Consensus Routing: The Internet as a Distributed System

J. John, E. Katz-Bassett, A. Krishnamurthy, T. Anderson and A. Venkataramani*

U. Washington and *U. Massachusetts, Amherst

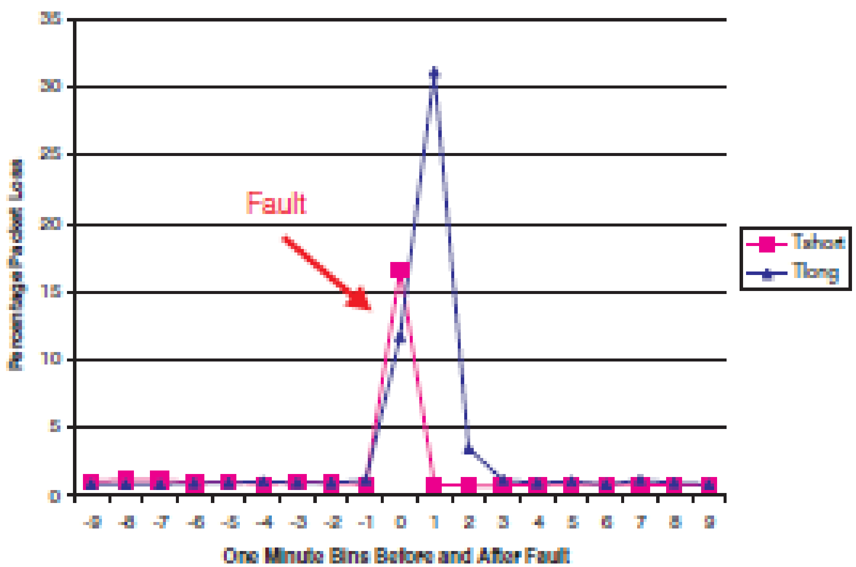
NSDI 2008 *Best Paper Award*

Motivation

- Internet routing traditionally favored responsiveness over consistency
  - How quickly the network reacts to changes, over ensuring packets traverse adopted routes
  - Router applies received updates immediately to its forwarding table before propagating it to others
- Responsiveness comes at the cost of availability
  - *A* things its route to a destination is via *B*, but *B* disagrees either
    - because *B*'s old route to that destination is via *A*, causing loops
    - because *B* does not have a current route to the destination, causing blackholes

Motivation

- BGP updates are known to cause up to 30% packet-losses for 2' after a routing change, even though physically routes exist



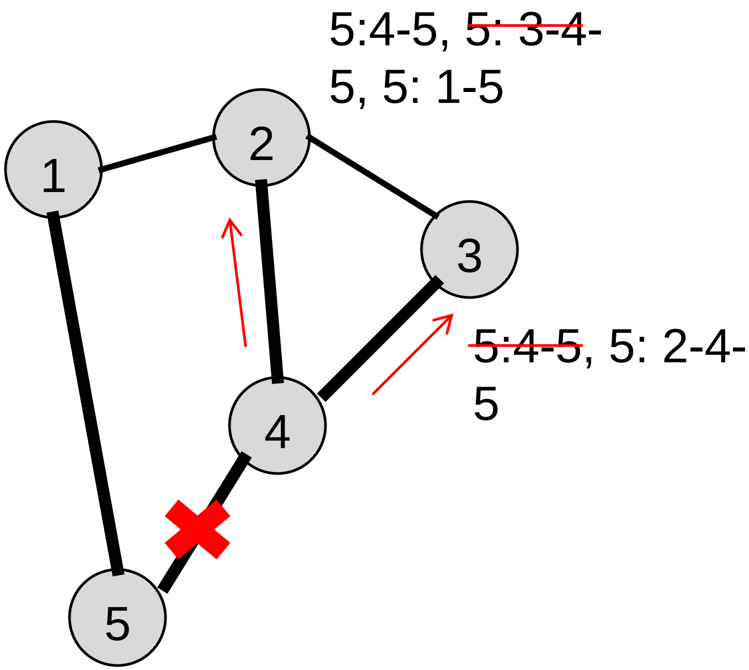
Average percentage of end-to-end loss of 512B ICMP packets to 100 web sites every second during the 10’ following two events (route updates)

