Improving the Reliability of Internet Paths with One-hop Source Routing

K. Gummadi, H. Madhyastha, S. Gribble, H. Levy, D. Wetherall
U. Washington
Proc. of OSDI 2004
Motivation

- Increasing interest on Internet reliability
  - Way bellow the 99.999% (“five 9s”) expected in the public-switched telephone network

- Previous approaches
  - Server replication (through clustering or CDNs) – expensive and commonly limited to high-end web sites
  - Multihoming (multiple ISP links) – only protection against single-link failures
  - Overlay networks – most previous approaches (RON) required non-scalable background monitoring
Key issues explored

- Failure characteristics of Internet paths and implication about effectiveness of overlay routing as a solution
  - Explored through a 7-days, large-scale measurement using PlanetLab as vantage point

- Can this be done in a more scalable manner?
  - One-hop source routing with random selection of intermediary node

- What benefits would end-users see?
  - Build and evaluated a prototype with a simple web-browsing workload
Path failures

- Week long measurement study
  - Probed 3,153 destinations from 67 Planetlab sites
  - Each destination is probed from exactly one node
- Goal is to answer
  - How often do paths fail? Where do failures occur? How long do failures last?
- Use 3153 destinations:
  - 378 popular web servers
  - 1,139 broadband hosts
  - 1,636 randomly selected IPs
- Probing approach
  - TCP ACKs at different frequencies, customized traceroute
  - Failure: 3 probes lost & failed traceroute
How often do paths fail

- Failures do happen, but not frequently
  - On average each path sees 6 failures/week
  - Server paths see 4 failures/week
  - Broadband paths see 7 failures/week

- Most paths see at least one failure in a week
  - 85% of all paths
  - 78% of server paths
  - 88% of broadband paths
Where do they fail

- For popular servers, few last-hop failures – good for SOSR
- For broadband nodes, last-hop failures dominate
Path downtime and failure duration

- Failure durations are highly skewed
  - Majority of failures are short
  - Median, significantly better than average, failure duration: 1-2 min for all paths
  - Median path availability: 99.9% for all paths

- A significant fraction of paths see long failures
  - Tend to occur on last-hop
  - Mean path availability: 99.6% for servers and 94.4% for broadband

- Failure duration
  - ~11’ on paths to servers
  - 84’ on paths to broadband hosts
Implications about overlay-based fix

- Failures happen often enough that they are worth fixing
- But, they are rare enough that recovery schemes should be inexpensive under normal conditions
- Failures near the end-nodes limit the performance of indirection routing
  - good news: servers see very few failures near end hosts
  - bad news: broadband hosts see many last_hop failures
Potential of SOSR

- Source routing can help recover from 66% of all failures on paths to servers
- Problem, again, is last hop
- Highly effective for core failures

<table>
<thead>
<tr>
<th>percent of failures that are recoverable</th>
<th>servers</th>
<th>broadband</th>
</tr>
</thead>
<tbody>
<tr>
<td>src_side</td>
<td>54%</td>
<td>55%</td>
</tr>
<tr>
<td>core</td>
<td>92%</td>
<td>90%</td>
</tr>
<tr>
<td>dst_side</td>
<td>79%</td>
<td>66%</td>
</tr>
<tr>
<td>last_hop</td>
<td>41%</td>
<td>12%</td>
</tr>
<tr>
<td>all</td>
<td>66%</td>
<td>39%</td>
</tr>
</tbody>
</table>
One-hope source routing

- Use default path under normal conditions
- When default path fails, source attempts to recover by routing through an intermediary
- You may need more than one attempt
Number of useful intermediaries

- 81% of the recoverable failures could be recovered through at least $\frac{21}{39}$ intermediaries.
- Still, ~6% could only be recovered through 1-5 nodes.

![Graph showing cumulative distribution function (CDF) of failures recovered vs. number of useful intermediaries.]

MSIT Peer-to-Peer Computing
Northwestern University
Effectiveness of Random-k

- Random-k: Pick K intermediaries at random, try them all, choose the first one to get through
- Random-4 delivers near-optimal success rate
  - 61% of all failure, 92% of all recoverable failures
  - Requires no a priori probing or state
User perceived benefits

- SOSR recovers from 56% of network failures
- But, can't recover from application failures
- 62% of wget + SOSR failures are application related

<table>
<thead>
<tr>
<th></th>
<th>network level failures</th>
<th>application level failures</th>
<th>HTTP refused</th>
<th>HTTP refused</th>
<th>HTTP timeout</th>
<th>HTTP error codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>wget</td>
<td></td>
<td></td>
<td>328</td>
<td>40</td>
<td>78</td>
<td>35</td>
</tr>
<tr>
<td>wget SOSR</td>
<td></td>
<td></td>
<td>145</td>
<td>41</td>
<td>101</td>
<td>96</td>
</tr>
</tbody>
</table>
Conclusions

- Failures happen, but they are short and infrequent and many occur on last-hop for broadband paths
- Recovery must be cheap in the common case
- A simple schemes can work, Random-4, no probing, realizes the potential of any scheme
- Web users see only 20% fewer failures
Question 3

- *Could SOSR solve performance faults?*