Failure Isolation in the Wide Area

Ethan Katz-Bassett, David Choffnes, Colin Scott, Harsha Madhyastha, Arvind Krishnamurthy and Tom Anderson

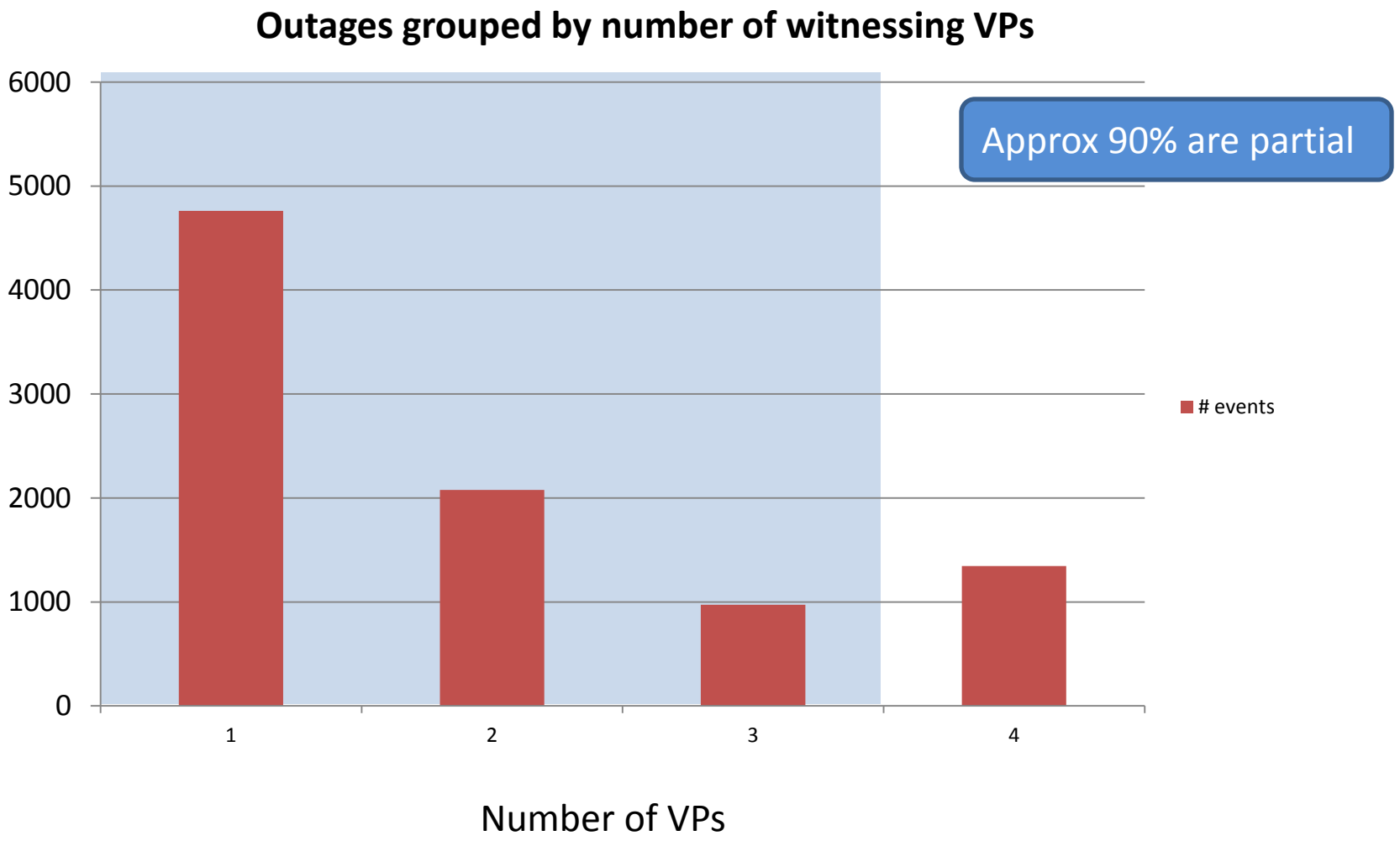
University of Washington

*funded by NSF

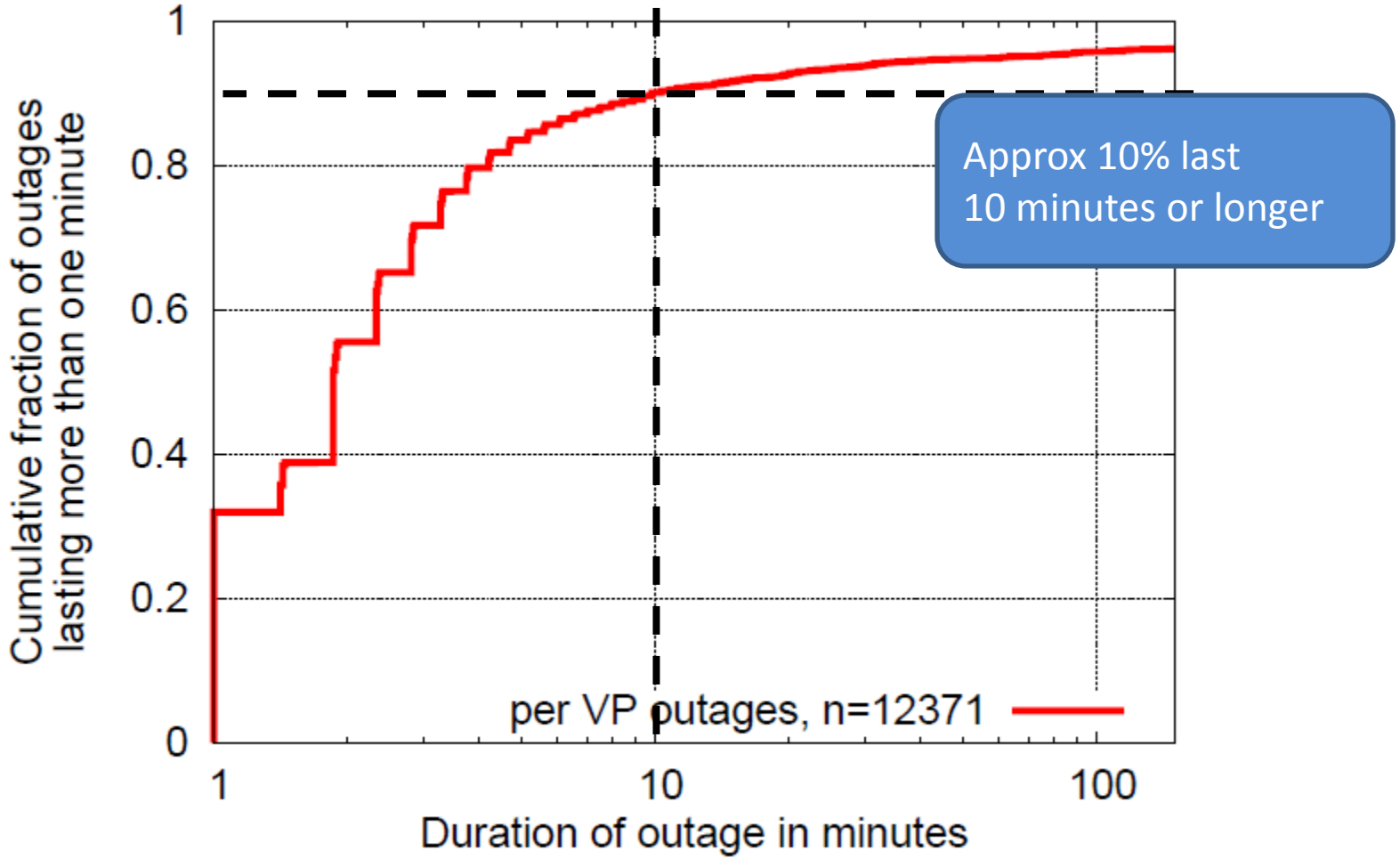
Outages happen.

