Reverse Traceroute

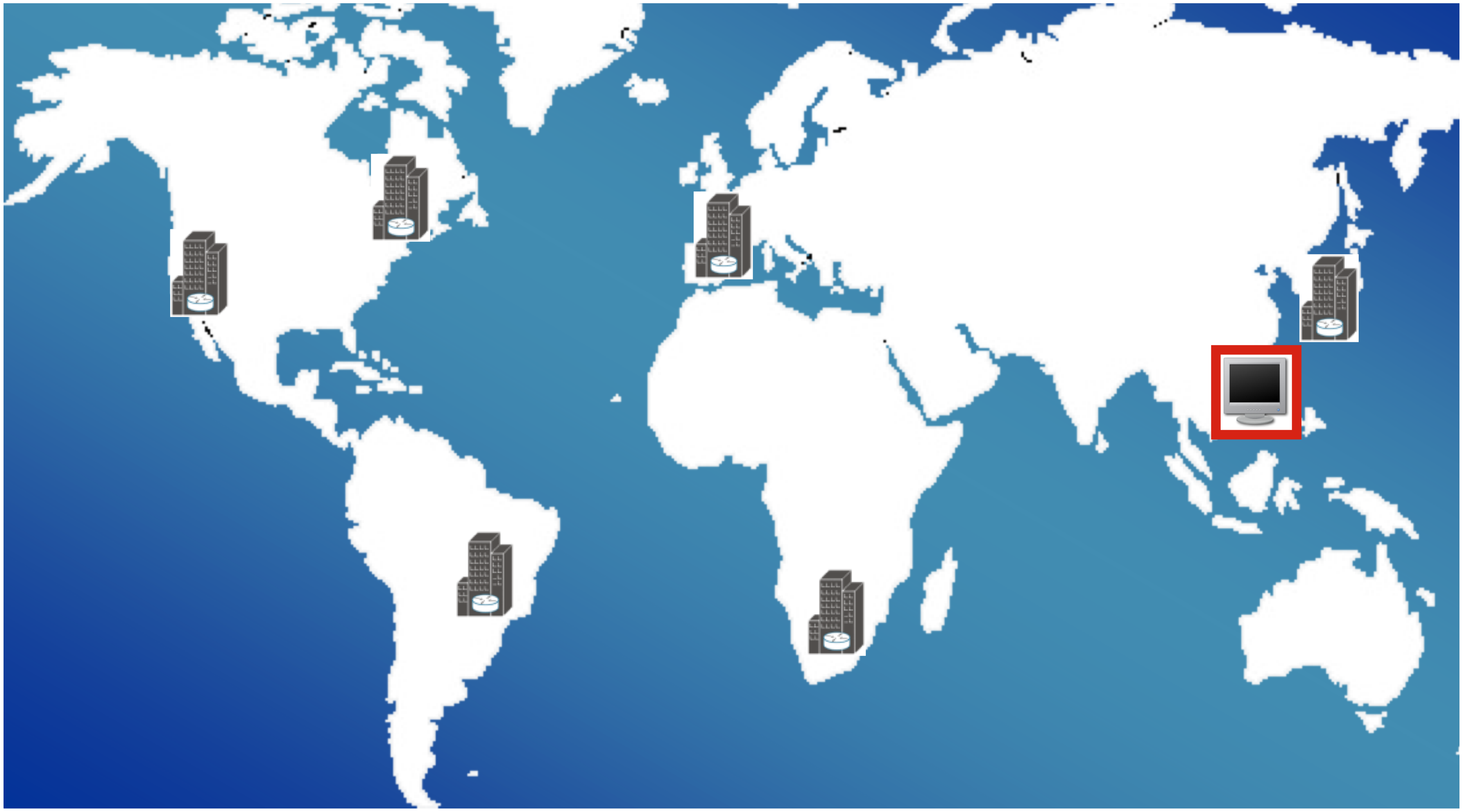
Ethan Katz-Bassett, Harsha V. Madhyastha,
Vijay K. Adhikari, Colin Scott,
Justine Sherry, Peter van Wesep,
Arvind Krishnamurthy, Thomas Anderson

NSDI, April 2010

This work partially supported by Cisco, Google, NSF

Data Centers Need Reverse Paths

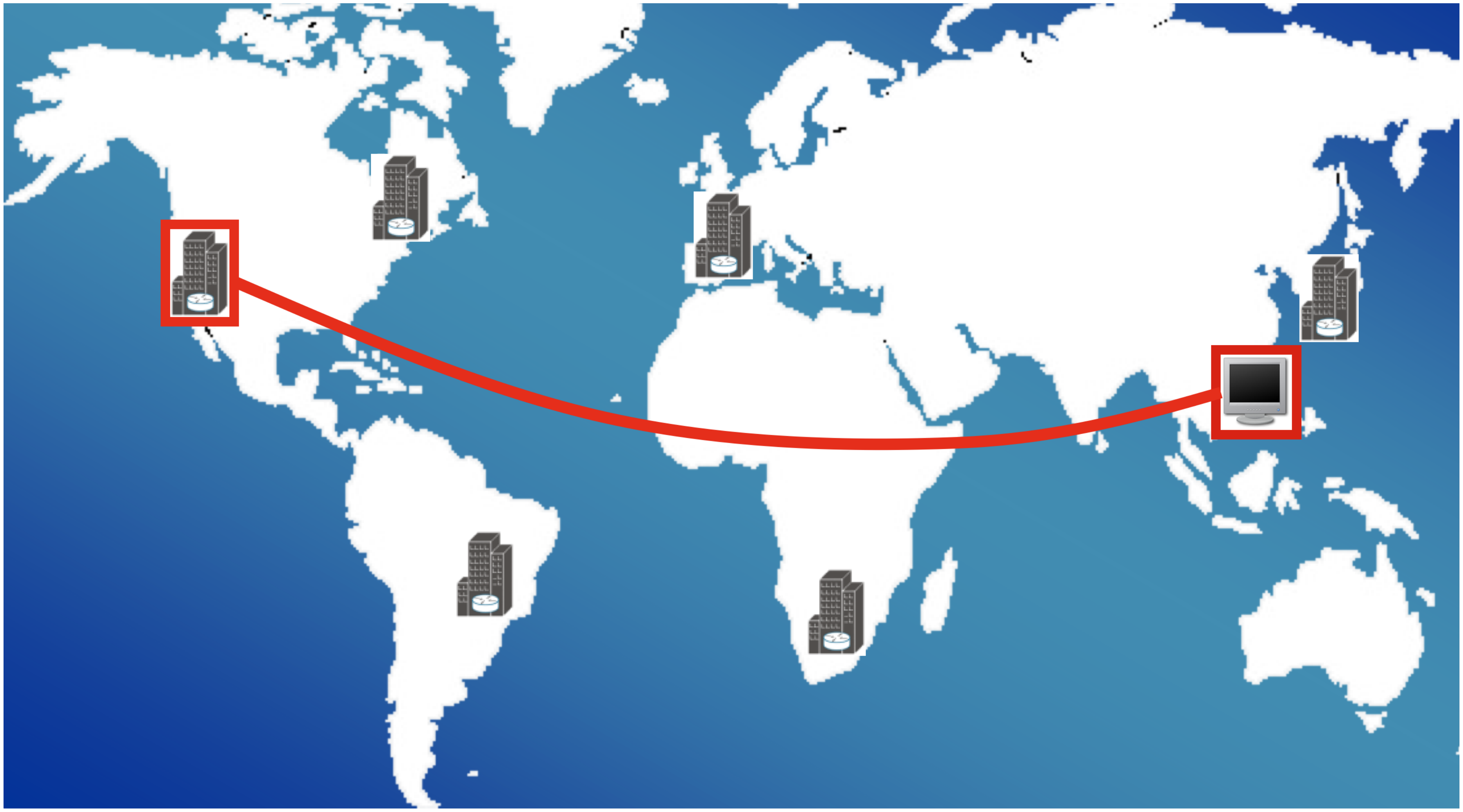
Actual problem encountered by Google [IMC 2009]



Clients in Taiwan experiencing 500ms network latency

Data Centers Need Reverse Paths

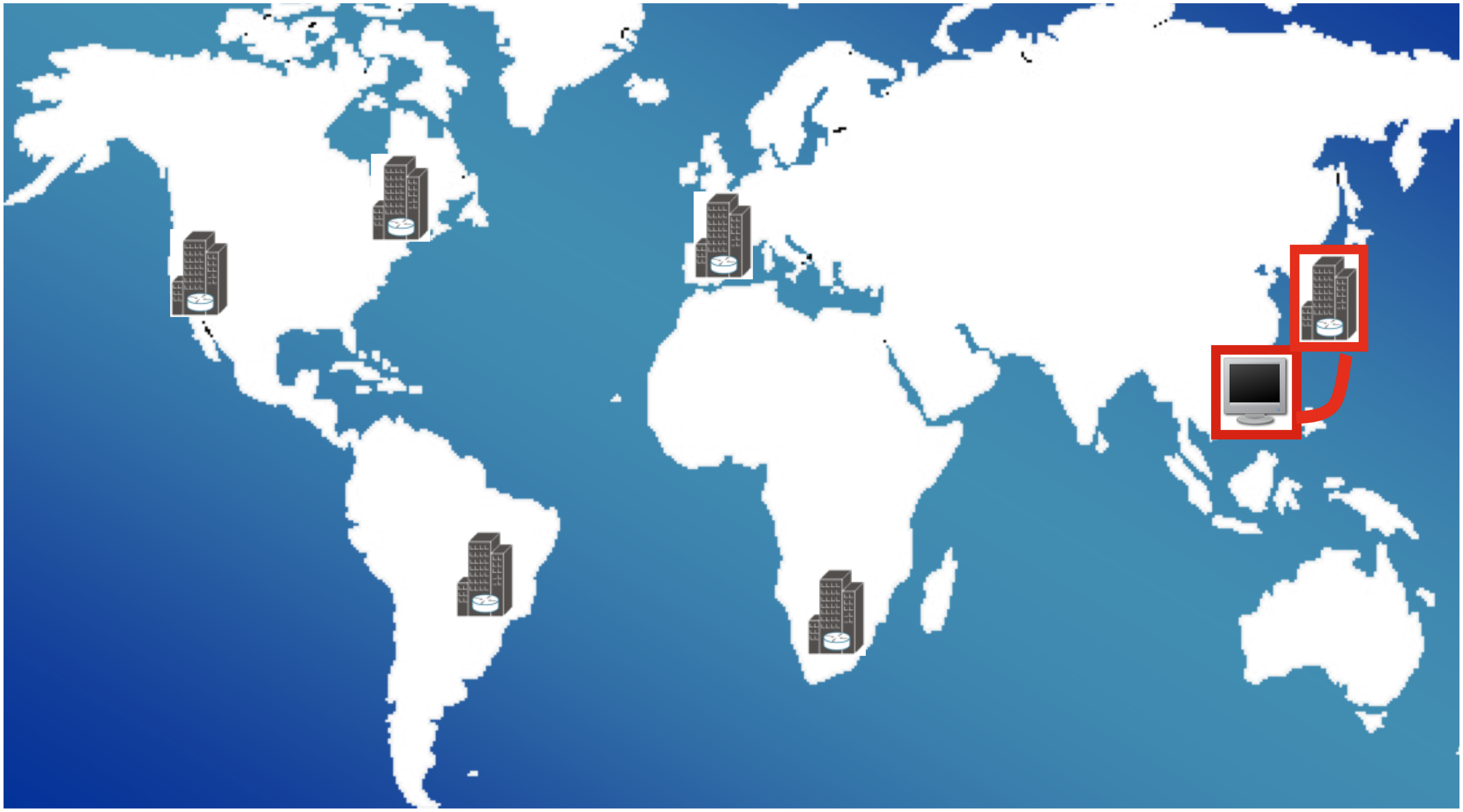
Is client served by distant data center?



