Towards a Highly Available Internet

Tom Anderson
University of Washington

Joint work with: John P. John, Ethan Katz-Bassett, Dave Choffnes, Colin Dixon, Arvind Krishnamurthy, Harsha Madhyastha, Colin Scott, Justine Sherry, Arun Venkataramani, and David Wetherall

Financial support from: NSF, Cisco, Intel, and Google

Internet-based real-time health?

Continuous Blood Glucose Monitor

Glucose
Measurement

Compare with
trend, history
for this patient,
history for
others...

Insulin Infusion Pump

Insulin Dosage

Internet Routing

Primary goal of the Internet is availability

- “There is only one failure, and it is complete partition”
  Clark, *Design Philosophy of the Internet Protocols*

Physical path => route

route => efficient data path

efficient data path => data flows

Internet routing today

Physical path =X> route

- 10-15% of BGP updates cause loops and inconsistent routing tables
- Loops account for 90% of all packet losses in core

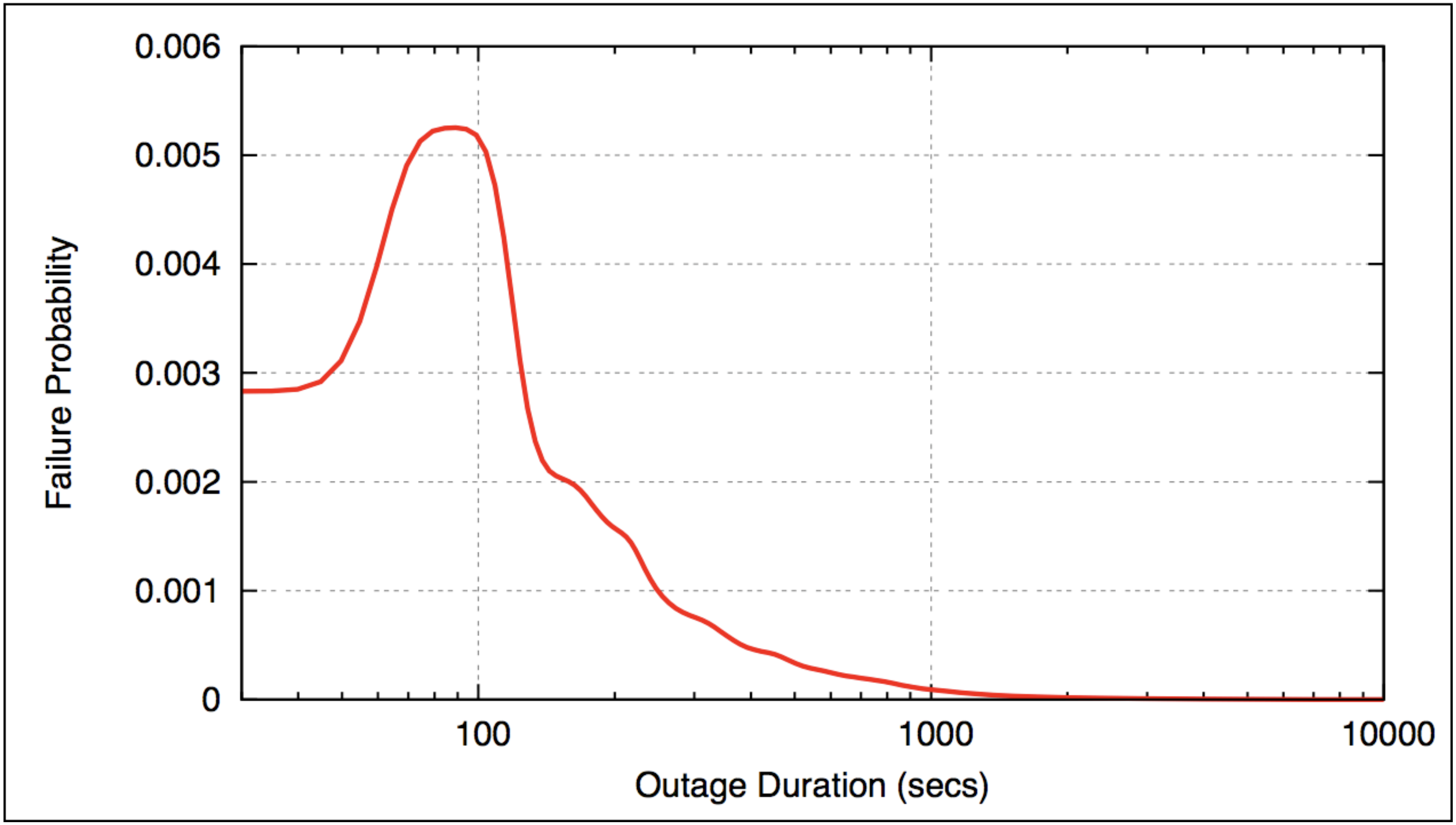
Route =X> efficient data path

- 40% of Google clients have > 400ms RTT

Efficient data path =X> data flows

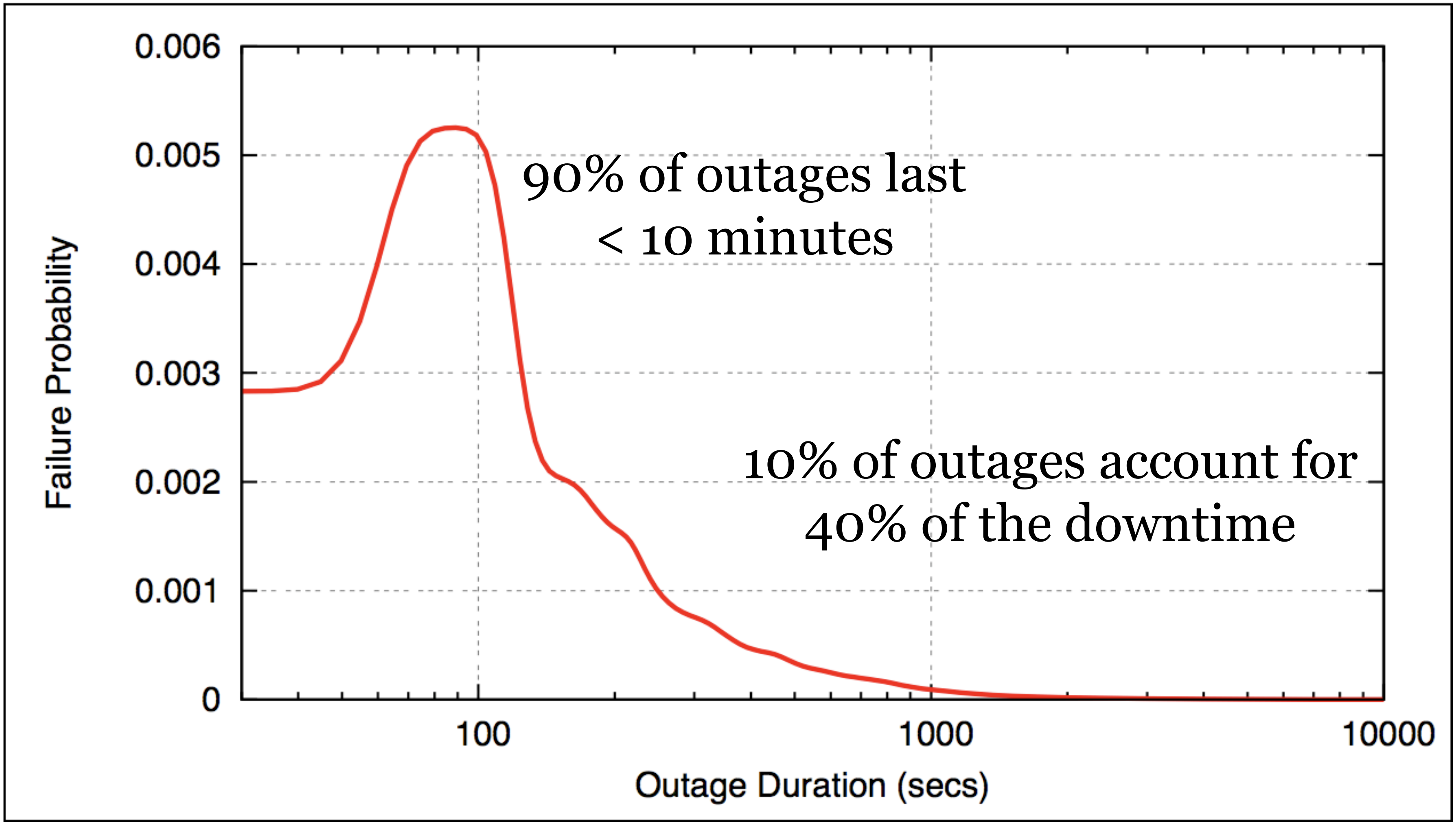
- Large scale botnets => almost every service vulnerable to large scale Internet denial of service attacks

Characterizing Internet Outages



Two month study: more than 2M outages

Characterizing Internet Outages



Two month study: more than 2M outages

Roadmap

Brief primer on Internet routing

Interdomain routing convergence (consensus routing)

- Towards high availability at a fine-grained time scale [NSDI 08]

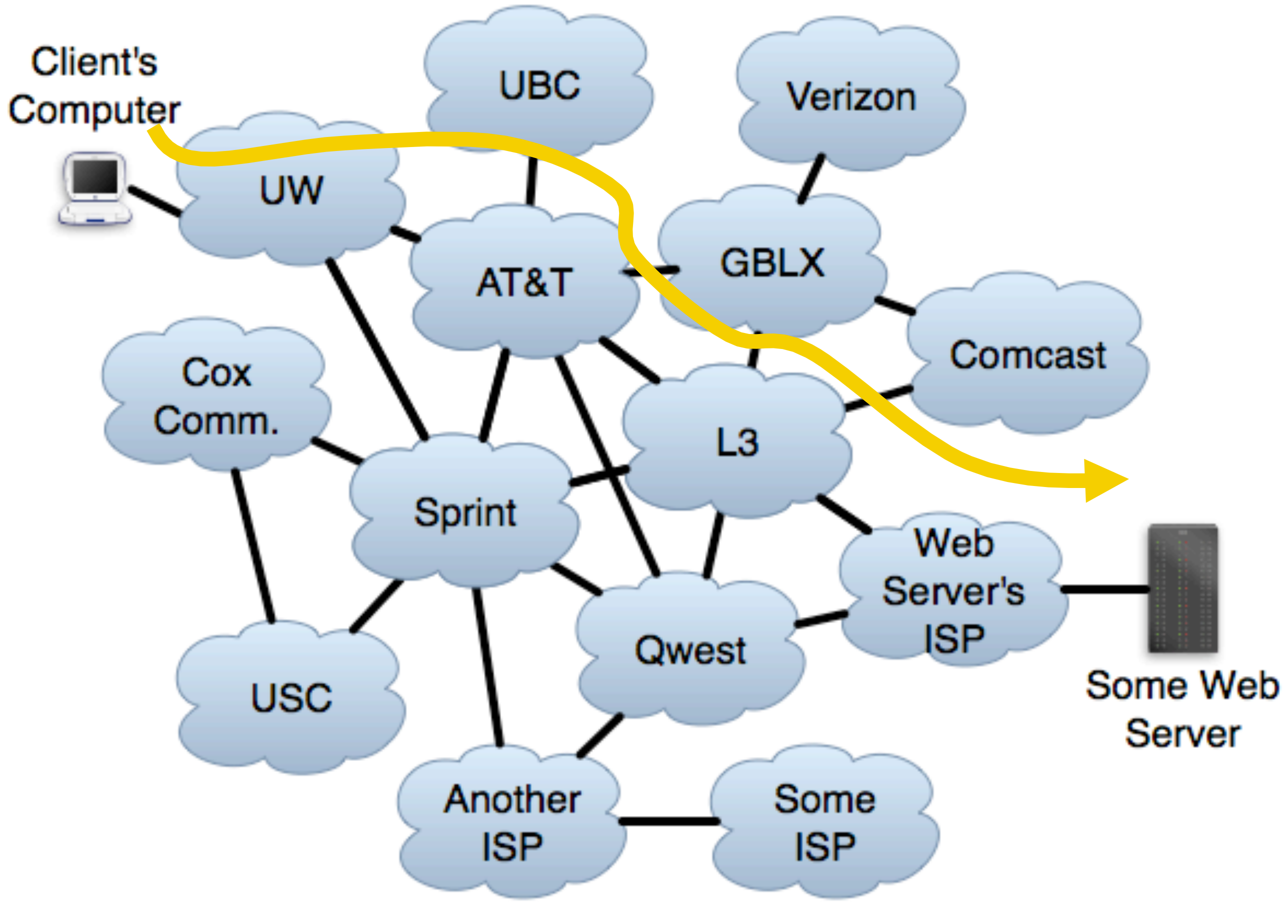
Interdomain routing diagnosis (Hubble/reverse traceroute)

- Towards high availability at a long time scale [NSDI 08, NSDI 10]

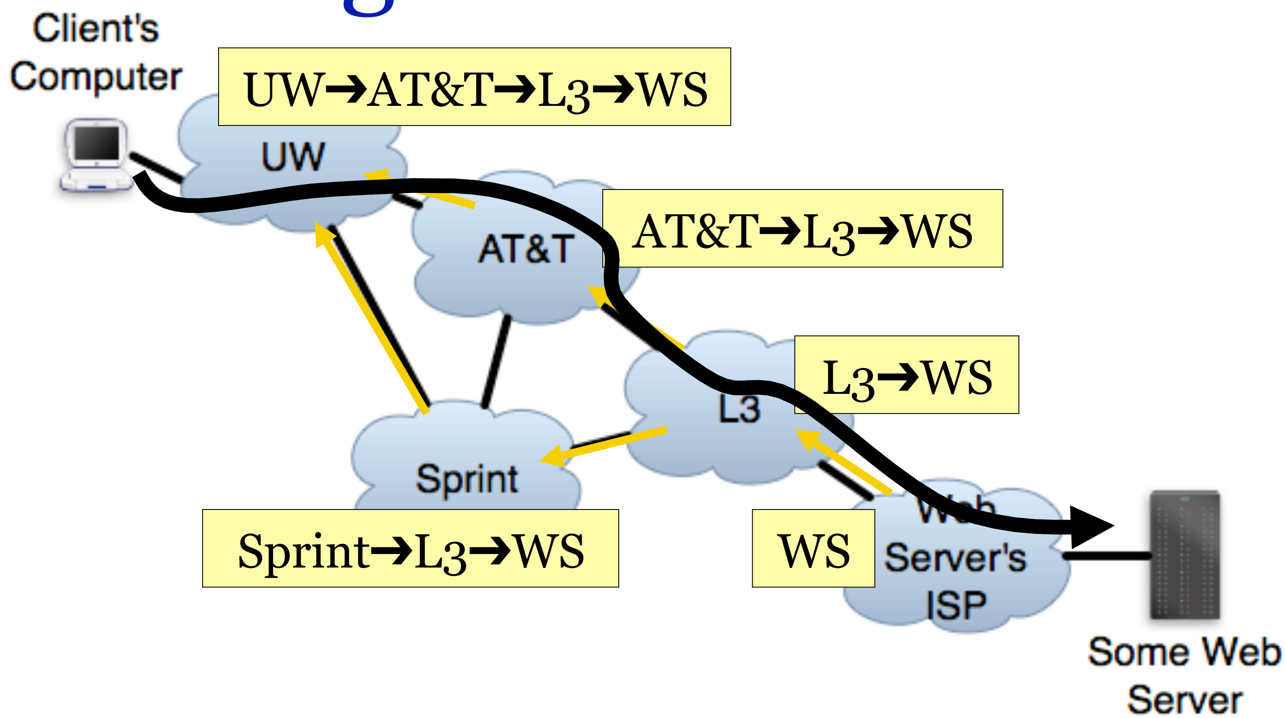
Distributed denial of service protection (phalanx)

- Towards withstanding million node botnets [NSDI 08]

Federation of Autonomous Networks



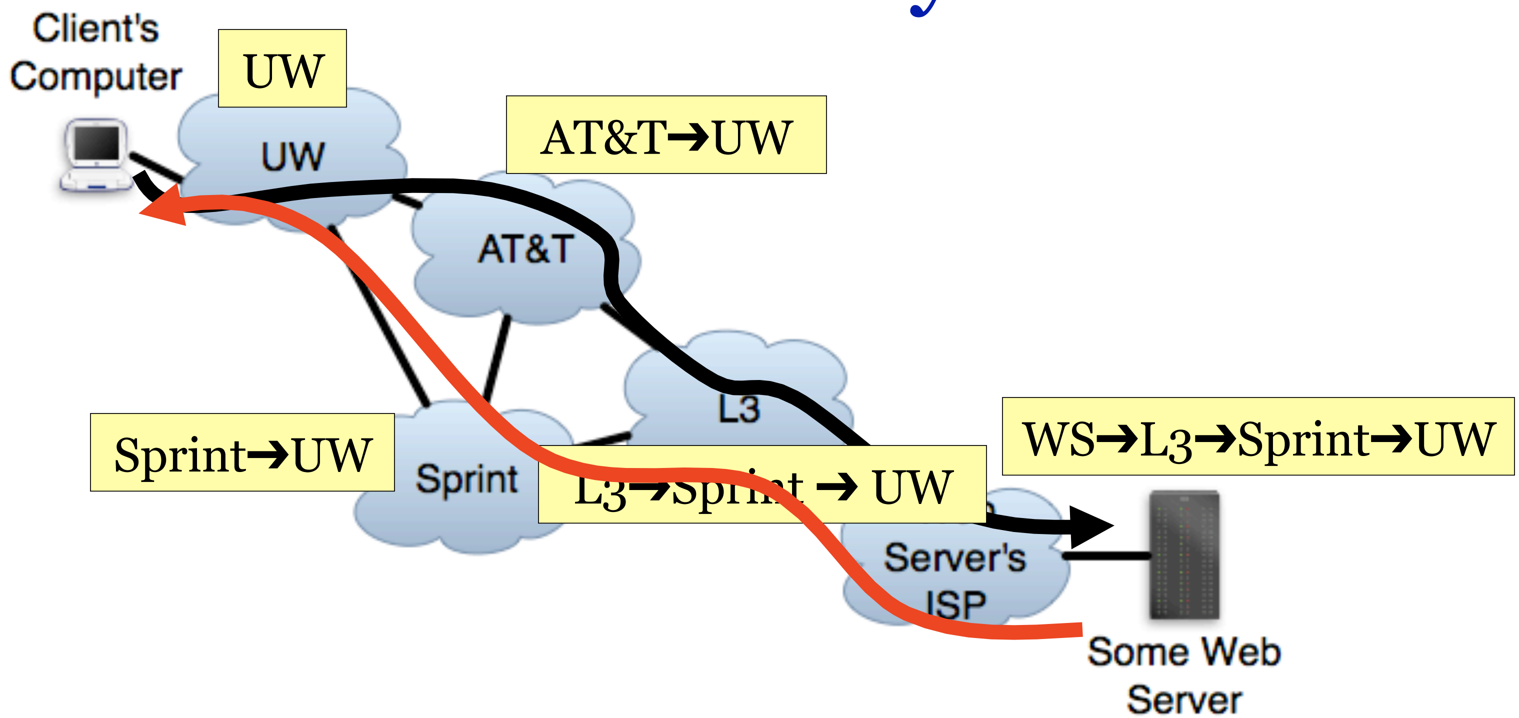
Establishing Inter-Network Routes



Border Gateway Protocol (BGP)

- Internet's interdomain routing protocol
- Network chooses path based on its own opaque policy
- Forward your preferred path to neighbors

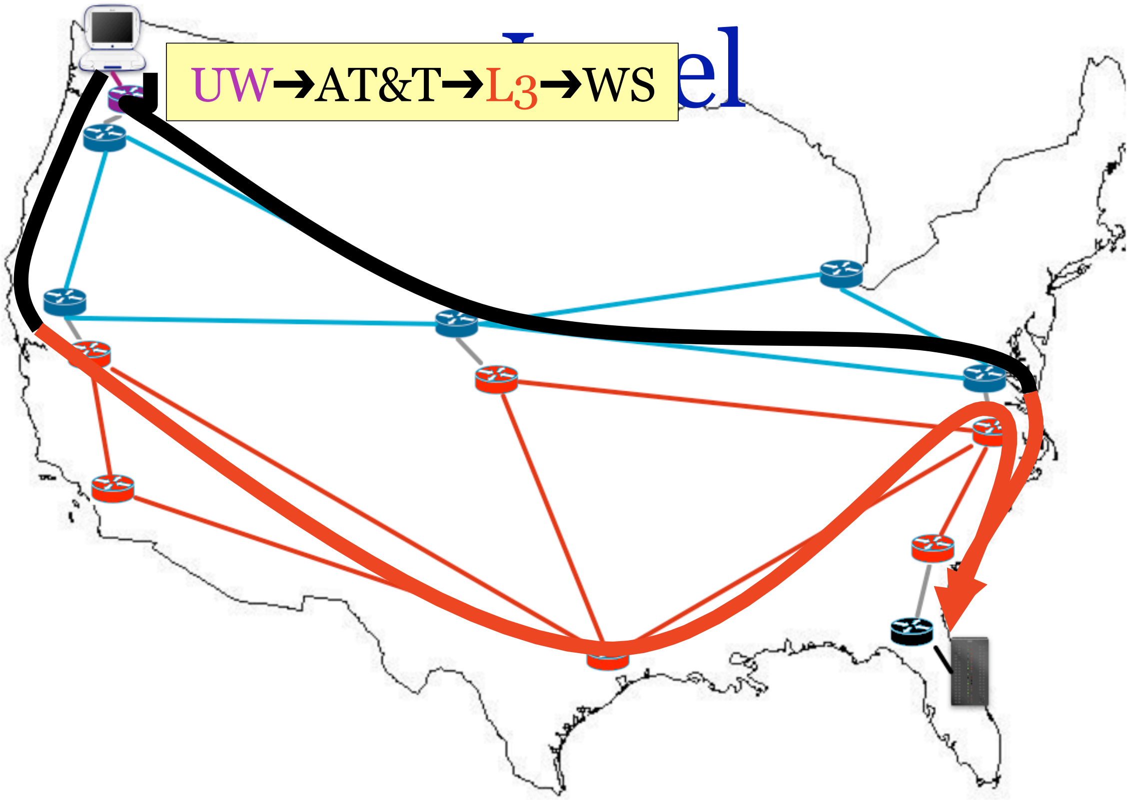
BGP Paths Can Be Asymmetric



Asymmetric paths are a consequence of policy

- Available paths depend on policy at other networks
- Network chooses path based on its own opaque policy ($$)
- Allowing policy-based decisions leads to asymmetry

From Interdomain Path to Router-Level



- Each ISP decides how to route across its network and where to hand traffic to next ISP
- End-to-end depends on interdomain + intradomain
  - Performance and availability stem from these decisions

Roadmap

Brief primer on Internet routing

Interdomain routing convergence (consensus routing)

- *Towards high availability at a fine-grained time scale [NSDI 08]*

Interdomain routing diagnosis (Hubble/reverse traceroute)

- Towards high availability at a long time scale [NSDI 08, NSDI 10]

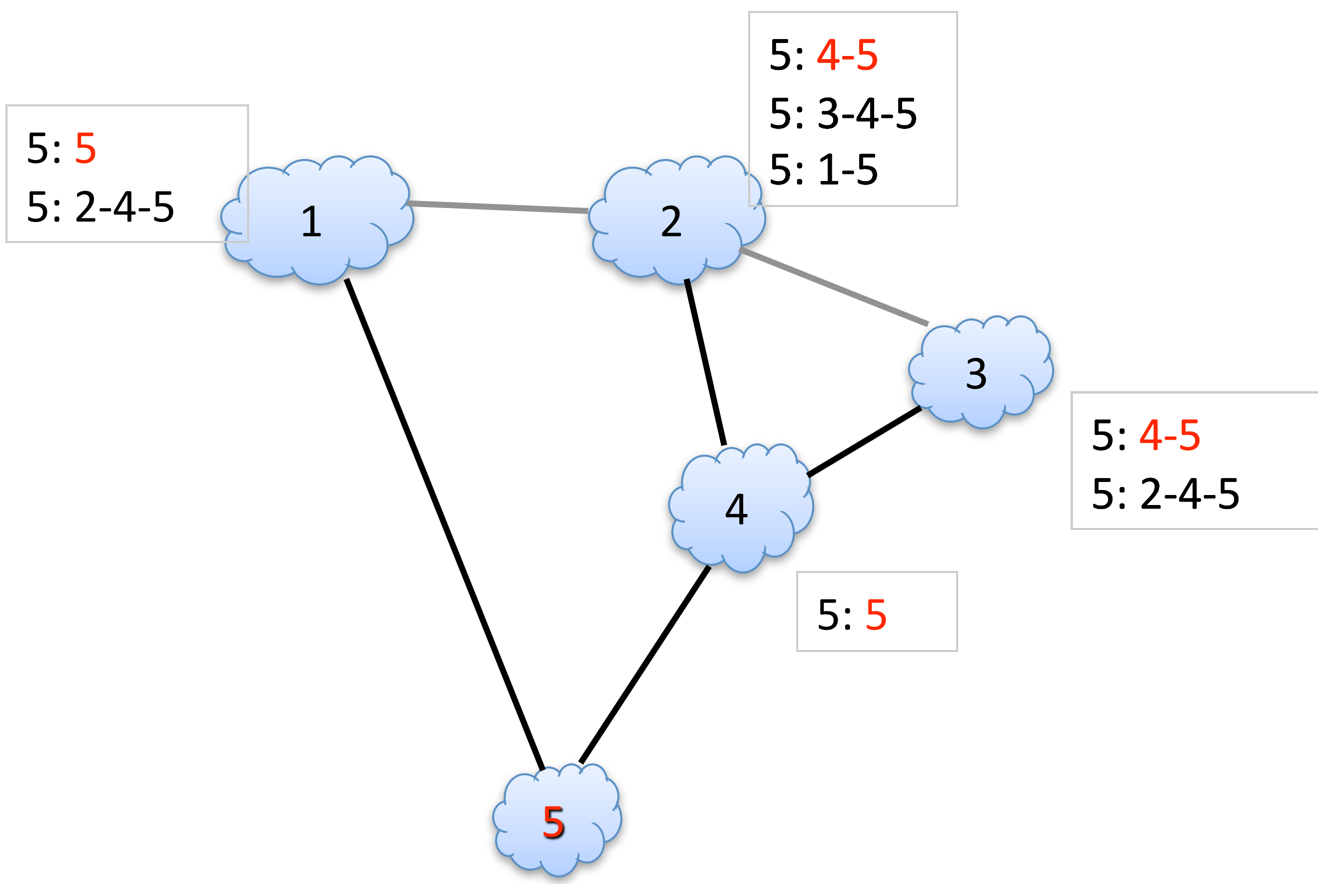
Distributed denial of service protection (phalanx)

- Towards withstanding million node botnets [NSDI 08]

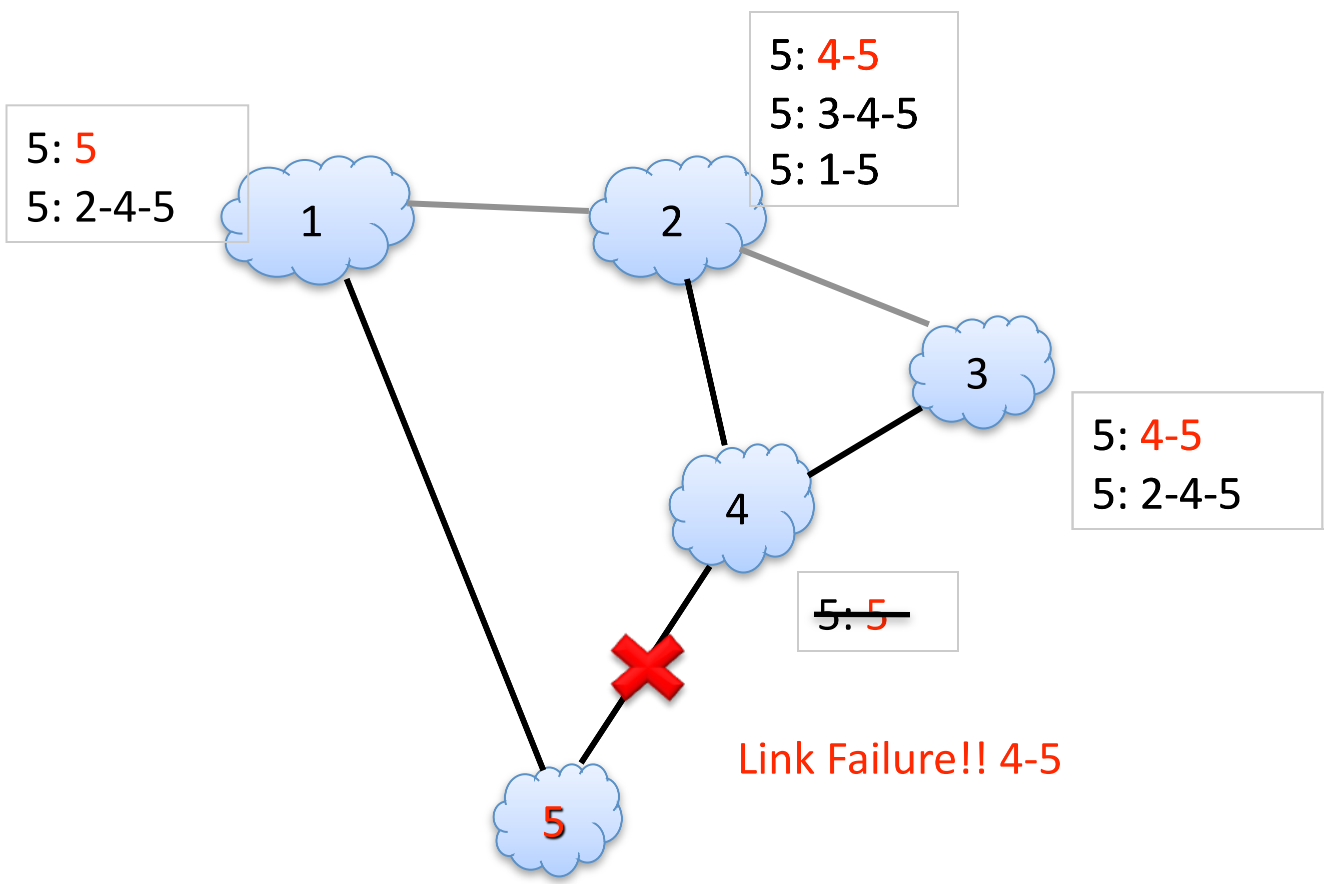
Border Gateway Protocol

- Key idea: *opaque policy routing* under local control
  - Preferred routes visible to neighbors
  - Underlying policies are not visible
- Mechanism:
  - ASes send their most preferred path (to each IP prefix) to neighboring ASes
  - If an AS receives a new path, *start using it right away*
  - Forward the path to neighbors, with a *minimum inter-message interval*
    - essential to prevent exponential message blowup
  - Path eventually propagates in this fashion to all AS's