- They’re expensive, embarrassing and annoying
- They take a long time to fix
  - Alert
  - Troubleshoot
  - Repair
- Lack of good tools for wide-area isolation
- Some examples…

Many outages and most are partial



And can be surprisingly long-lasting



Improving outage response time

- Move from *human* to *computer* timescale
  - Detection
    - Hubble, NEWS
  - **Isolation**
  - Remediation

What we know about outages

- Hubble told us they can be …
  - Frequent and long-lasting
    - confirmed with EC2 study
  - Invisible to BGP feeds
  - Partial
  - Unidirectional
  - In ASes outside of source and destination

But *where* are the outages?

- Can't fix a problem if you don't know where

- State of the art: traceroute
  - Only tells part of the story
  - Even with control of source and destination
  - Especially *without* control of destination

Example confusion (12/16/10)

"It seems traffic attempting to pass through Level3's network in the Washington, DC area is getting lost in the abyss. Here's a trace from VZ residential FIOS to www.level3.com:" – Outages.org list

User 1

```
 1    Wireless_Broadband_Router.home [192.168.3.254]
 2     L100.BLTMMD-VFTTP-40.verizon-gni.net [96.244.79.1]
 3    G10-0-1-440.BLTMMD-LCR-04.verizon-gni.net [130.81.110.158]
 4    so-2-0-0-0.PHIL-BB-RTR2.verizon-gni.net [130.81.28.82]
 5    so-7-1-0-0.RES-BB-RTR2.verizon-gni.net [130.81.19.106]
 6    0.ae2.BR2.IAD8.ALTER.NET [152.63.34.73]
 7     ae7.edge1.washingtondc4.level3.net [4.68.62.137]
 8    vlan80.csw3.Washington1.Level3.net [4.69.149.190]
 9     ae-92-92.ebr2.Washington1.Level3.net [4.69.134.157]
10     *        *        *     Request timed out.
```

User 1: Broken link is in DC

Example confusion (12/16/10)

"It seems traffic attempting to pass through Level3's network in the Washington, DC area is getting lost in the abyss. Here's a trace from VZ residential FIOS to www.level3.com:" – Outages.org list

User 2

```
 1  192.168.1.1 (192.168.1.1)
 2  l100.washdc-vfttp-47.verizon-gni.net (96.255.98.1)
 3  g4-0-1-747.washdc-lcr-07.verizon-gni.net (130.81.59.152)
 4  so-3-0-0-0.lcc1-res-bb-rtr1-re1.verizon-gni.net (130.81.29.0)
 5  0.ae1.br1.iad8.alter.net (152.63.32.141)
 6  ae6.edge1.washingtondc4.level3.net (4.68.62.133)
 7  vlan90.csw4.washington1.level3.net (4.69.149.254)
 8  ae-71-71.ebr1.washington1.level3.net (4.69.134.133)
 9  ae-8-8.ebr1.washington12.level3.net (4.69.143.218)
10  ae-1-100.ebr2.washington12.level3.net (4.69.143.214)
11  ae-6-6.ebr2.chicago2.level3.net (4.69.148.146)
12  ae-1-100.ebr1.chicago2.level3.net (4.69.132.113)
13  ae-3-3.ebr2.denver1.level3.net (4.69.132.61)
14  ge-9-1.hsa1.denver1.level3.net (4.68.107.99)
15  4.68.94.27 (4.68.94.27)
16  4.68.94.33 (4.68.94.33)
17  * * *
```

User 1: Broken link is in DC

User 2: It's in Denver?

Is this even the same problem?
What if it's on the reverse path?
(and paths aren't symmetric)

System for wide-area failure isolation

- Goal: Detect and isolate outages online

- What kind of outages?
  - Long lasting, partial and avoidable
- What kind of isolation?
  - IP link or ASN
- How quickly?
  - Within seconds or small numbers of minutes

Overview

- Detection
  - Target selection
  - Implementation
- Isolation

Types of outages we detect

- Focus on long-lasting, avoidable and high-impact outages
  - Long-lasting: not fixing itself (needs some help)
  - Avoidable: requires path diversity, no stub ASes
  - High impact: outages in PoPs affecting many paths

Experimentation platform

- Monitoring VPs: geographically diverse (~12)

- CloudFront PoP (16)
  - Correlate with app-layer outages
- Popular PoPs wrt # intersecting paths (83)
  - And targets on “other” side of PoPs (185)
- PlanetLab hosts (76)
  - Ground-truth isolation

Detection implementation

- *Partial* outages
  - 2+ sources reach the destination
  - 2+ sources see no ping response 4 consecutive times (8 minutes)
- Reducing noise
  - Destination is consistently reachable from 1+ sources (filter out lossy links)
  - 1+ sources *without* connectivity has seen at least one ping response from destination in the past

Overview

- Detection
- Isolation
  - Approach
  - System design
  - Early results

What we want out of isolation

- Direction (forward or reverse)
- Narrowly determine location (link or ASN)
- Online (allow for immediate action)