Clients in Taiwan experiencing 500ms network latency

Data Centers Need Reverse Paths

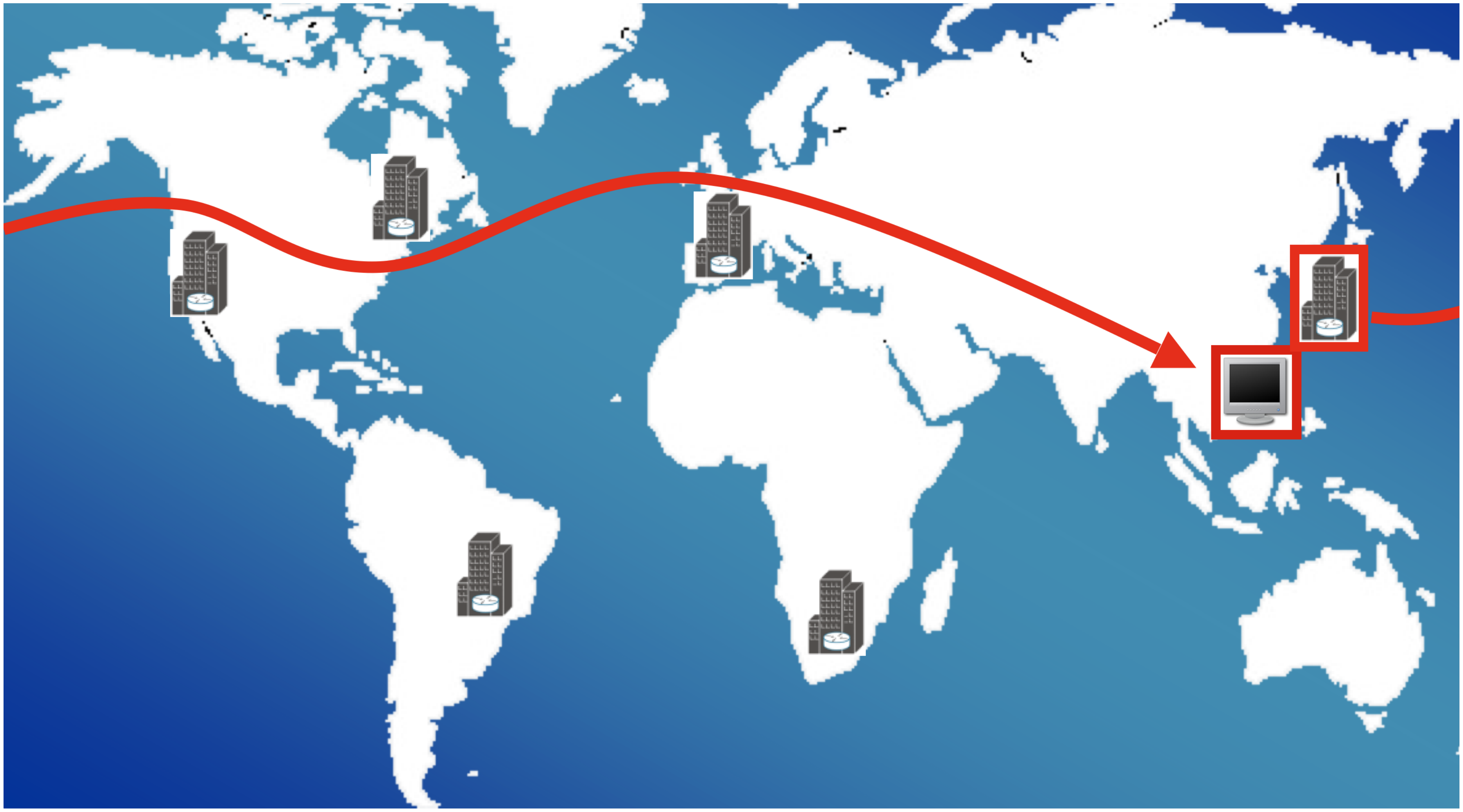
Is client served by distant data center? Check logs: **No**



Clients in Taiwan experiencing 500ms network latency

Data Centers Need Reverse Paths

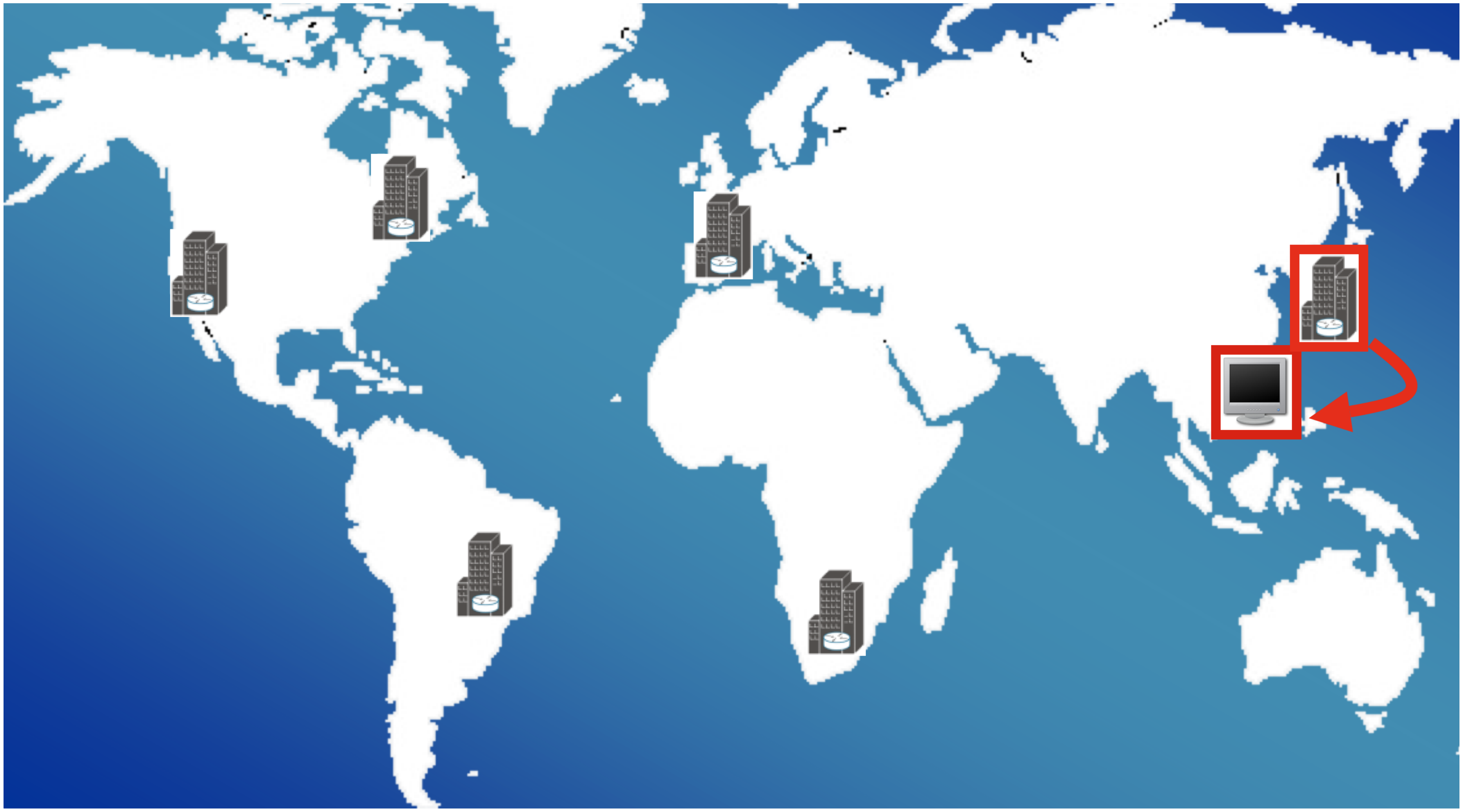
Is path from data center to client indirect?



Clients in Taiwan experiencing 500ms network latency

Data Centers Need Reverse Paths

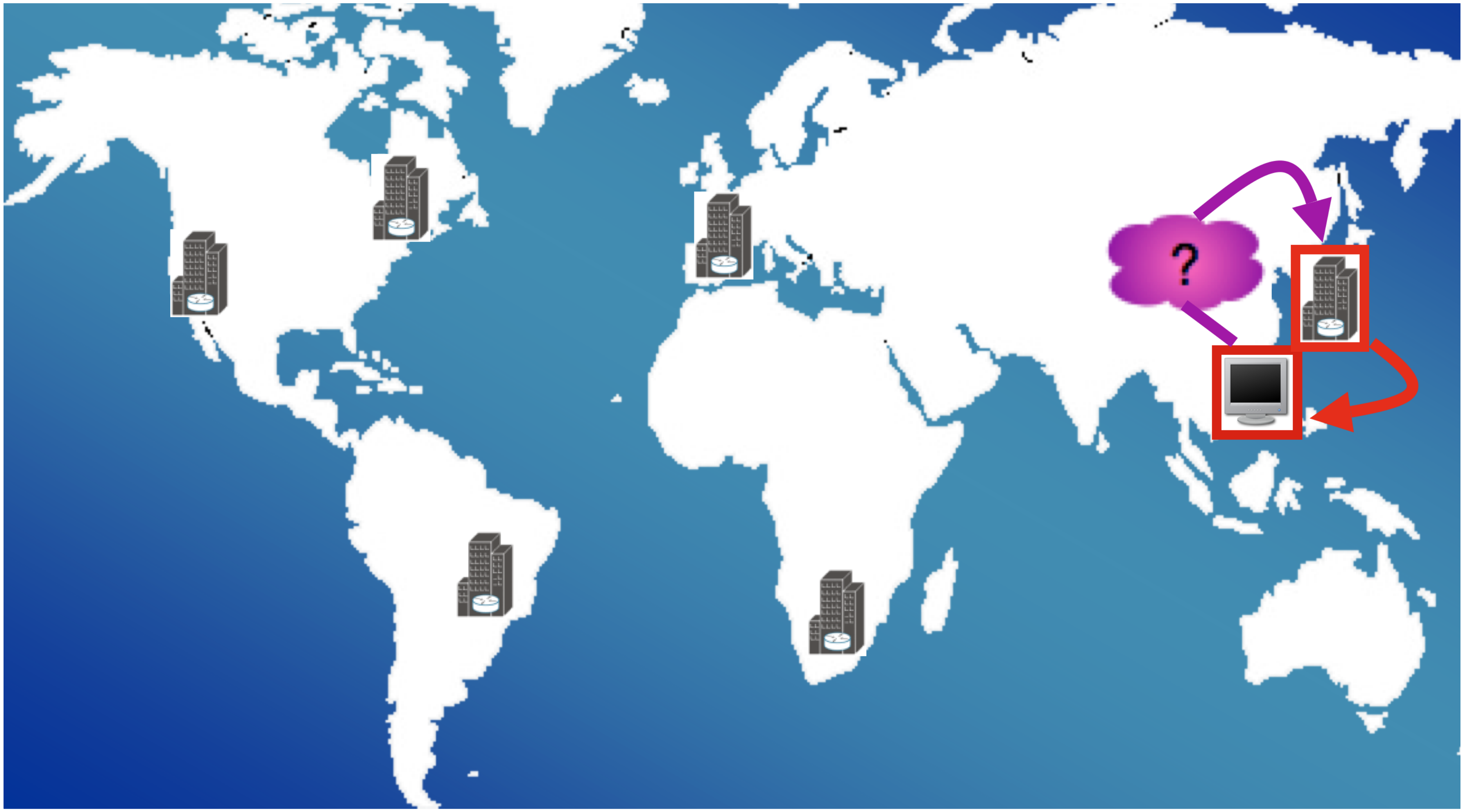
Is path from data center to client indirect? Traceroute: **No**