Labovitz et al., *Delayed Internet routing convergence*, SIGCOMM 2000

- Transient loops account for 90% of all packet loss according to a Sprint network study

A routing loop example

2 (3) prefers the path through 3 (2)



Link failure causing BPG loops at 2 and 3

MRAI (Minimal Route Advertisement Interval) timer prevents 2 and 3 from advertising the new adopted paths

When timer expires, both discover the alternate paths through 1 that existed all along

Bold lines are selected paths

Consensus routing

- A *consistency first* approach to routing that cleanly separates safety and liveness concerns
  - Safety (*nothing bad ever happens*)
    - All the routers use a consistent route towards a destination
  - Liveness (*something good eventually happens*)
    - System reacts quickly to failures or policy changes
- To ensure both
  - Run a distributed coordination algorithm to ensure globally consistent view of routing state
  - Forward packets using one of two logically distinct modes
    - Stable – only use consistent routes
    - Transient – heuristically forward packets when no stable route is available

Stable mode

- Upon receiving an update, do not immediately adopt it
  - Processes it using its policy engine and logs the new route, then forwards it to its neighbors
- Periodically, all routers engage in a coordination algorithm to determine the most recent set of complete updates
  - Based on Chandy-Lamport snapshot algorithm
  - Lamport’s Paxos consensus algorithm
- Routers use output to compute a set of *stable forwarding tables (SFT)*

Stable mode

- Coordination proceeds in epochs, ensuring that in each one, all ASes have a consistent set of SFTs
- The k^{th} epoch consists of
  1. Update log – Routers process and log route updates (w/o modifying SFT)
  2. Distributed snapshot – ASes take a distributed snapshot
  3. Frontier computation
     1. Aggregation – ASes send snapshots to consolidators
     2. Consensus – Consolidators run Paxos to agree upon a global view and set of updates globally incomplete (*I*)
     3. Flood – Consolidators flood *I* and set of ASes, *S*, that successfully responded to the snapshot
  4. SFT computation – Each AS computes next SFT
  5. View change – Routers maintain current and previous SFT and marks forwarded packets

Stable mode – 1.Update log

- Routers maintain
  - Routing Information Base (RIB) – including, for each prefix, the most recent update, locally selected best route, and route advertised to each neighbor
  - History – for each prefix a chronological list of received and selected routes
  - Stable Forwarding Table – for each prefix the next-hop interfaces corresponding to the stable routes

Stable mode – 1.Update log

- Consensus routing maintains the invariant
  - if a router *A* adopts a new route to a dest, all routers that had received the update through *A* have processed the update
- Triggers – used to maintain the invariant
  - A GID for a set of causally related events propagating through the network
  - A tuple *(originating as number, trigger number)*
  - In BGP, each updates announces a route and implicitly withdraws a previous one; triggers track the withdrawal
  - To ensure consistency of routes, AS does not adopt a new route until it knows that the trigger associated with the update is complete

Stable mode – 2. Distributed snapshot

- To transition between epochs, take a snapshot
- Local state at A consist of
  - Sequence of triggers in A's history
  - Set of incomplete updates
    - Incomplete because the update is being processed by the AS
    - AS is waiting for update to a neighboring AS (for MRAI to expire)
    - The update is in transit from a neighboring AS
- Use Chandy-Lamport to take snapshot
  - To initiate a snapshot, save local state and send maker to all neighbors
  - Upon receiving a marker on channel *c*
    - If it hasn't recorded state, do that, and record state of *c* as empty
    - Record state of *c* as sequence of messages received on *c* after recordings its state and before receiving the marker