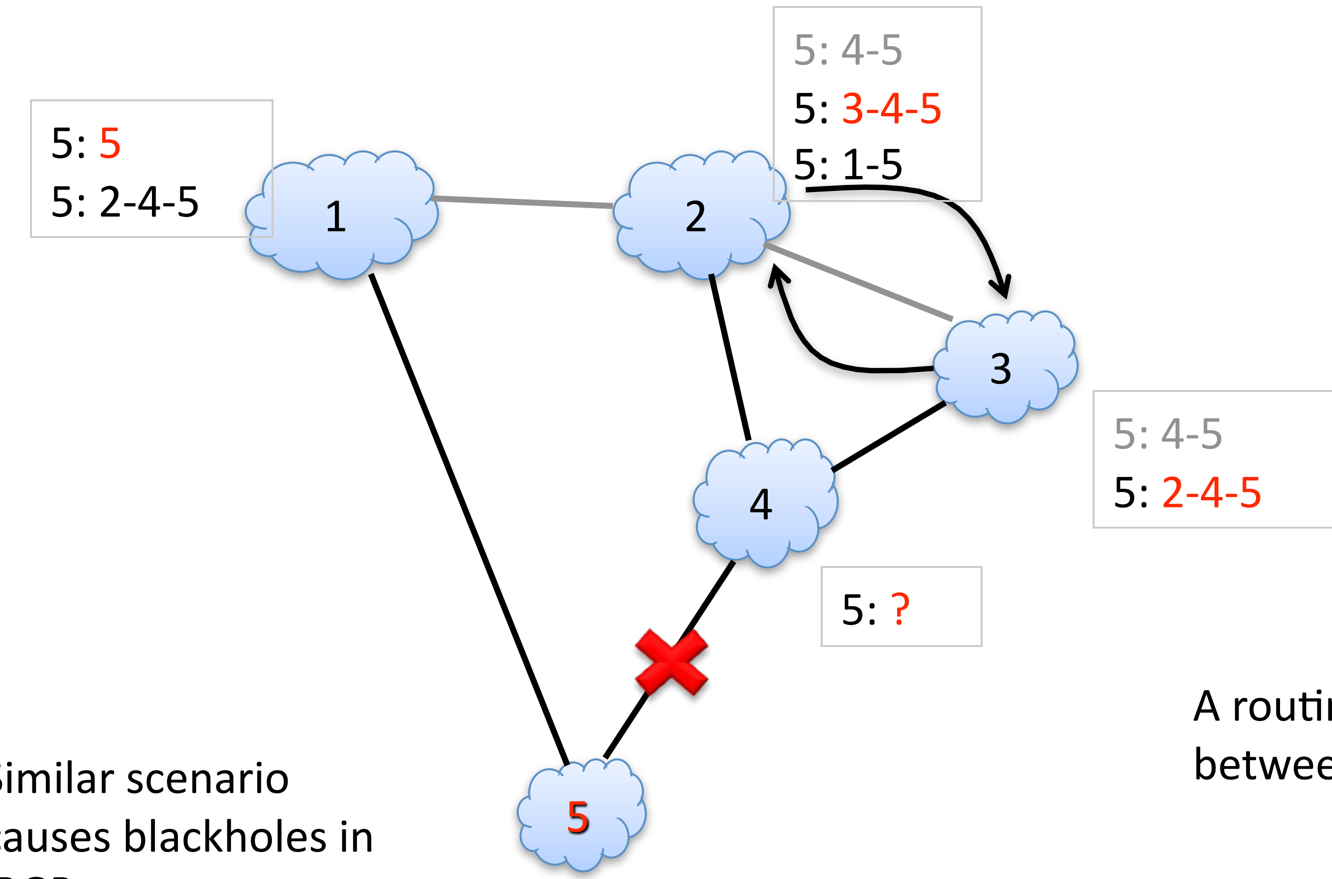
Failures Cause Loops in BGP



Failures Cause Loops in BGP



Failures Cause Loops in BGP

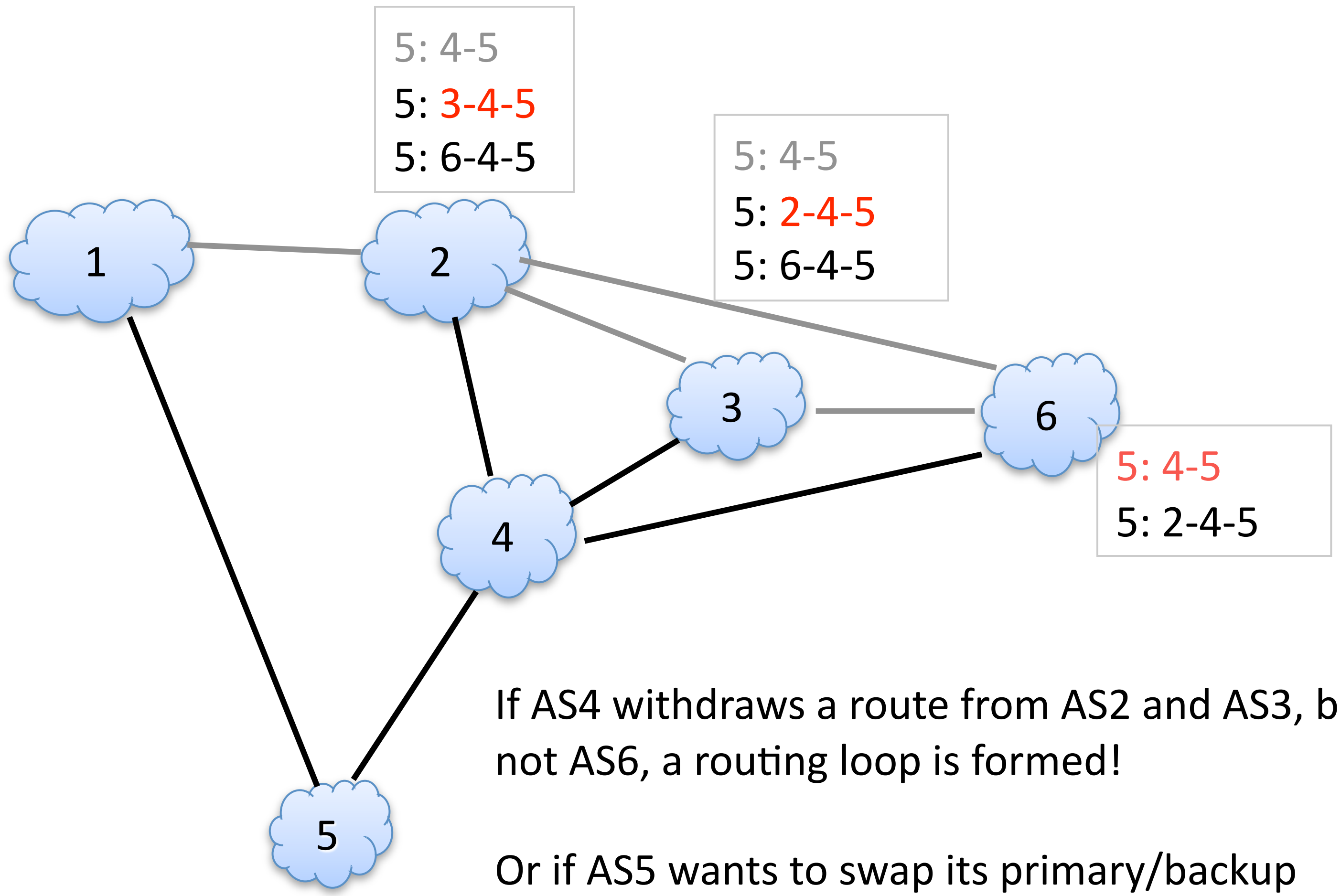


AS2 and AS3 now switch to next best path

A routing loop is formed between AS2 and AS3!

Similar scenario causes blackholes in iBGP

Policy Changes Cause Loops in BGP



If AS4 withdraws a route from AS2 and AS3, but not AS6, a routing loop is formed!

Or if AS5 wants to swap its primary/backup provider from 4 -> 1, or 1->4, a loop is formed

The Internet as a Distributed System

BGP mixes liveness and safety:

- Liveness: routes are available quickly after a change
- Safety: only policy compliant routes are used

BGP achieves neither!

- Messages are delayed to avoid exponential blowup
- Updates are applied asynchronously, forming temporary loops and blackholes

This is a distributed state management problem!

Consensus Routing

Separate concerns of liveness and safety

- Different mechanism is appropriate for each

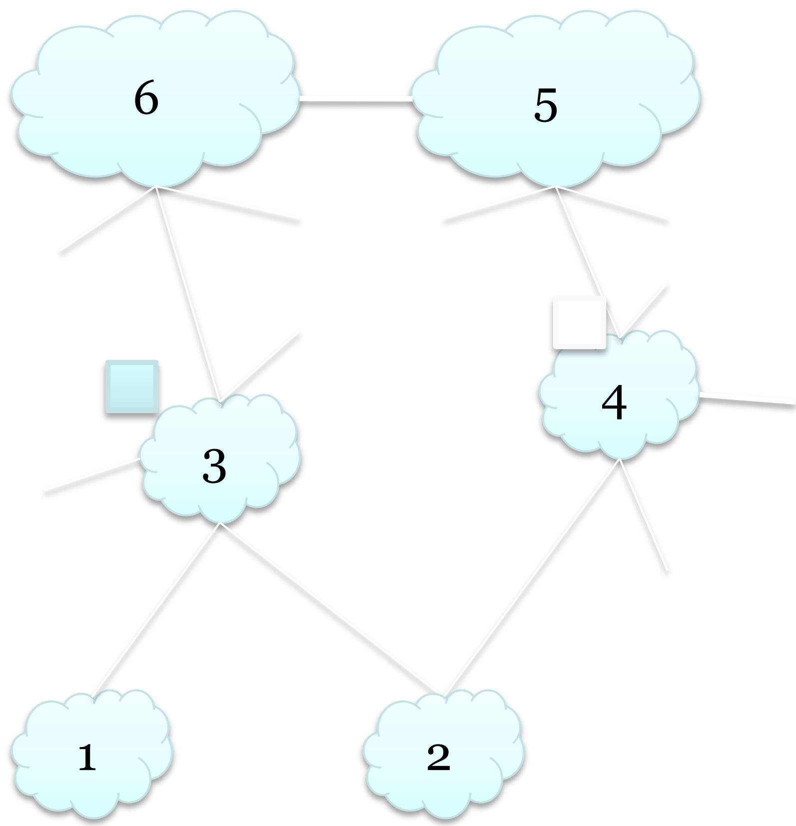
Liveness: routing system adapts to failures quickly

- Dynamically re-route around problem using known, stable routes (e.g., with backup paths or tunnels)

Safety: forwarding tables are always consistent and policy compliant

- AS's compute and forward routes as before, including timers to reduce message overhead
- Only apply updates that have reached everywhere
- Apply updates at the same time everywhere

Mechanism

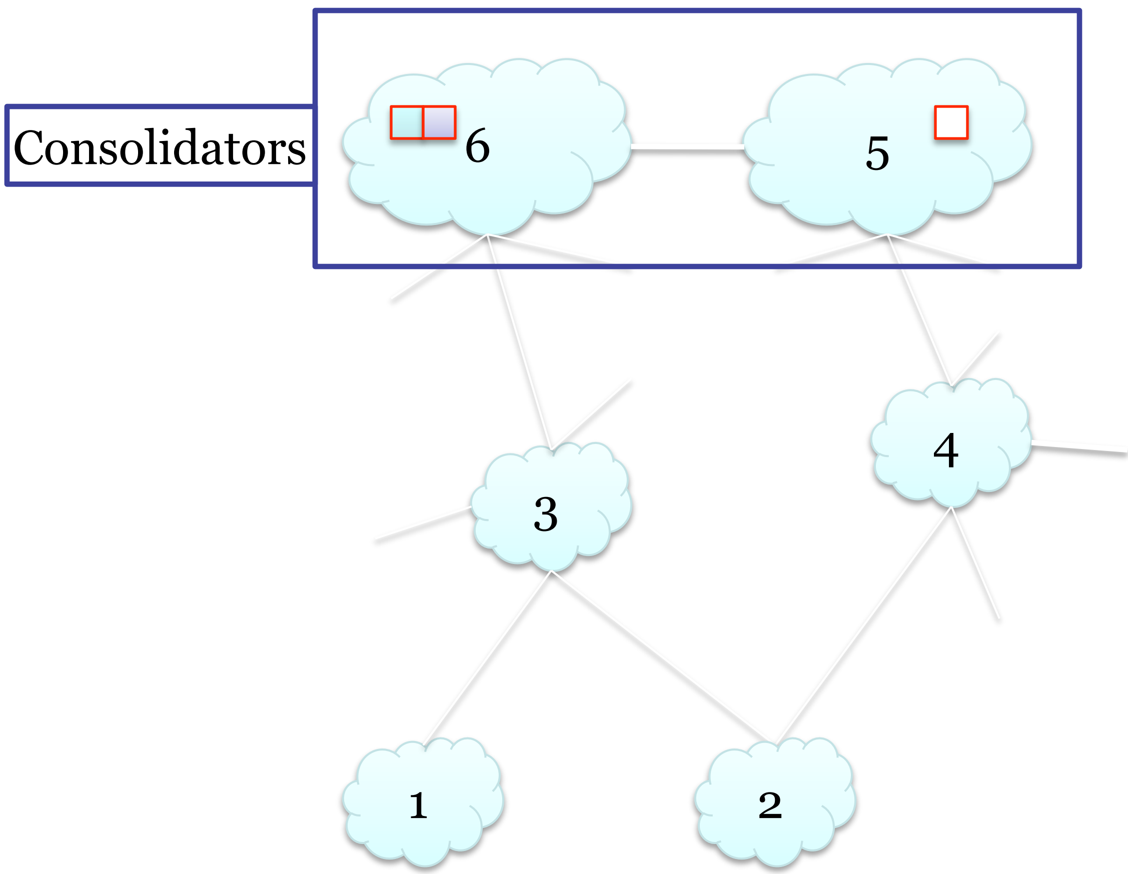


1. Run BGP, but don't apply the updates

Periodically, a **distributed snapshot** is taken

Updates in transit, or being processed are marked **incomplete**

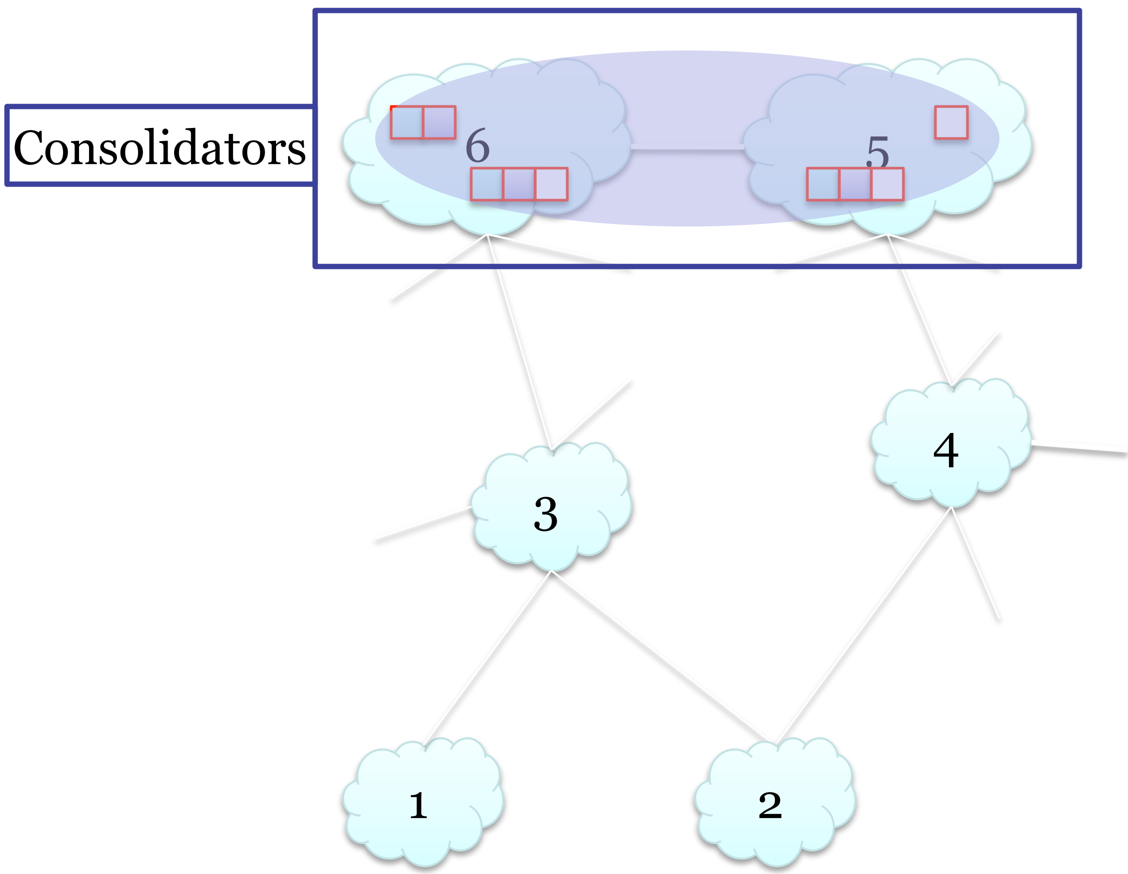
Mechanism



1. Run BGP, but don't apply the updates
2. Distributed Snapshot

ASes send list of incomplete updates to the consolidators

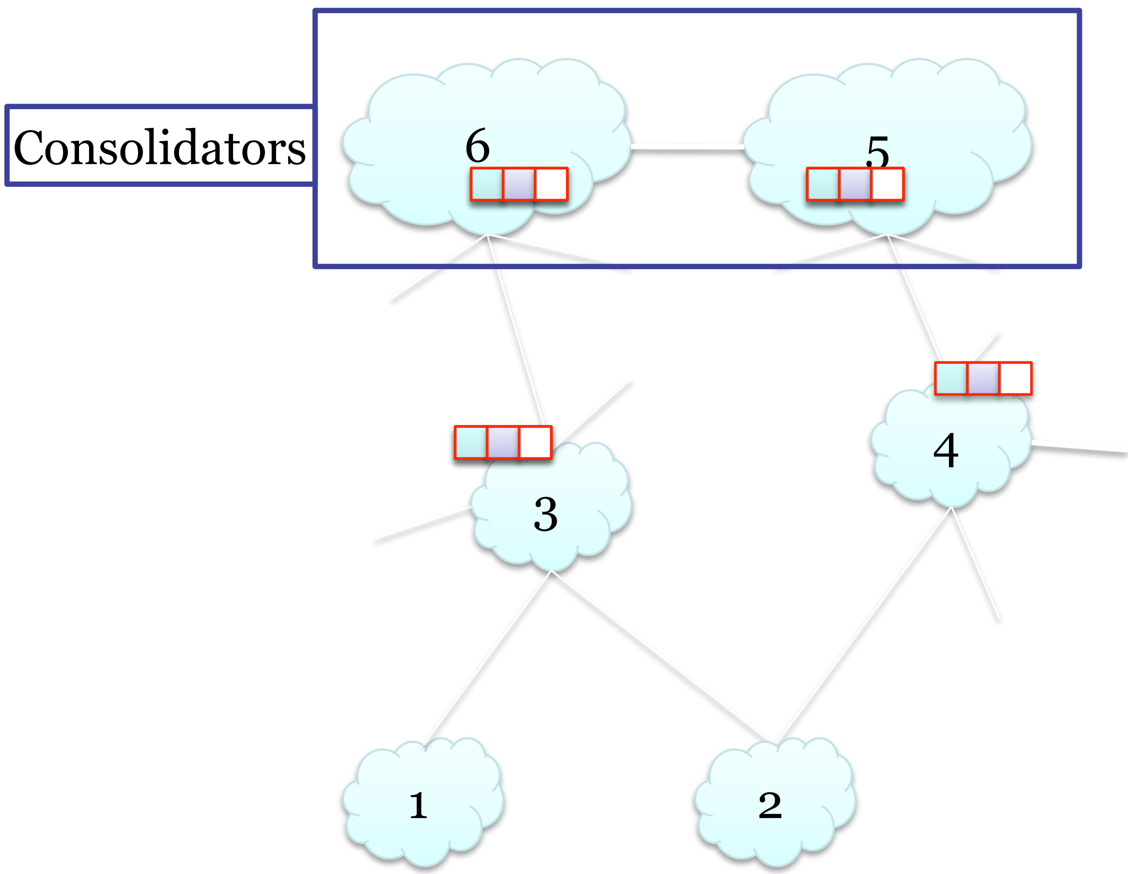
Mechanism



1. Run BGP, but don't apply the updates
2. Distributed Snapshot
3. Send info to consolidators

Consolidators run a consensus algorithm to agree on the set of incomplete updates

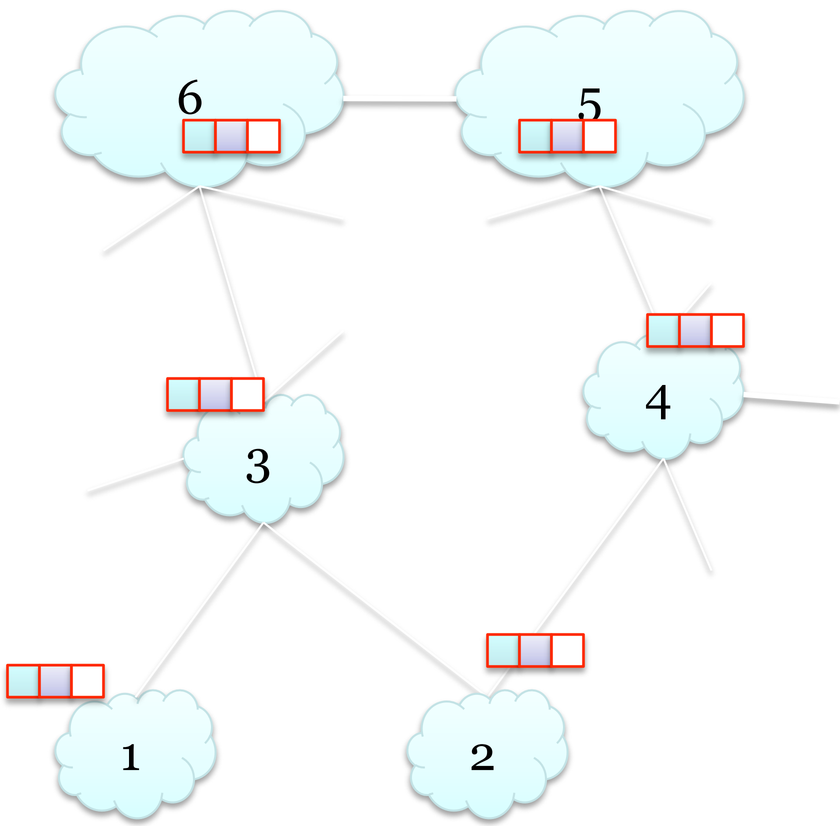
Mechanism



1. Run BGP, but don't apply the updates
2. Distributed Snapshot
3. Send info to consolidators

Consolidators **flood** the incomplete set to all the ASes

Mechanism



1. Run BGP, but don't apply the updates
2. Distributed Snapshot
3. Send info to consolidators
4. Consensus
5. Flood

Apply **completed** updates

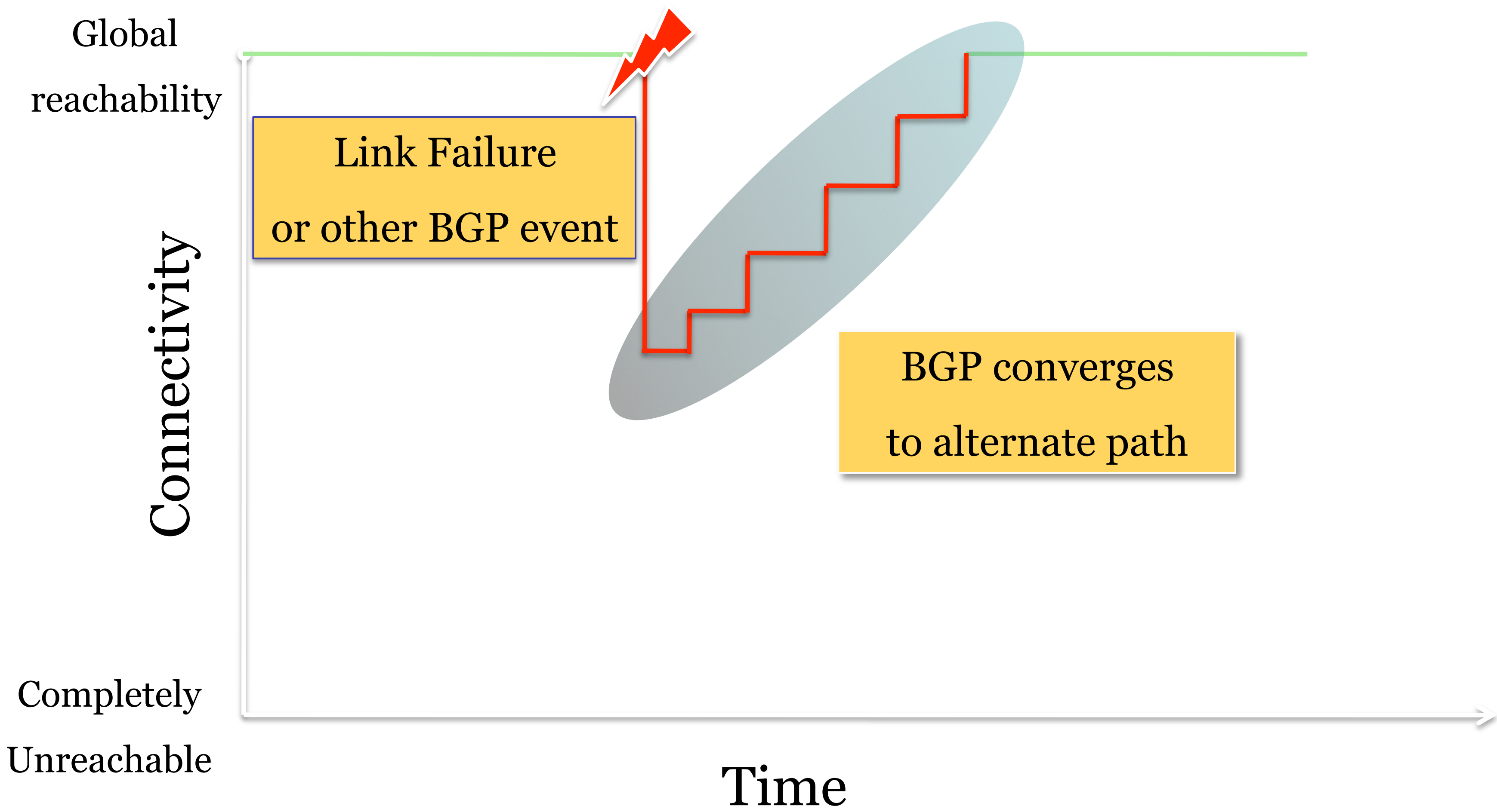
Liveness

Problem: Upon link failure, need to wait till path reaches everyone

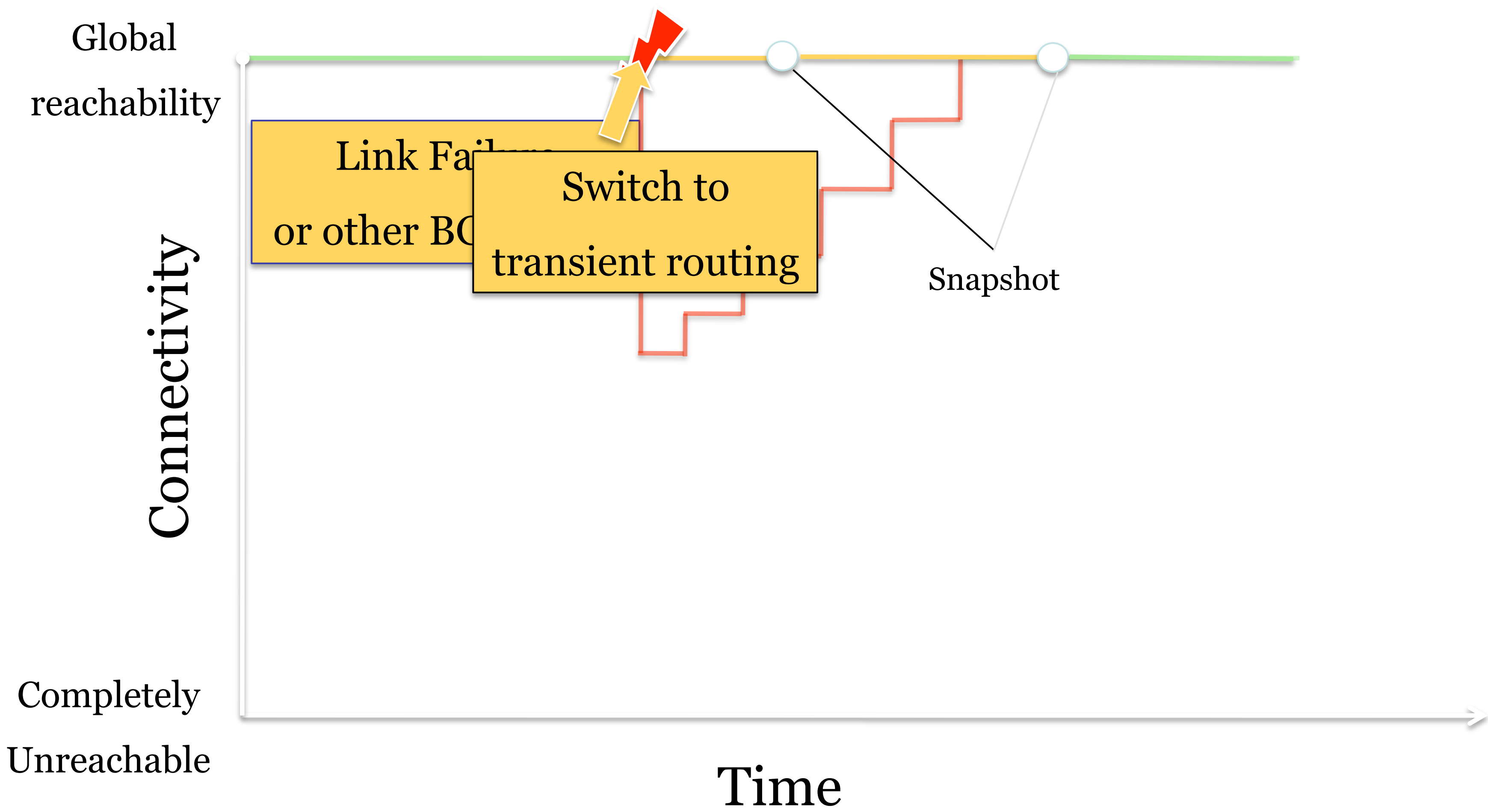
Solution: Dynamically re-route around the failed link

