Path Splicing

Murtaza Motiwala, Megan Elmore, Nick Feamster and Santosh Vempala
College of Computing, Georgia Tech
http://www.gtnoise.net/splicing

Presented by Alex Wang
CS 217B
UCLA, Spring 2009
What is Path Splicing?

• **Scalable** mechanism for providing nodes access to alternate paths

• **Main Idea:**
  – Generate alternate paths by running multiple routing protocol instances on the same topology
  – Allow traffic to switch between paths at intermediate hops
  – Enable end systems the control to switch paths
Why Path Splicing?

- **Multipath routing** typically computes $k$ edge-disjoint paths
  - Nodes can become disconnected if at least one link on each path fails, even if the graph is still connected

- **Path splicing** allows traffic to change paths at intermediate nodes
  - $K$ nodes must fail at **one cut** to create a disconnection
Outline

• Routing protocol design goals and metrics
• Intradomain Path Splicing
• Interdomain Path Splicing
• Evaluation Summary
Routing Protocol Design Goals

• **High reliability**
  – Maintain information about connectivity between pairs of nodes, even with failures

• **Fast recovery**
  – Allow end systems to discover and use alternate paths

• **Small stretch**
  – Alternate paths should not be much longer than the default path

• **Control to end systems**
  – End systems should have some control over the paths that traffic uses
Design Metrics

• **Reliability**
  
  – Need a more concrete notion than “operational” reliability
  
  – Measure reliability in terms of the fraction of node pairs that become disconnected when a fraction of edges fail. (Or vice versa, with edges to nodes)
  
  – Definition (for edges):
    
    • “For a given graph $G$, and any $0 \leq p \leq 1$, let $R(p)$ denote the fraction of node pairs that are disconnected when each edge fails independently with probability $p$.
    
    • Reliability is then represented as a function $y = R(x)$, where $x$ ranges from $0$ to $1$."


Design Metrics

• **Novelty**
  – Paths not necessarily completely node disjoint
  – Measure the diversity of paths by quantifying the fraction of edges between two paths that are distinct (Can also be applied to vertices)
  – Definition:
    • “Given a (source, destination) pair, let $P_s$ be the path with fewer edges and $P_l$ be the path with more edges.
    • Formally, novelty is $1 - \frac{|P_l \cap P_s|}{|P_s|}$”
Design Metrics

- **Recovery Time**
  - System should quickly, scalably and simply provide working paths to nodes during failure.
  - Measure the time it takes for a pair of nodes to establish a working path after a failure occurs.
  - Definition:
    - “Recovery time is the time that the routing system takes to re-establish connectivity between a (source, destination) pair after the existing path has failed.”
Design Metrics

• Stretch
  – Measure and quantify the additional latency incurred by alternate paths over the default path
  – Definition:
    • “Stretch is defined as the ratio of the latency on a path (between a pair of nodes) in the perturbed topology to the ratio of the shortest path (between the same pair of nodes) in the original topology.”
End System Control

• End systems should have some control over path selection, but not too much control
  – If a certain path is deemed to be non-functional or detrimental, an application may want to signal the use of a different path
  – Too much control may inhibit traffic engineering goals of network operators
Intradomain Path Splicing

• Create a set of “slices” for the network
  – “A slice is essentially a set of shortest path trees for a particular view of the network graph”

• Discover alternate paths for slices by creating routing trees based on random link-weight perturbations
Slices for Splicing

- Different slices reflecting different trees rooted at the same node
- Traffic can reach the node by traversing one or more trees
Implementation

• End systems insert a splicing header in between the network and transport headers
  – Independent decisions can be made at each hop

• Forwarding scheme:
  – Read the rightmost \( \log(k) \) bits to determine the forwarding table to use
  – Shift the bitstream right by \( \log(k) \) bits to allow subsequent hops to do the same operation