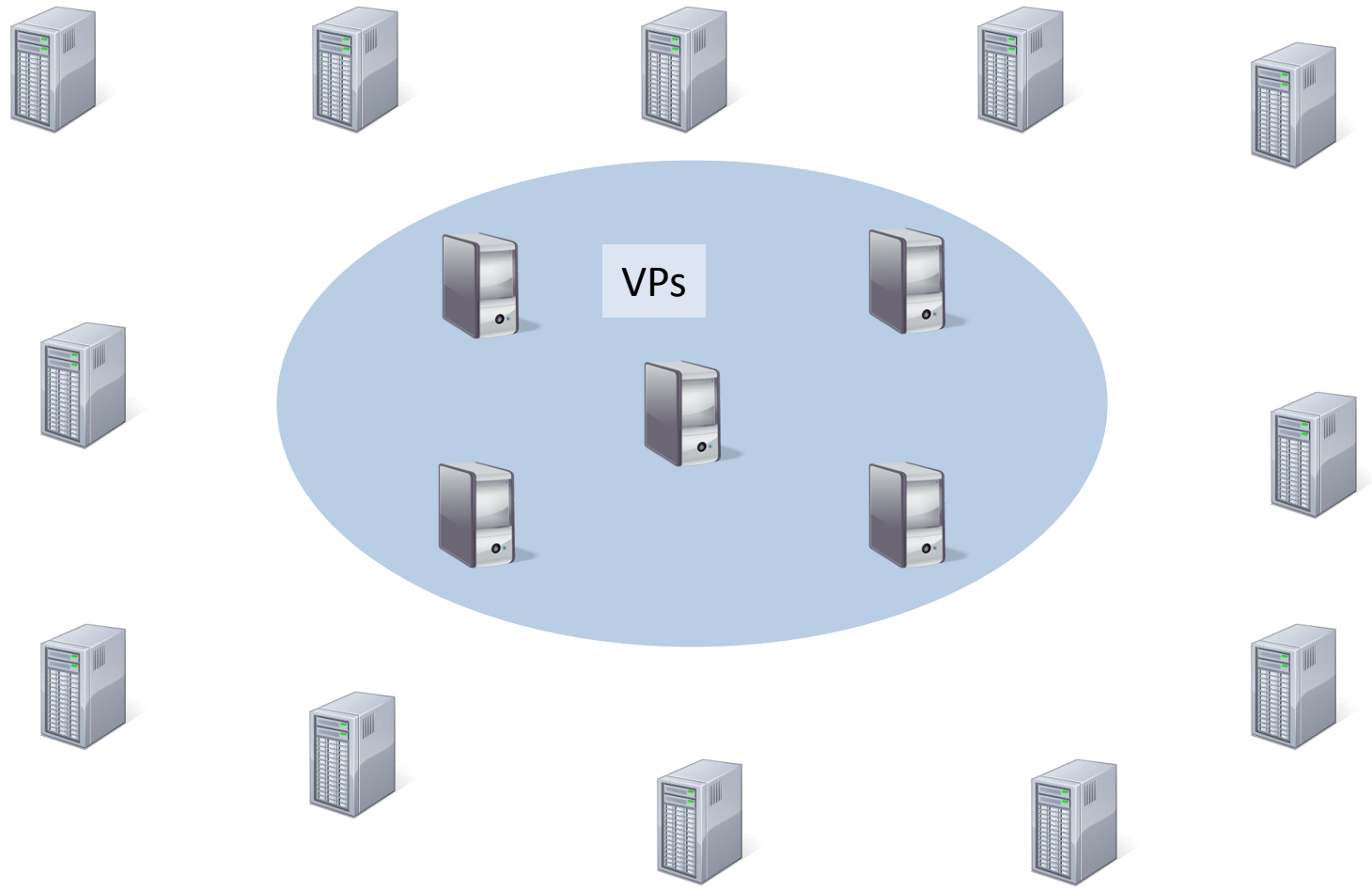
Isolation approach

- When outage between two endpoints occurs:
  - What were the previously working *paths*?
  - What are the *current* working *hops*?
  - Combine to infer likely problem links/networks

Enabling isolation during outages

- Atlas of path information to “seed” isolation
  - Rapidly refreshed, historical path information
  - Forward & reverse traceroute (intermediate hops)
  - Historical alternative paths
- Measurements during outages
  - Forward hops: spoofed forward traceroute
  - Pings to historical hops (fwd and rev)
  - Reverse hops: reverse traceroute