- Failure carrying packets (FCP)
- Pre-computed backup paths
- Detour routing

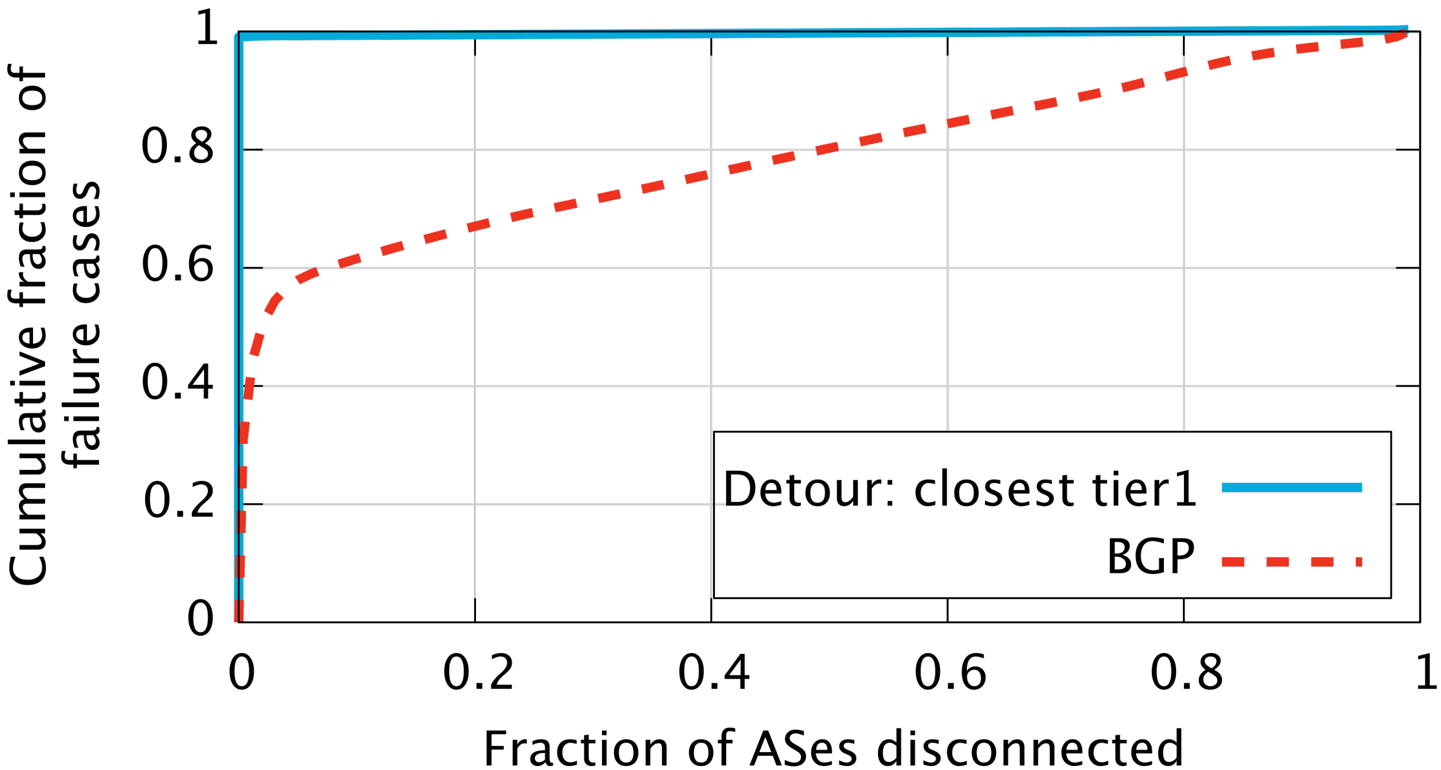
BGP


Global
reachability
Link Failure
or other BGP event
Connectivity
BGP converges
to alternate path
Completely
Unreachable
Time

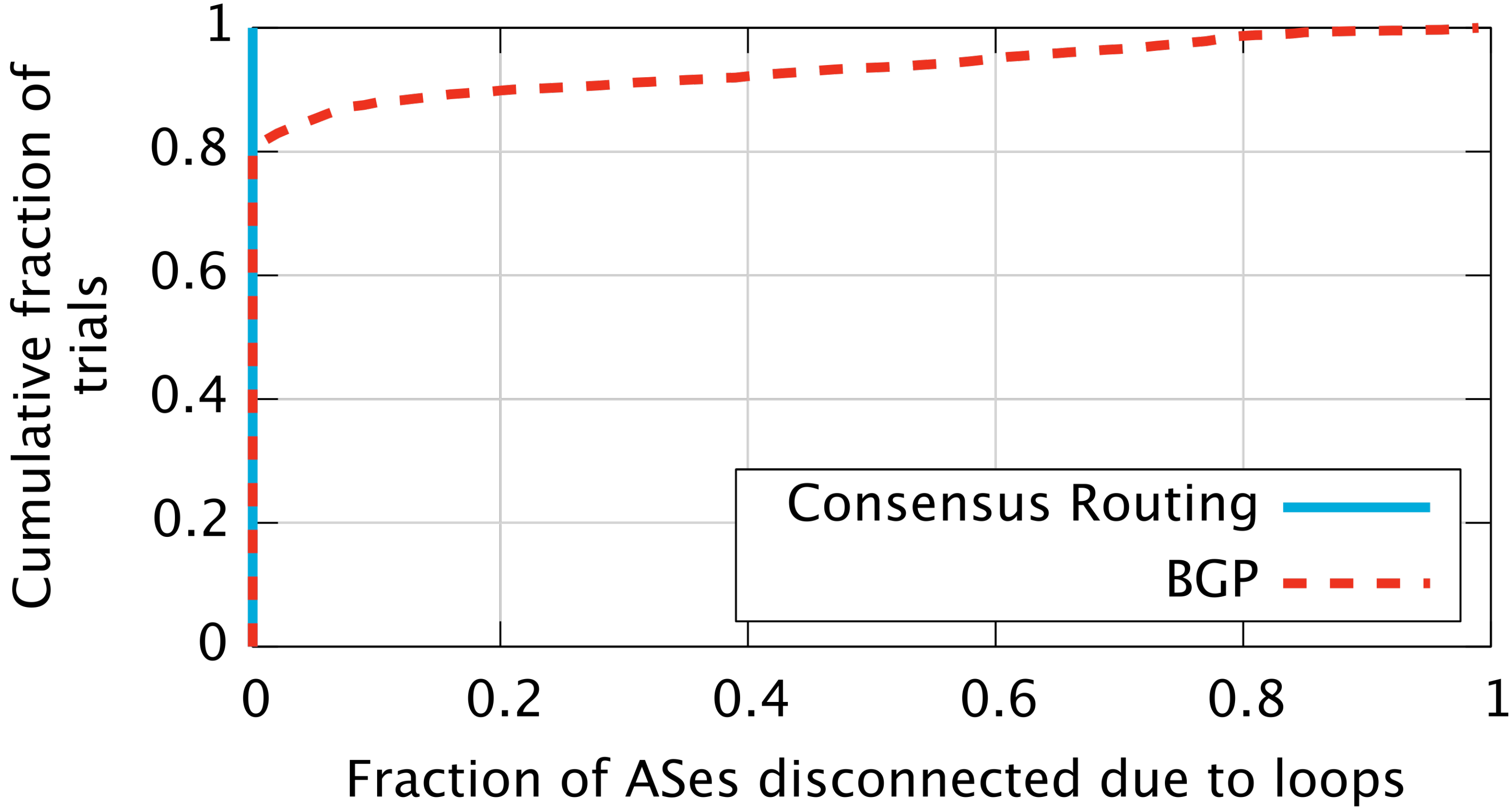
Consensus Routing



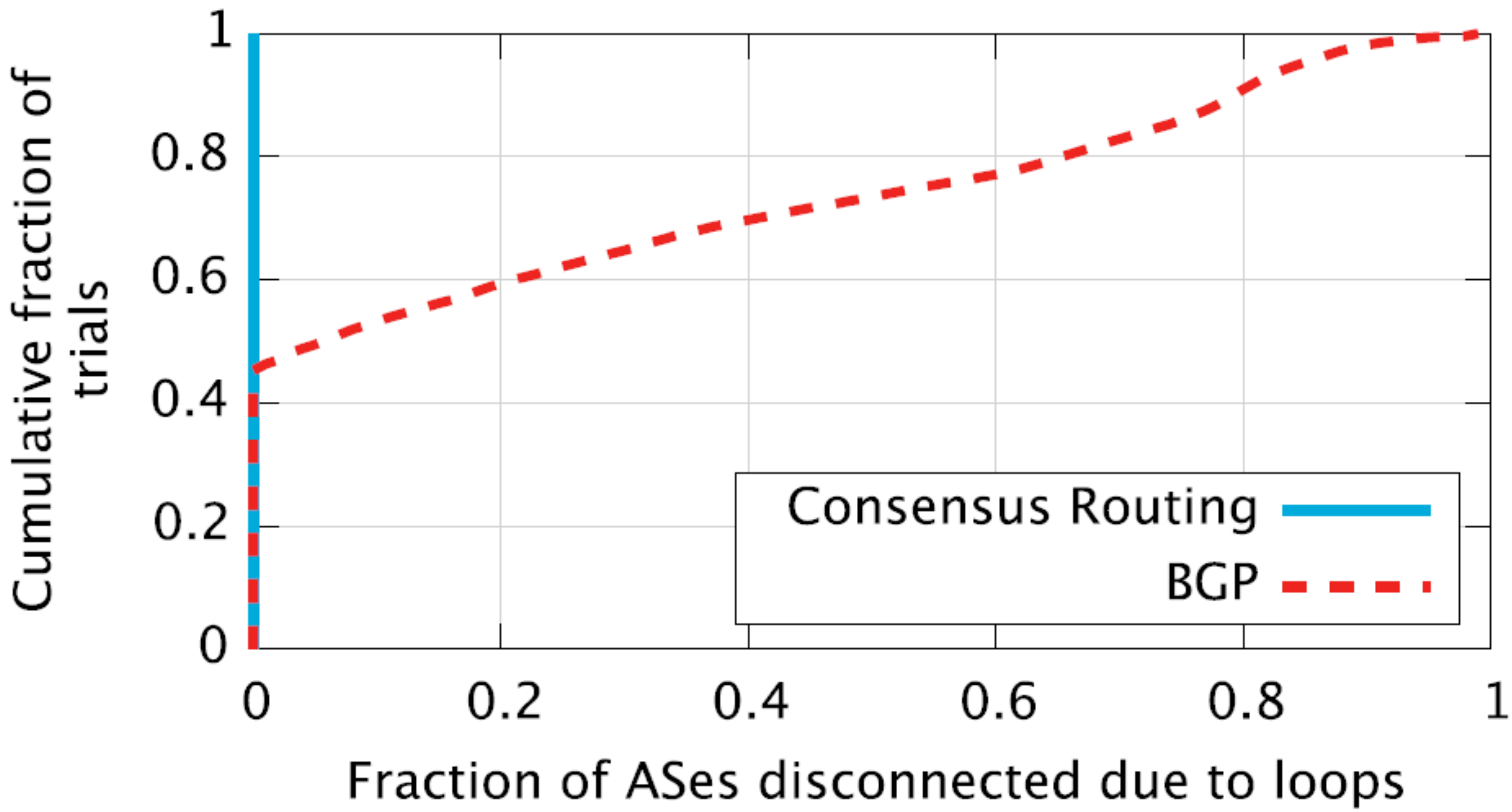
Availability After Failure



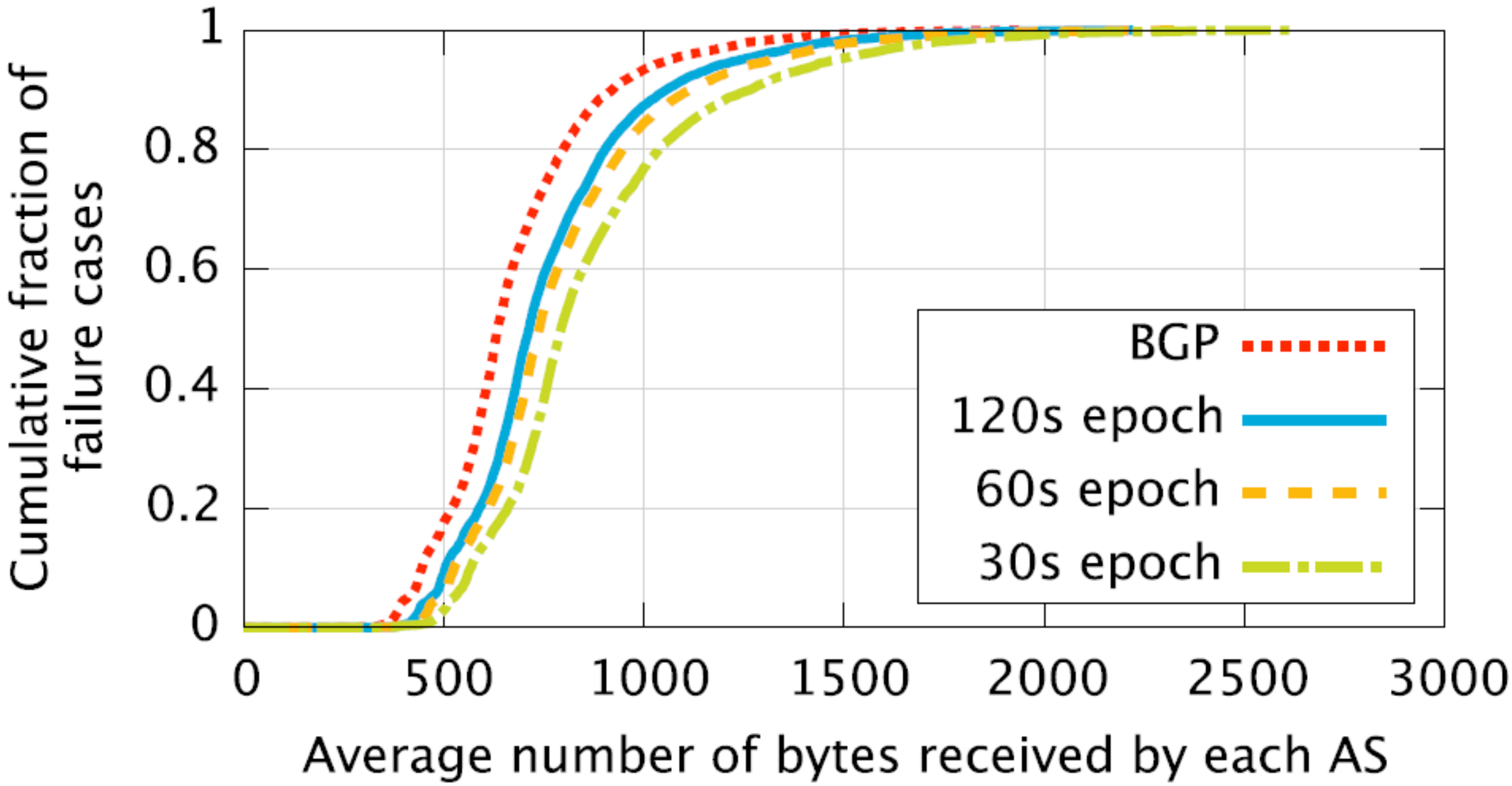
BGP loops, path prepending



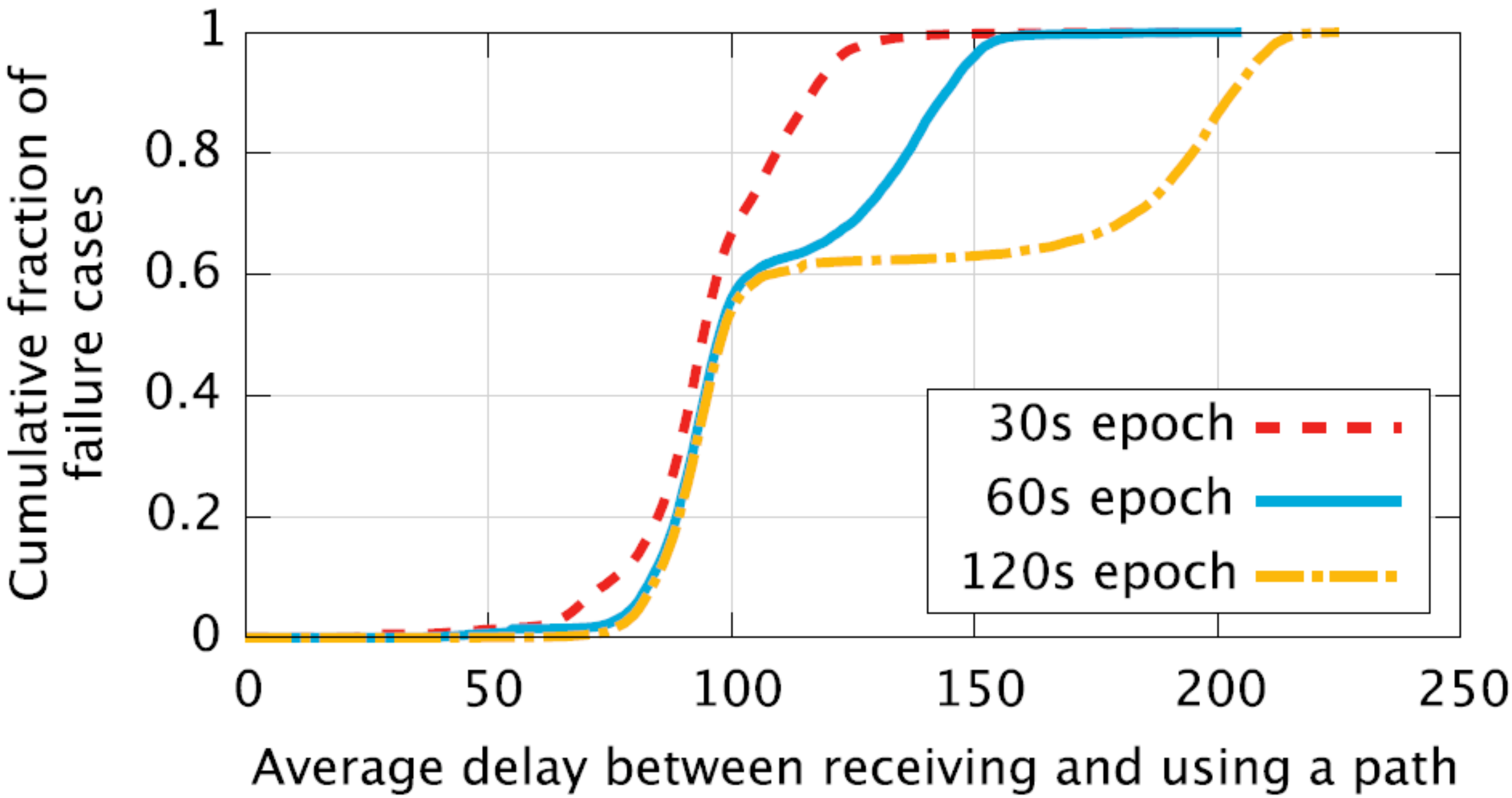
BGP loops, prefix engineering



Control traffic overhead



Average delay in reaching consensus



Roadmap

Brief primer on Internet routing

Interdomain routing convergence (consensus routing)

- Towards high availability at a fine-grained time scale [NSDI 08]

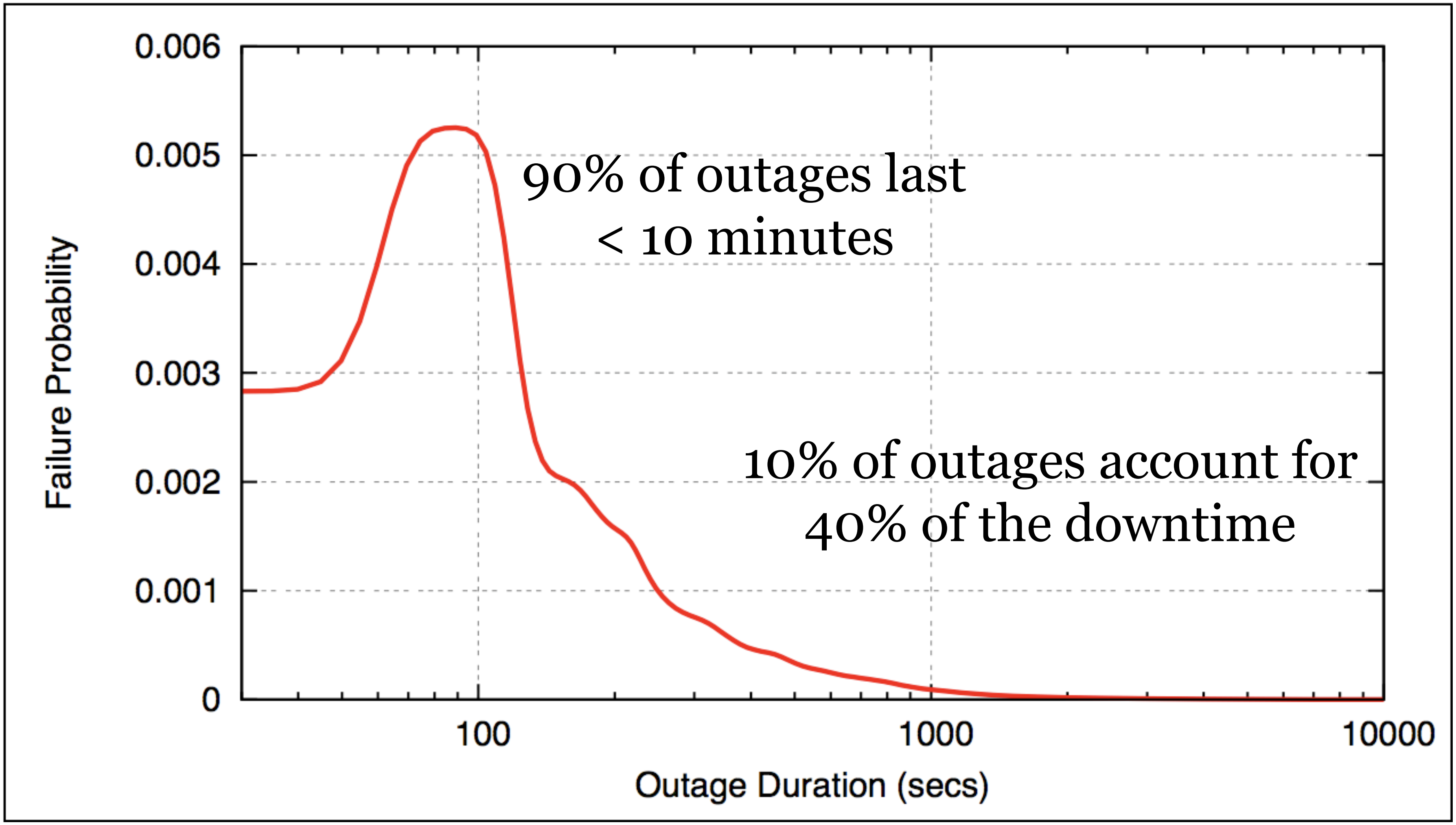
Interdomain routing diagnosis (Hubble/reverse traceroute)

- *Towards high availability at a long time scale [NSDI 08, NSDI 10]*

Distributed denial of service protection (phalanx)

- Towards withstanding million node botnets [NSDI 08]

Characterizing Internet Outages



Two month study found more than 2M outages

Current Troubleshooting: Traceroute

To troubleshoot these routing problems, network operators need better tools

- Protocols do not provide much visibility
- Networks do not have incentive to divulge

Traceroute: measures route from the computer running traceroute to anywhere

- Provides no information about reverse path

"The number one go-to tool is traceroute."

NANOG Network operators troubleshooting tutorial, 2009.

Data Centers Need Better Tools

Clients in Taiwan experiencing 500ms network latency

Data Centers Need Better Tools

Is client served by distant data center?

Clients in Taiwan experiencing 500ms network latency

Data Centers Need Better Tools

Is client served by distant data center? Check logs: **No**

Clients in Taiwan experiencing 500ms network latency

Data Centers Need Better Tools

Is path from data center to client indirect?

Clients in Taiwan experiencing 500ms network latency

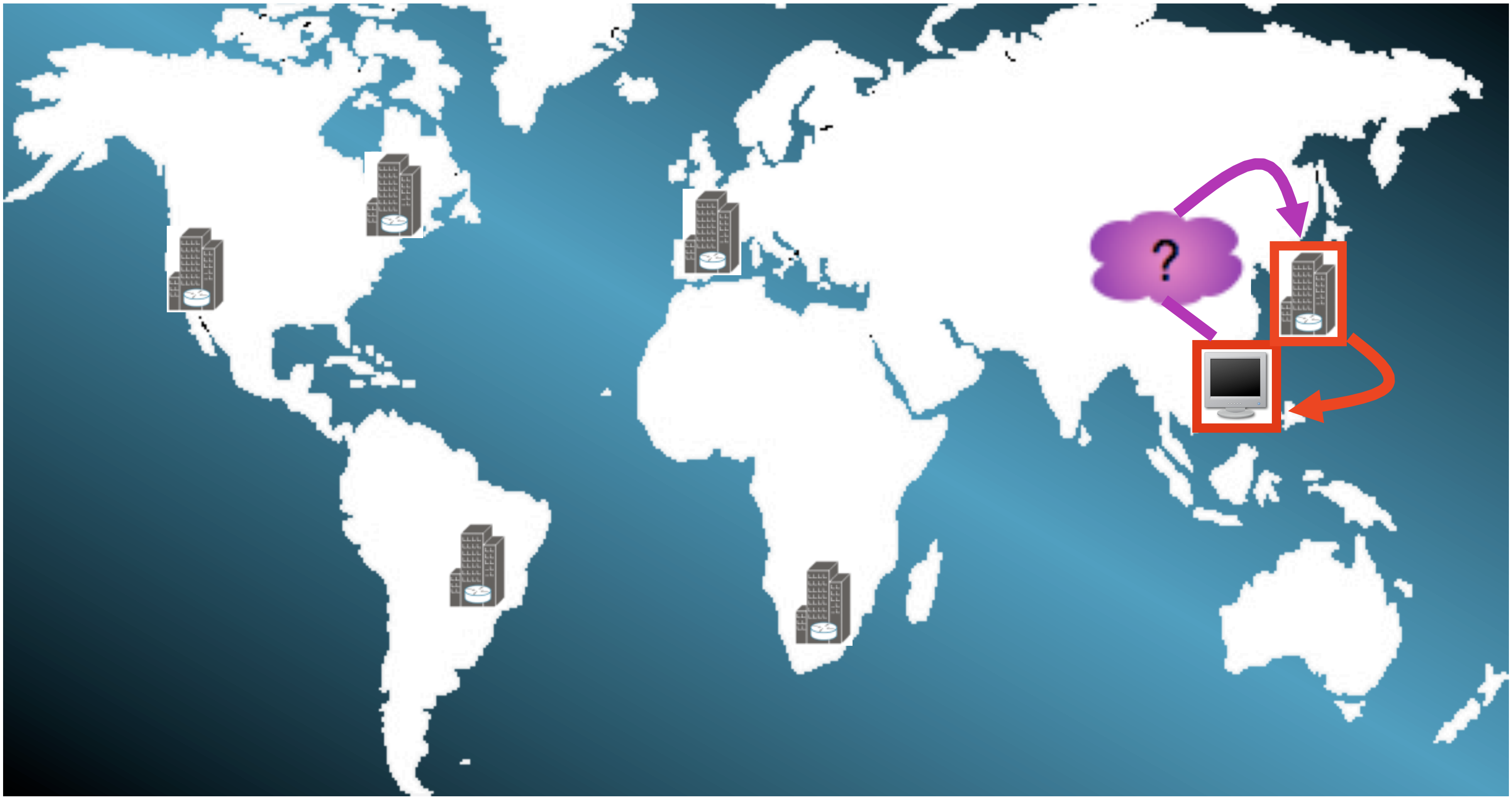
Data Centers Need Better Tools

Is path from data center to client indirect? Traceroute: **No**

Clients in Taiwan experiencing 500ms network latency

Data Centers Need Better Tools

*Is **reverse path** from client back to data center indirect?*



Clients in Taiwan experiencing 500ms network latency

Data Centers Need Better Tools

Is ***reverse path*** *from client back to data center indirect?*

Clients in Taiwan experiencing 500ms network latency

Want path from **D** back to **S**, don't control **D**

KEY IDEAS FOR REVERSE TRACEROUTE

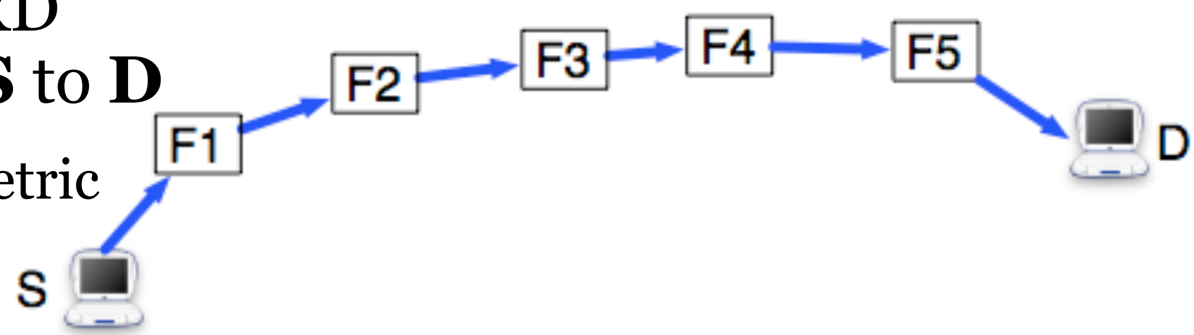
Technique does not require control of destination