Stable mode – 3. Frontier computation

- After snapshot, each AS sends it to all consolidators
  - Snapshot report – set of incomplete triggers and saved sequence of triggers
- Consensus
  - Consolidators wait for bit, then exchange snapshot reports
  - Run Paxos to agree upon the set of ASes *S* by exchanging snapshot reports
  - After consensus, each computes *I*, the consolidated set of incomplete triggers in the network
    - A trigger *t* is incomplete if neither *t* nor any trigger it depends on is incomplete
    - A trigger is incomplete if present incomplete in some node
- Flood – Consolidators flood *I* and set of ASes, *S*, that successfully responded to the snapshot

Stable mode – 4. SFT computation

- After receiving *I* , each AS builds a new SFT
  1. Save current SFT
  2. For each destination prefix *p*
     1. Find the latest selected update *u = (t,r)* in *p*'s *History* such that *t* is complete
     2. Adopt *r* as the route to *p* in the new SFT
     3. Drop all records before *u* from *p*'s *History*
- If any adopted path contains an AS whose snapshot was excluded by consensus, the corresponding route is replaced by *null* in the SFT

Stable mode – 5.View change

- The end of this process marks the end of epoch k^{th} and the beginning of $(k+1)^{th}$
- Since there are no synch clocks, ASes maintain and use both SFTs
- For packet forwarding
  - Once a router has computed the new $(k+1)^{th}$ SFT, it starts forwarding routes along the new routes
  - If a packet reaches a router that has not finished computing $(k+1)^{th}$ SFT, the router sets a bit in the packet header and everybody routes using k^{th} SFT from then on
  - This ensure loop-free forwarding
  - If you get a package routed using an older SFT, treat it as if the corresponding route were *null*

Transient mode

- Forwarding switches to transient mode when no stable route is available
  - Due to failure of next-hop router
  - A no-null route has not yet propagated or some router was slow to submit snapshot report
- Uses different schemes to handle this
  - Routing deflection
  - Detour routing
  - Backup routes
- Consensus routing provides a mechanism that reliable indicate when to switch to transient and back, allows different schemes to co-exist

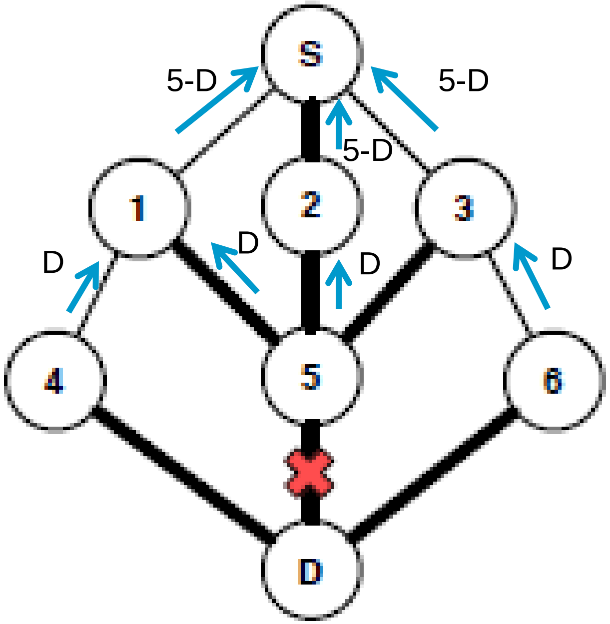
Routing deflections

- When packet finds a failed link
  - Router deflects packet to a neighboring AS after consulting its RIB to identify one that announced a different valid route to destination
  - If no neighboring AS has announced one, backtrack
- Still, this is not enough to ensure reachability
  - You still need the other schemes

1: D:1-4-D, *1-5-D*

S: D: S-1-5-D, S-2-5-D, S-3-5-D

All routes go through 5-D!