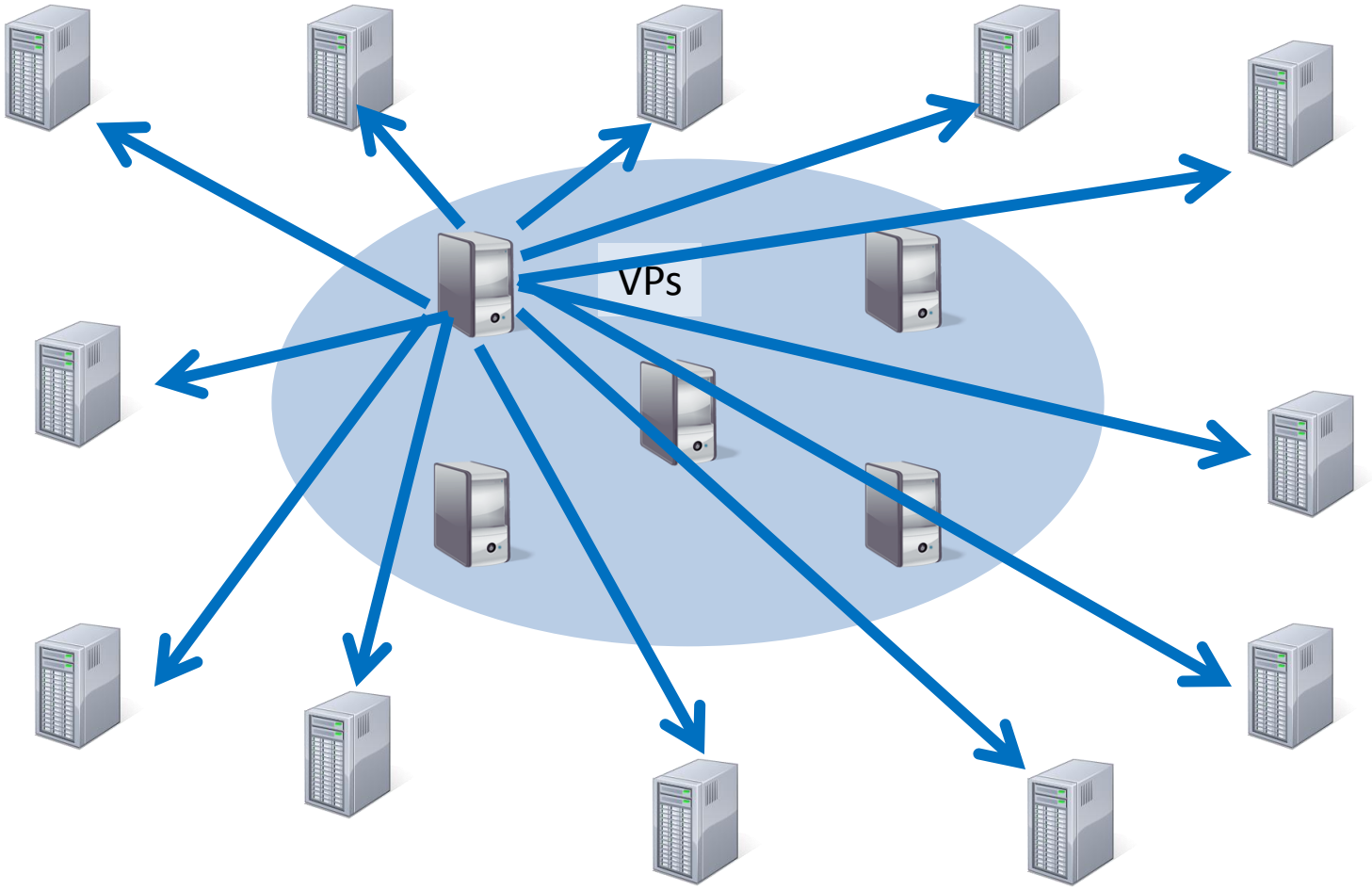
Isolation system



Traceroute atlas

- Forward traceroutes to all targets
  - Updated every 5 minutes

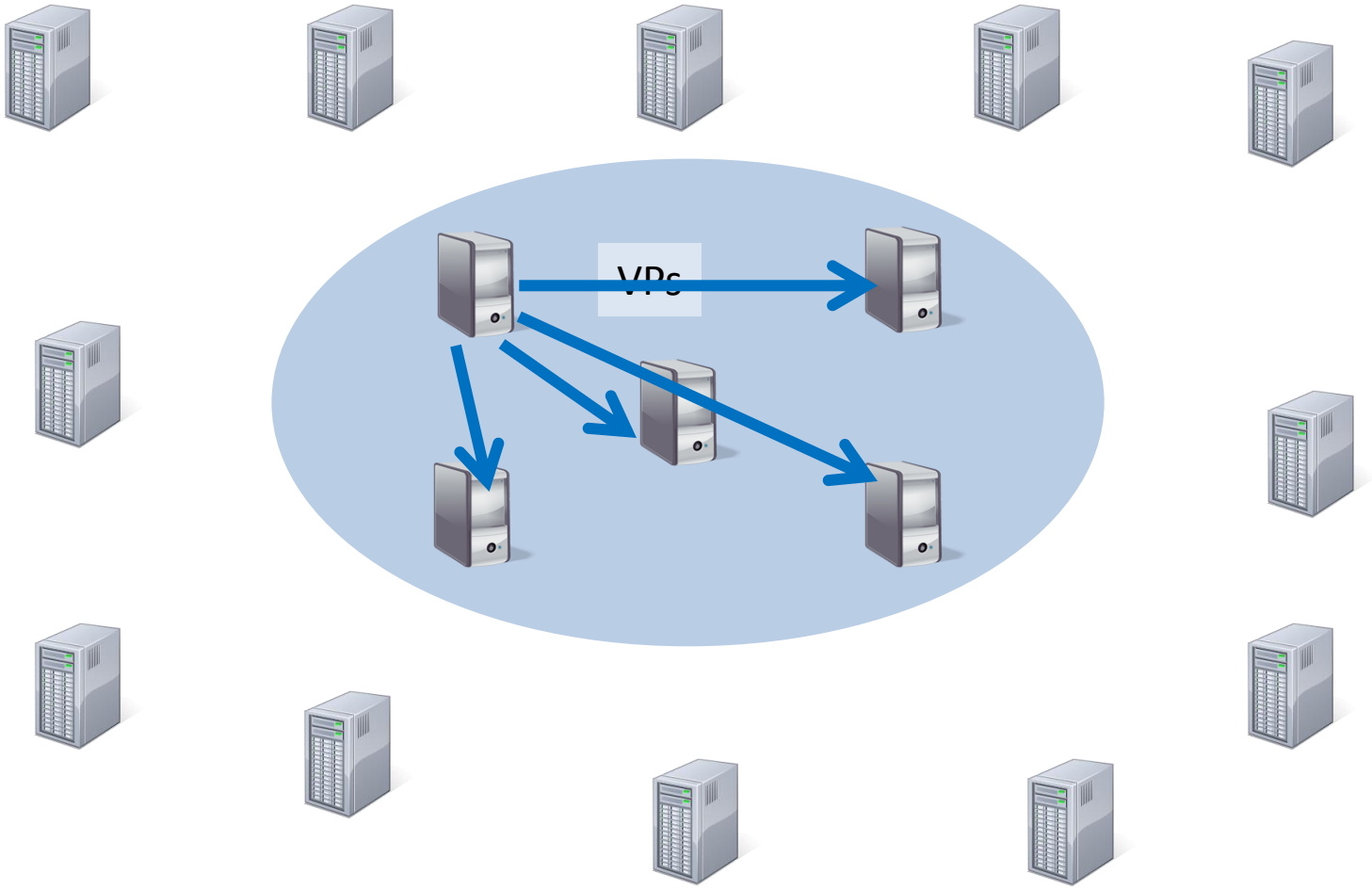
Each host traceroutes each target



Traceroute atlas

- Forward traceroutes to all targets
  - Updated every 5 minutes
- Traceroutes toward measurement sources
  - Rounds start every 5 minutes
  - Maximum staleness: 15 minutes
- *Opportunities for optimization*
  - Great motivation for work on path-measurement efficiency

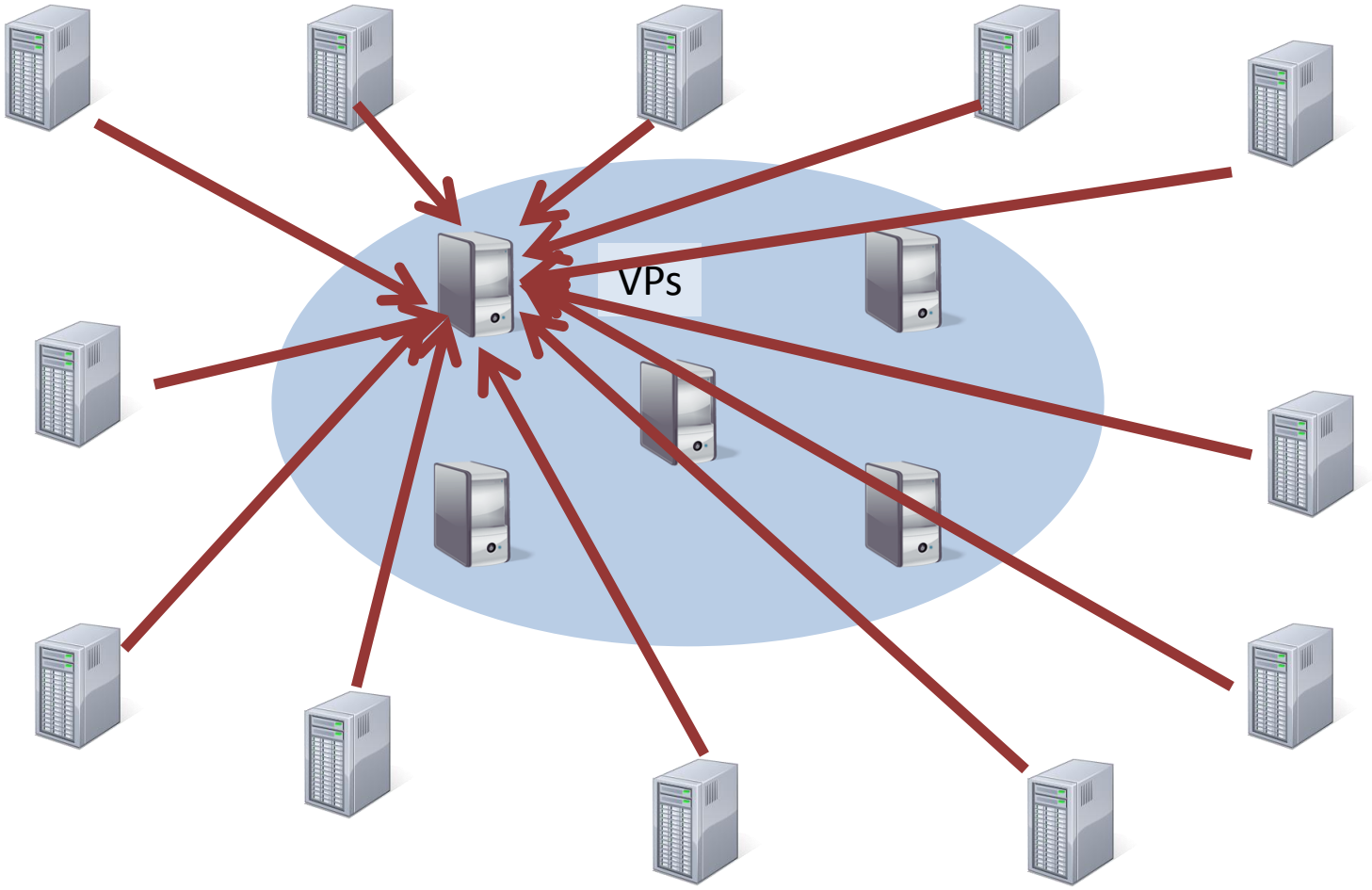
All VPs traceroute each other



Traceroute atlas

- Forward traceroutes to all targets
  - Updated every 5 minutes
- Traceroutes toward measurement sources
  - Rounds start every 5 minutes
  - Maximum staleness: 15 minutes
- Reverse path measurements
  - Use reverse traceroute technique…