Want path from **D** back to **S**, don't control **D**

Can issue FORWARD traceroute from **S** to **D**

- But likely asymmetric

Can't use traceroute on reverse path



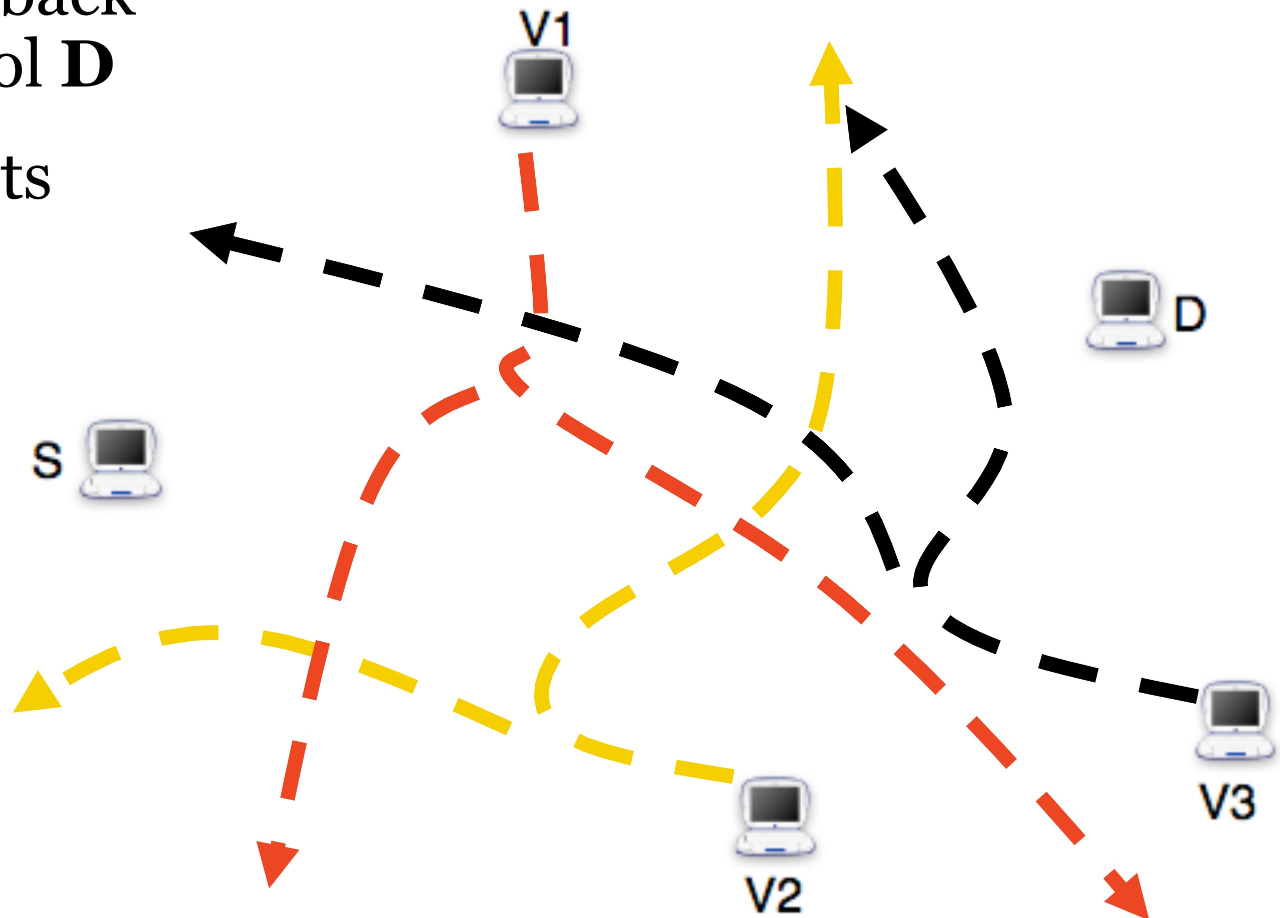
KEY IDEAS FOR REVERSE TRACEROUTE

Technique does not require control of destination

Want path from **D** back to **S**, don't control **D**

Set of vantage points

- Can measure an atlas of routes



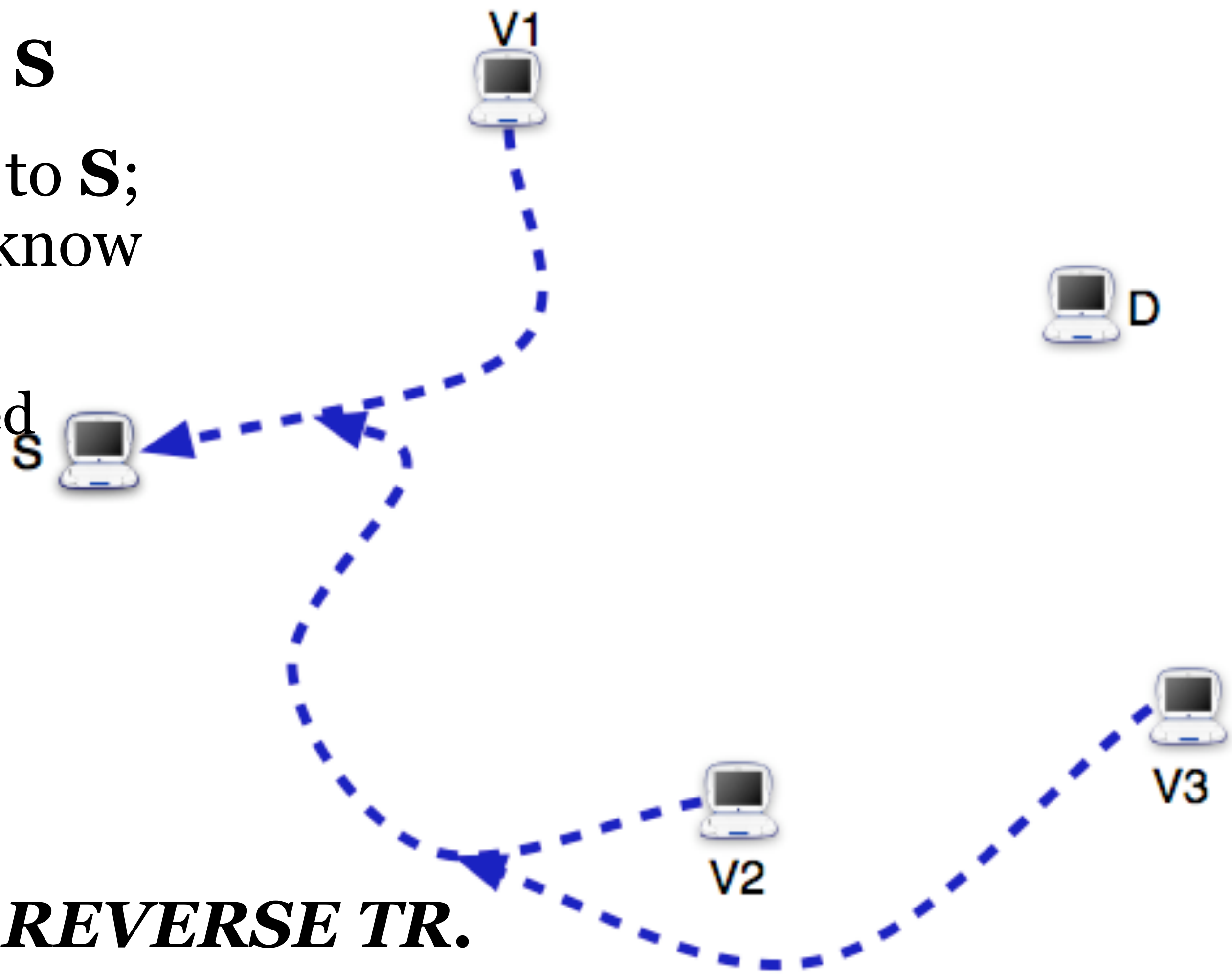
KEY IDEAS FOR REVERSE TR.

Multiple VPs combine for view unattainable from any one

Traceroute from all vantage points to **S**

Gives atlas of paths to **S**; if we hit one, we know rest of path

- Destination-based routing

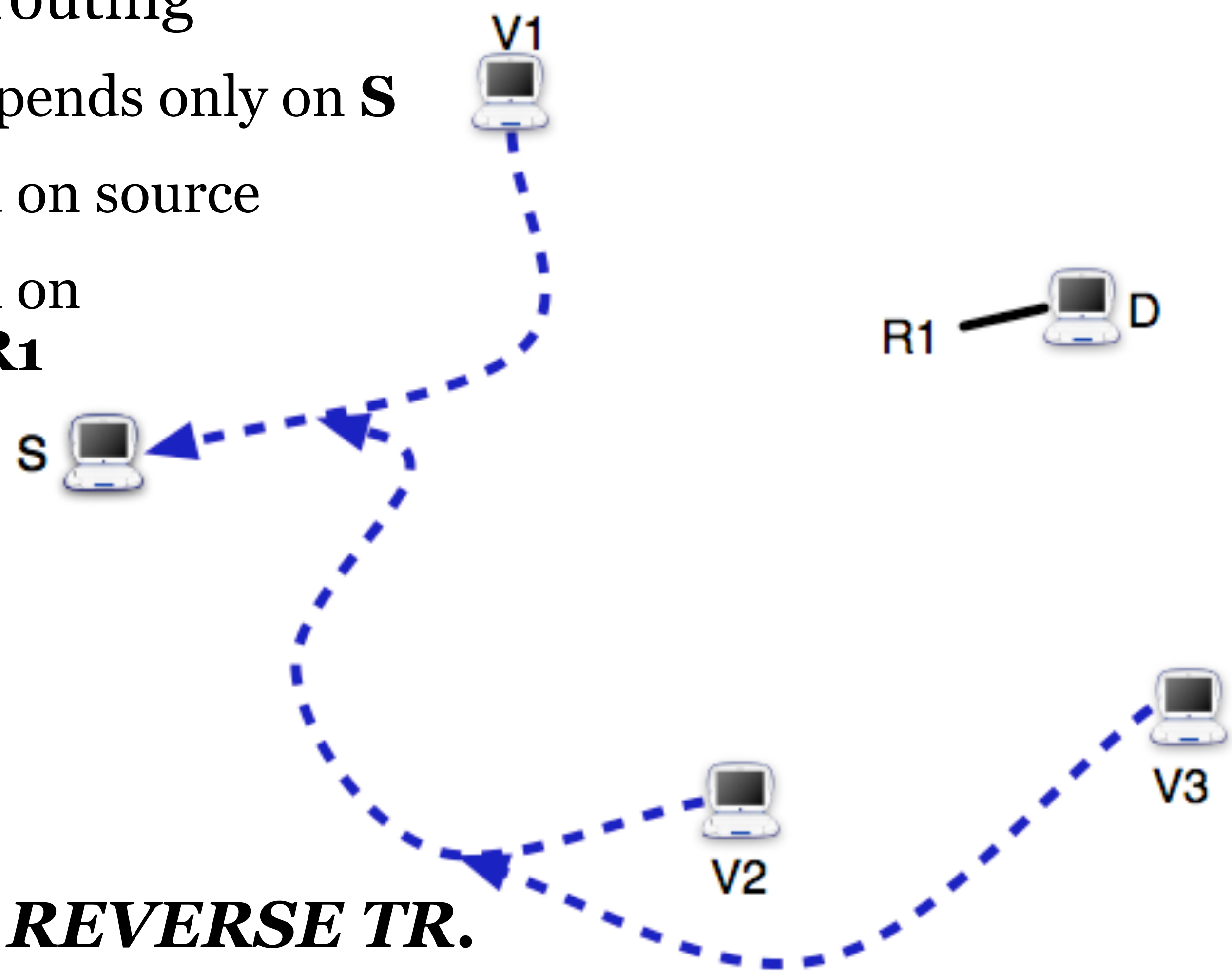


KEY IDEAS FOR REVERSE TR.

Traceroute atlas gives baseline we bootstrap from

Destination-based routing

- Path from **R1** depends only on **S**
- Does not depend on source
- Does not depend on path from **D** to **R1**

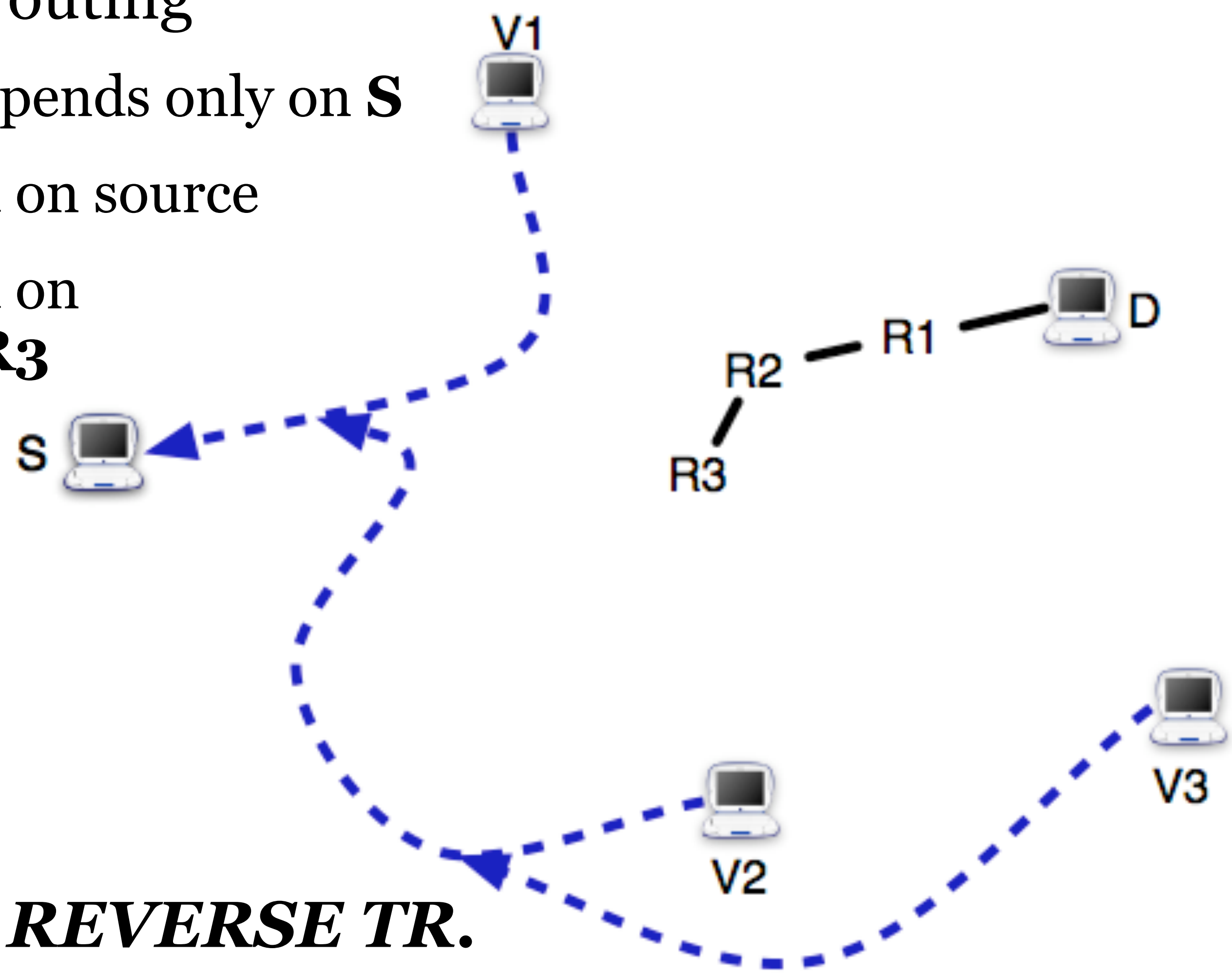


KEY IDEAS FOR REVERSE TR.

Destination-based routing lets us stitch path hop-by-hop

Destination-based routing

- Path from **R3** depends only on **S**
- Does not depend on source
- Does not depend on path from **D** to **R3**

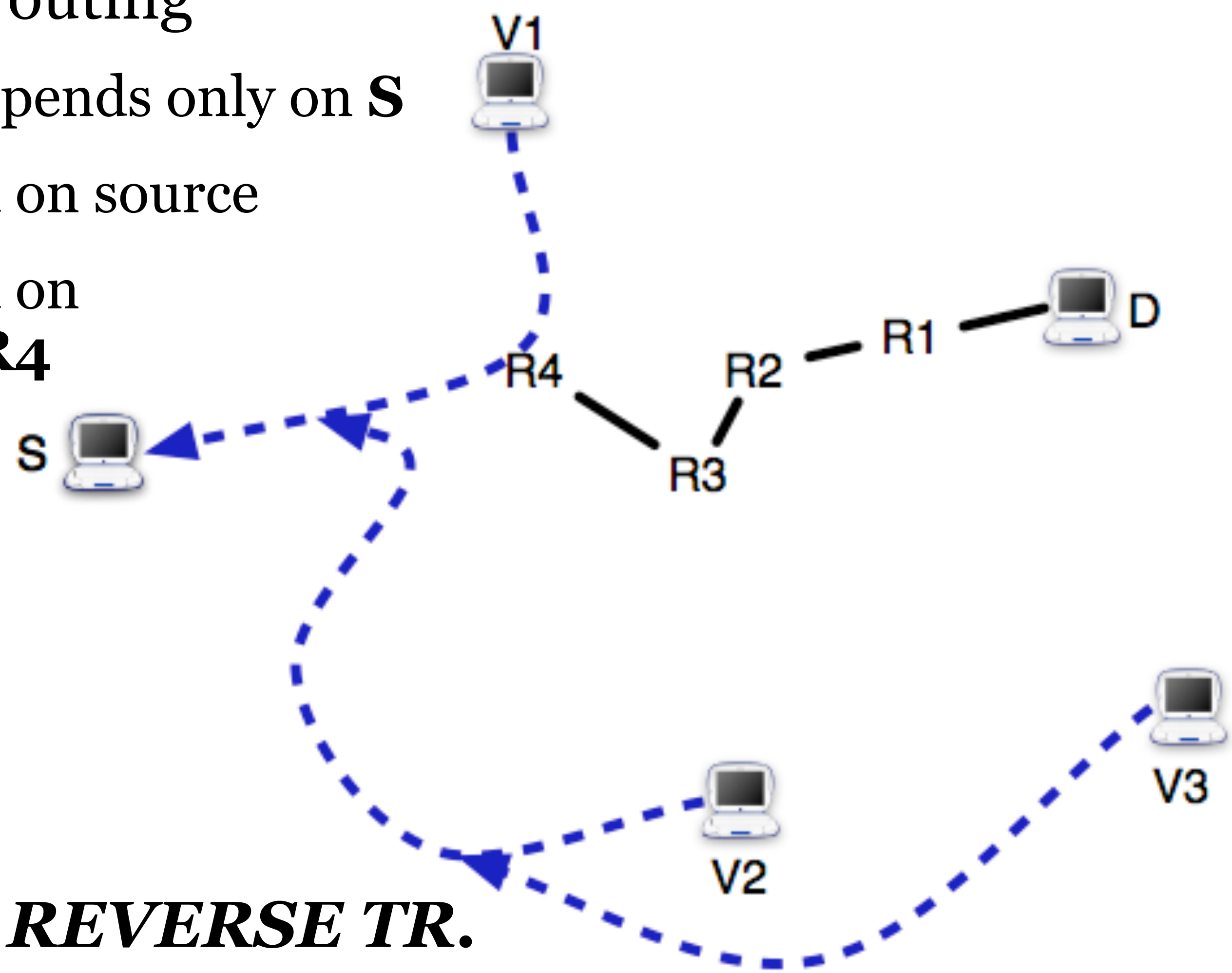


KEY IDEAS FOR REVERSE TR.

Destination-based routing lets us stitch path hop-by-hop

Destination-based routing

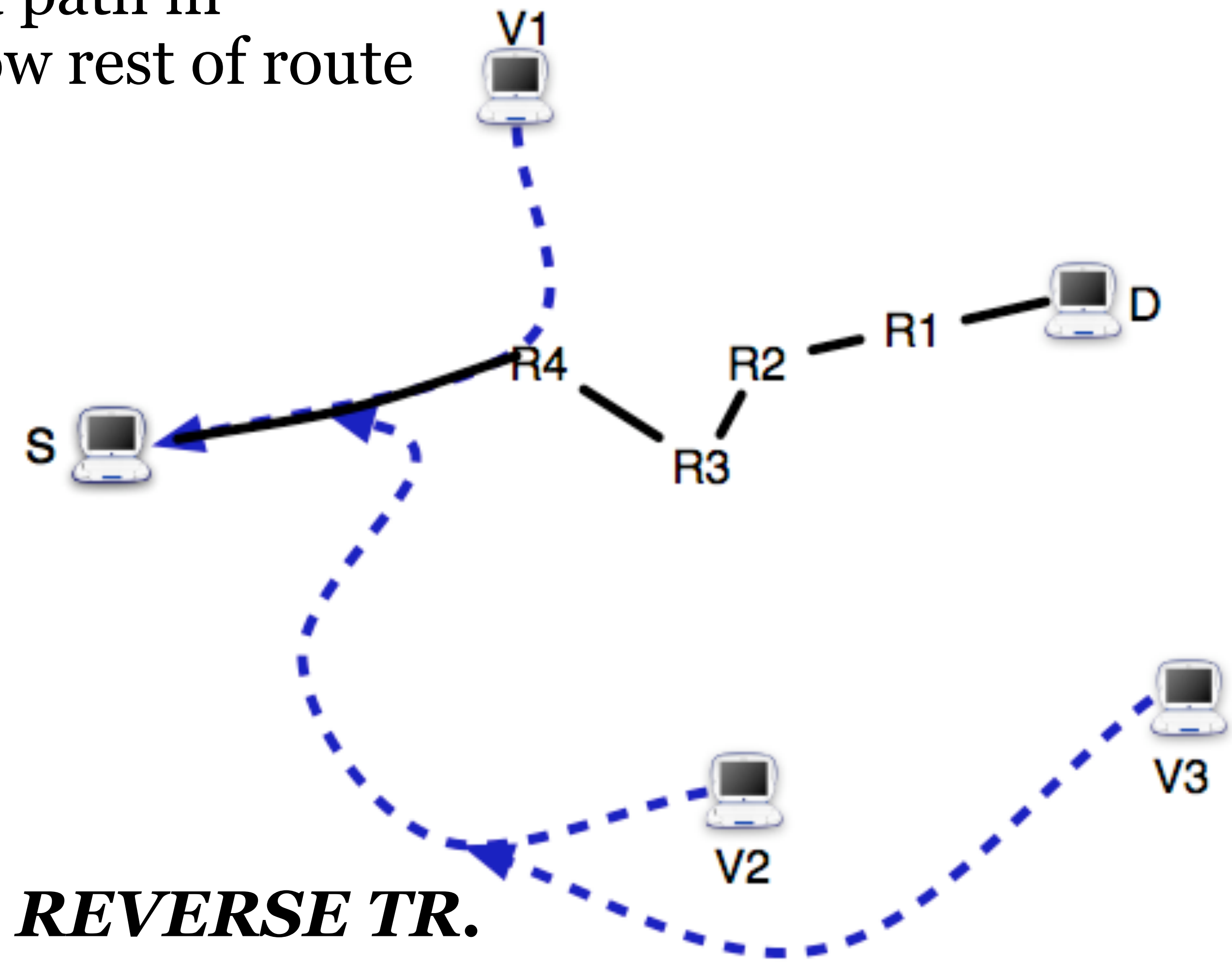
- Path from **R4** depends only on **S**
- Does not depend on source
- Does not depend on path from **D** to **R4**



KEY IDEAS FOR REVERSE TR.

Destination-based routing lets us stitch path hop-by-hop

Once we intersect a path in our atlas, we know rest of route



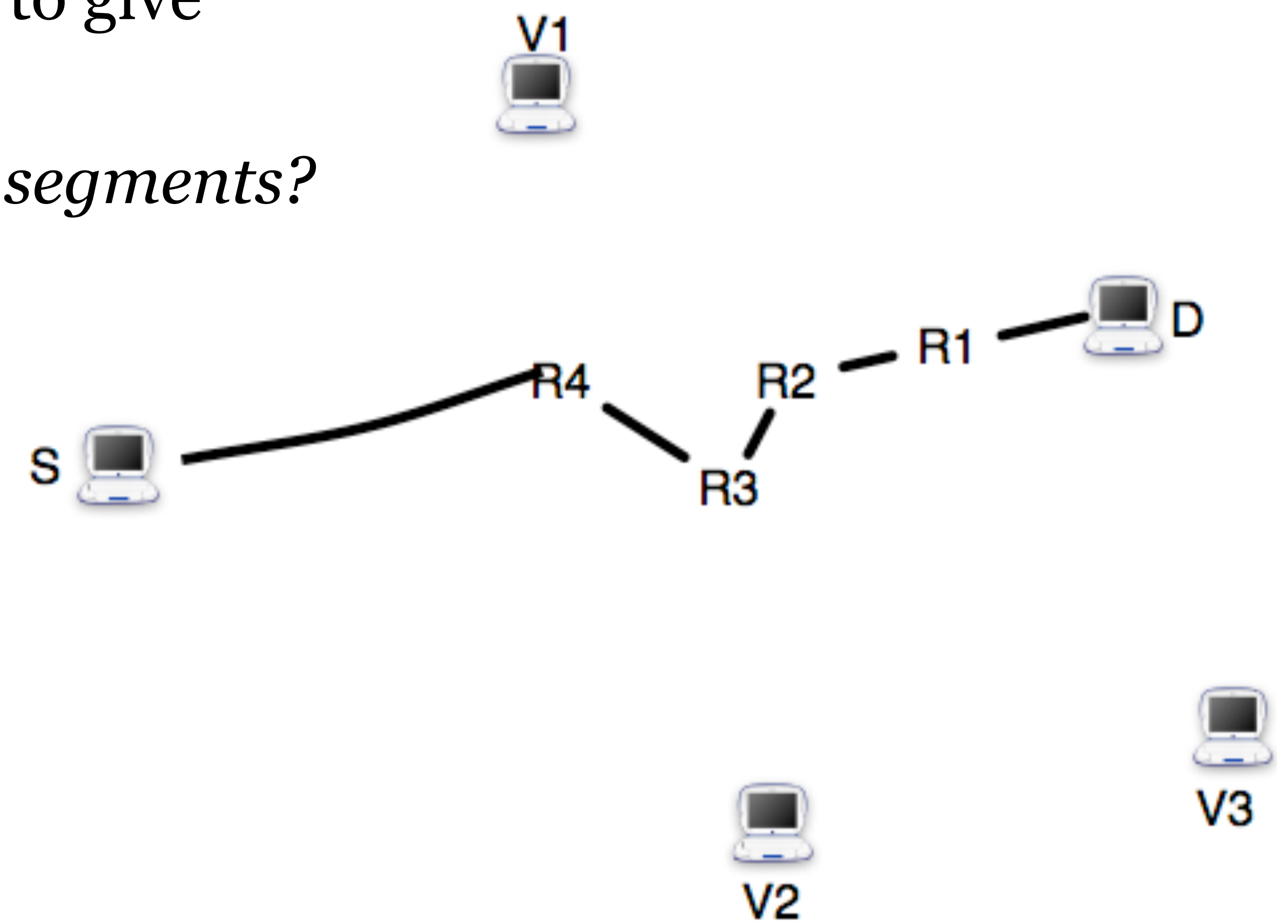
KEY IDEAS FOR REVERSE TR.

Destination-based routing lets us stitch path hop-by-hop

Traceroute atlas gives baseline we bootstrap from

Segments combine to give complete path

But how do we get segments?



KEY IDEAS FOR REVERSE TR.

Destination-based routing lets us stitch path hop-by-hop

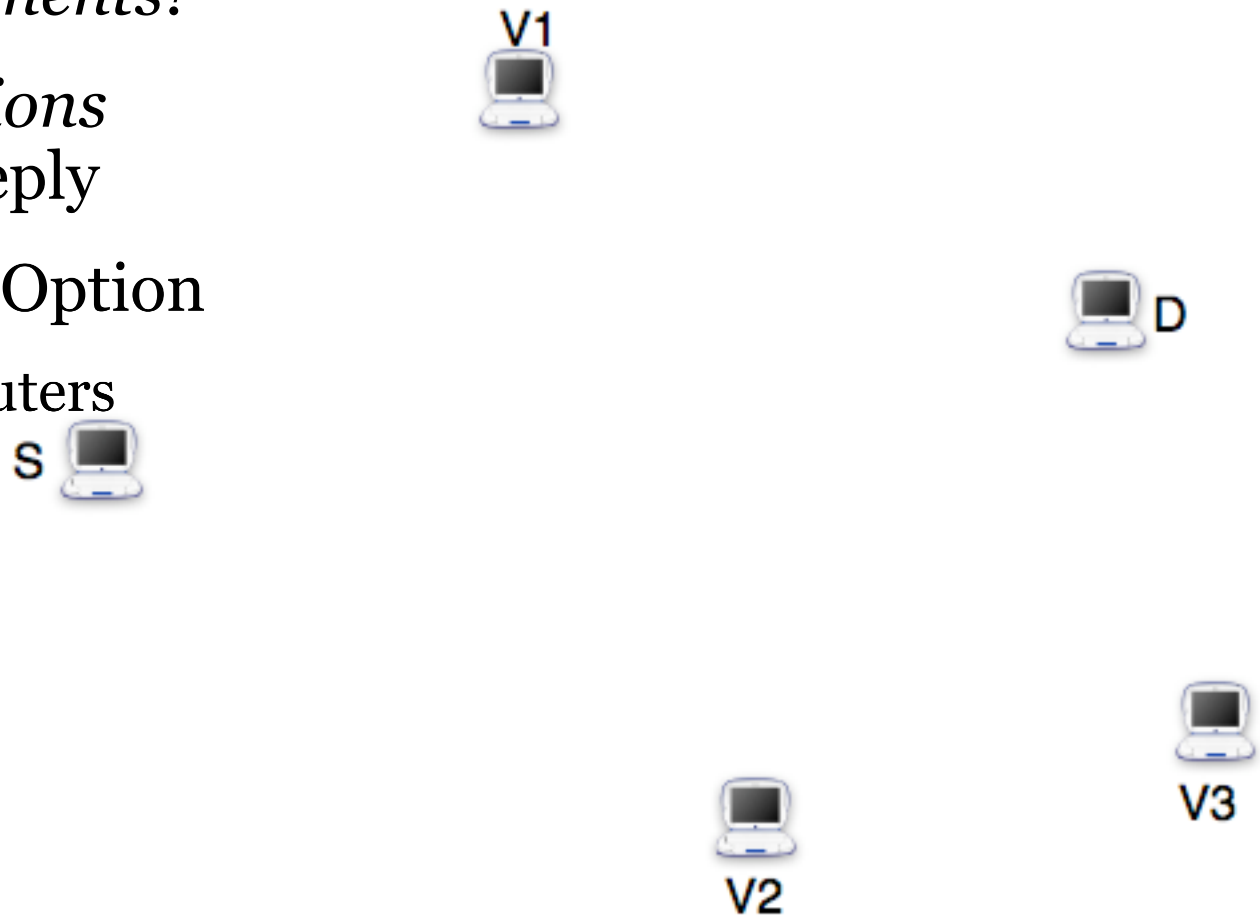
Traceroute atlas gives baseline we bootstrap from

How do we get segments?

Unlike TTL, *IP Options* are reflected in reply

Record Route (RR) Option

- Record first 9 routers
- If **D** within 8, reverse hops fill rest of slots



KEY IDEAS FOR REVERSE TR.

