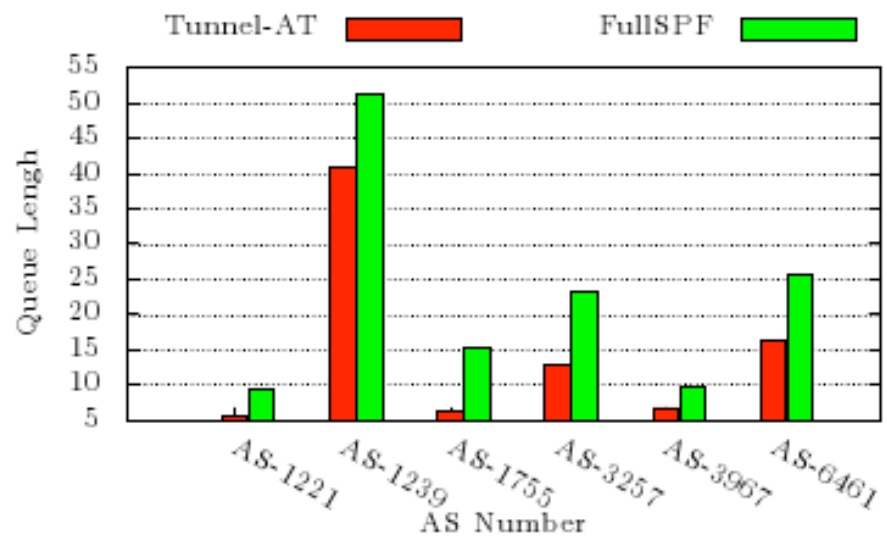
- Almost optimal path length:



Evaluation

- Complexity is less than one full SPT





Thanks

Some Properties

- Property 1: If destination d1 is the parent of d2 in the same ATTree, d2 can be protected in the same way as d1.
- Property 2: If destination d1 is the parent of d2, d1 and d2 are not in the same ATTree, but they are in the same detaching tree, then d2 can be protected in the same way as d1.
- Property 3: If destination d1 is the parent of d2, d1 and d2 are not in the same detaching tree, then d2 can be protected in the same way as d1 if reprotection is used.
- In summary, a destination can be protected in the same way as its parent.