Each VP measures reverse paths



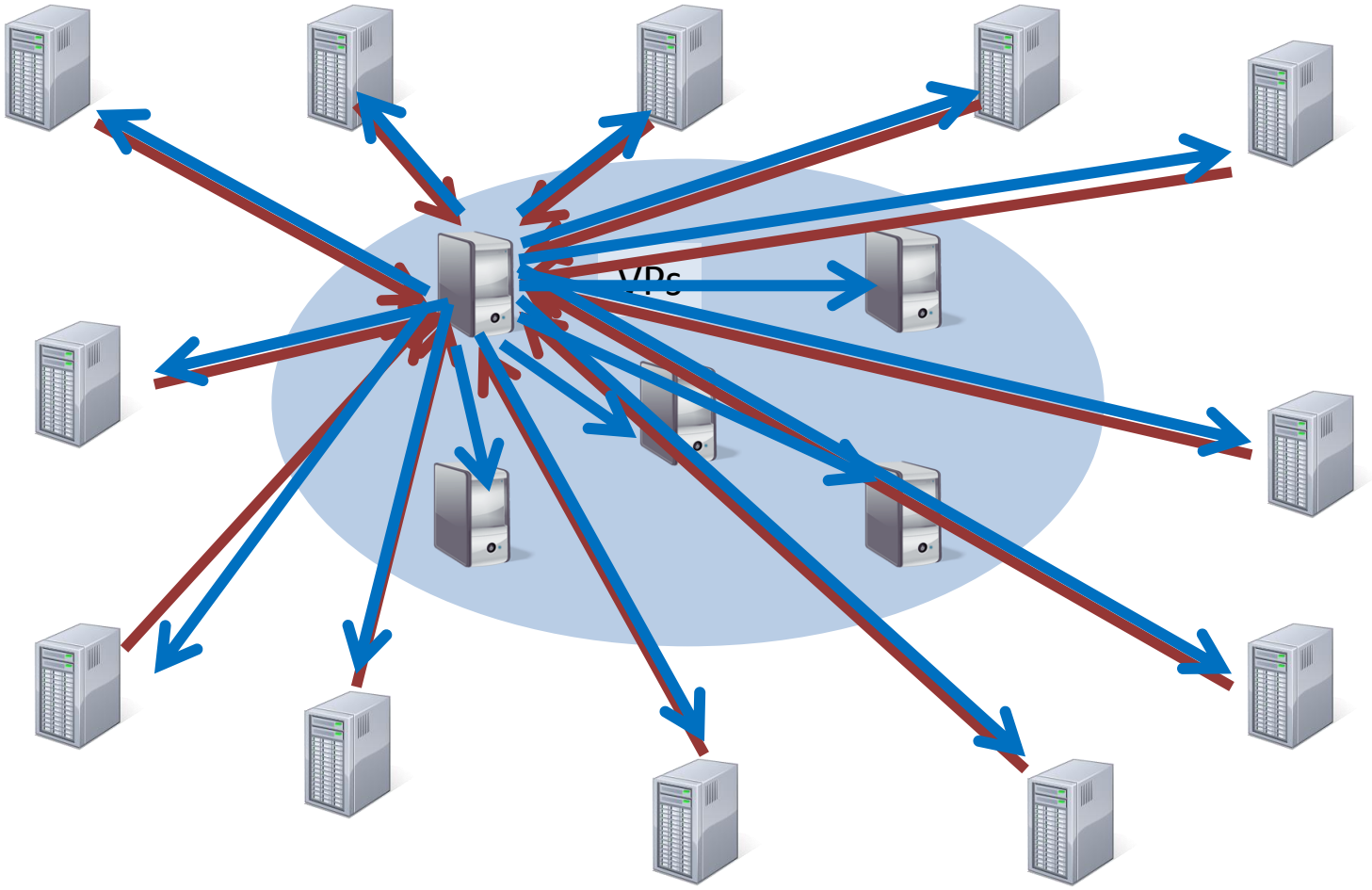
Reverse traceroutes

- Reverse path info generally requires
  - IP options support along the path
  - Limited spoofing
  - A lot of trial and error
- Comparison
  - Fwd traceroute
    - 10s of measurements
    - Usually done in a few seconds (less than a minute at most)
  - Reverse traceroute (unoptimized)
    - ~40 measurements
    - 100s of seconds (median: 851 seconds when done in bulk)

Scaling reverse traceroute

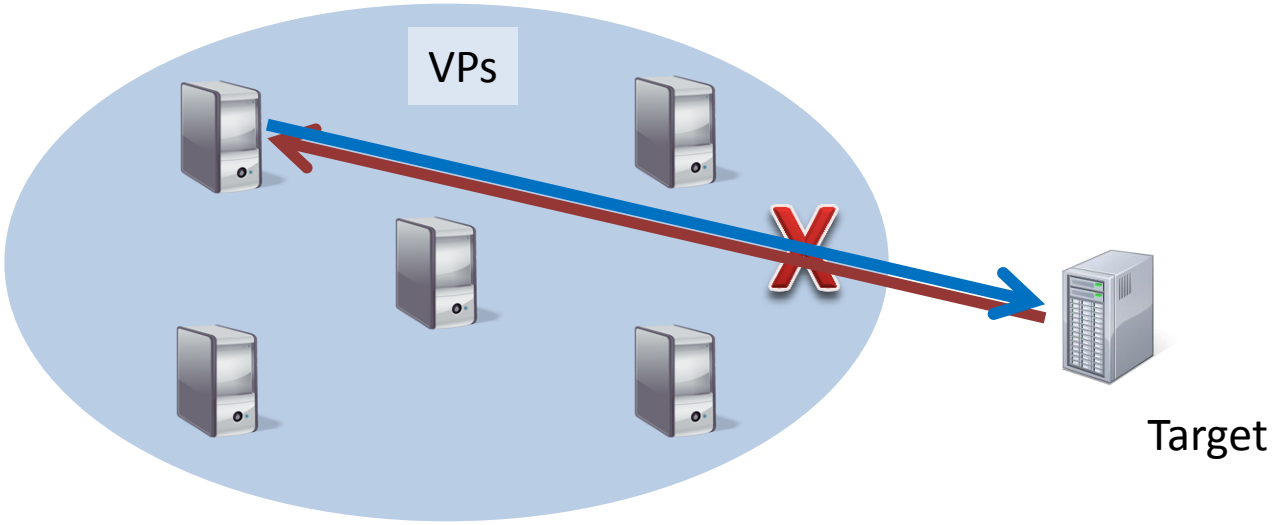
- Feedback loop for retaining path knowledge
  - Path-segment caching layer
  - Batching/staging measurements
  - Clearing bottlenecks
    - Determining when to spoof
    - Identifying successful spoofers
    - Avoiding probes to unresponsive routers
- Results (amortized averages)
  - Without optimizations: 53 seconds per revtr
  - With optimizations: 1-2 seconds (15 meas per revtr)