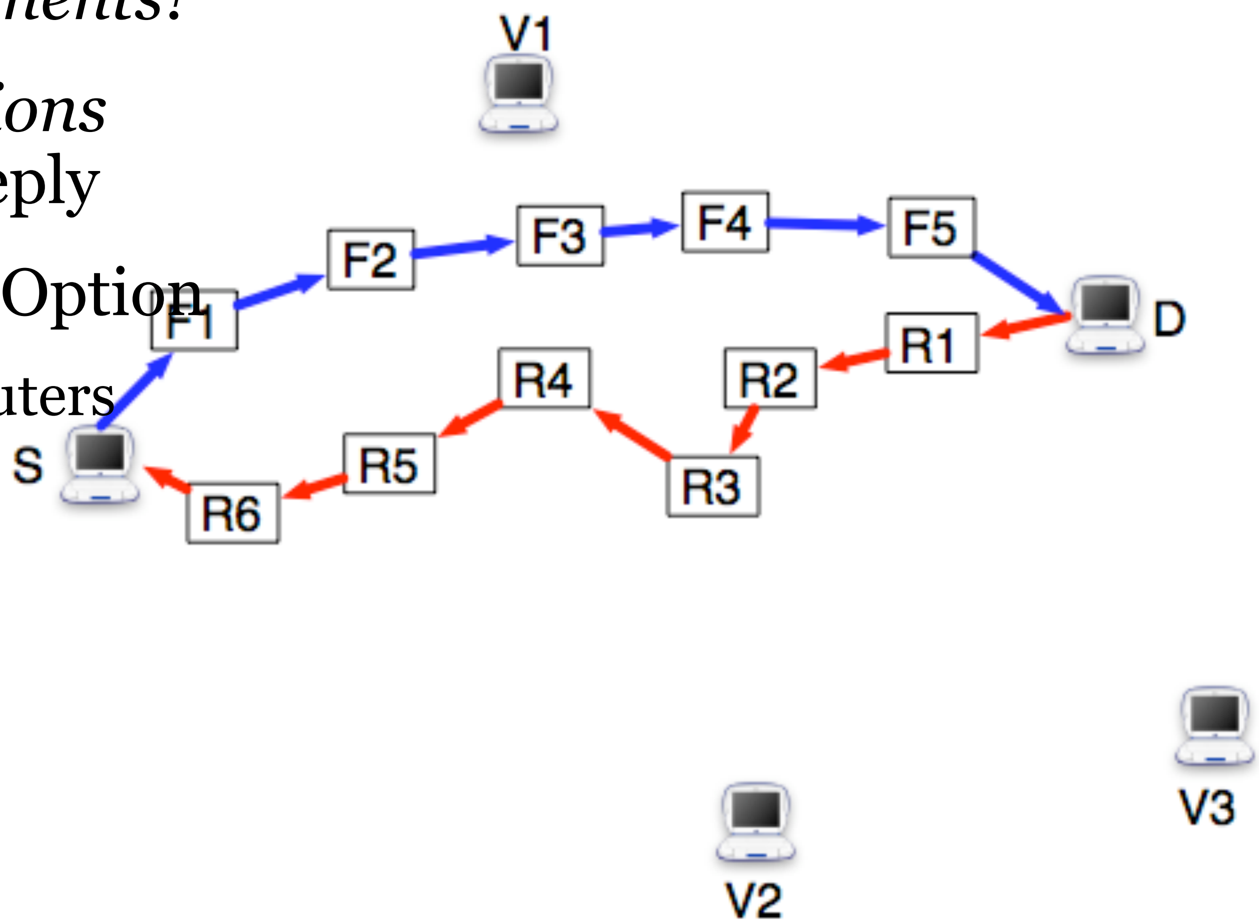
IP Options work over forward and reverse path

How do we get segments?

Unlike TTL, *IP Options* are reflected in reply

Record Route (RR) Option

- Record first 9 routers
- If **D** within 8, reverse hops fill rest of slots



KEY IDEAS FOR REVERSE TR.

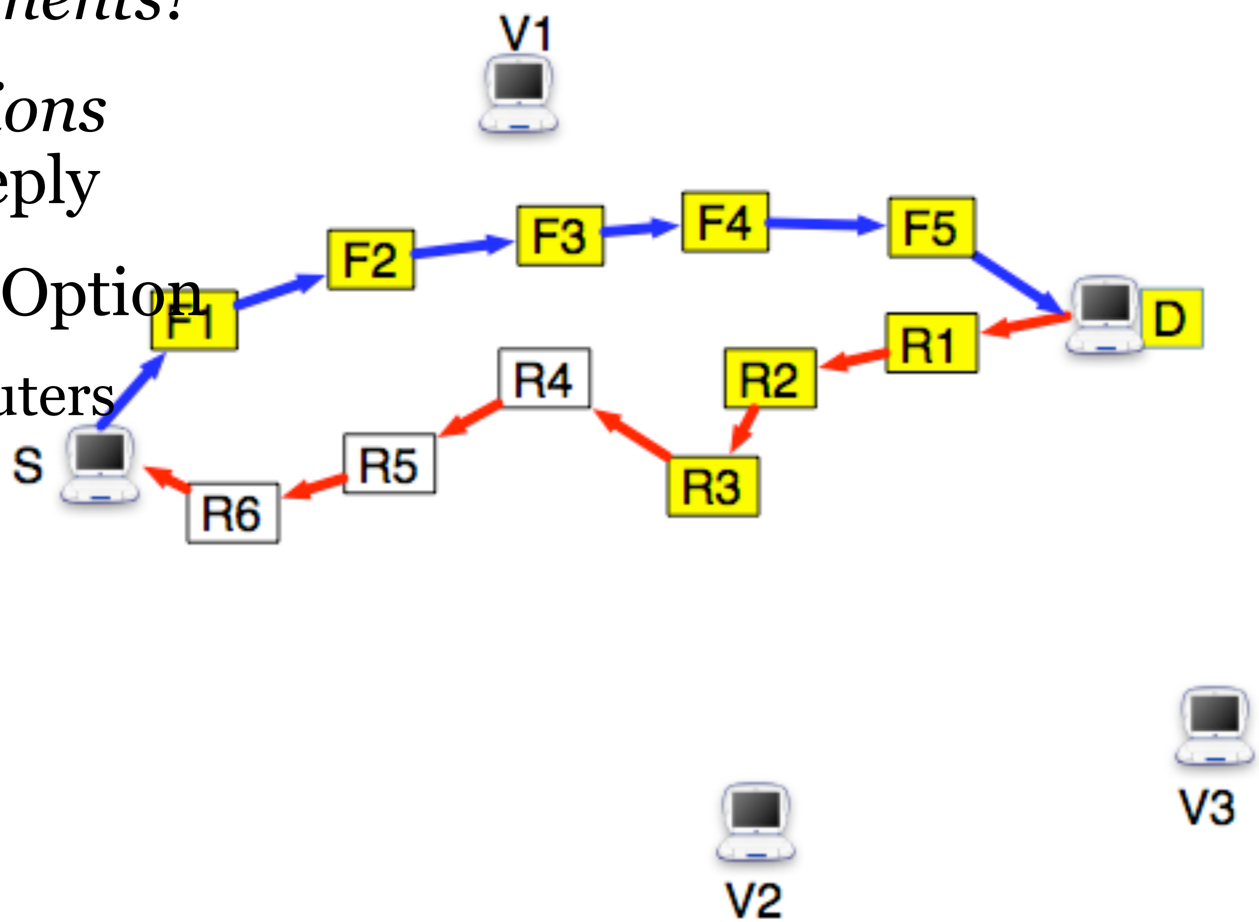
IP Options work over forward and reverse path

How do we get segments?

Unlike TTL, *IP Options* are reflected in reply

Record Route (RR) Option

- Record first 9 routers
- If **D** within 8, reverse hops fill rest of slots
- … but average path is 15 hops, 30 round-trip

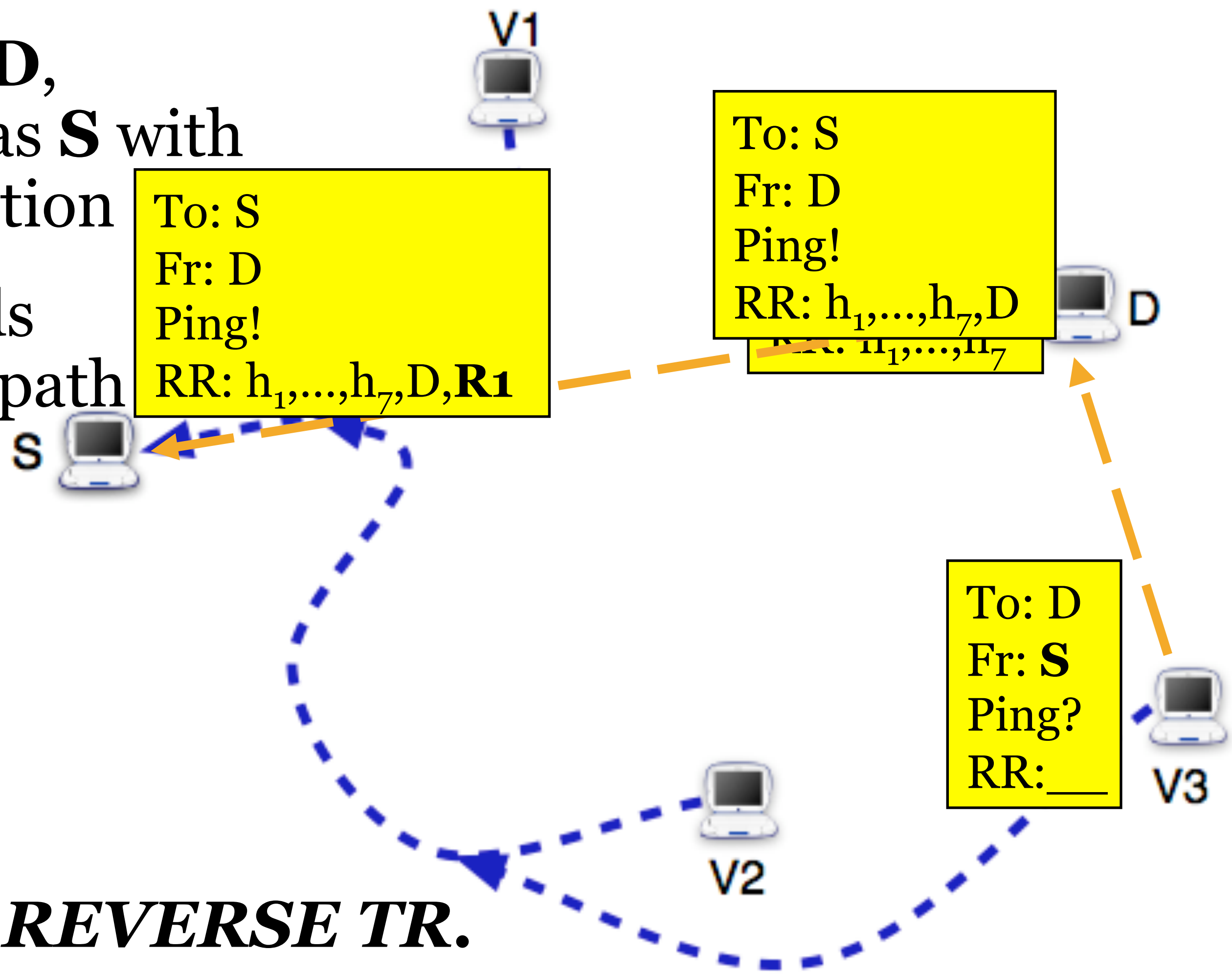


KEY IDEAS FOR REVERSE TR.

IP Options work over forward and reverse path

From vantage point
within 8 hops of **D**,
ping **D** spoofing as **S** with
Record Route Option

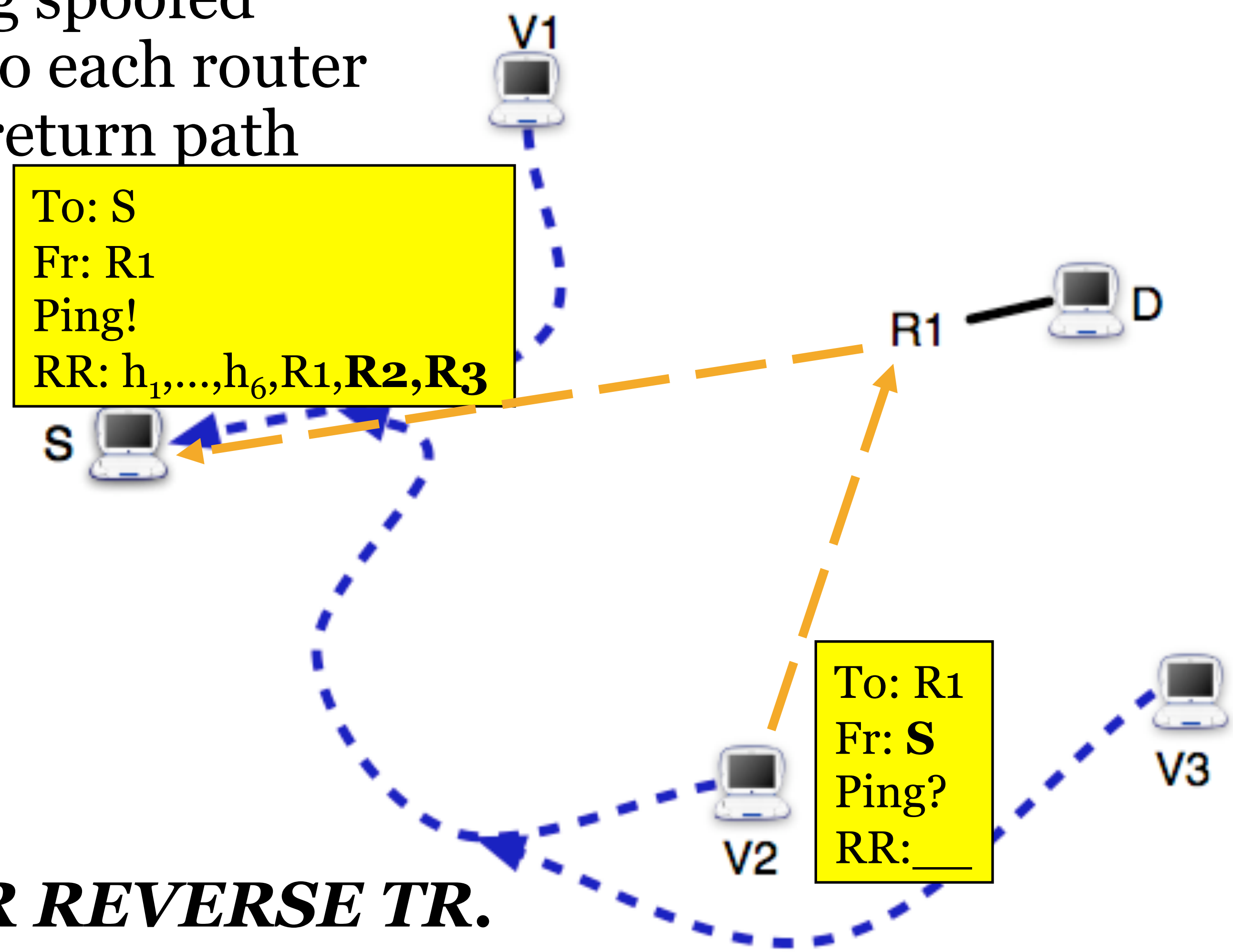
D's response records
hop(s) on return path



KEY IDEAS FOR REVERSE TR.

Spoofing lets us use vantage point in best position

Iterate, performing spoofed Record Routes to each router we discover on return path



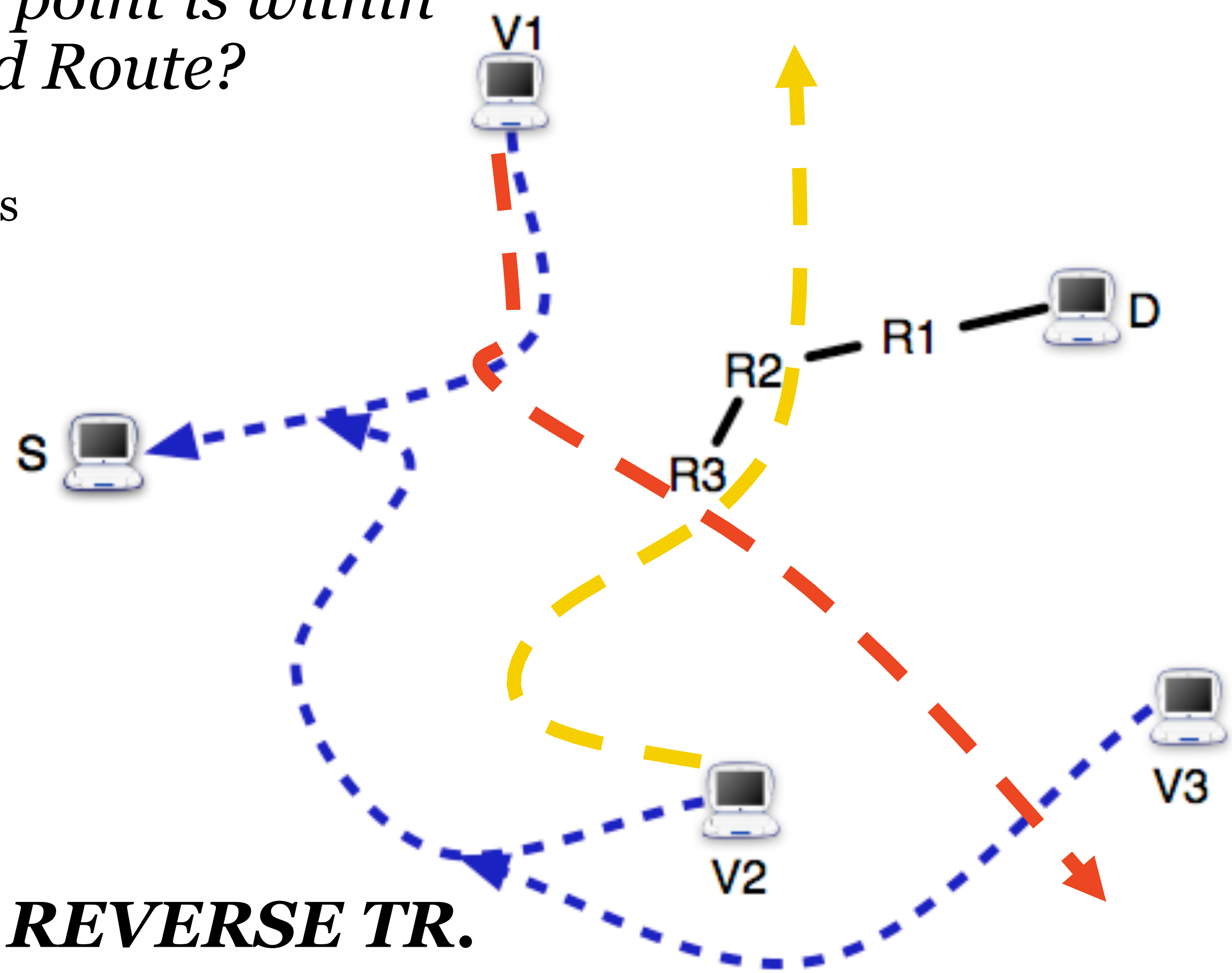
KEY IDEAS FOR REVERSE TR.

Spoofing lets us use vantage point in best position

Destination-based routing lets us stitch path hop-by-hop

What if no vantage point is within 8 hops for Record Route?

Consult atlas of known paths to find adjacencies



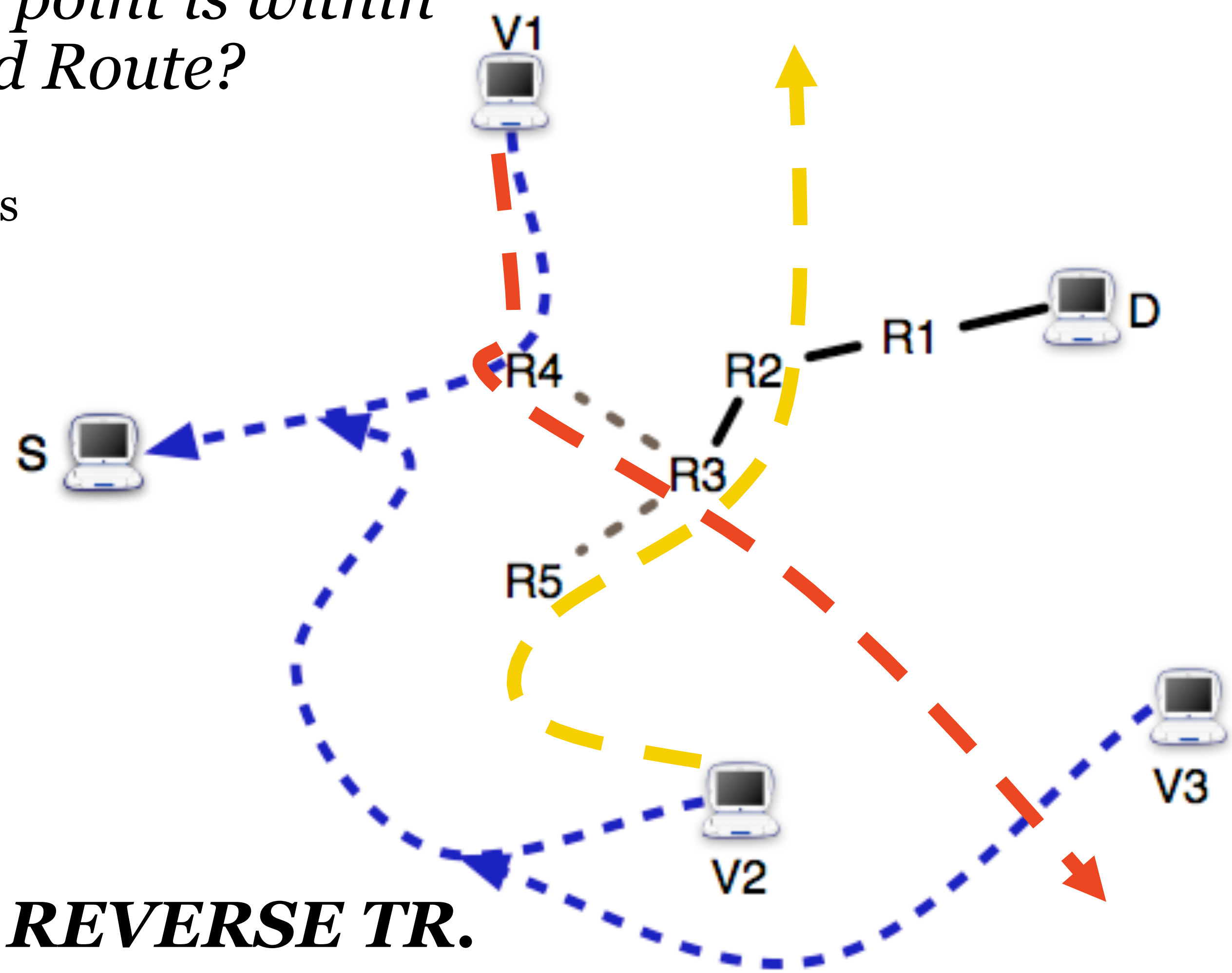
KEY IDEAS FOR REVERSE TR.

Spoofing lets us use vantage point in best position

Destination-based routing lets us stitch path hop-by-hop

What if no vantage point is within 8 hops for Record Route?

Consult atlas of known paths to find adjacencies



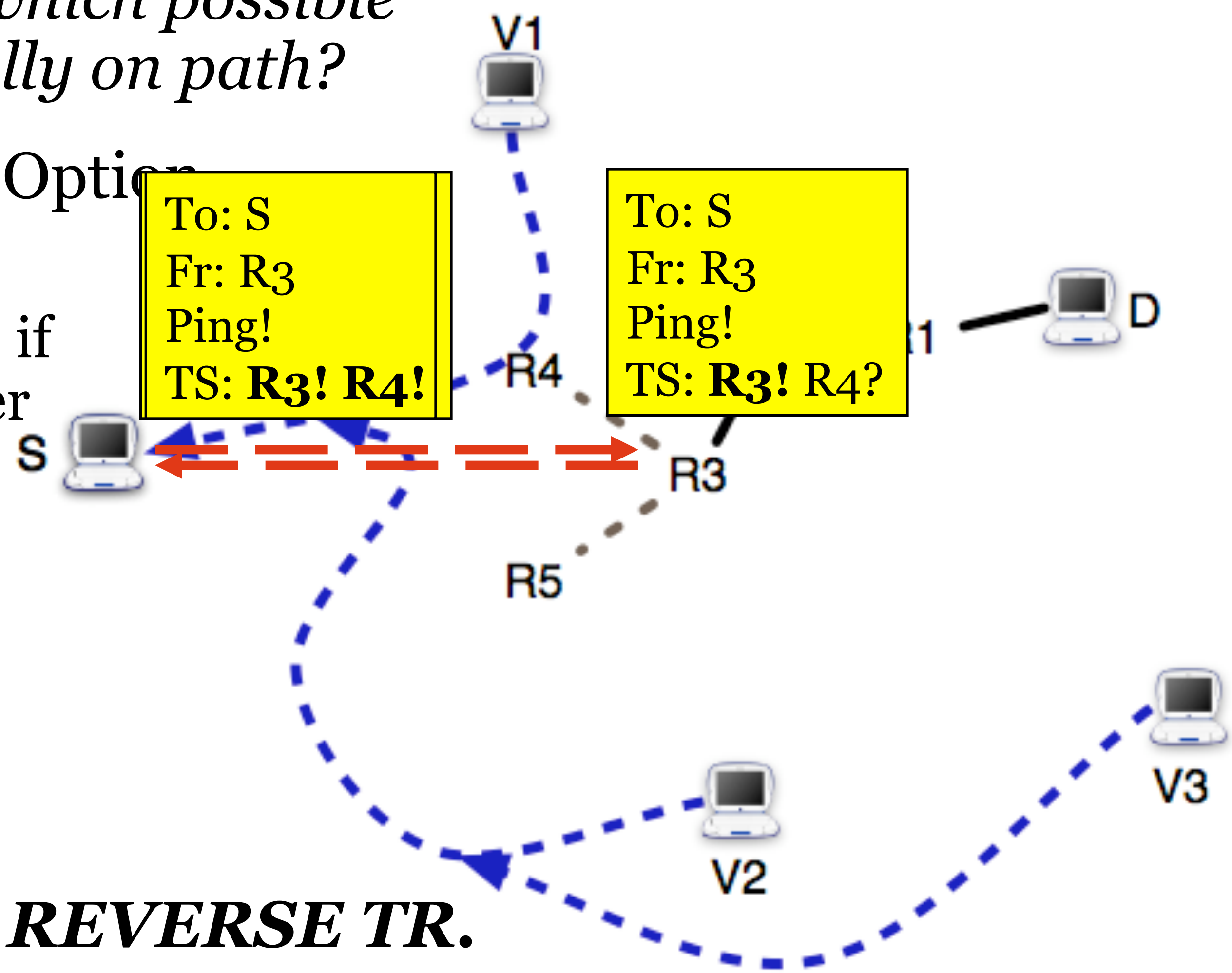
KEY IDEAS FOR REVERSE TR.

Known paths provide set of candidate next hops

How do we verify which possible next hop is actually on path?

IP Timestamp (TS) Option

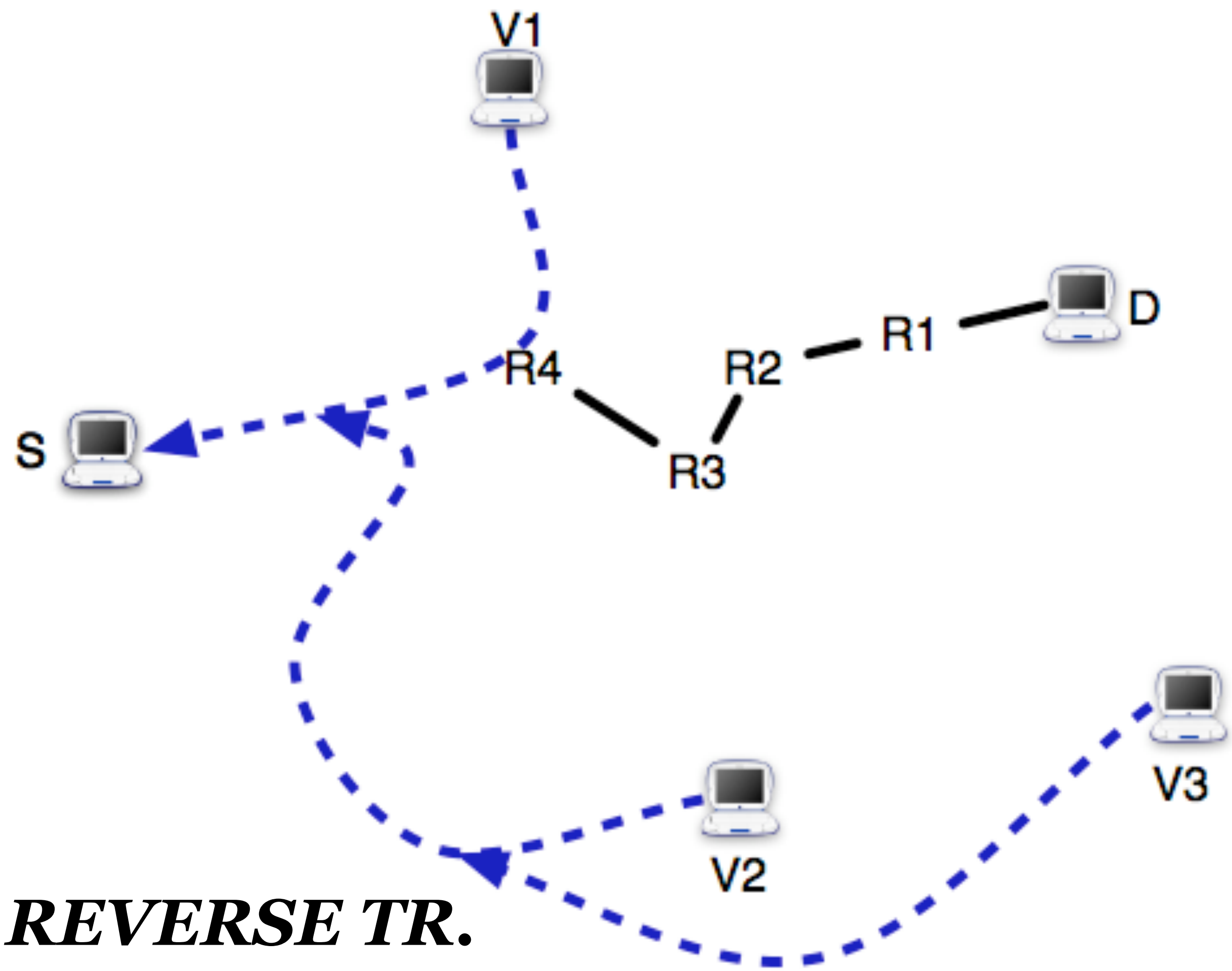
- Specify ≤ 4 IPs, each timestamps if traversed in order



KEY IDEAS FOR REVERSE TR.