Clients in Taiwan experiencing 500ms network latency

Data Centers Need Reverse Paths

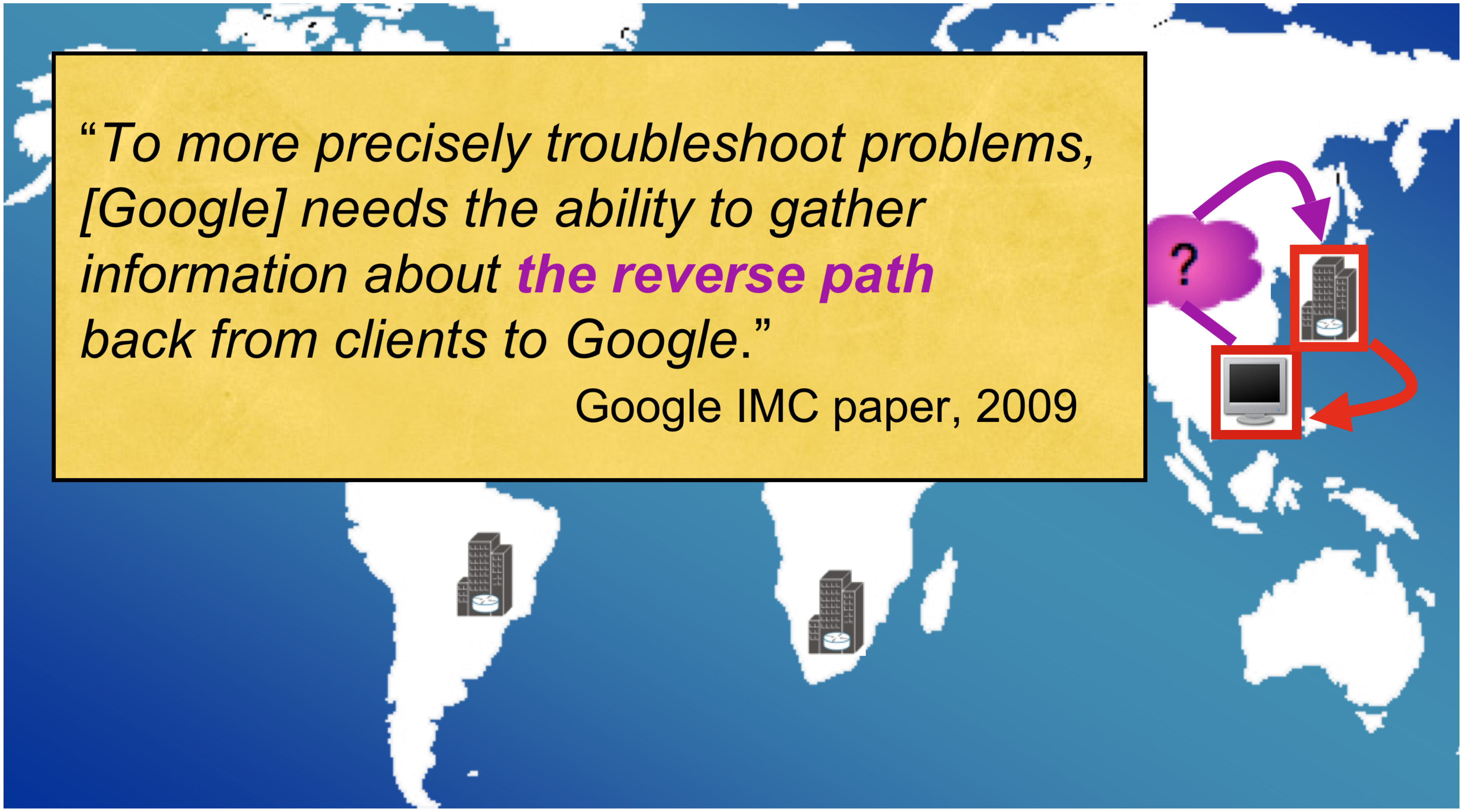
*Is **reverse path** from client back to data center indirect?*



Clients in Taiwan experiencing 500ms network latency

Data Centers Need Reverse Paths

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



*"To more precisely troubleshoot problems, [Google] needs the ability to gather information about **the reverse path** back from clients to Google."*

Google IMC paper, 2009

Clients in Taiwan experiencing 500ms network latency

Researchers Need Reverse Paths, Too

The inability to measure reverse paths was the biggest limitation of my previous systems:

- Geolocation constraints too loose [IMC '06]
- Hubble can't locate reverse path outages [NSDI '08]
- iPlane predictions inaccurate [NSDI '09]

Other systems use sophisticated measurements but are forced to assume symmetric paths:

- Netdiff compares ISP performance [NSDI '08]
- iSpy detects prefix hijacking [SIGCOMM '08]
- Eriksson et al. infer topology [SIGCOMM '08]

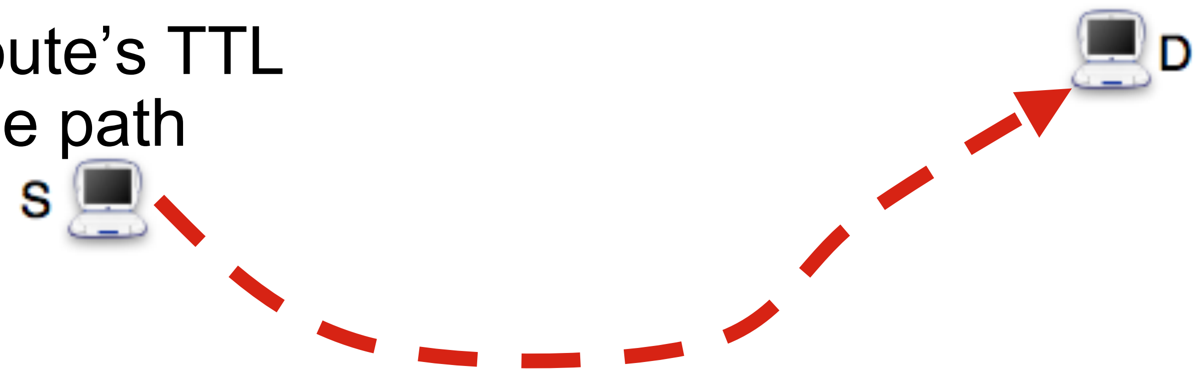
Everyone Needs Reverse Paths

"The number one go-to tool is traceroute.
Asymmetric paths are the number one plague.
The reverse path itself is completely invisible."

NANOG Network operators troubleshooting tutorial, 2009.

Goal: **Reverse traceroute**,
without control of destination and
deployable today without new support

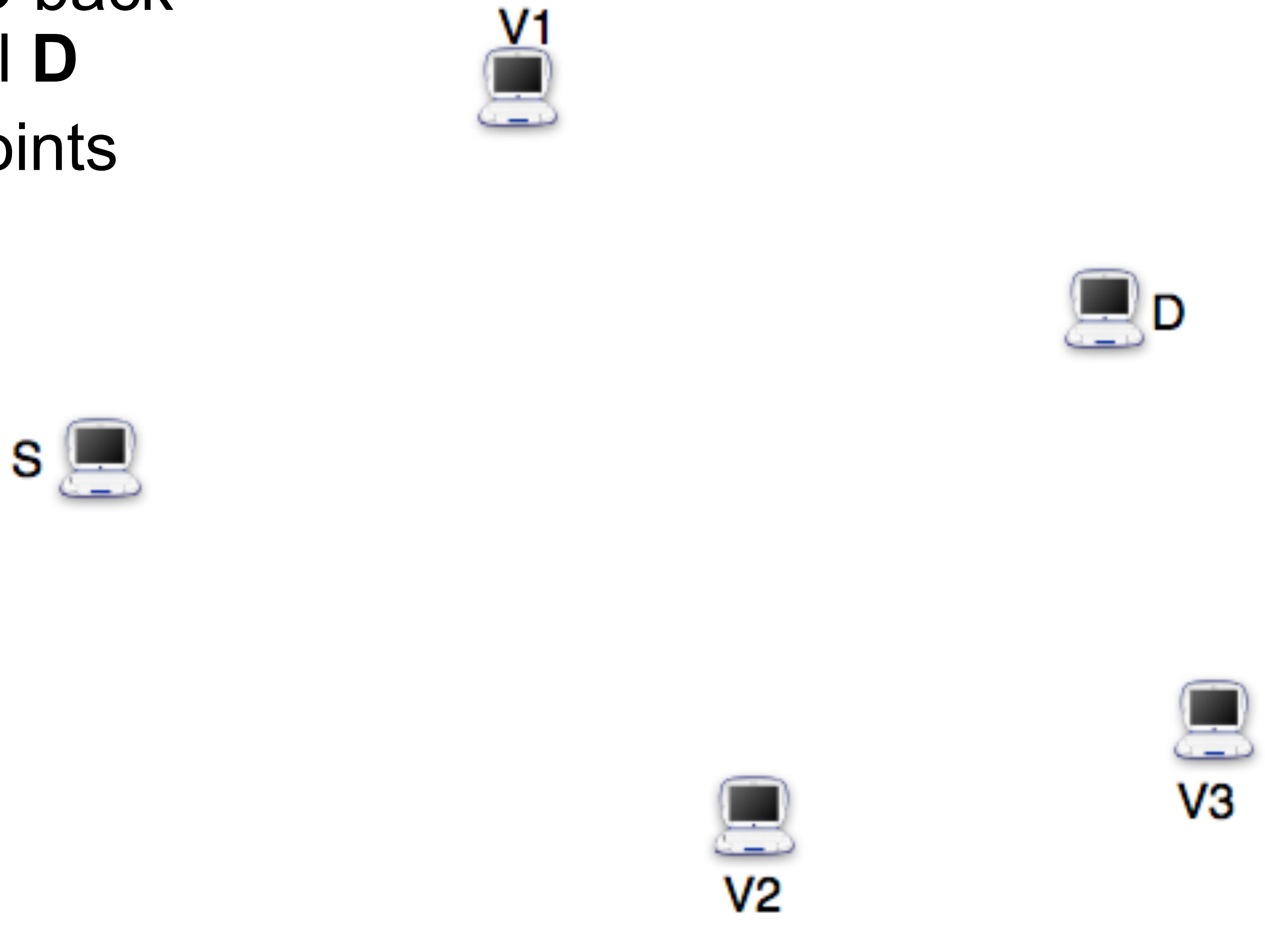
- Want path from **D** back to **S**, don’t control **D**
- Traceroute gives **S** to **D**, but likely asymmetric
- Can’t use traceroute’s TTL limiting on reverse path