Atlas



Destinations

Measurements during outages



Spoofed forward traceroutes

- Problem: traceroute can't measure working forward path during reverse path outage
  - Need tool that avoids reverse path

- SFT: TTL-limited probes spoofed as another VP
  - Select VPs that are likely to be reachable
  - Yields forward hops during reverse-path outage
  - Can provide more information than traceroute, even during forward/bidirectional failures

Simple (real) example

plgmu4.ite.gmu.edu to pl2.bit.uoit.ca

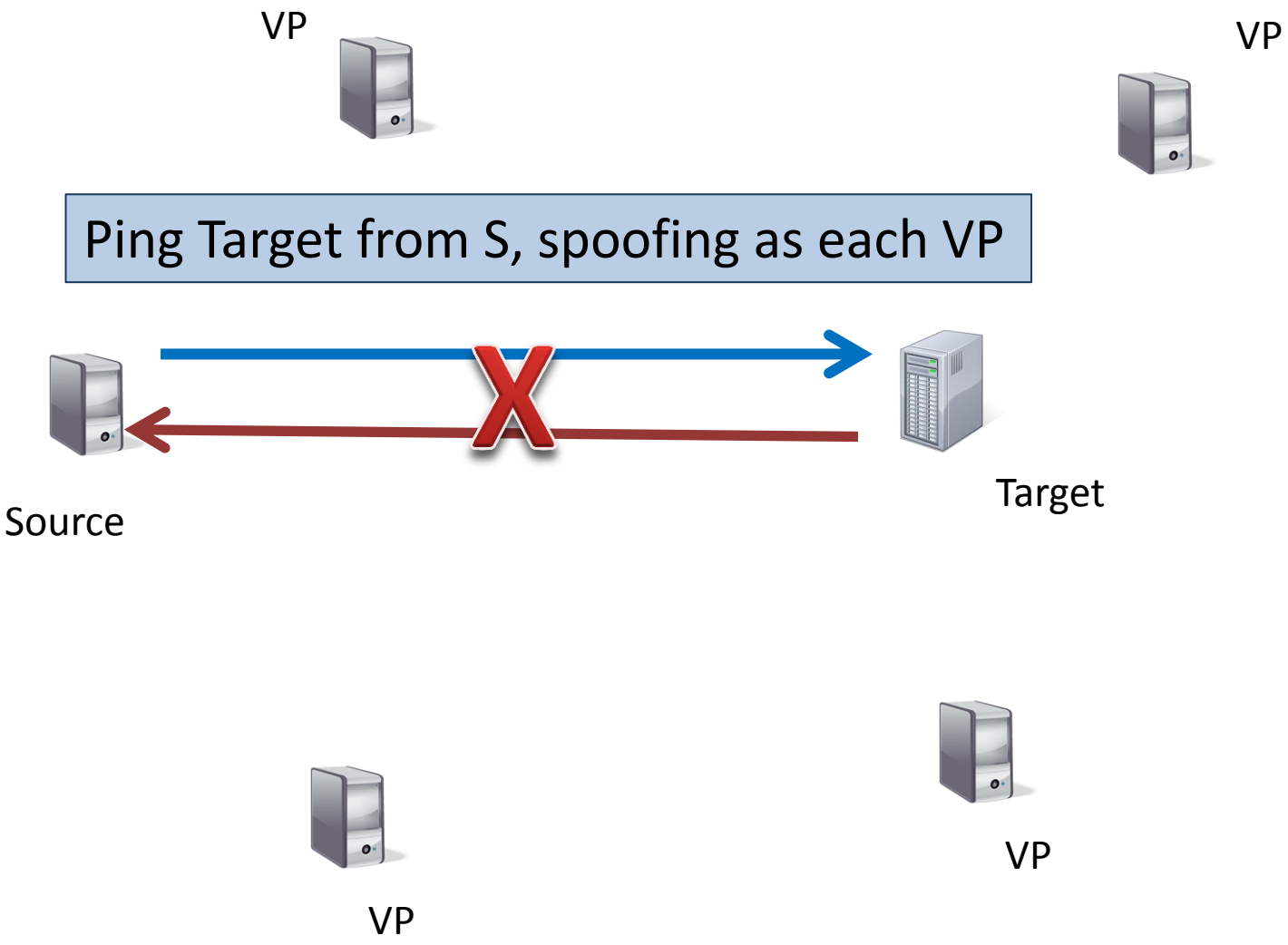
Normal traceroute

1. 199.26.254.65
2. 10.255.255.250
3. 192.70.138.121
4. 192.70.138.110
5. 216.24.186.86
6. 216.24.186.84
7. 216.24.184.46
8. * * *
9. * * *
10. * * *
11. * * *
12. * * *