Known paths provide set of candidate next hops

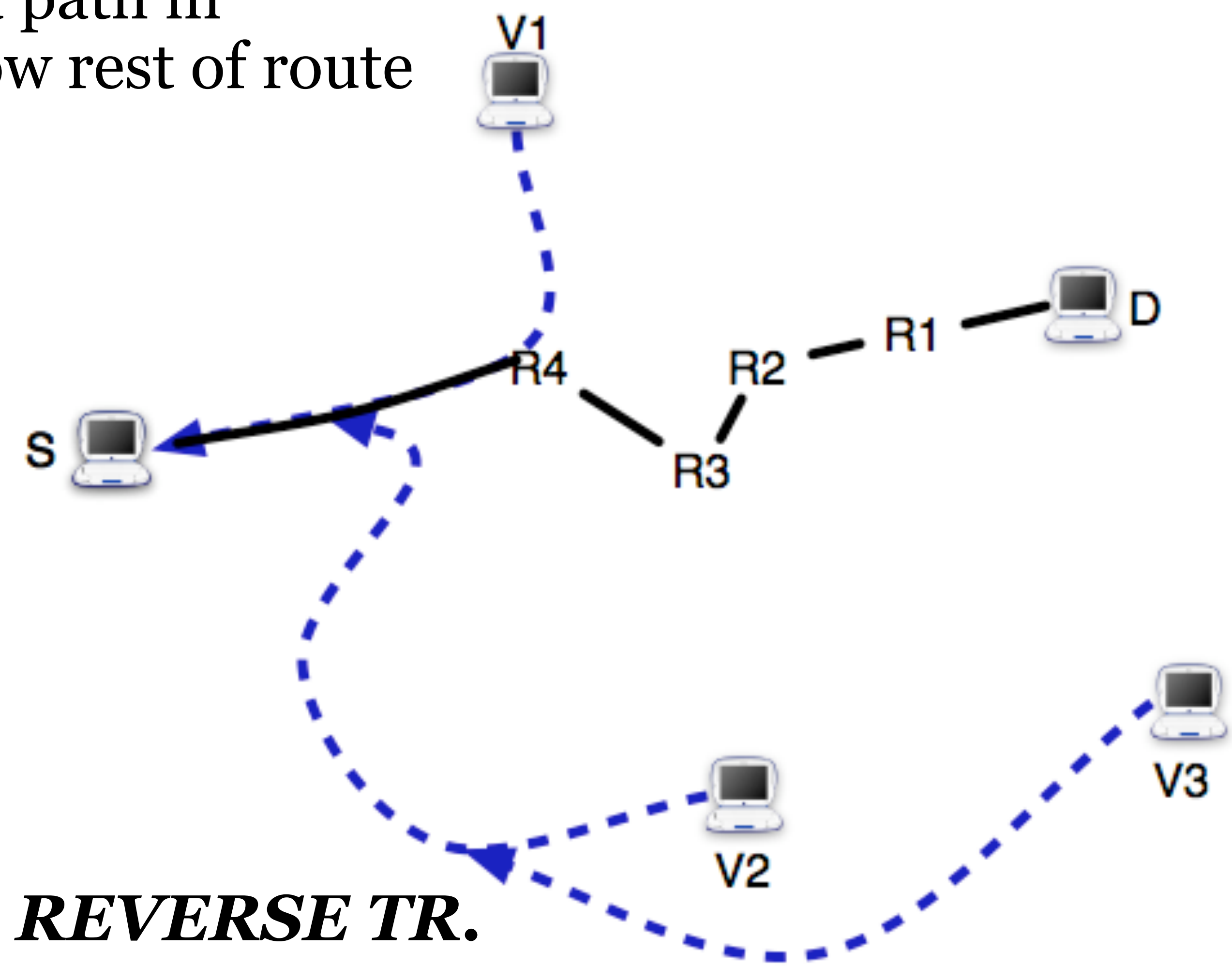
IP Options work over forward and reverse path



KEY IDEAS FOR REVERSE TR.

Destination-based routing lets us stitch path hop-by-hop

Once we intersect a path in our atlas, we know rest of route

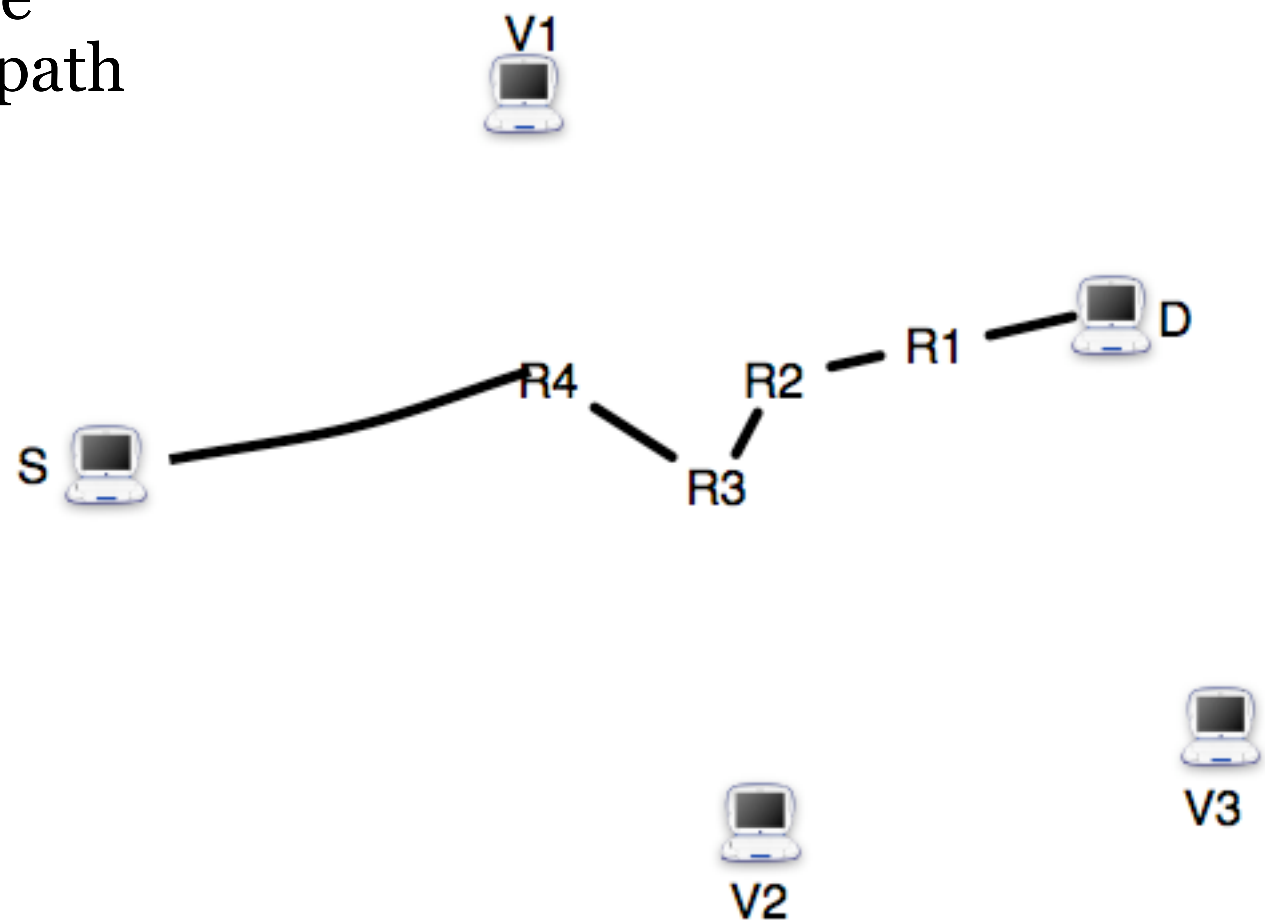


KEY IDEAS FOR REVERSE TR.

Destination-based routing lets us stitch path hop-by-hop

Traceroute atlas gives baseline we bootstrap from

Techniques combine
to give complete path



KEY IDEAS FOR REVERSE TR.

Destination-based routing lets us stitch path hop-by-hop

Traceroute atlas gives baseline we bootstrap from

Key Ideas For Reverse Traceroute

Works without control of destination

Multiple vantage points

Traceroute atlas provides:

- Baseline paths
- Adjacencies

Stitch path hop-by-hop

IP Options work over forward and reverse path

Spoofing lets us use vantage point in best position

Additional techniques to address:

Accuracy: Some routers process options incorrectly

Coverage: Some ISPs filter probe packets

Scalability: Need to select vantage points carefully

Deployment

Coverage tied to set of vantage points (VPs)

Current deployment:

- VPs: ~90 PlanetLab / Measurement Lab sites
- Sources: PlanetLab sites
- Try it at http://revtr.cs.washington.edu

Evaluation

Quick summary:

Coverage: The combination of techniques is necessary to get good coverage

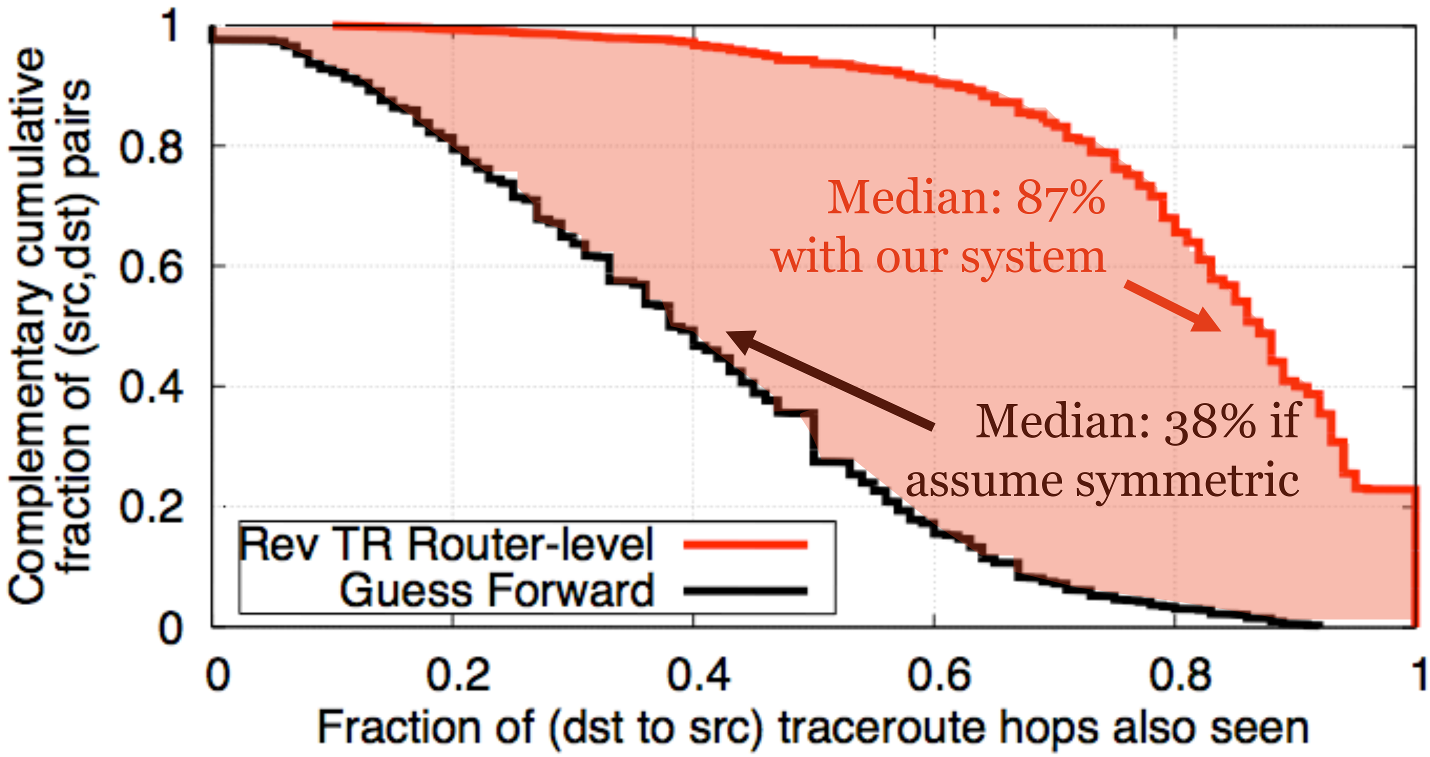
Overhead: Reasonable overhead, 10x traceroute (in terms of time, # of probes)

Next:

Accuracy: Does it yield the same path as if you could issue a traceroute from destination?

- 2200 PlanetLab to PlanetLab paths
- Allows comparison to direct traceroute on “reverse” path

Does it give the same path as traceroute?



We identify most hops seen by traceroute

Why we do not always see all the traceroute hops:

1. Hard to know if 2 IPs actually are the same router
2. Coverage will improve further with more vantage points

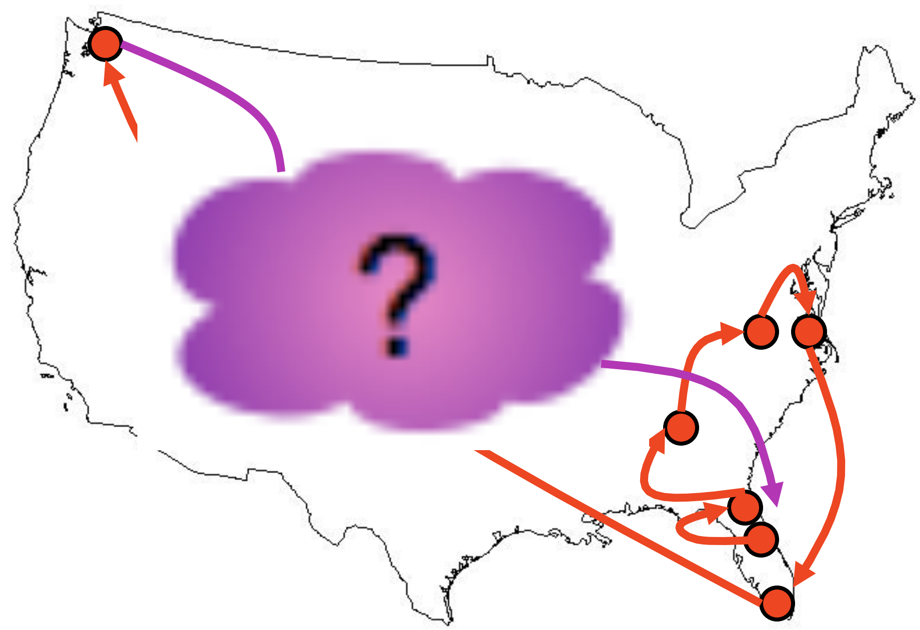
Example of debugging inflated path

150ms round-trip time Orlando to Seattle, 2-3x expected

- E.g., Content provider detects poor client performance

(Current practice) Issue traceroute, check if indirect

Hop no.	DNS name / IP address
1	132.170.3.1
2	198.32.155.89
3	**JAX-FL**. . . net.flrnet.org
4	**ATLANTA**ix.cox.com
5	**ASH**. . . as.cox.net
6	core2. . . **WDC**.pnap.net
7	cr1.**WDC**. . . internap.net
8	cr2-cr1.**WDC**. . . internap.net
9	cr1.**MIA**. . . internap.net
10	cr1.**SEA**. . . internap.net



Indirectness: FL→DC→FL

But only explains half of latency inflation

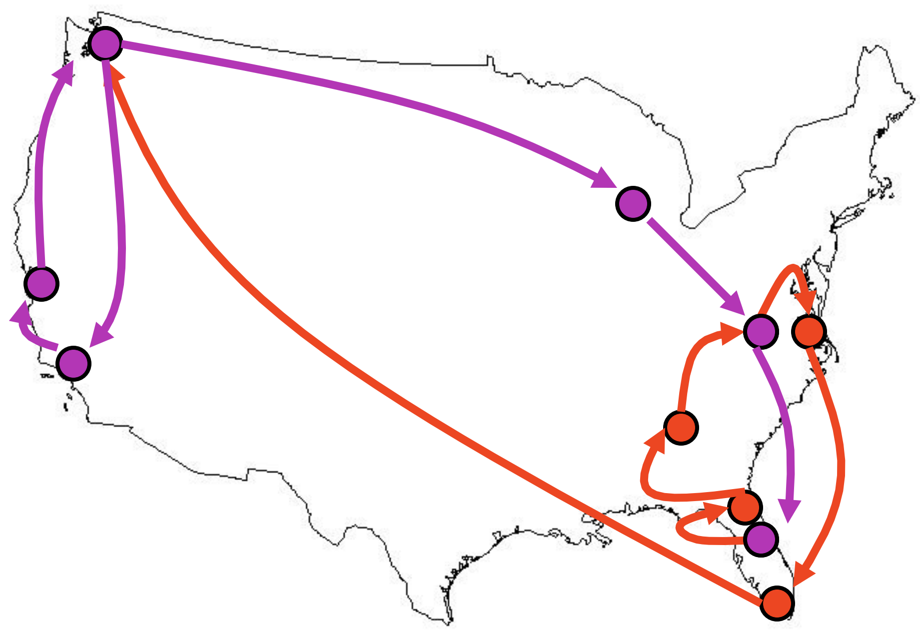
Example of debugging inflated path

(Current practice) Issue traceroute, check if indirect

- Does not fully explain inflated latency

(Our tool) Use reverse traceroute to check reverse path

Hop no.	DNS name / IP address
1	cr1.**SEA**...internap.net
2	cr1.**SEA**...internap.net
3	internap...**LSANCA**01.transitrail.net
4	te4...**LSANCA**01.transitrail.net
5	te4...**PLALCA**01.transitrail.net
6	te4...**STTLWA**01.transitrail.net
7	te4...**CHCGIL**01.transitrail.net
8	te2...**ASBNVA**01.transitrail.net
9	132.170.3.1
10	planetlab2.eecs.**UCF.EDU**



Indirectness: WA →LA→WA

Bad reverse path causes inflated round-trip delay

Operators Struggle to Locate Failures

"Traffic attempting to pass through Level3's network in the Washington, DC area is getting lost in the abyss. Here's a trace from Verizon residential to Level3."

Outages mailing list, December 2010

<u>Mailing List User 1</u>

1 Home router

2 Verizon in Baltimore

3 Verizon in Philly

4 Alter.net in DC

5 Level3 in DC

6 * * *

7 * * *

<u>Mailing List User 2</u>

1 Home router

2 Verizon in DC

3 Alter.net in DC

4 Level3 in DC

5 Level3 in Chicago

6 Level3 in Denver

7 * * *

8 * * *

How Can We Locate a Problem?

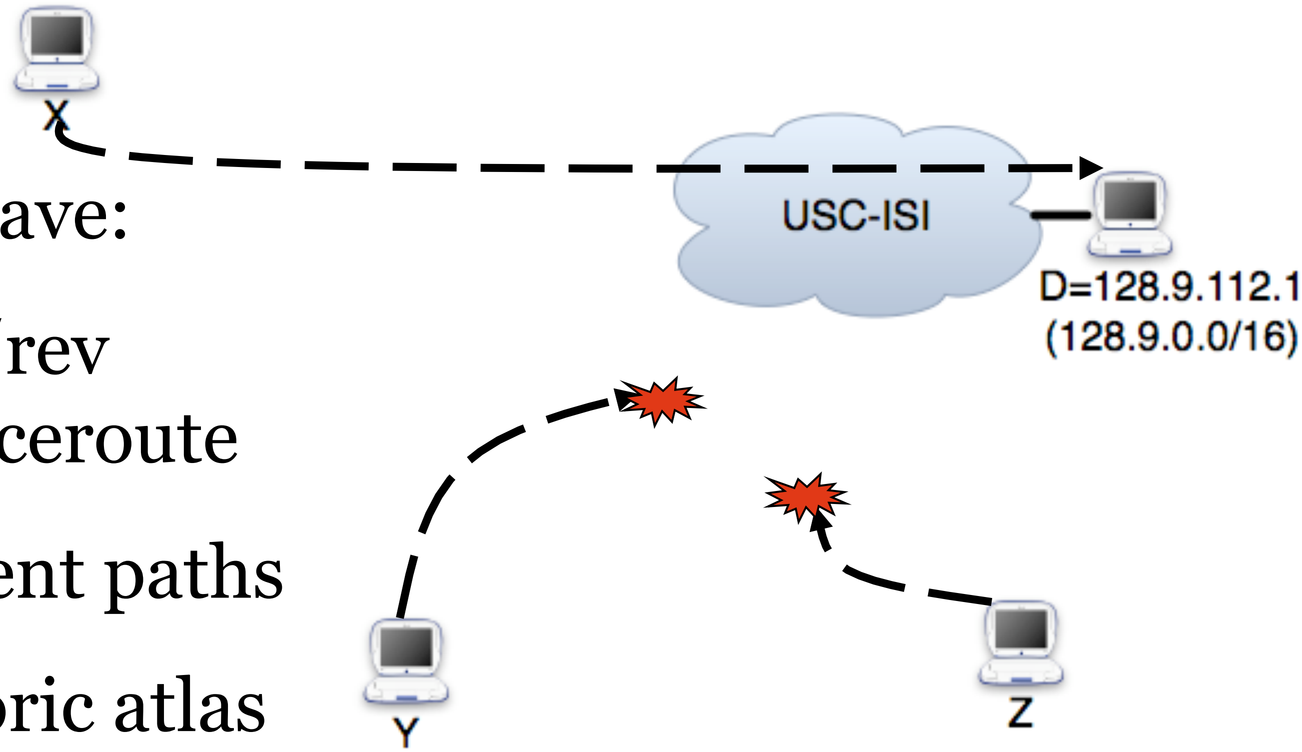
We have:

Fwd/rev traceroute

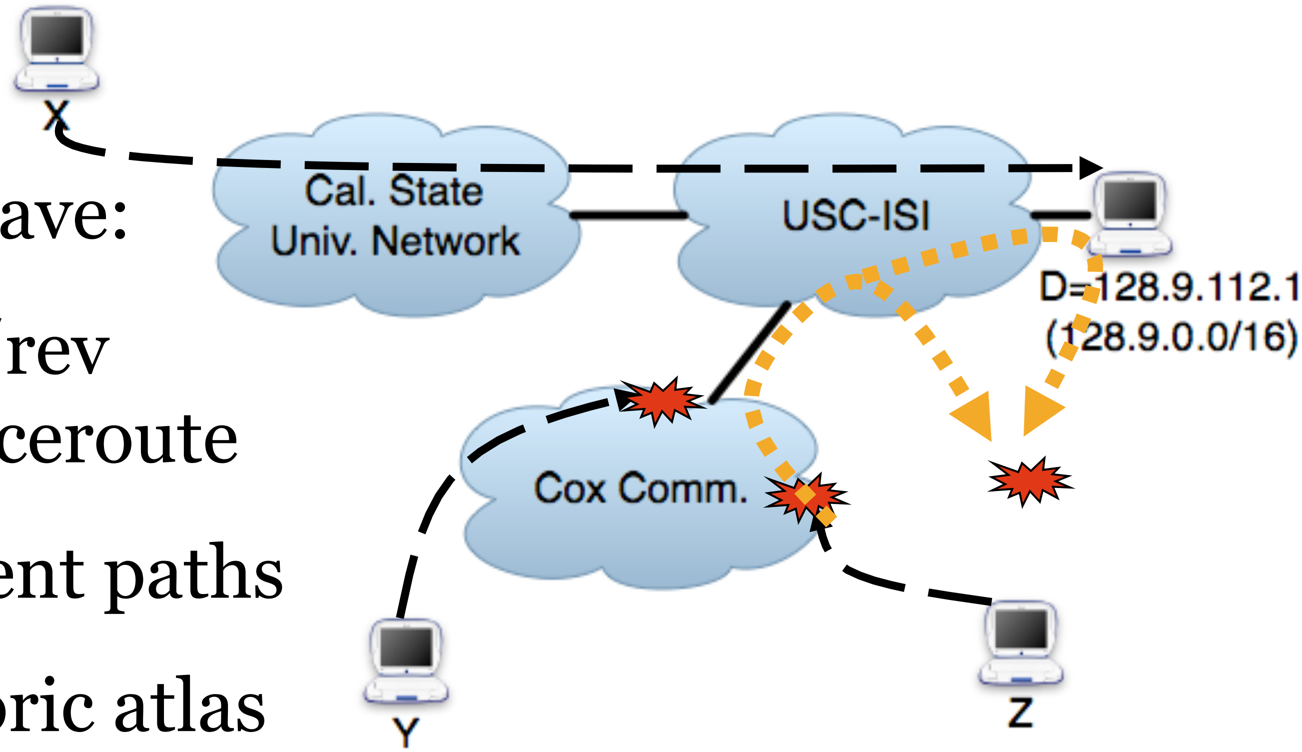
Current paths

Historic atlas

Group paths



How Can We Locate a Problem?



We have:

Fwd/rev traceroute

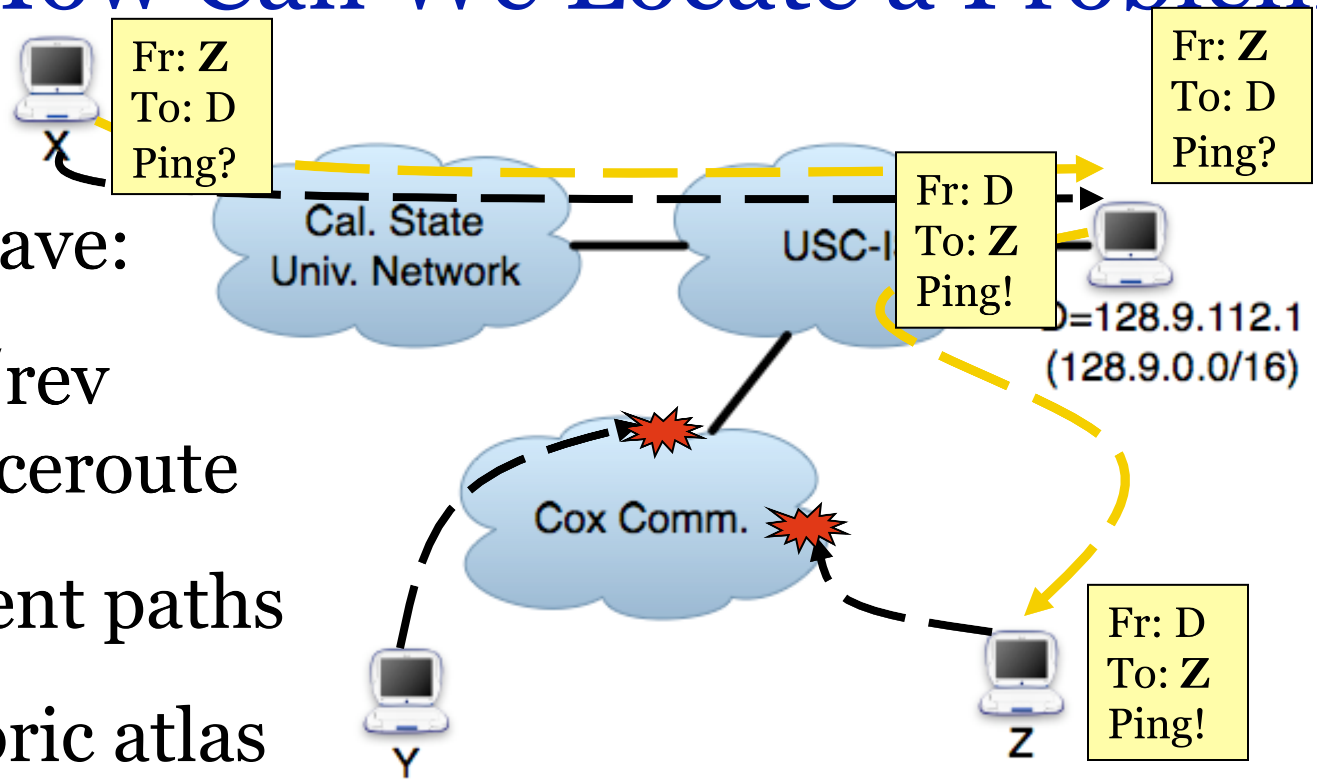
Current paths

Historic atlas

Group paths – Looks like Cox failure, but:

- Failure could be on reverse path
- Cannot tell which ISP is responsible, as paths may be asymmetric

How Can We Locate a Problem?



We have:

Fwd/rev traceroute

Current paths

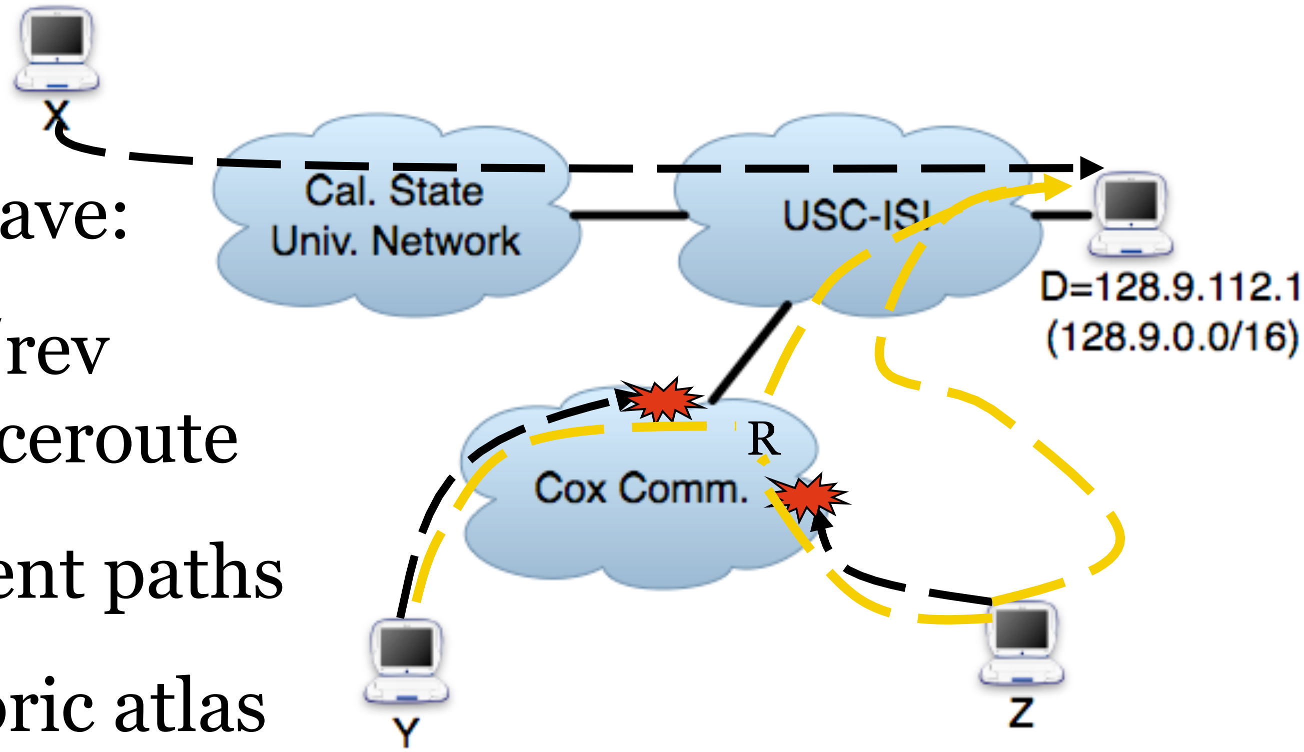
Historic atlas

Group paths

Use Reverse Traceroute to isolate direction

- Also lets us measure working direction

How Can We Locate a Problem?