Other transient schemes

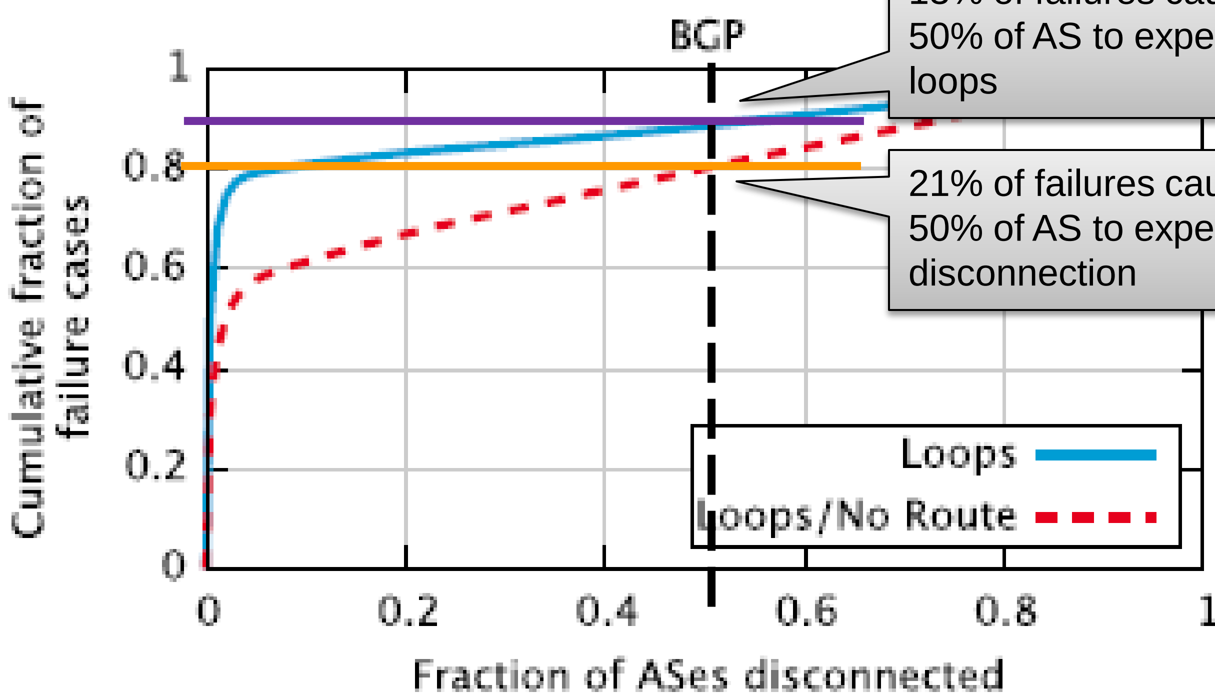
- Detour routing
  - After finding a failed link, select a neighboring AS and tunnels transient packets to it
  - If detouring AS is a Tier-1, high chance of delivering the packet
  - A new business model?
- Backup routes
  - Use pre-computed backup path to forward the packet (one approach to compute them: RBGP)

Evaluation

- Simulation
  - CAIDA AS-level graphs gathered from RouteViews BGP tables
    - Links annotated with inferred business relationships
  - Simulate route selection and exchange of route updates accounting for MRAI timers
  - Use standard "valley free" export policies and follow standard route selection criteria (customer > peers > providers)
- Using XOPR to measure implementation overhead
- Using PlanetLab and simulation to measure cost of consensus