KEY IDEA

- Technique does not require control of destination

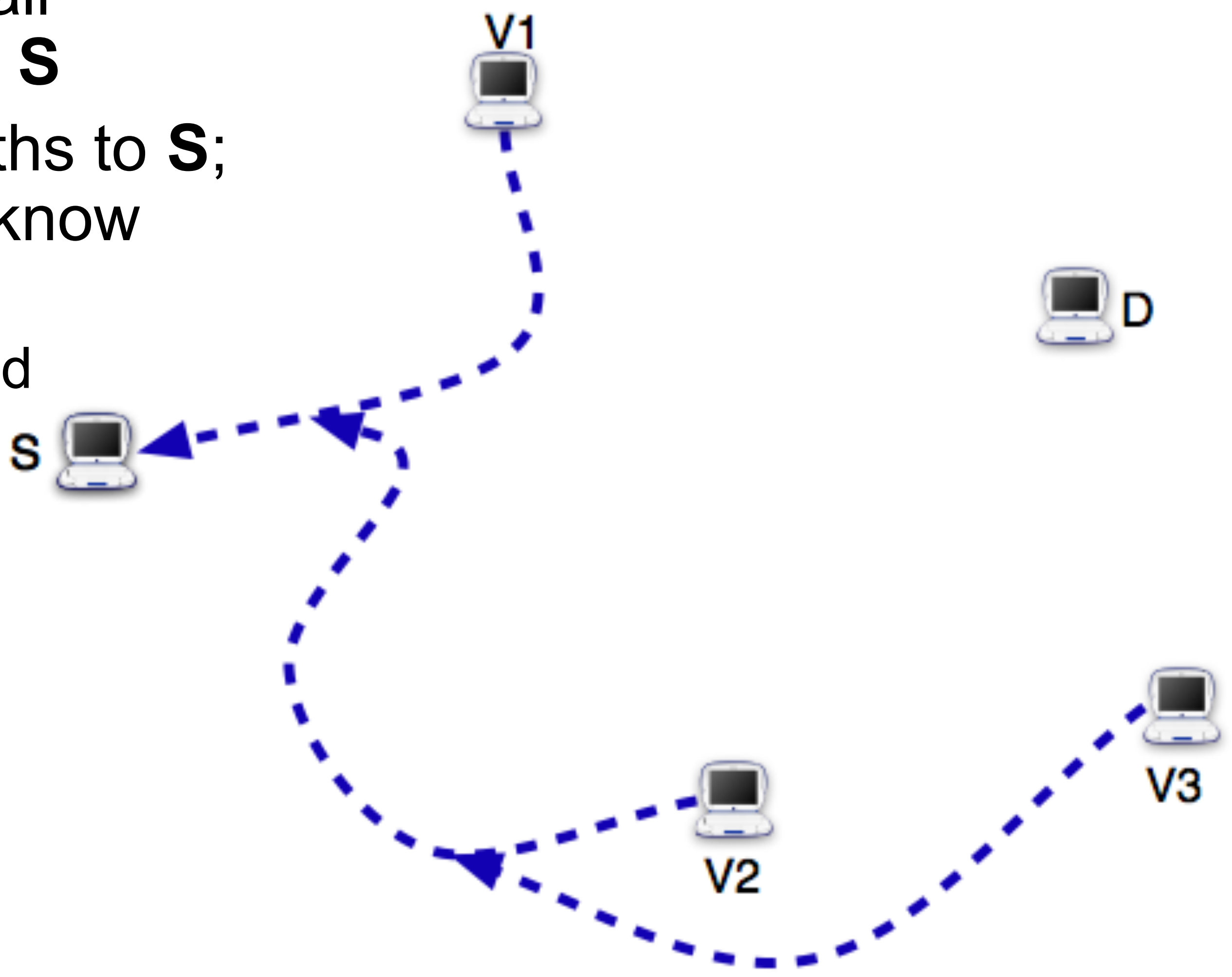
- Want path from **D** back to **S**, don't control **D**
- Set of vantage points



KEY IDEA

- Multiple VPs combine for view unattainable from any one

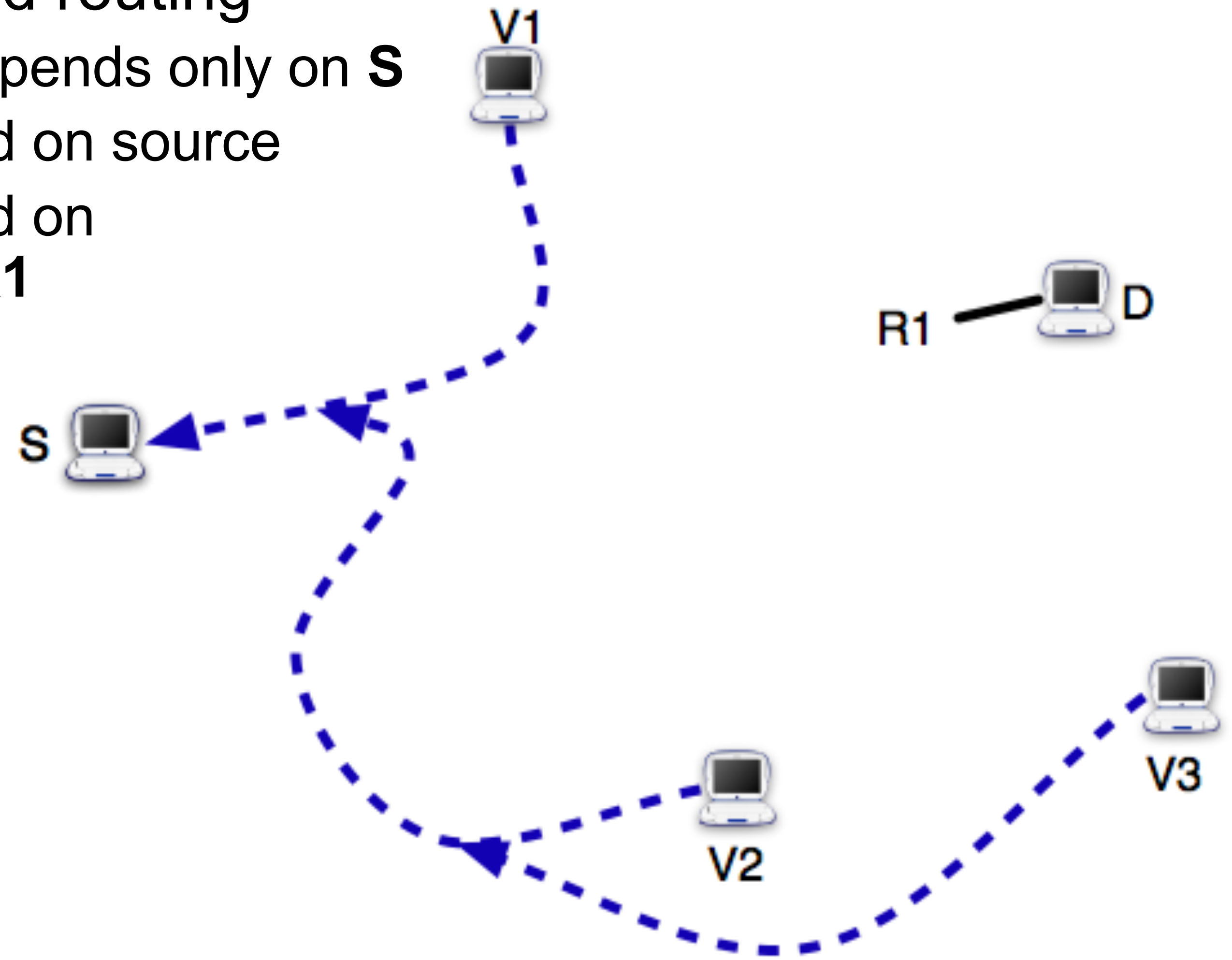
- Traceroute from all vantage points to **S**
- Gives atlas of paths to **S**; if we hit one, we know rest of path
  - Destination-based routing



KEY IDEA

- Traceroute atlas gives baseline we bootstrap from

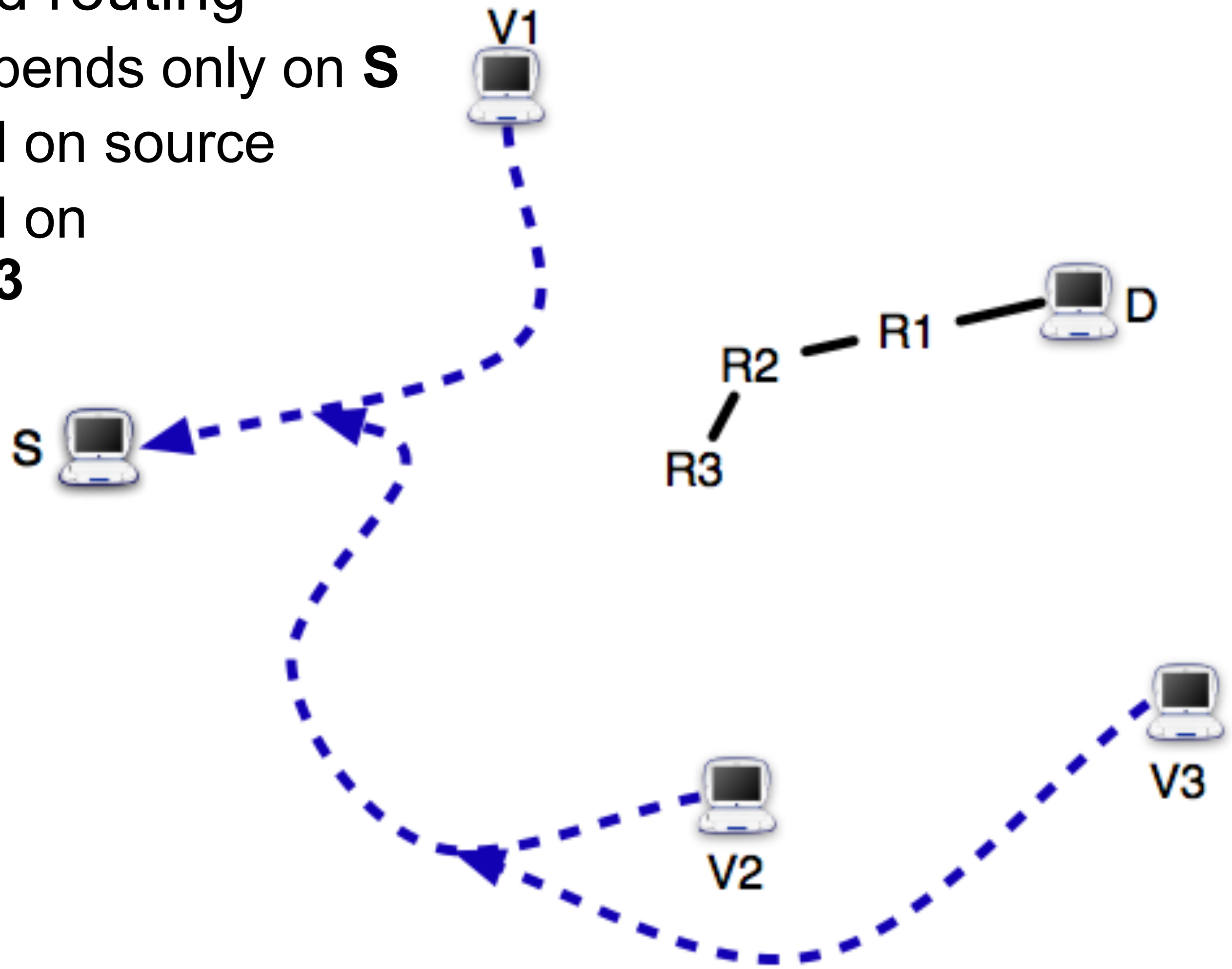
- Destination-based routing
  - Path from **R1** depends only on **S**
  - Does not depend on source
  - Does not depend on path from **D** to **R1**