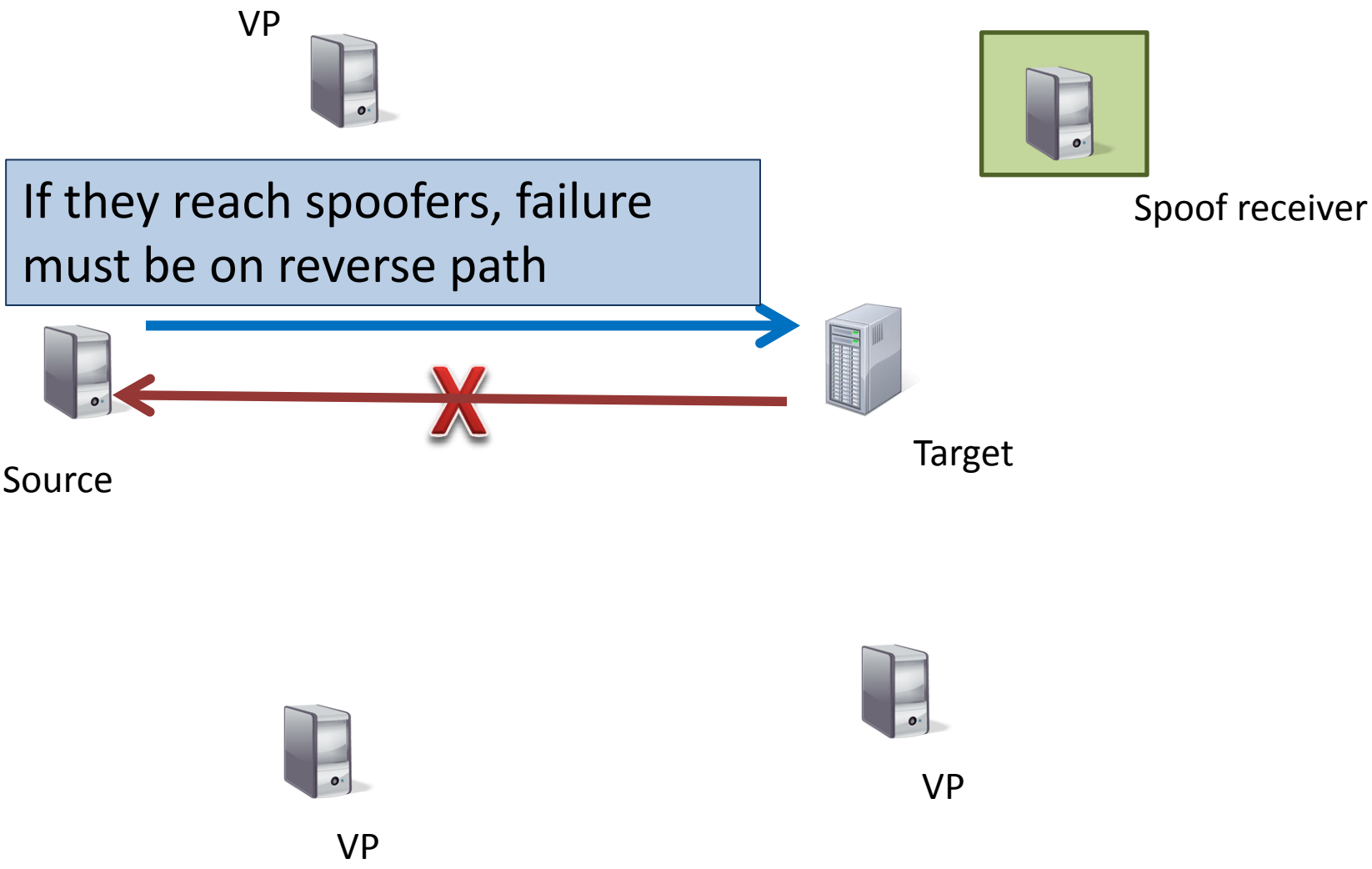
Spoofed traceroute

1. 199.26.254.65
2. 10.255.255.250
3. 192.70.138.121
4. 192.70.138.110
5. 216.24.186.86
6. 216.24.186.84
7. 216.24.184.46
8. 205.189.32.229
9. 66.97.16.57
10. 66.97.23.238
11. pl2.bit.uoit.ca (205.211.183.4)