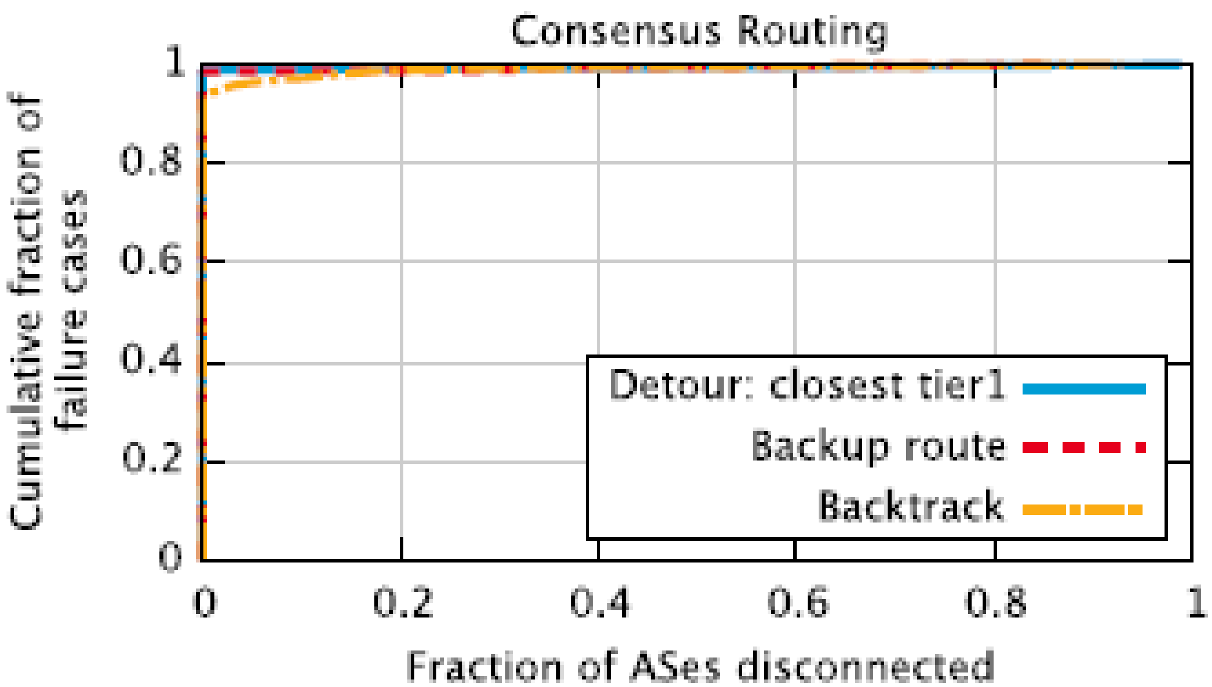
Link failures

- After reaching stable state, fail one link of a multi-homed stub AS
  - Multi-home stub AS – one with 1+ provider and no customers
    - Why? There's a valid physical route after one link fails
  - For each failure, fraction of AS disconnected at some point