<table>
<thead>
<tr>
<th>IP Header</th>
<th>011001100 ...</th>
<th>Transport Header</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Each hop examines/removes \( \log(k) \) bits
Implementation

• Splicing bits carry *no explicit semantics*
  – End hosts don’t need to learn details of actual paths
  – Bits act as an opaque identifier for some path

• Incremental deployment
  – Routers with splicing can inspect the splicing header
  – Routers w/o splicing continue forwarding based on the IP header
Implementation

• When failure occurs, traffic must be redirected to a different slice
  – Easily performed by changing bits in the splicing header
  – Redirection can be performed by either end systems or intermediate nodes
Possible Optimizations

• Single routing protocol instance
  – Nodes generate variants of the topology locally to compute multiple forwarding tables

• Single forwarding table
  – One table for all slices, maintaining a separate column to record valid slices for an entry

• Embed splicing bits into the IP header
  – Utilize type-of-service and IP ID fields in IP header
Interdomain Path Splicing

• Each router learns one BGP route to each destination \textit{per session}

• Most BGP-speaking routers already have multiple BGP sessions to neighboring routers

• Existing routers may \textit{already} learn multiple diverse routes for each destination

• Insert the best $k$ routes for each destination, rather than just a single best route
Interdomain Path Splicing

- Use splicing bits to determine how to forward a packet in both ingress and egress routers
  - Ingress routers learn multiple paths to a destination from border routers using iBGP
  - Egress routers learn multiple routes to a destination from border routers of neighboring ASes via eBGP
Implementation

- Divide splicing bits into segments to separate interdomain and intradomain paths
- Interdomain paths must comply with ISPs’ business policies
  - “Policy” bit to limit how splicing bits can be used
Practical Concerns

• Forwarding loops
  – Finite loops can occur since a single path is built from multiple routing trees
  – Can be mitigated by limiting the number of times a packet can switch slices

• AS-level forwarding consistency
  – Violation of protocol semantics if non-announced routes are used
  – Non-default path used only if splicing bits explicitly specify it, or if default path fails
Evaluation Summary

• **Reliability with splicing approaches optimal**
  – For intradomain splicing, 5 slices and for interdomain splicing, only 2 slices achieve near-optimal reliability.

• **Splicing has fast recovery**
  – An end system can recover from failure in about 2 trials when trying splicing bits at random.

• **Perturbations achieve high novelty with low stretch**
  – Intradomain splicing has an average stretch of 20% while gaining 80% paths which are different from the original.
  – For interdomain, the average hop stretch is only 3.8% when 5% of AS links have failed.
Evaluation Summary

• Splicing provides better recovery than routing deflections
  – Path splicing with only 5 slices can provide better recovery than routing deflections with bounded stretch.
  – Path splicing generally provides much shorter recovered paths, and the recovered paths have much lower variance in terms of stretch.

• Splicing is incrementally deployable
  – Splicing offers significant benefits even if only a fraction of ASes deploy it.
Evaluation Summary

- **Loops are rare**
  - Forwarding loops are transient and infrequent.
  - In intradomain splicing, only 1 loop longer than 2 hops and *no persistent loops* were observed, even with 10% of links failed.

- **Splicing causes minimal disruption to traffic**
  - Splicing does not have much adverse effect on traffic in the network.
  - Evaluation using real traffic data on Abilene showed that total load on links increases only by 4% on average.
Open Issues

• Changes required for deployment
  – Routers
    • Forwarding plane changed to support multiple routes, and select a route based on splicing bits
  – End Systems
    • Path splicing relies on failure detection mechanisms before finding a new working path

• Adversarial Concerns
  – Attacker could potentially waste resources by setting splicing bits to induce loops
    • Arbitrary loops can’t be created
“Secret Sauce”? 

• Proposes splicing in general as a primitive before describing their implementation 
• Well-defined design goals 
• Targeted and thorough evaluation 
• Addresses possible concerns 
• Brushes off negative points well
Questions