KEY IDEA

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

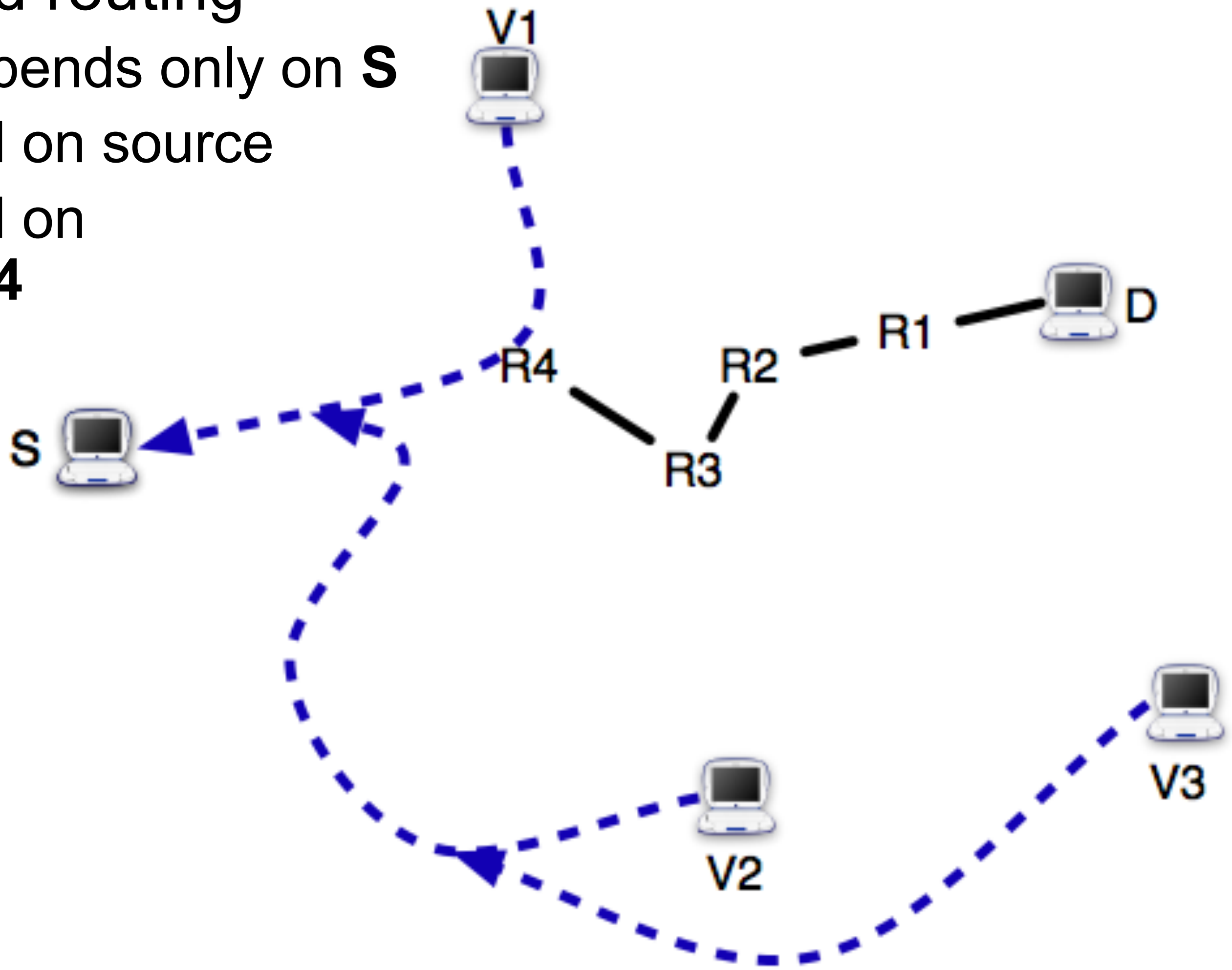
- Destination-based routing
  - Path from **R3** depends only on **S**
  - Does not depend on source
  - Does not depend on path from **D** to **R3**



KEY IDEA

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

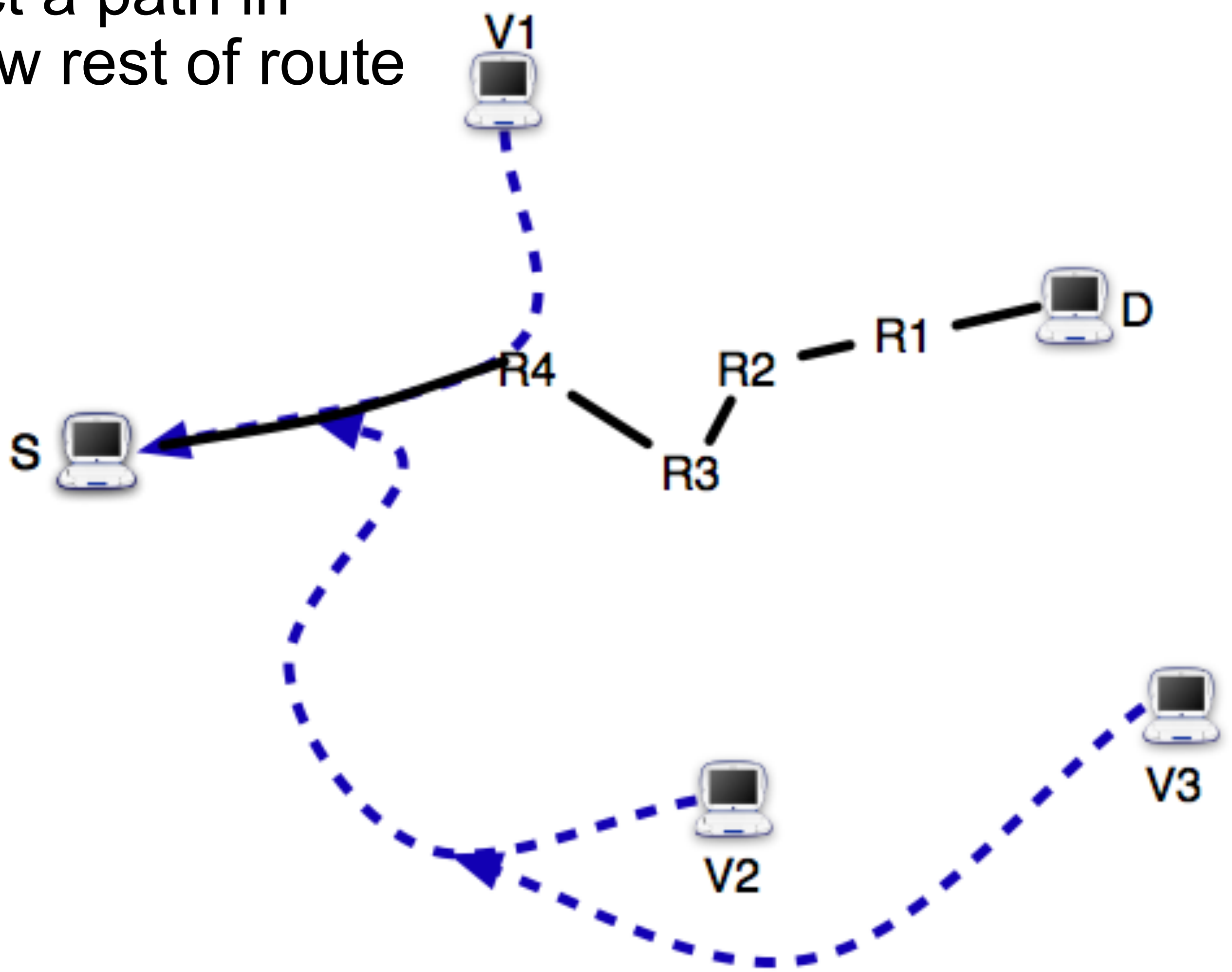
- Destination-based routing
  - Path from **R4** depends only on **S**
  - Does not depend on source
  - Does not depend on path from **D** to **R4**