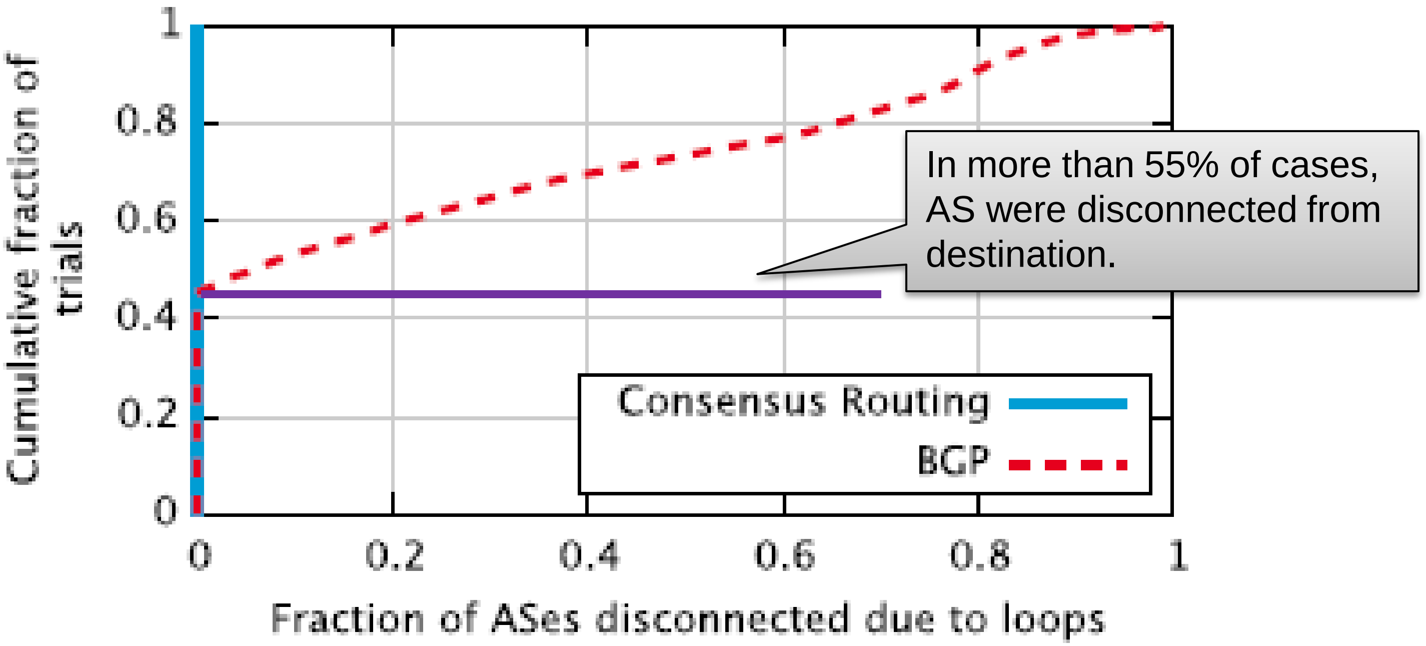
Link failures

- Consensus routing with different transient forwarding schemes
  - Simplest form, backtracking, enable continuous connectivity to at least 74% of ASes following 99% failures
  - Detouring/backup route maintains complete connectivity following 98.5/98% of failures



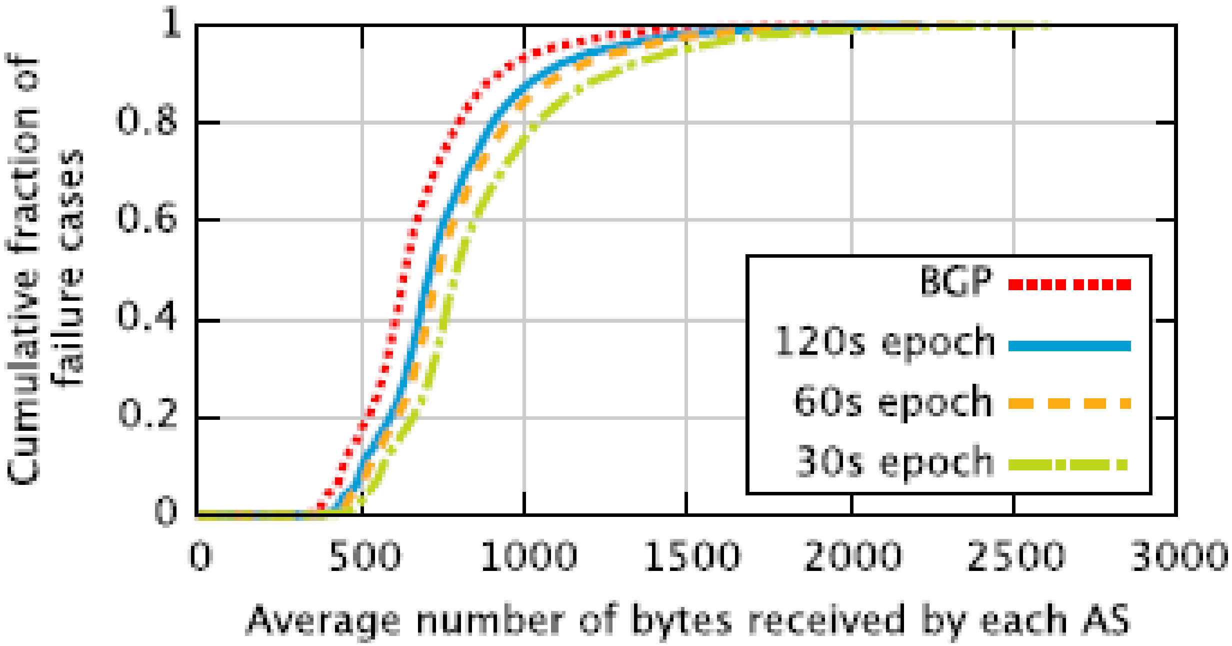
Traffic engineering

- Subprefix-based traffic engineering using ASes with 3+ providers
- In each run, pick one AS and one of its providers and withdraw the subprefix from each of the other providers
- Consensus routing transitions from one consistent state to another, avoiding transient loops