We have:

Fwd/rev traceroute

Current paths

Historic atlas

Group paths

Use Reverse Traceroute to isolate direction

Use historic atlas to reason about what changed

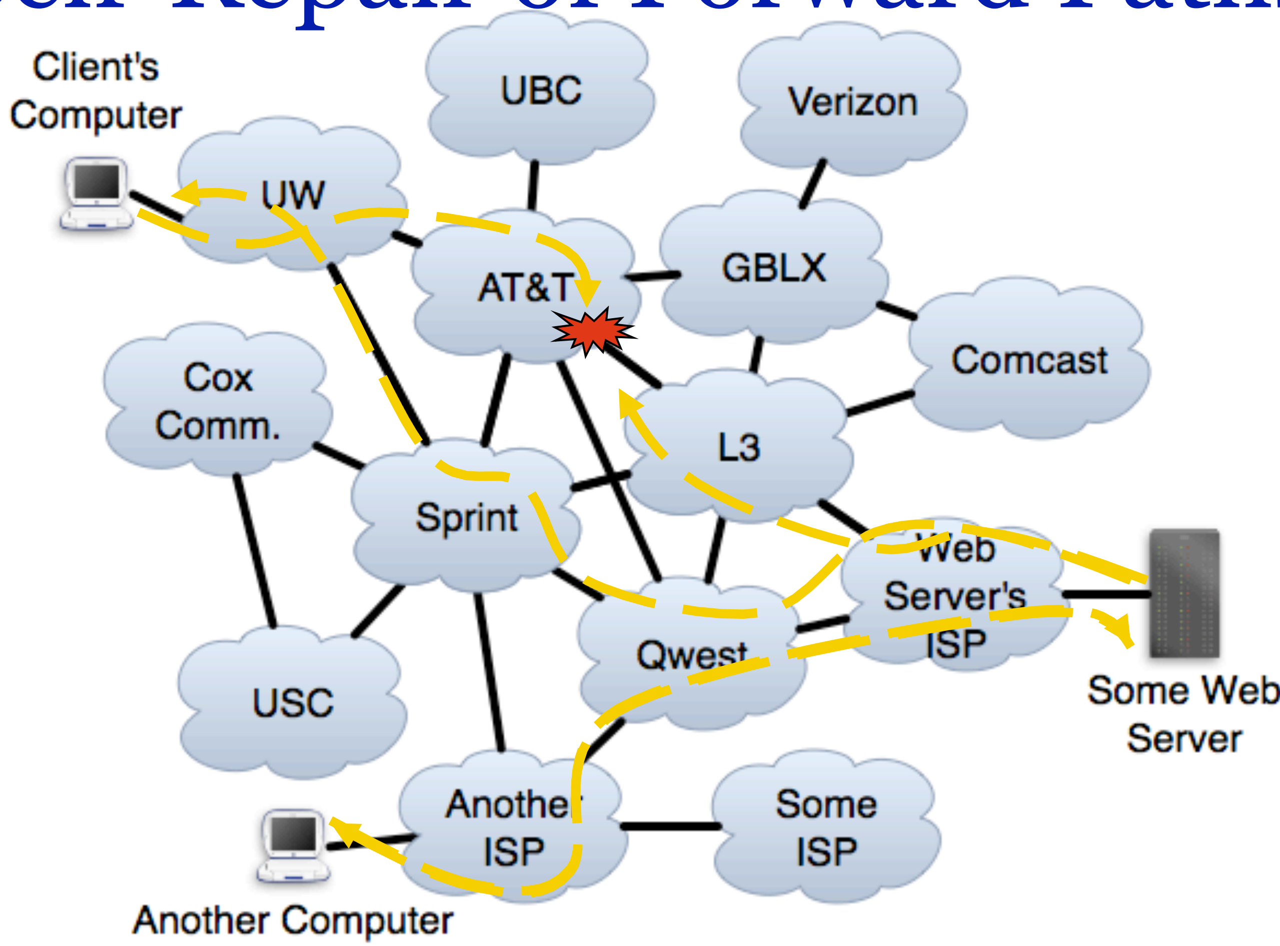
Partial Outages: An Opportunity

Initial version of isolation system running continuously. Preliminary results:

Working routes exist, even during failures

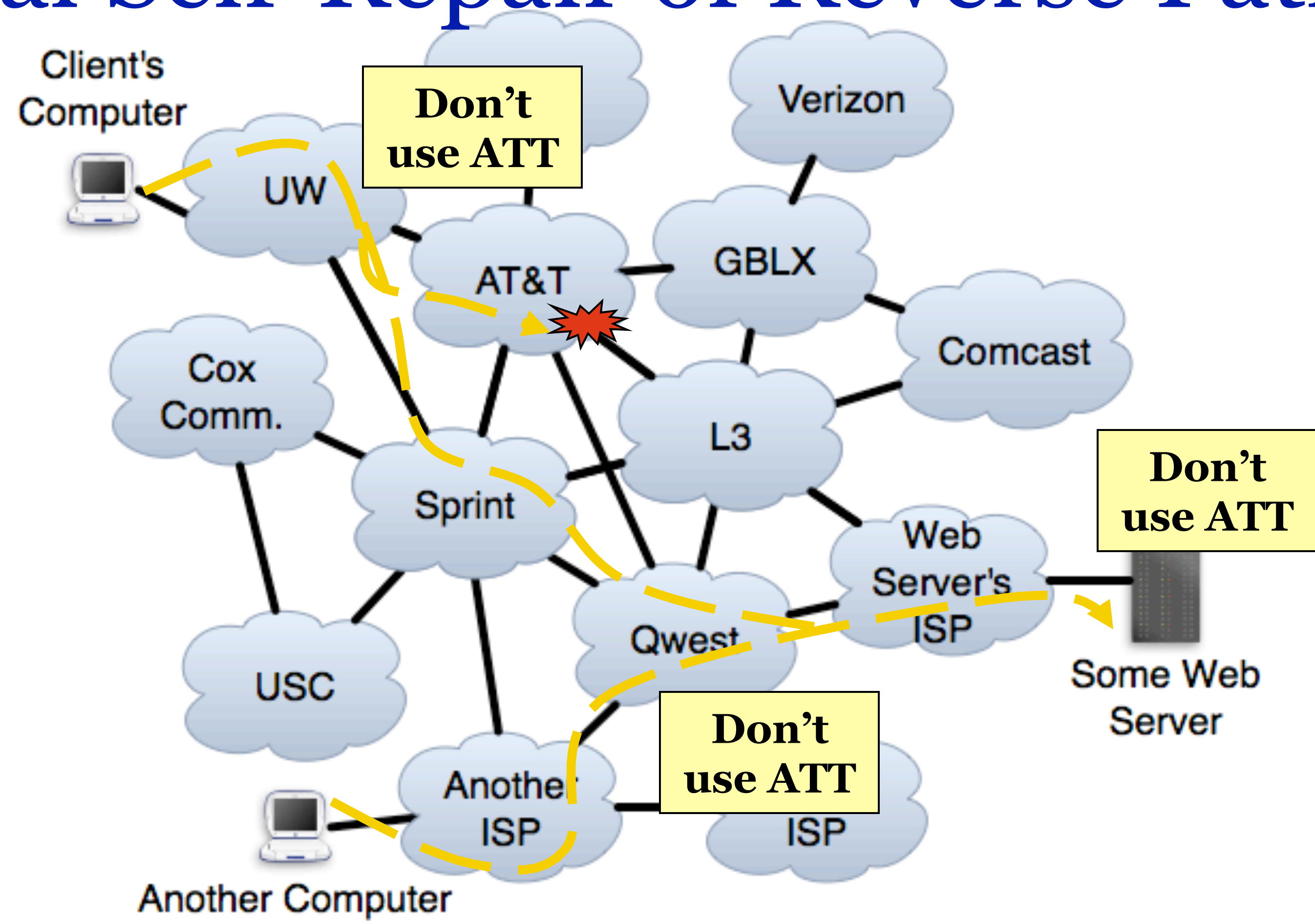
- 68% of black holes are partial
  - Paths from some vantage points fail, others work
- Can't be explained by hardware failure: misconfiguration or result of policy
- 69% are one-way failures, other direction work

Self-Repair of Forward Paths



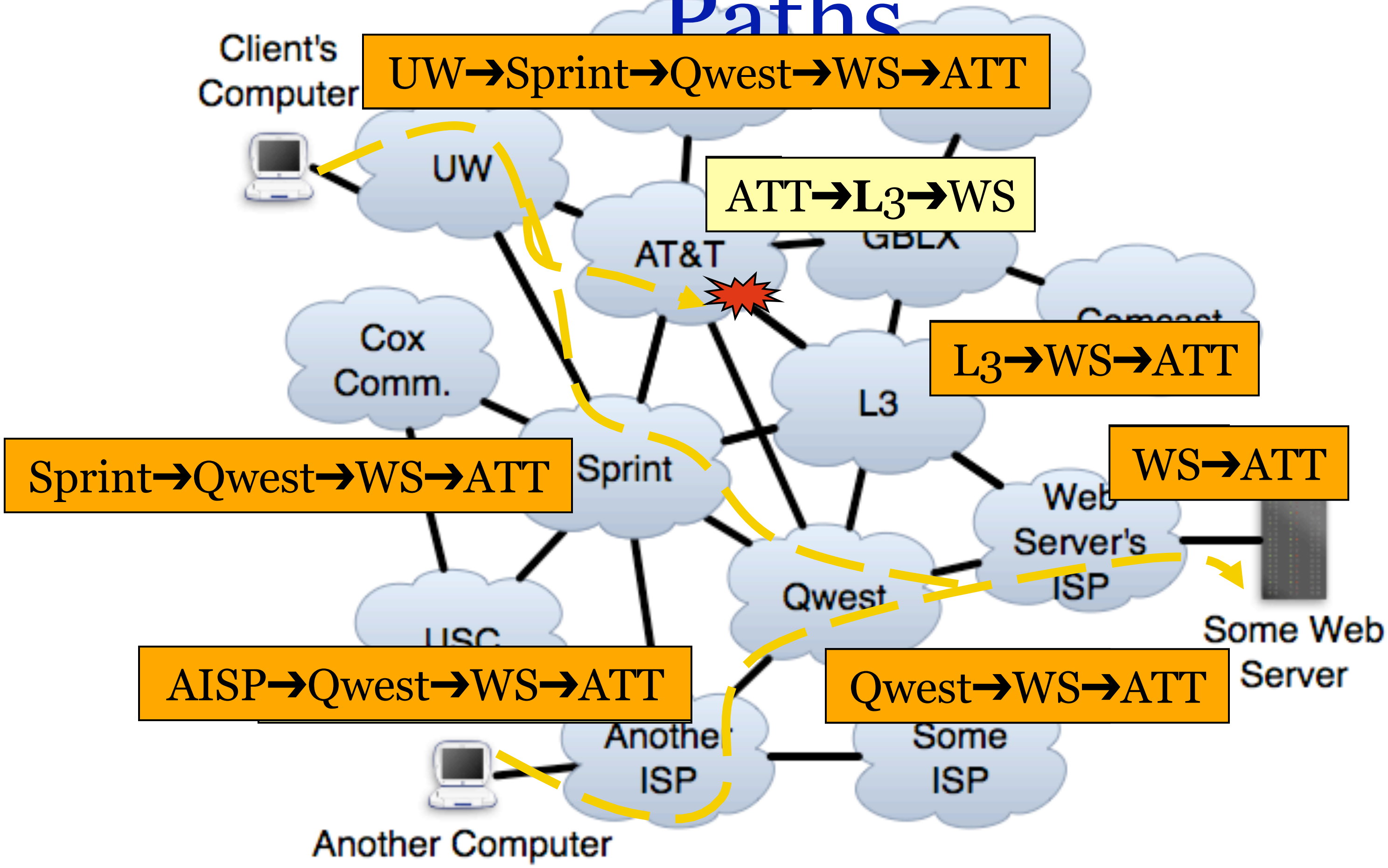
Straightforward: Choose a different path or data center.

Ideal Self-Repair of Reverse Paths



We want a way to signal to ISPs which networks to avoid.

Practical Self-Repair of Reverse Paths



Use BGP loop prevention to force switch to working path.

Remediation Goals

Without control of the network causing a failure, automatically reroute traffic in a way that is:

Effective: Allows networks to avoid failure

Non-disruptive: Little effect on working paths

Predictable: Understandable effect, and reverts when no longer needed

BGP loop-prevention as our basic mechanism, with:

Proposed techniques for each of 3 properties

Experiments in progress

Summary

Substantial improvements in Internet availability are both needed, and possible

Interdomain routing convergence (consensus routing)

- Towards high availability at a fine-grained time scale

Interdomain routing diagnosis (Hubble/reverse traceroute)

- Towards high availability at a long time scale

Distributed denial of service protection (phalanx)

- Towards withstanding million node botnets

Final Thought

"A good network is one that I never have to think about" – Greg Minshall

Botnets are Big

Botnet: Group of infected computers controlled by a hacker to launch various attacks

- Infected via viruses, trojans and worms
- Botnets patch the vulnerability to let the hacker maintain control
- Self-sustaining economy in attack technologies

Total bots:

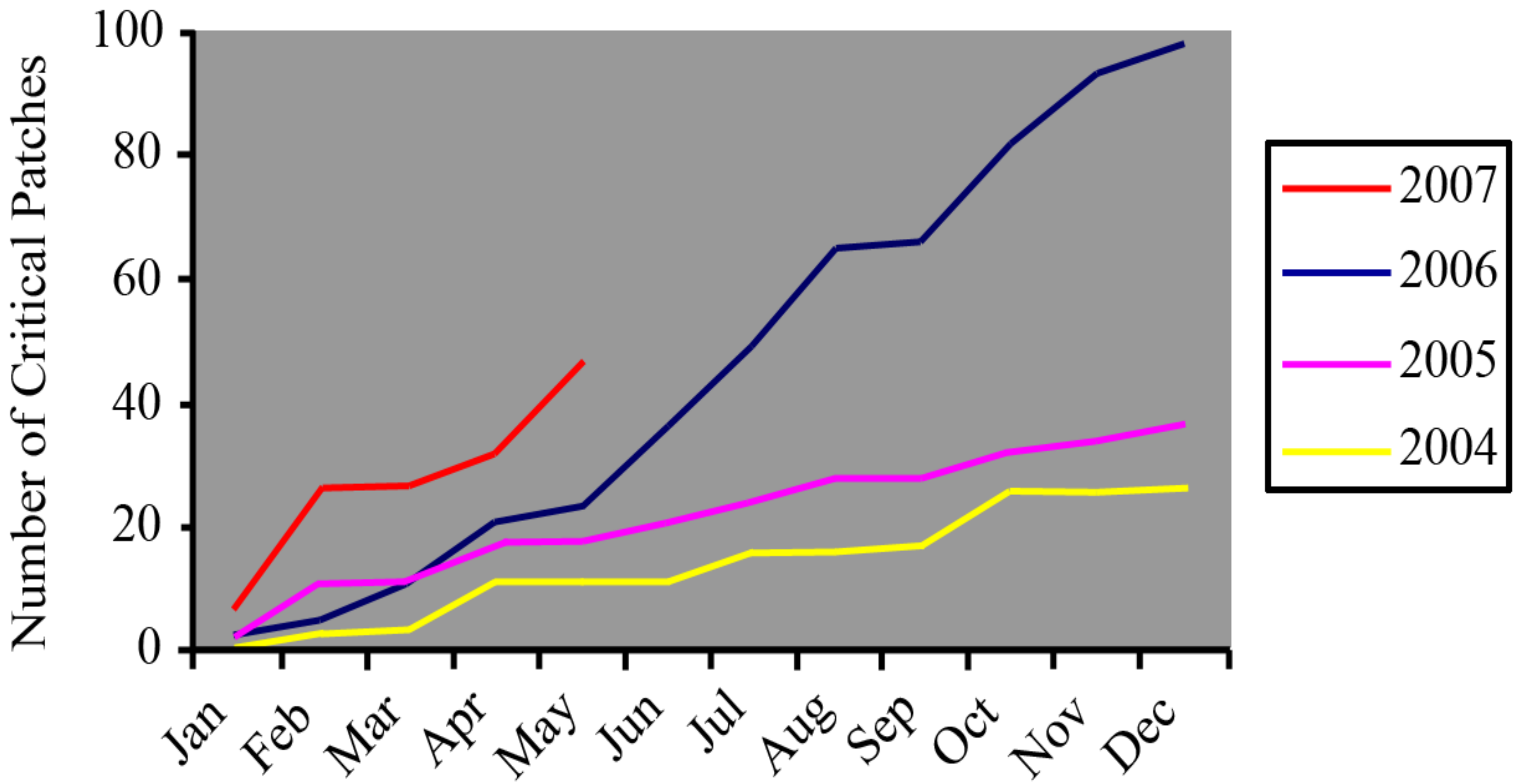
- 6 million [Symantec]
- 150 million [Vint Cerf]

Single botnets have numbered 1.5 million

Back of the envelope: 4.5 Tb/s attack possible today

- If average bot matches bittorrent distribution

Plenty of Vulnerabilities



Solution Space

Many research proposals for in-network changes (traceback, pushback, AITF, TVA, SIFF, NIDS, …)

- But a million node botnet => need near complete deployment
- Plus a terabit/sec can overwhelm any NIDS

For read-only data, Akamai is an effective solution

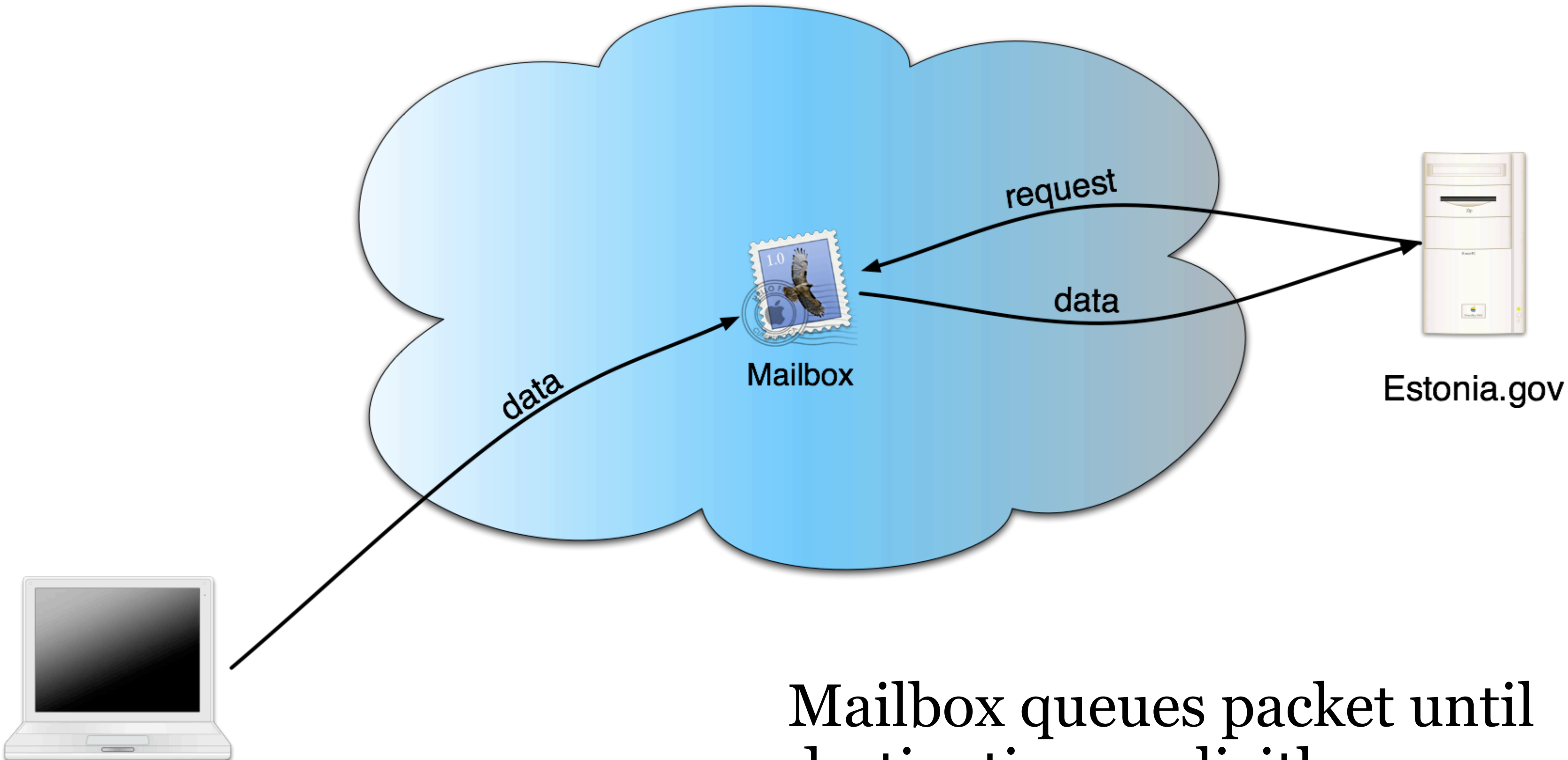
- Put a copy of the data on every Akamai node
- Works today for most US government web sites

Many services aren't read-only:

- Estonia (egovt), IRS e-filing, Amazon, eBay, Skype, etc.

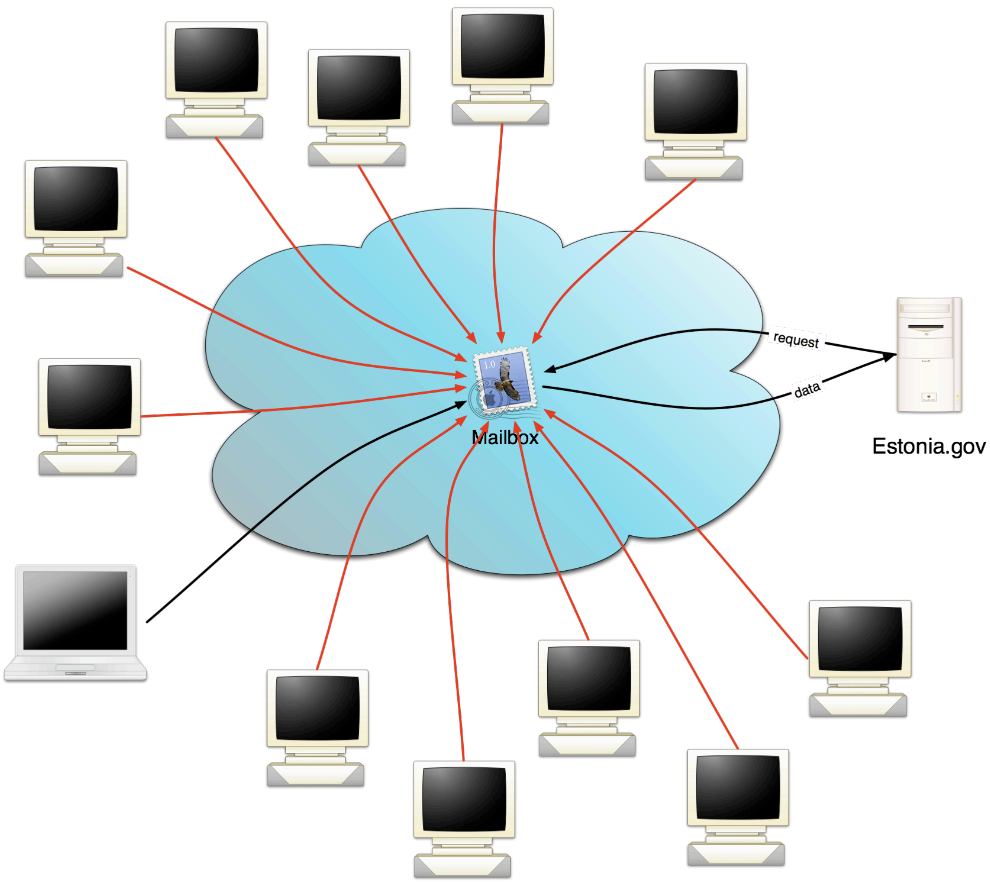
What if we had a swarm for this case?

Single Mailbox



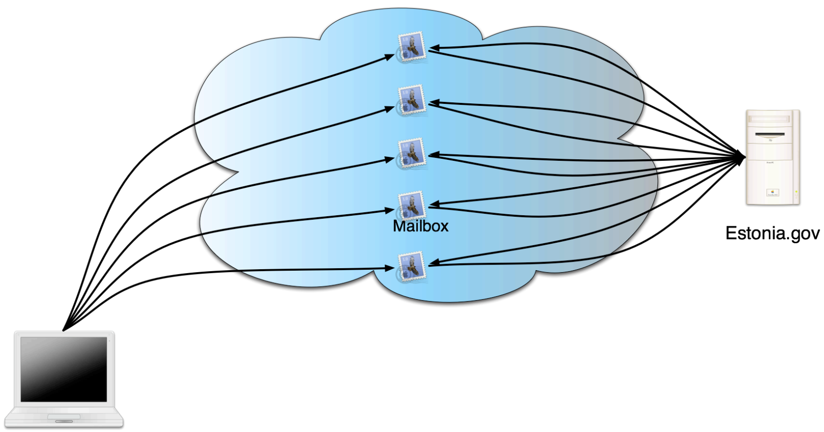
Mailbox queues packet until destination explicitly requests it

Single Mailbox



If the botnet can discover the mailbox, game over

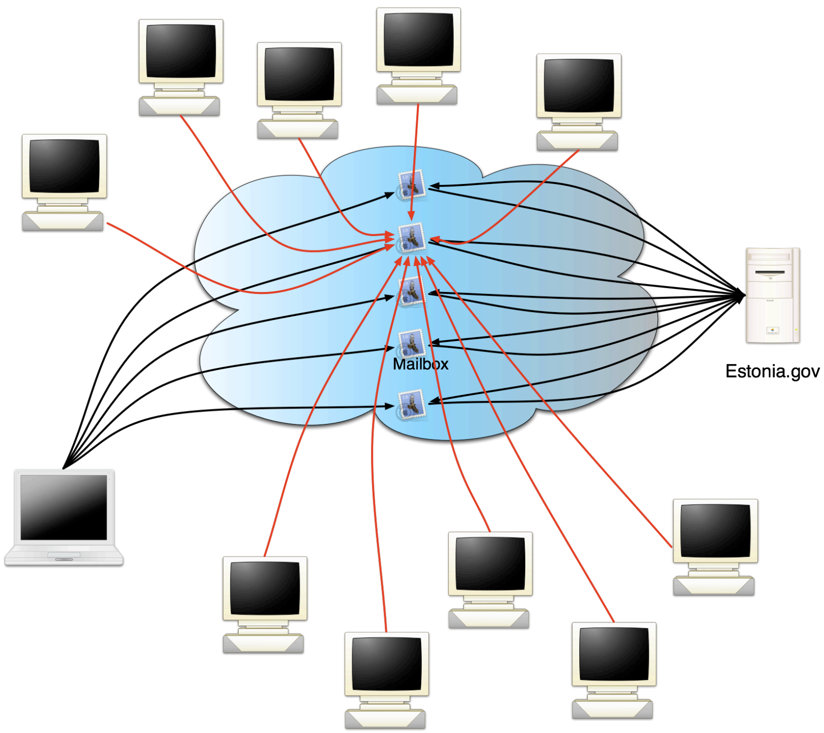
Many Mailboxes



Source sends packets through a random sequence of mailboxes

Sequence known to destination, but not to attacker

Many Mailboxes

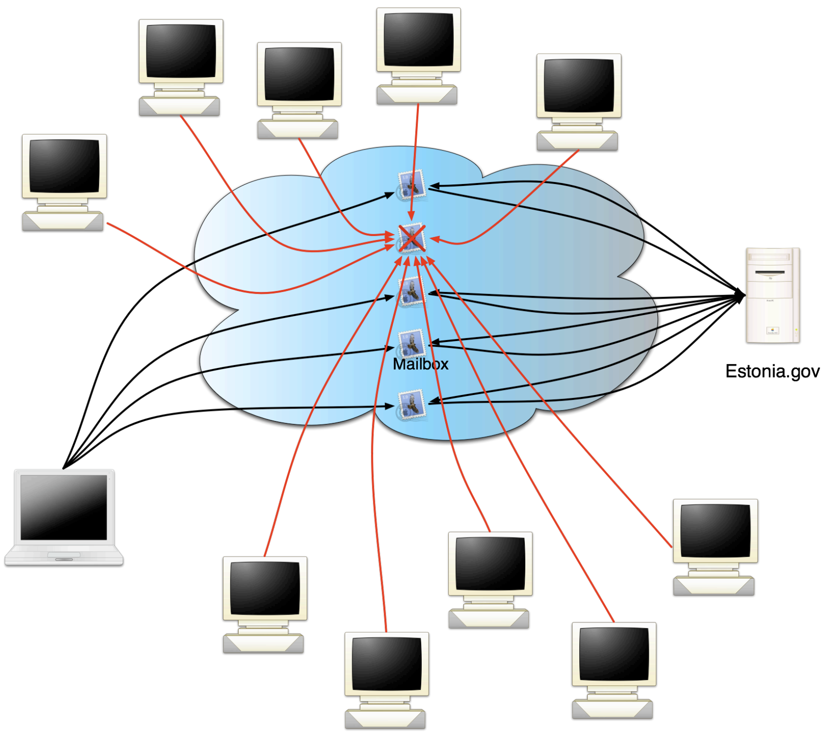


Source sends packets through a random sequence of mailboxes

Sequence known to destination, but not to attacker

Botnet can take down one mailbox

Many Mailboxes



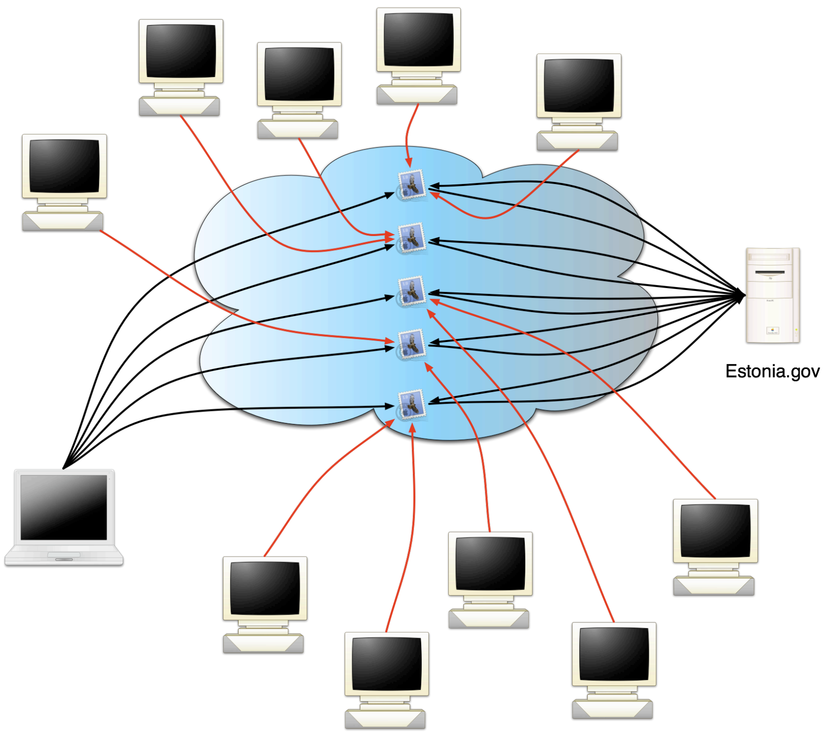
Source sends packets through a random sequence of mailboxes