KEY IDEA

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

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

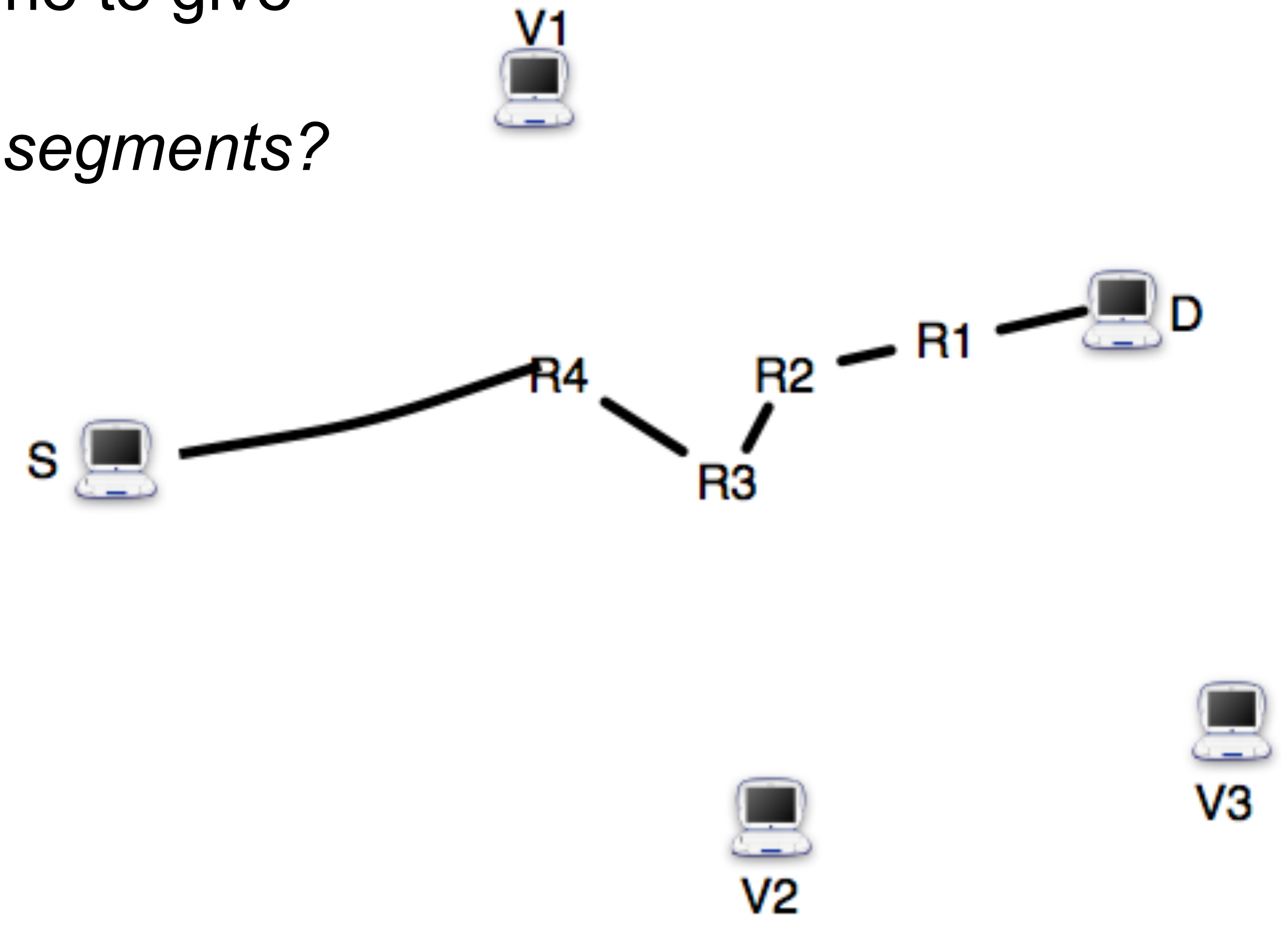


KEY IDEAS

- 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?

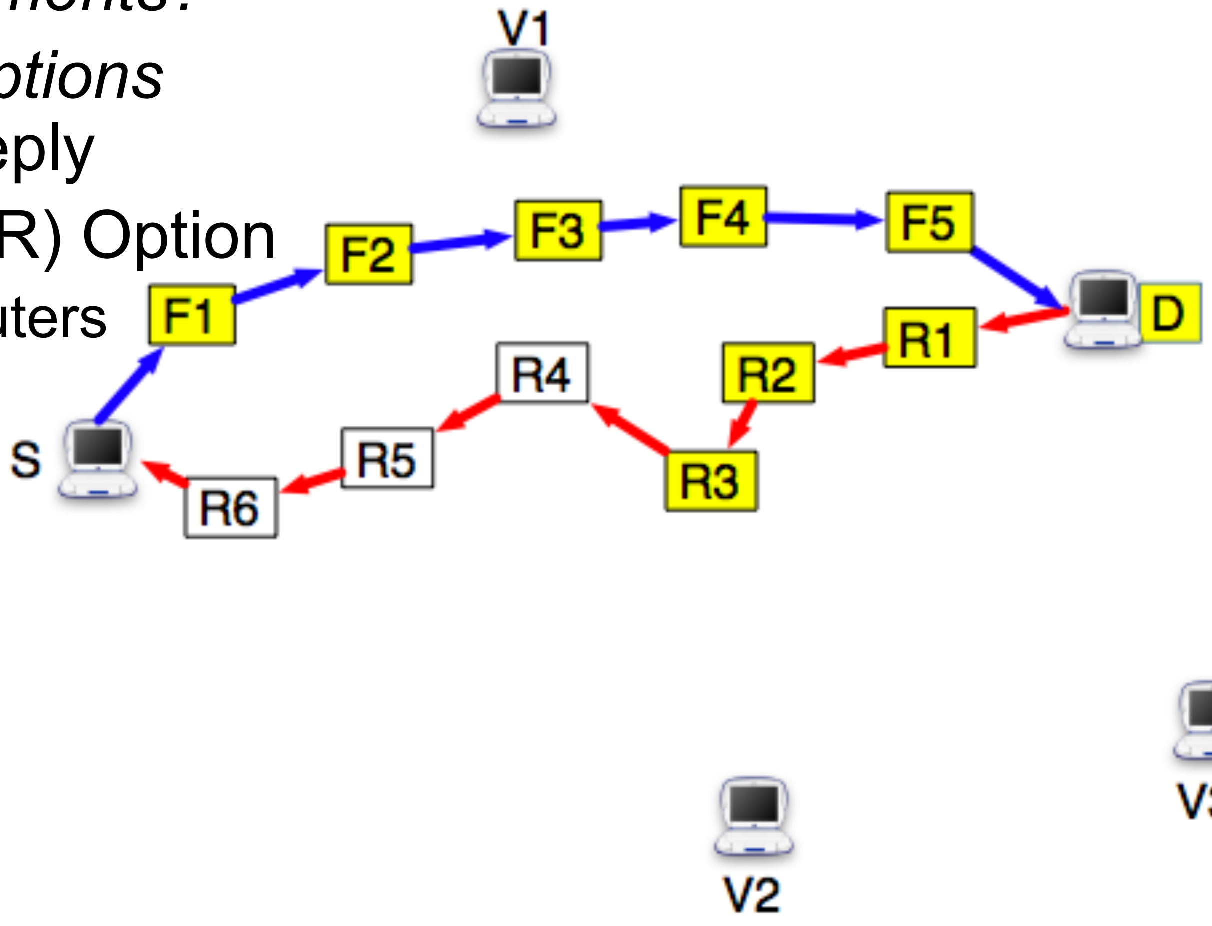


KEY IDEAS

- 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

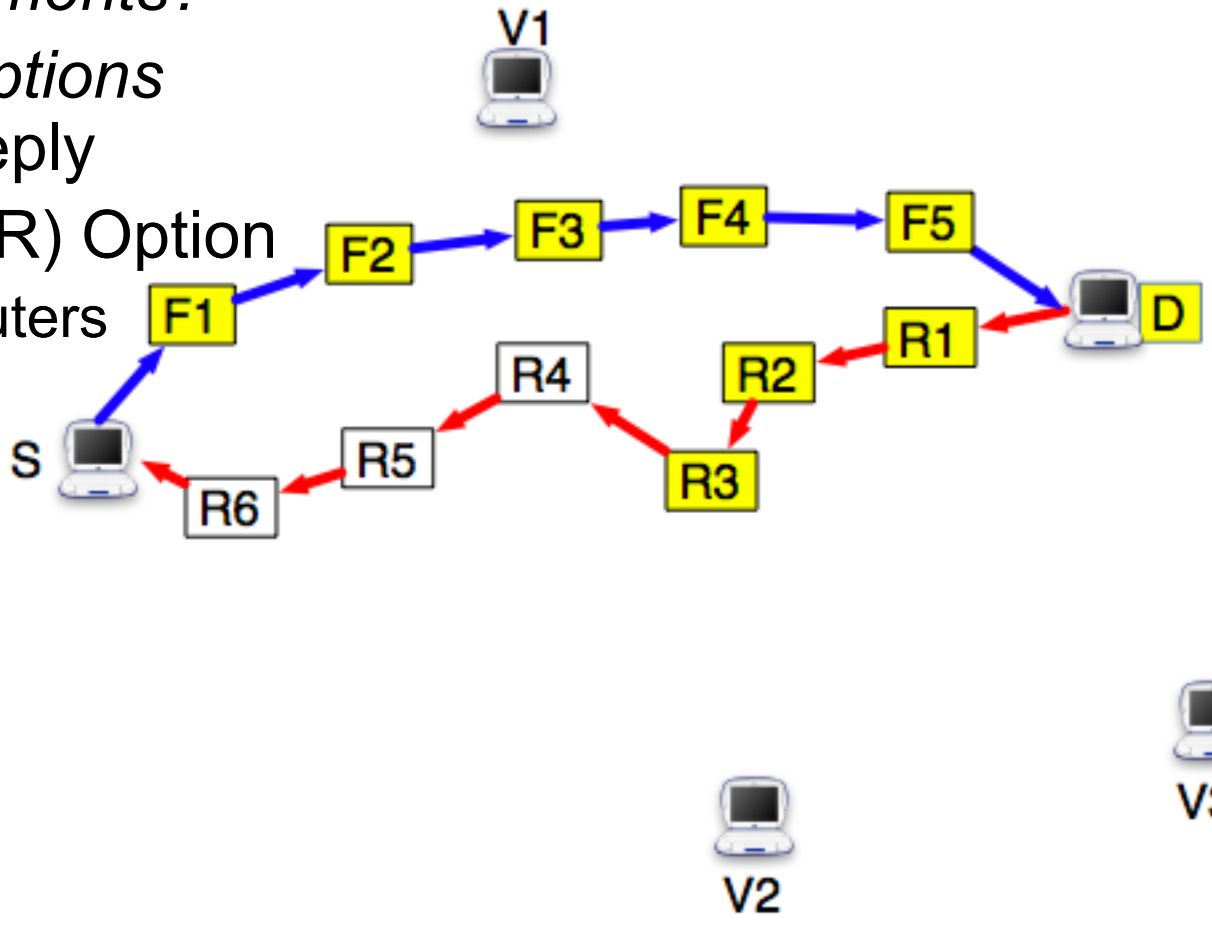


KEY IDEA

- IP Options work over forward and reverse path

How do we get segments?