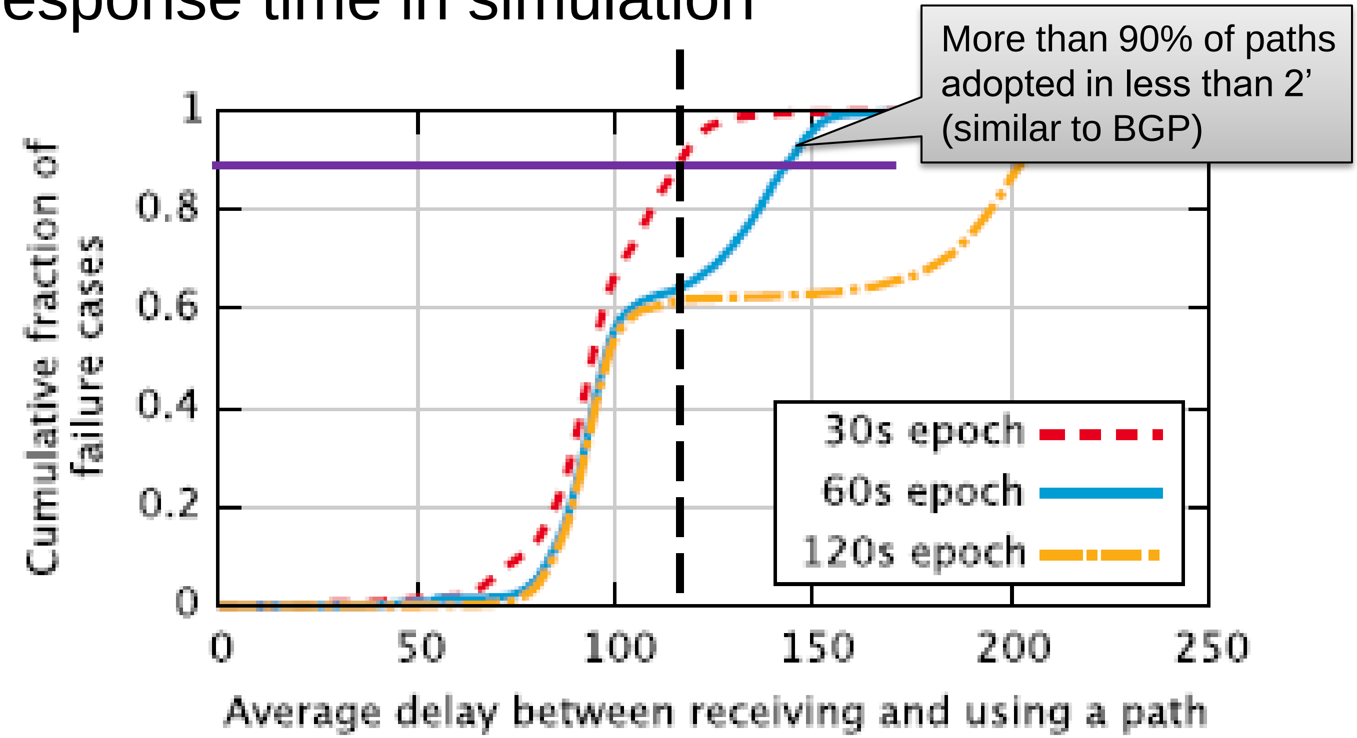
Overhead – additional traffic

- Consensus routing needs extra control traffic to take a distributed snapshot & flood incomplete triggers
  - Negligibly overhead due to BGP large updates



Overhead - time

- Consolidators have to reach an agreement on the set of snapshots that will be considered for computing SFTs
- Response time in simulation



Summary

- There's a general agreement on the need for higher availability
- Simply waiting for things to get better won't do; any BGP-like protocol is fundamentally susceptible to long periods of convergence
- Consensus routing aims toward improved availability by applying classical distributed systems concepts