Sequence known to destination, but not to attacker

Botnet can take down one mailbox

But communication continues

Many Mailboxes



Source sends packets through a random sequence of mailboxes

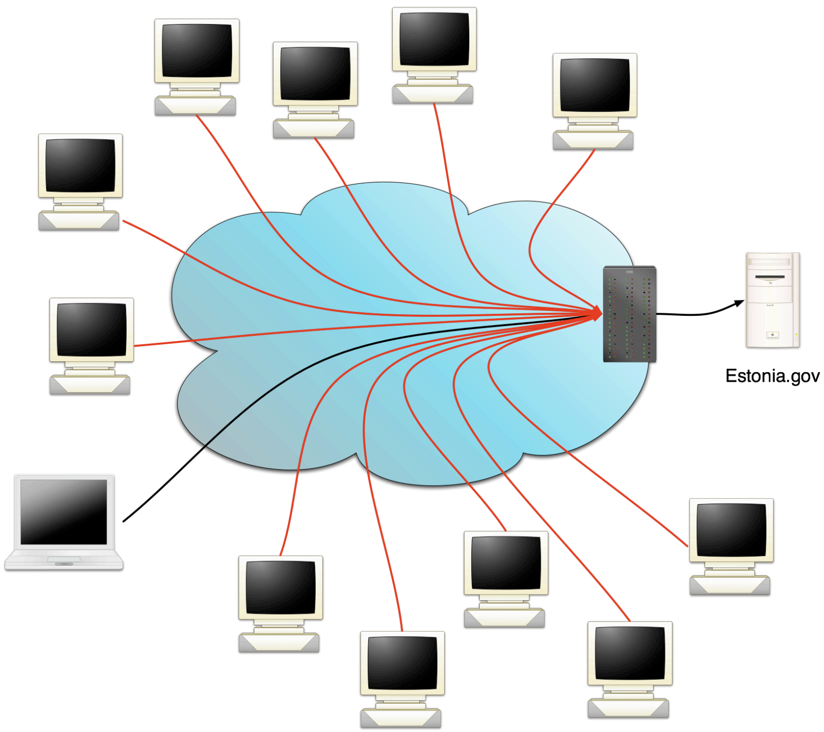
Sequence known to destination, but not to attacker

Botnet can take down one mailbox

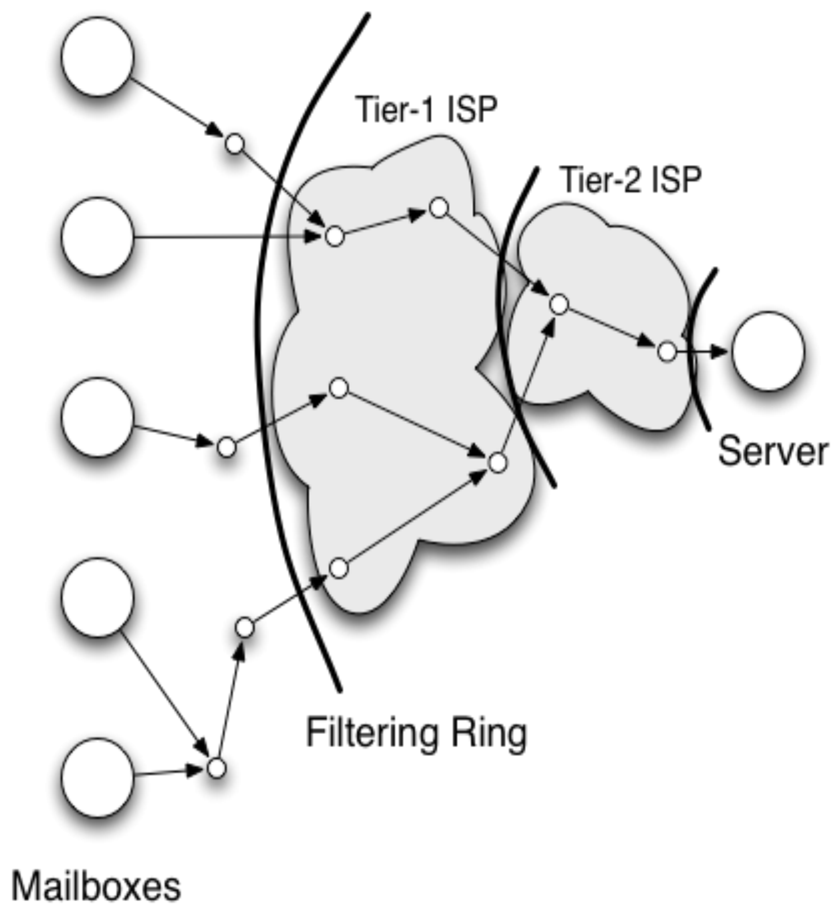
But communication continues

Diluted attacks against all mailboxes fail

Why not just attack the server?



Filtering Ring



Each request has a nonce

Exit router keeps a list of requests

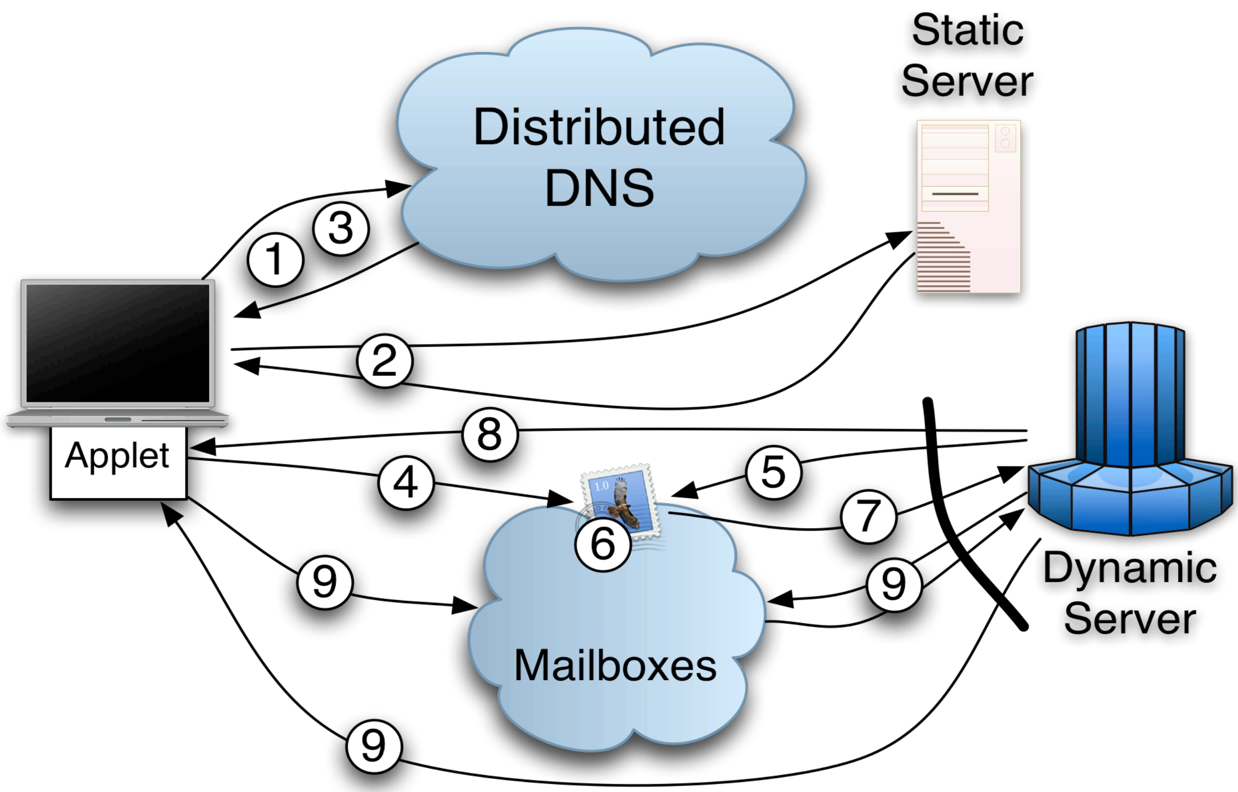
Drop all incoming pkts without the nonce

Remove the nonce once used

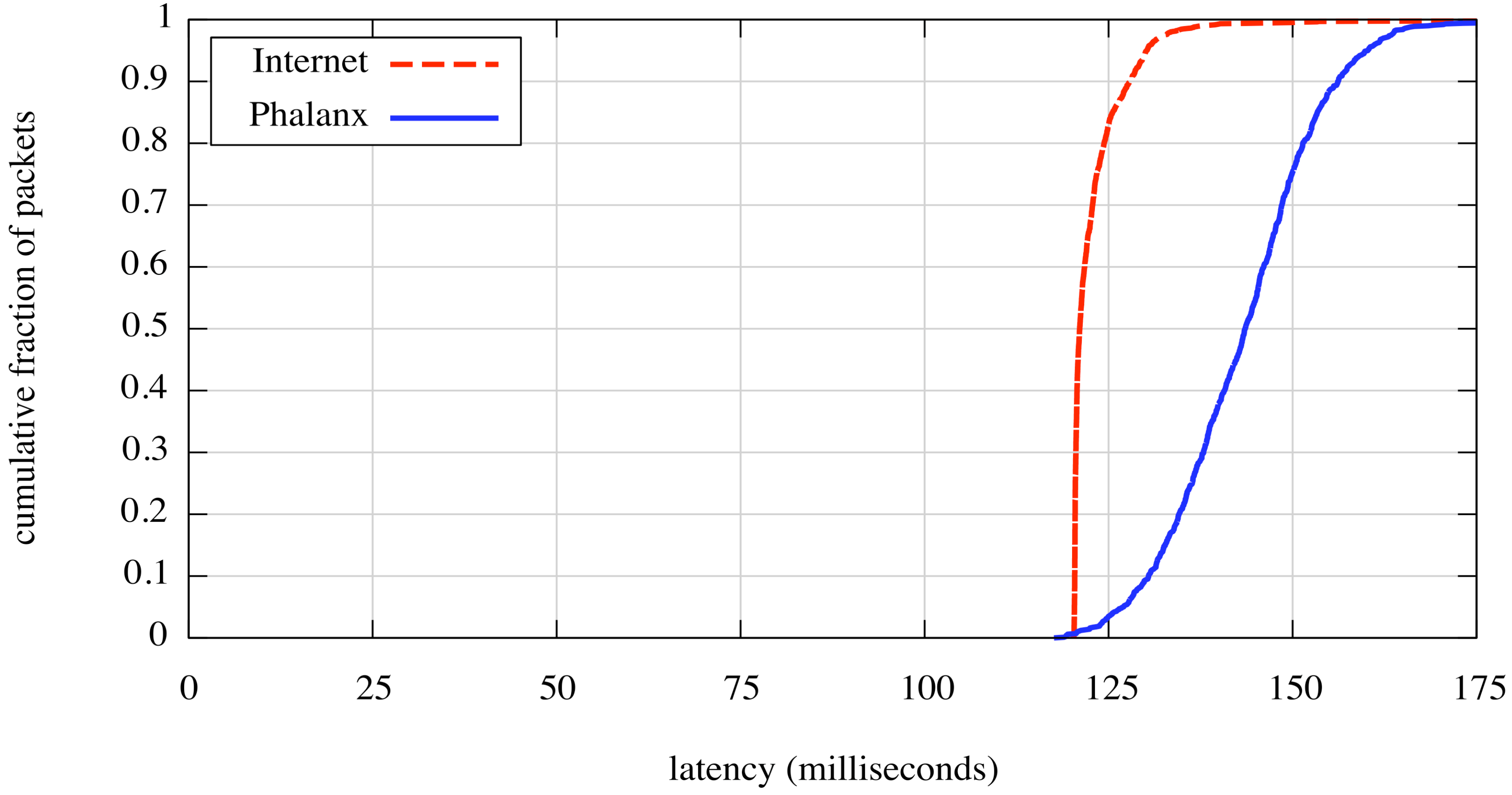
Efficient implementation using bloom filters

Attack needs to flood all border routers of an ISP to be effective

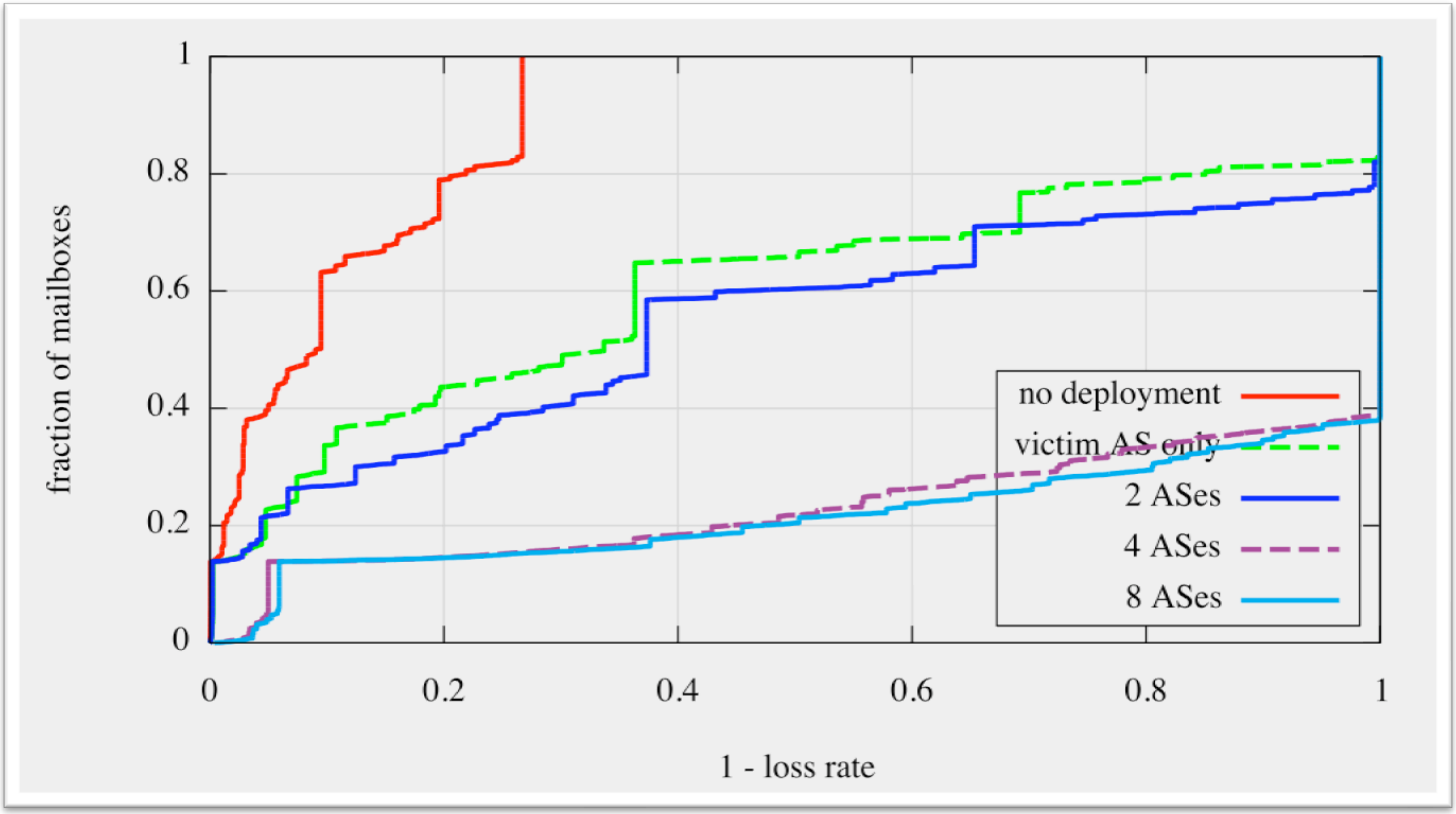
Phalanx Example



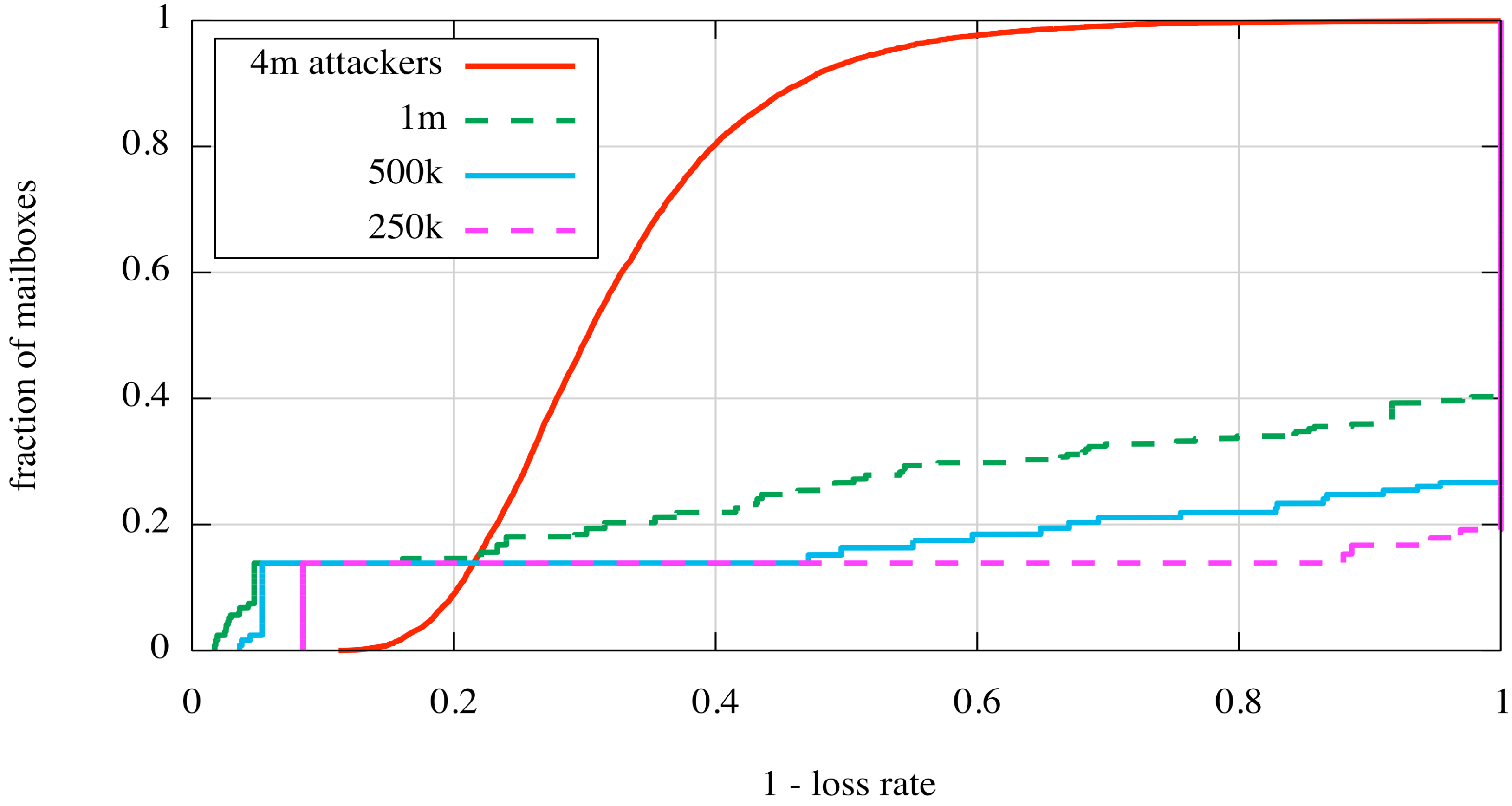
Phalanx Latency Penalty



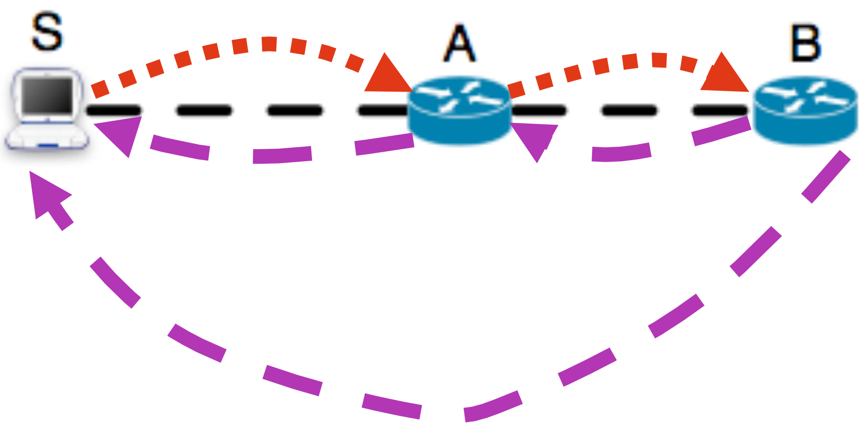
Phalanx vs. In Network Solutions



Phalanx Scalability



Measuring Link Latency



Many applications want link latencies

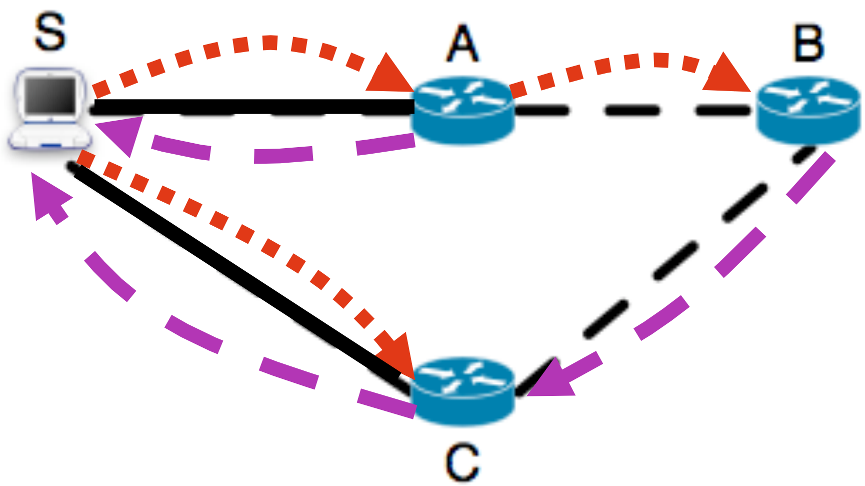
- IP geolocation, ISP performance, performance prediction, …

Traditional approach is to assume symmetry:

$$\text{Delay}(A,B) = (\ \text{RTT}(S,B) - \text{RTT}(S,A)\ ) / 2$$

Asymmetry skews link latency inferred with traceroute

Reverse Traceroute Detects Symmetry

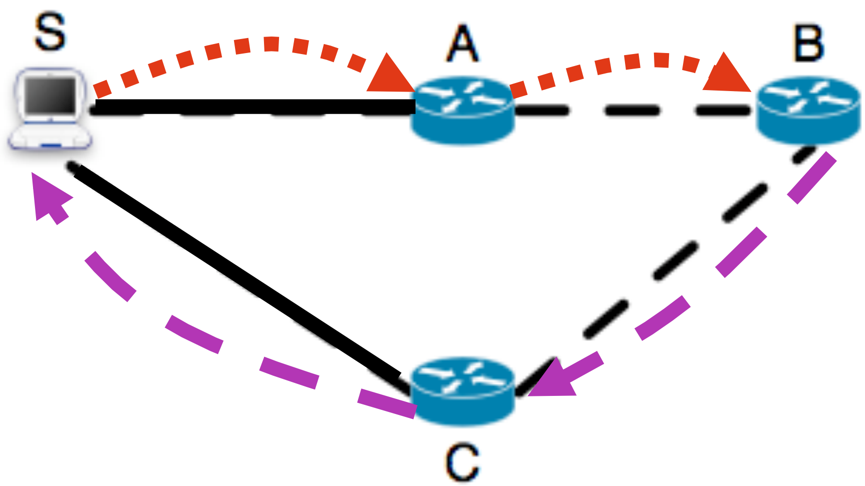


Solved
(S,A)
(S,C)

Reverse traceroute identifies symmetric traversal

- Identify cases when RTT difference is accurate
- We can determine latency of (**S,A**) and (**S,C**)

Reverse TR Constrains Link Latencies

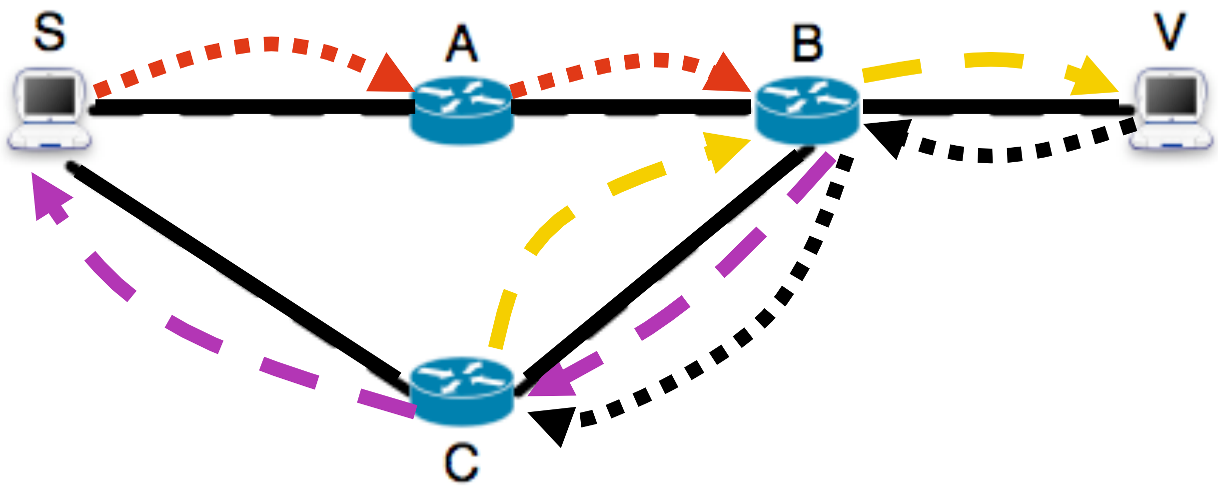


Solved
(S,A)
(S,C)

Build up system of constraints on link latencies of all intermediate hops

- Traceroute and reverse traceroute to all hops
- RTT = Forward links + Reverse links

Reverse TR Constrains Link Latencies

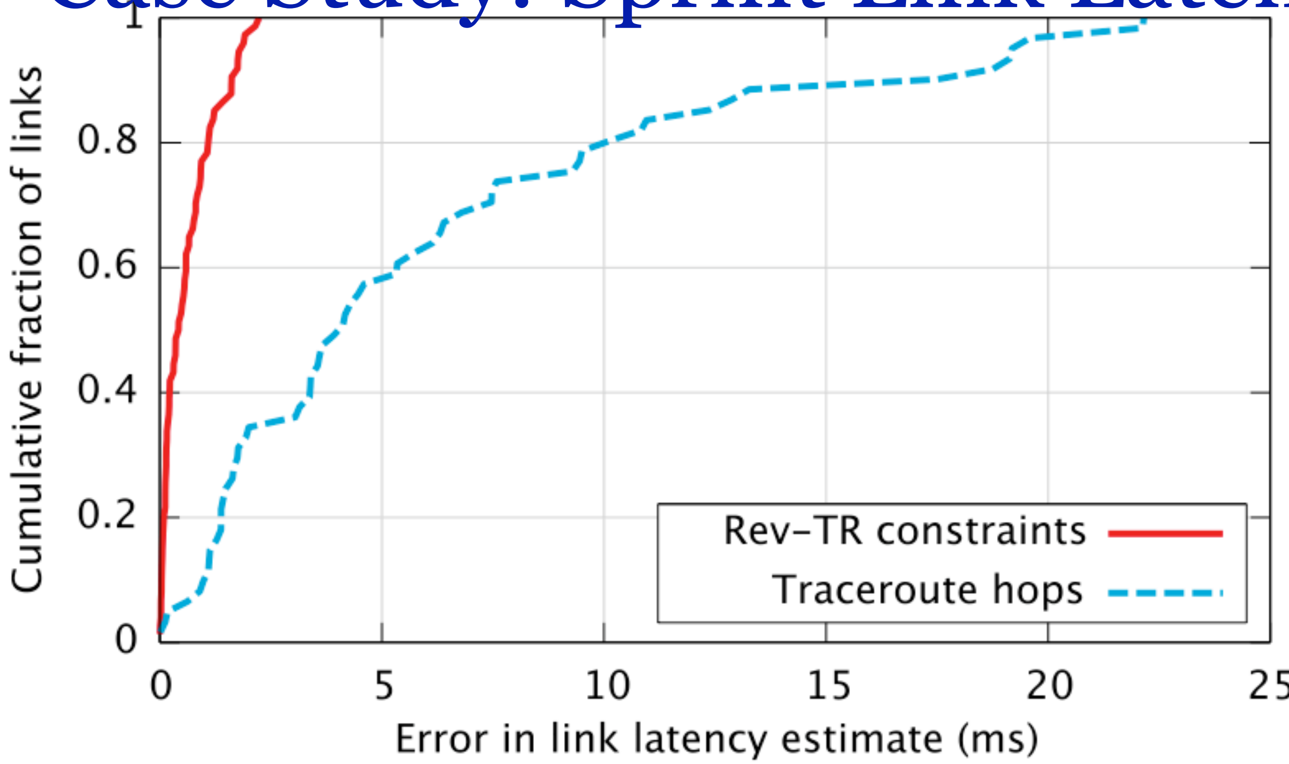


Solved
(S,A)
(S,C)
(V,B)
(B,C)
(A,B)

Build up system of constraints on link latencies of all intermediate hops

- Traceroute and reverse traceroute to all hops
- RTT = Forward links + Reverse links

Case Study: Sprint Link Latencies



Reverse traceroute sees 79 of 89 inter-PoP links, whereas traceroute only sees 61

Median (0.4ms), mean (0.6ms), worst case (2.2ms) error all 10*x* better than with traditional approach