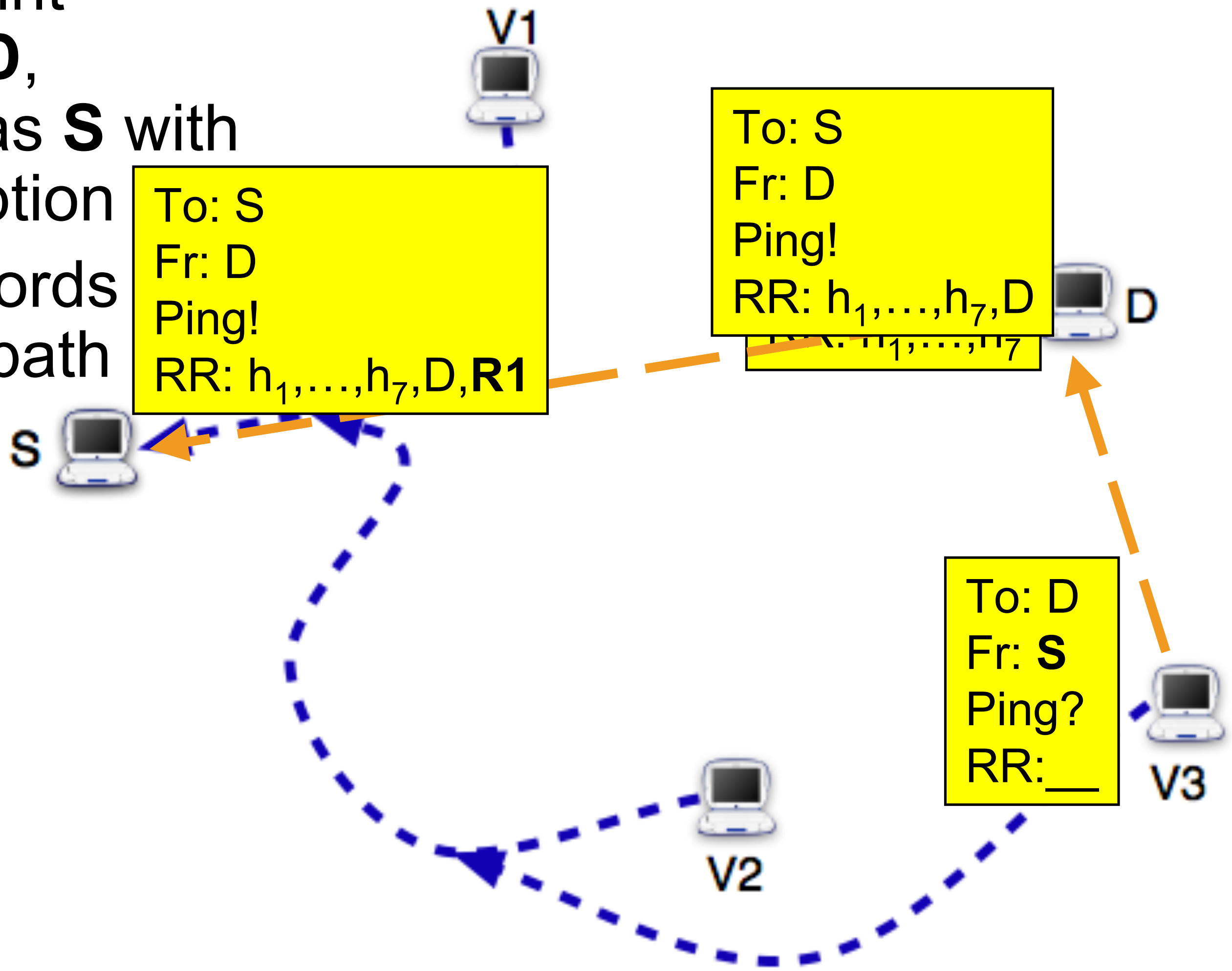
- Unlike TTL, *IP Options* are reflected in reply
- Record Route (RR) Option
  - Record first 9 routers
  - If **D** within 8, reverse hops fill rest of slots
  - … but average path is 15 hops, 30 round-trip



KEY IDEA

- IP Options work over forward and reverse path

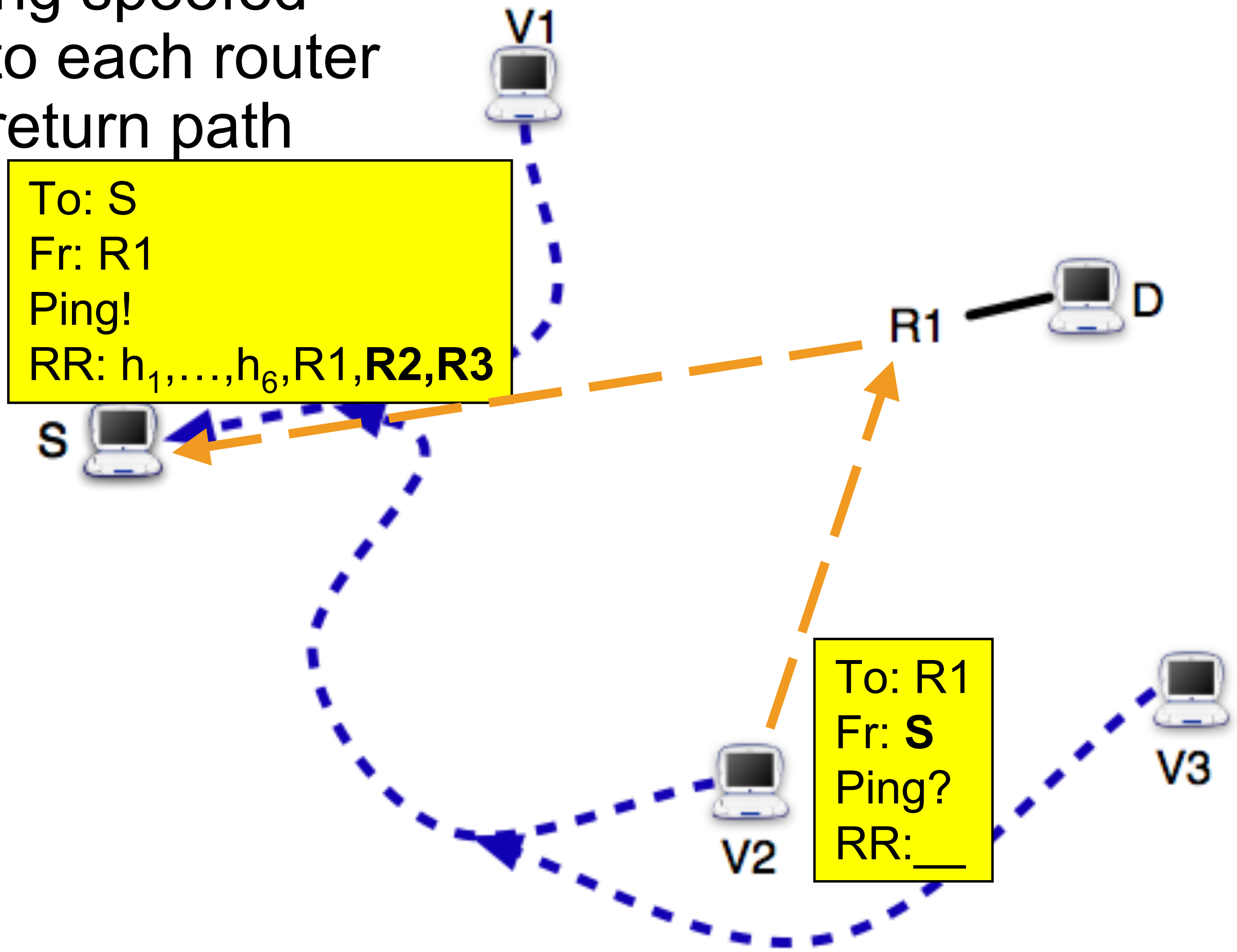
- From vantage point within 8 hops of **D**, ping **D** spoofing as **S** with Record Route Option
- **D**'s response records hop(s) on return path



