SDProber: Software Defined Prober

Sivaramakrishnan Ramanathan\textsuperscript{1}, Yaron Kanza\textsuperscript{2} and
Balachander Krishnamurthy\textsuperscript{2}

\textsuperscript{1}University of Southern California
\textsuperscript{2}AT&T Labs Research
Delay Measurements

• Persistent delays in networks cause adverse effects
  • Disrupts quality of service in applications impacting revenue
• Delay measurements needs to be done constantly
• Trade-off between detection time and measurement cost
  • Lack of measurements increases detection time
  • Frequent measurements affect the network
• Network operator needs to balance between measurement cost and detection time
Measurements with bounds

- Bounding measurements can help in balancing the measurement cost and detection time to the network operator
  - **Lower bound** specifies the minimum number of inspections which needs to be performed on link
  - **Upper bound** limits the number of inspections performed on link
- Existing tools such as ping and traceroute cannot apply such bounds
  - Depends on the underlying routing algorithm which is **inflexible**
  - Finding the optimal solution with pre-defined path solution is **NP-hard**
SDProber – Software defined prober

• SDProber allows adaptable and efficient delay measurements in networks with bounded constraints
• SDProber uses probe packets to estimate the time taken for traversing every link
• Probes in SDProber take a pseudo-random walk in a weighted graph
  • Avoids complex computation
• Weights are adapted to satisfy rate constraints on links
  • send more probe packets to links where lower bounds are not satisfied
  • send less probe packets to links where upper bounds are satisfied
SDProber – System Overview

- Time of arrival from S1: $t_1$
- Time of arrival from S2: $t_2$
- Delay: $t_2 - t_1$

Diagram:
- S1
- S2
- Collector
- SDN Controller
- Probe Agent

Add mirroring/forwarding rules
SDProber – Pseudo random walk

• For every probe packet, the initial starting node and each traversing link are selected randomly

• By altering the initial node weights or weights on choosing the next node, SDProber can control how probe packets inspect the network
  • Easily adapts to changes in probing constraints or network
  • Reduces costs

• Implementation of Random walk is done using Openvswitches group tables and forwarding rules
SDProber – Pseudo random walk

- SDProber uses **binary exponential backoff** to adjust weights
- Initial node weights and link weights are adjusted
  - Doubled/halved when probing rates are less/more than constraint
Evaluation: Competing approaches

- Two approaches that use shortest path to send probe packets
- In each approach, the probe packet is mirrored at every node it traverses

Random Pair Selection (RPS)
- For every iteration, source and destination pairs are selected randomly and probe packets are sent through the shortest path between them
- At every iteration, source and destination pairs are selected from pairs which have not been selected before

Greedy Path Selection
- At each iteration, pairs of source and destination are selected such that the sum of min-rate values of all unvisited links on paths is maximum
Evaluation: Setup

• Tested on 196 nodes + 243 links real topology
• Probing iteration was launched every 30 seconds
• Evaluated results on three probing profiles (small, medium, large) having different min-rate, max-rate
  • Network operators could choose probing rates based on SLA or historically analyzing delays
Evaluation: Control over inspection

- Red line indicates the probability of traversing through link
- Blue line is the number of probe packets which traversed each link
  - Error is ±10% from expected value
Evaluation: Cost effectiveness

• For each method, we increased the number of emitted probe packets per iteration till min-rate constraints are satisfied
  • SDProber sends 4—12 times fewer probe packets than RPS and greedy
  • While satisfying min-rates on all links, SDProber sends 10—62 times fewer excess probe packets than RPS and greedy
Conclusion

• SDProber provides an efficient and flexible delay measurements with measurement constraints on inspection rates
• SDProber uses probe packets to estimate delay on links
• Probes take a pseudo random walk
• Weights are adapted using binary exponential backoff to satisfy inspection rate constraints
• Evaluated SDProber on a real world ISP topology to show SDProber’s control over probe packets and cost effectiveness
Evaluation: Detection time

- SDProber detects delays twice as fast as greedy
  - Links with low weights are visited last in greedy
- SDProber and RPS have comparable detection time
  - But RPS sends more packets to satisfy rate constraints
Evaluation: Learning

- Varied % of historically delayed links
- When there are more historically delayed links, increasing $\alpha$ reduces detection time by 2—6%
- When there are no historically delayed links, increasing alpha increases detection time by 4%

![Bar chart showing time (s) for different percentages of FDL and values of $\alpha$.]
Choosing probing profiles

• Using SLA associated with customers
  • 99.9% uptime equates to 45 minutes of down time per month
  • Network operators can set the min-rate of probing such that the delayed links are detected with guarantees

• Using historical delay data
  • Links which have history of congestion can be probed more
Guarantees and convergence

• SDProber attempts to satisfy rate constraints with minimum violations given several parameters (TTL, packets available per iteration)

• Inspecting each link can provide tighter guarantees
  • But is expensive, requires more probe agents and is inefficient

• Random walk provides guarantees on inspection, provided there are enough probe packets per iteration

• Binary exponential backoff helps in expediting the satisfaction of constraints
  • There is no convergence – measurements are continuous and constraints are used as a driving factor for faster detection with low costs
Timestamp

• SDProber detects persistent delays in WAN where delays are usually in milliseconds

• Delay between switches and collector can be estimated using ping
  • Delays on link could therefore be bounded
  • Using historical measurements, delays greater than a particular threshold can be alerted to the network operator

• Timestamping can be done on packets using INT
  • Requires that there is clock synchronization at all switches