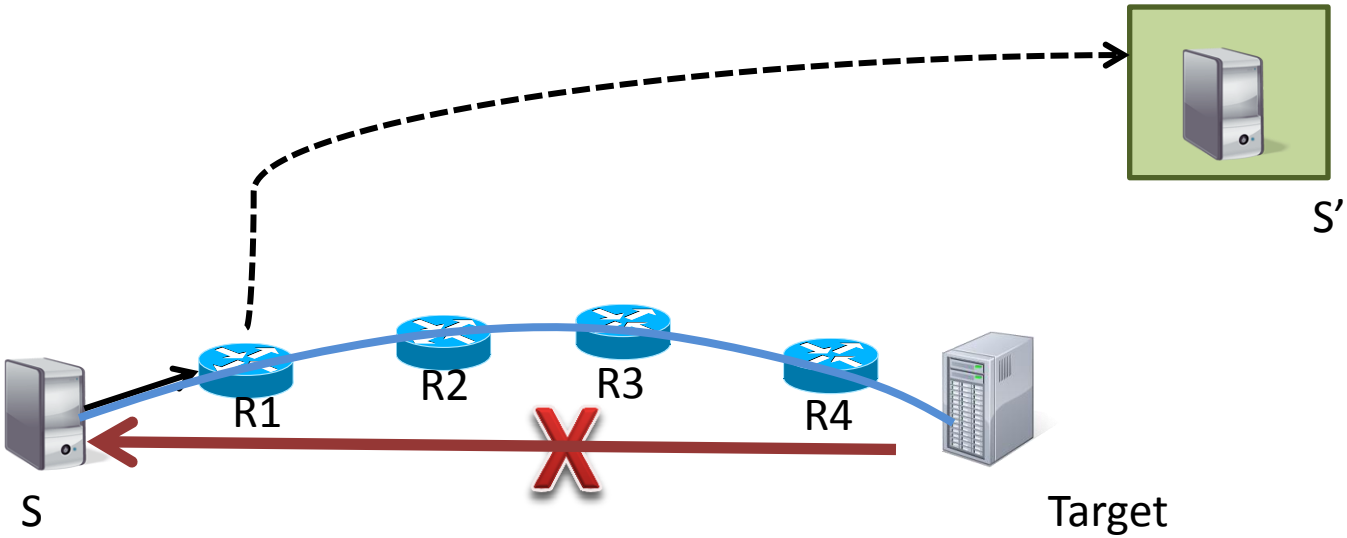
SFT during a failure



SFT during a failure

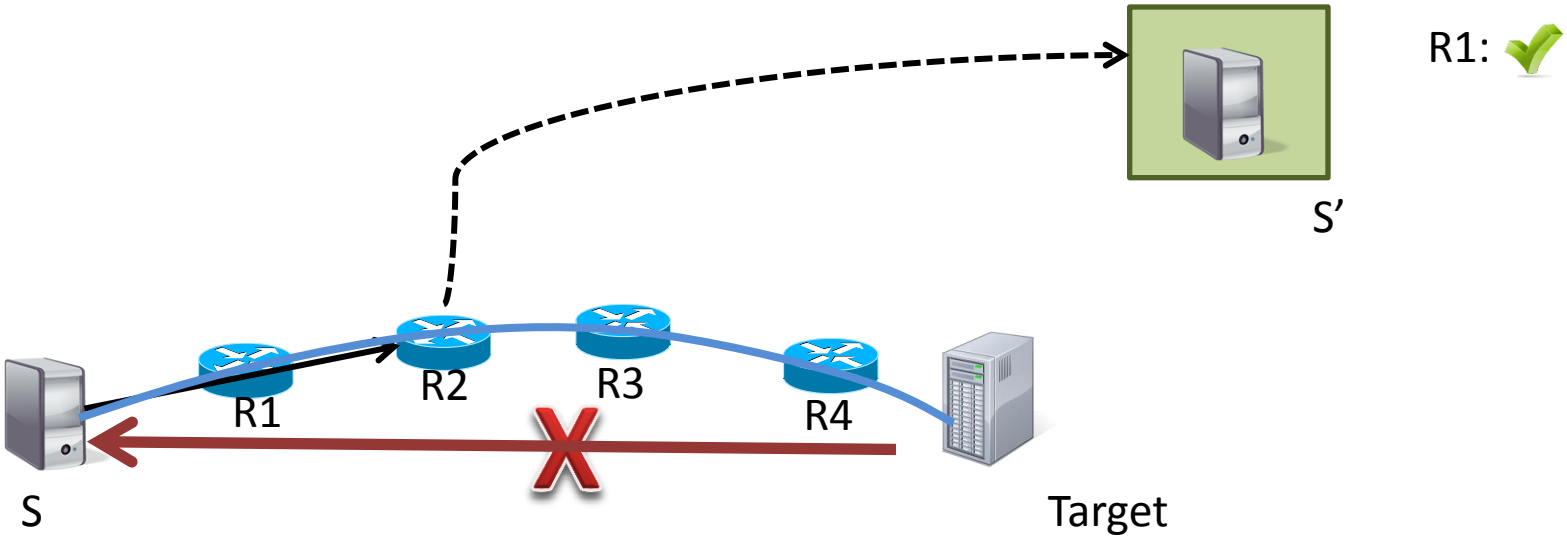


SFT during a failure



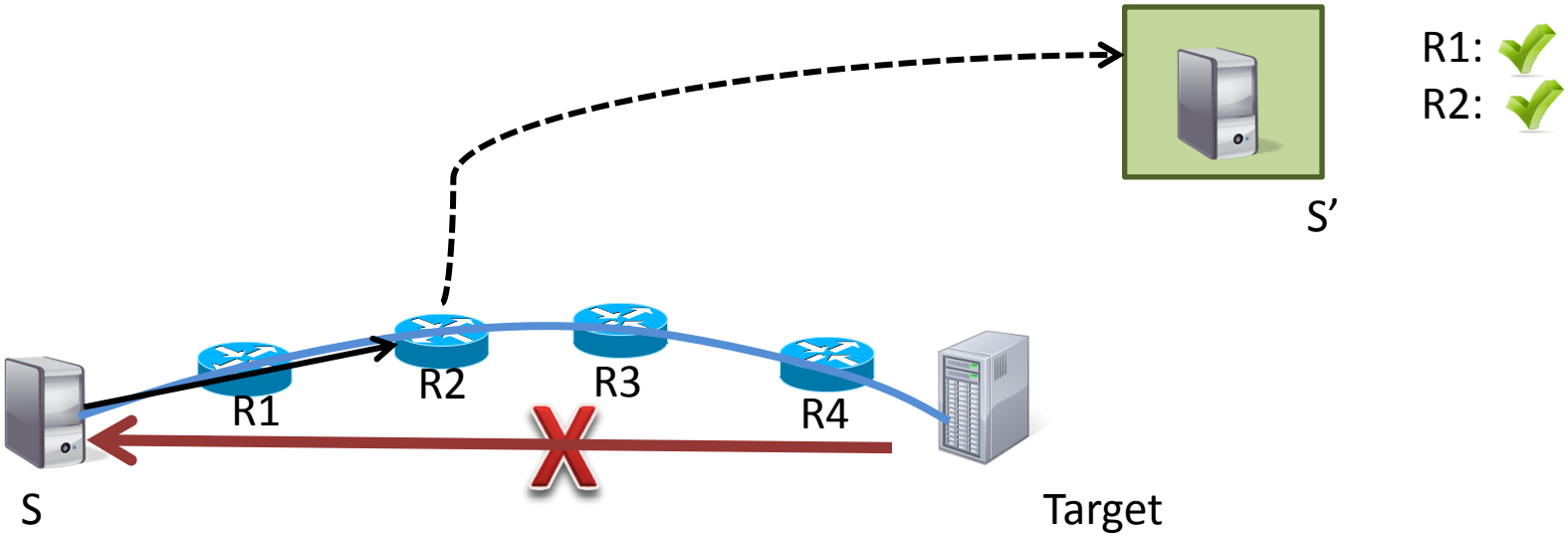
Ping T from S, spoofing as S' and using TTL=1

SFT during a failure



Ping T from S, spoofing as S' and using TTL=2

Test each reverse subpath

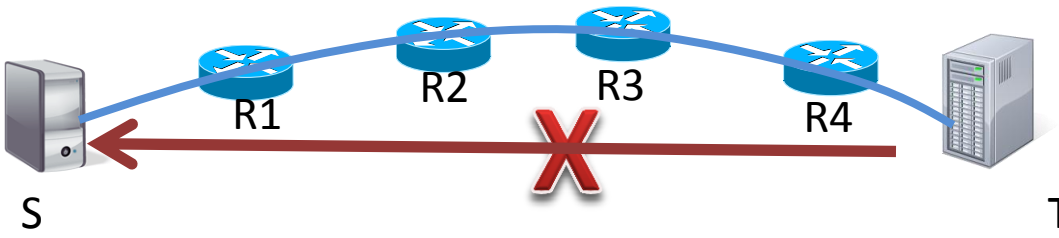


Now we know the current forward path

Test each reverse subpath

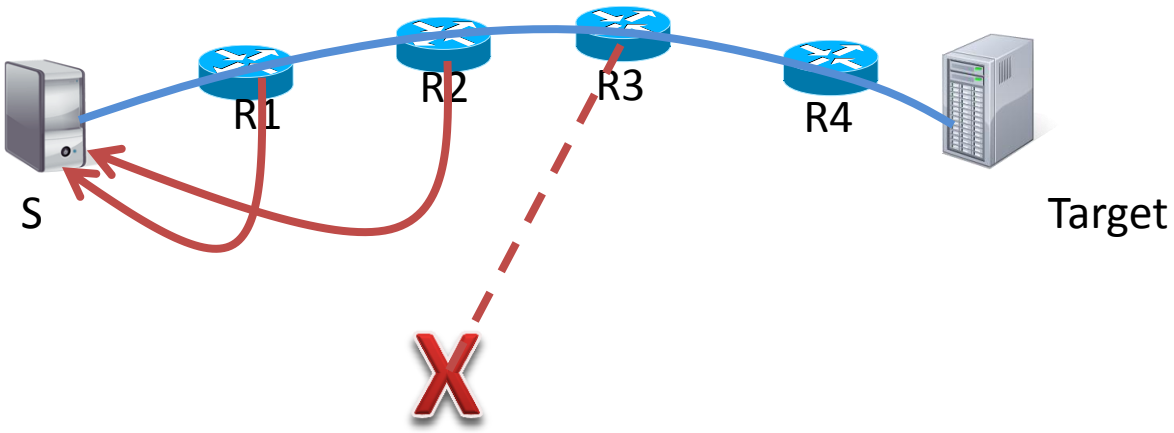
S’

R1: ✓
R2: ✓
R3: ✓
R4: ✓



Now we know the current forward path

Isolating on reverse path



Aha! It's the reverse path from R3

Putting it all together

- Find spoofing VPs that reach T
- Determine working direction (if any)
  - Forward: have S spoof toward T as VP
  - Reverse: VP spoof toward T as S
- Failure cases
  - Forward-only : spoof traceroute
  - Reverse-only: reverse traceroute to each fwd hop
  - Bi-directional: spoof traceroute

Early results

- Location (~2500 total)
  - PL/Mlab: 1241
  - Top 100: 1220
  - CloudFront: 38
- Duration: Average is 453 seconds
- Directionality
  - Forward: 860
  - Reverse: 130
  - Bi-directional: 439
  - The rest were indeterminate (different path, fixed by time of isolation, …)

Evaluation plan

- Coverage
  - How much of the network can we monitor?
  - How precise is isolation?
- Effectiveness
  - When affecting CDN, try application layer
  - Corroborate with NANOG
  - Post to outages.org

Summary

- System for wide-are failure isolation
  - Detection at fine granularity
  - Algorithm for isolation
    - Historical, rapidly refreshed path atlas
    - Spoofed probing to measure during outage

- Ongoing work
  - Refining isolation
  - Identifying opportunities for automatic remediation