KEY IDEA

- Spoofing lets us use vantage point in best position

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

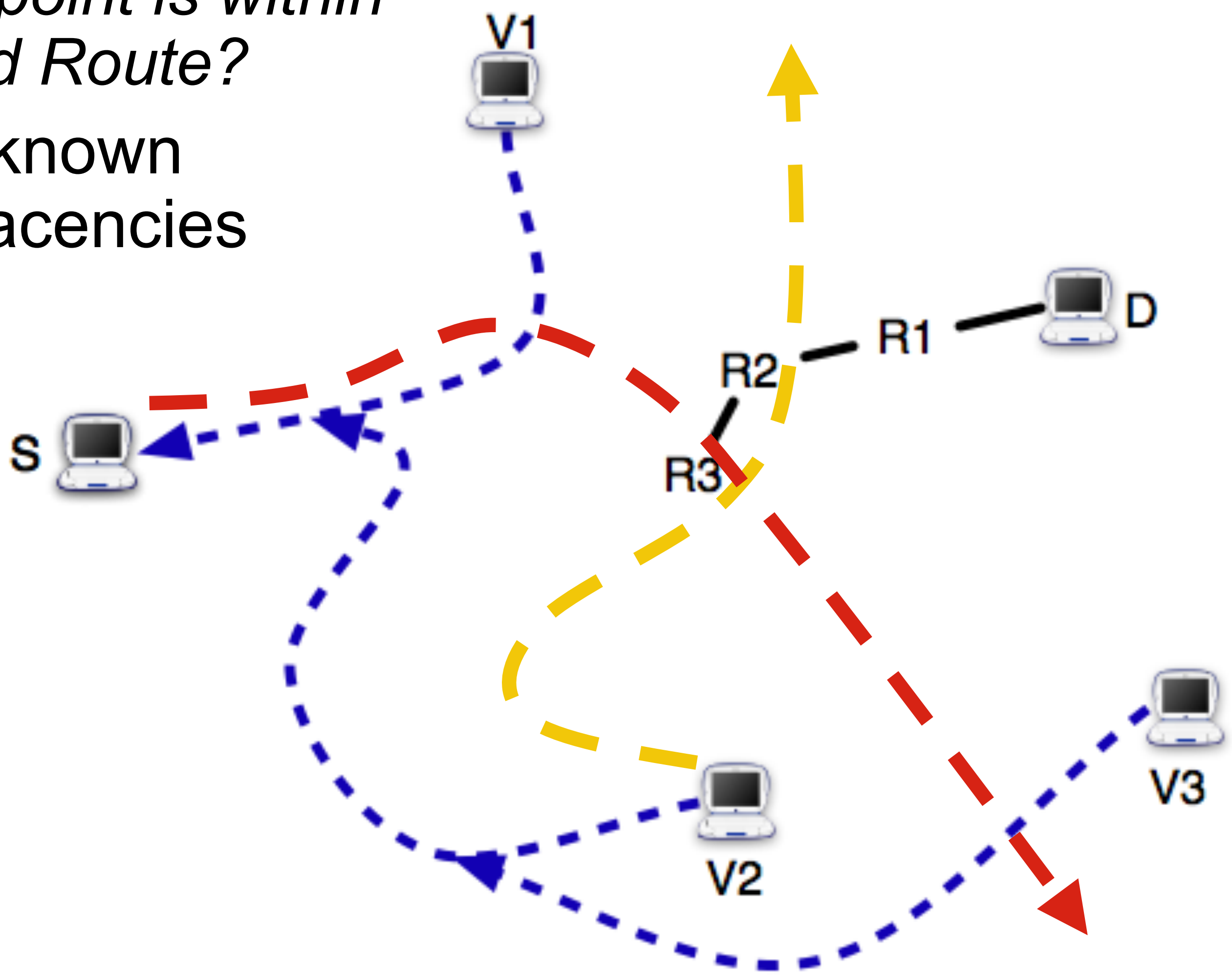


KEY IDEAS

- 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

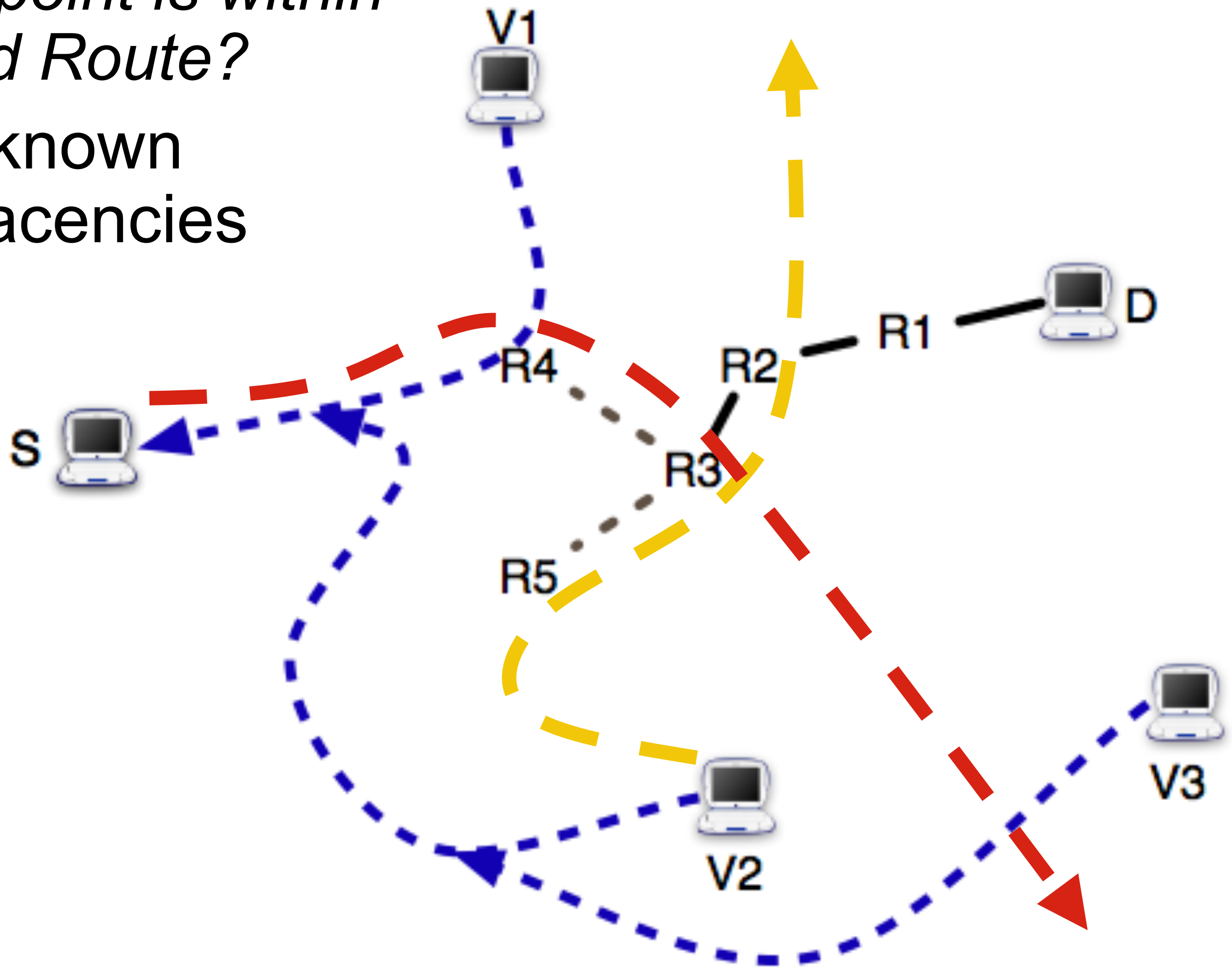


KEY IDEAS

- 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

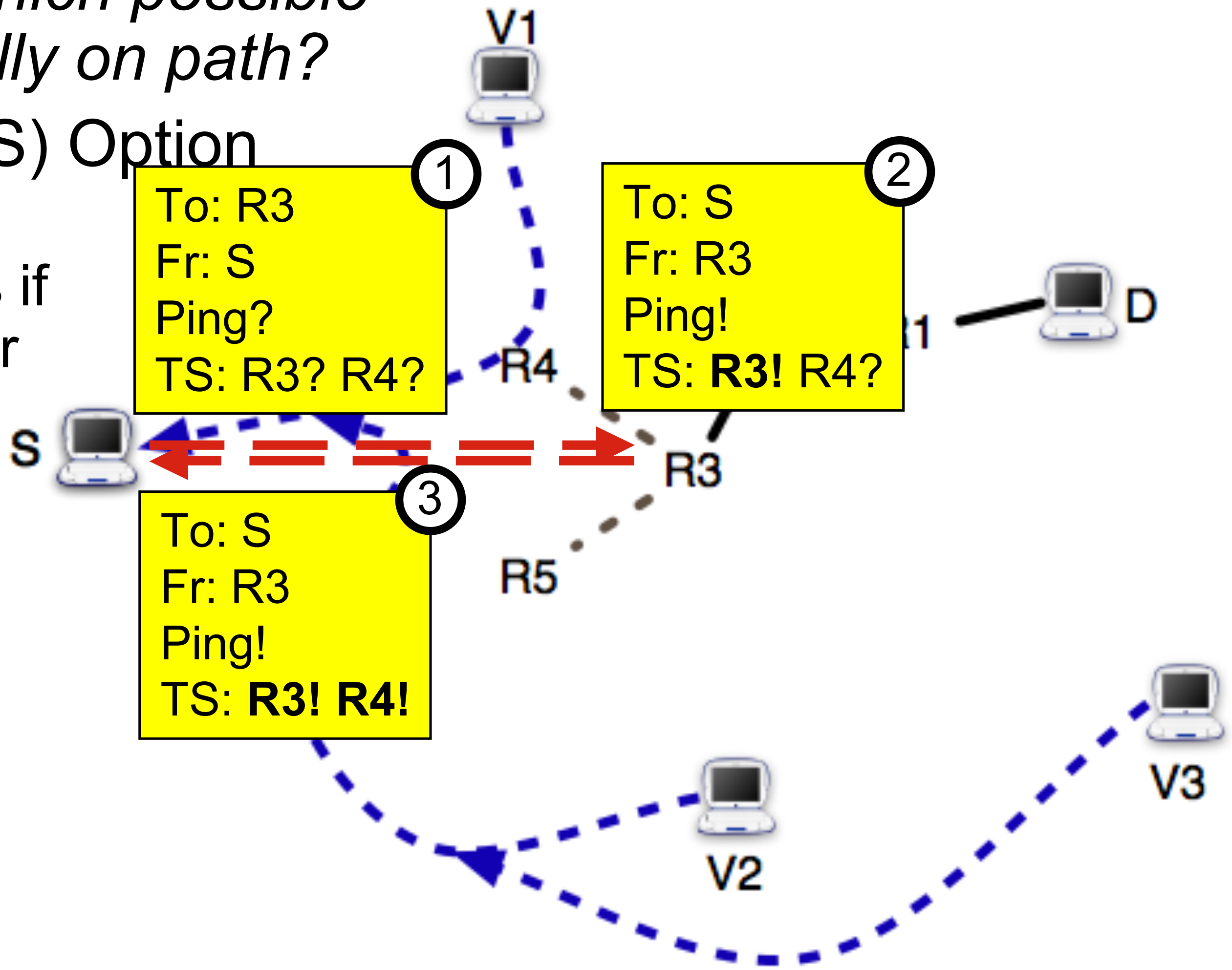


KEY IDEA

- Known paths provide set of possible next hops to guess

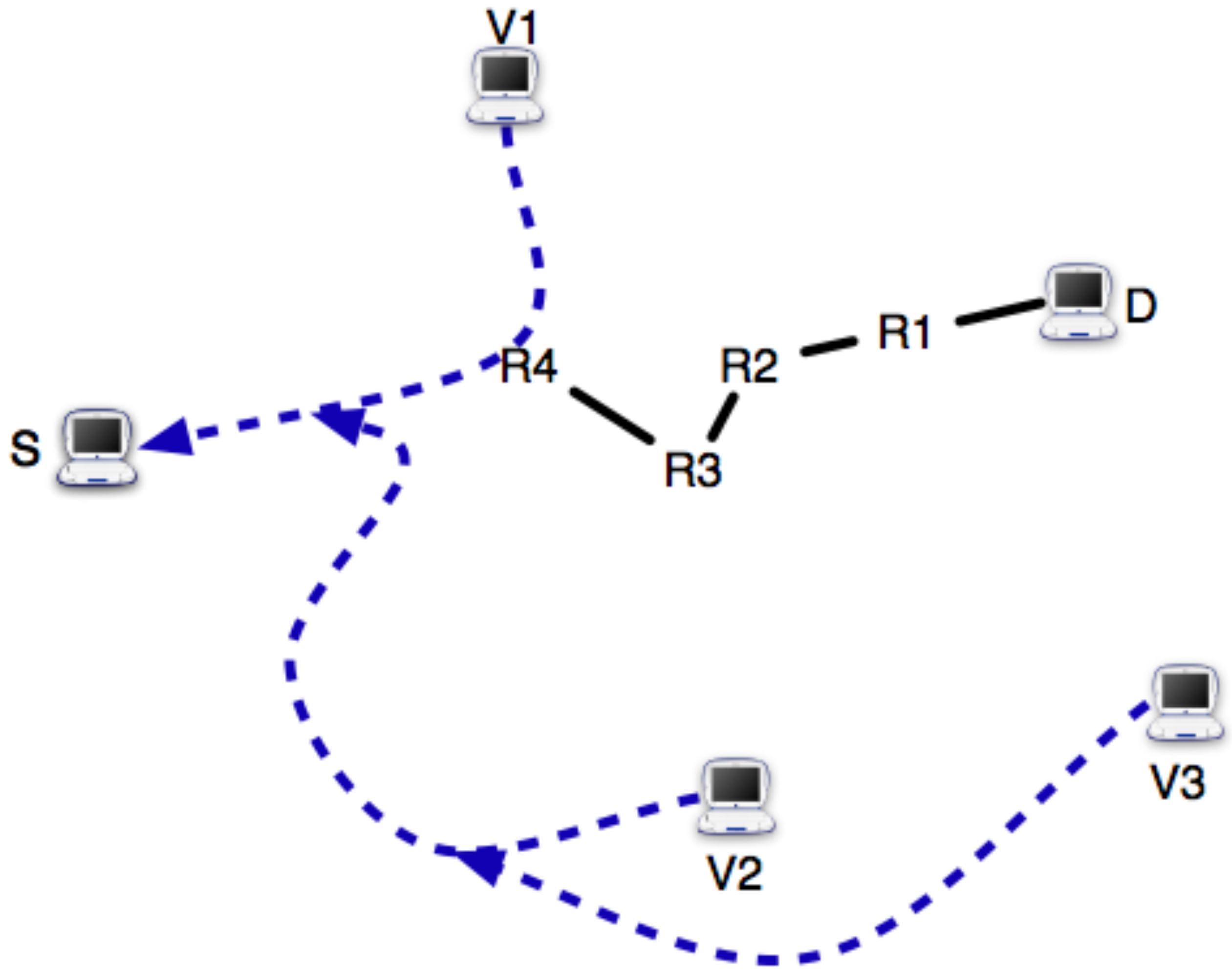
How do we verify which possible next hop is actually on path?

- IP Timestamp (TS) Option
  - Specify ≤ 4 IPs, each timestamps if traversed in order



KEY IDEAS

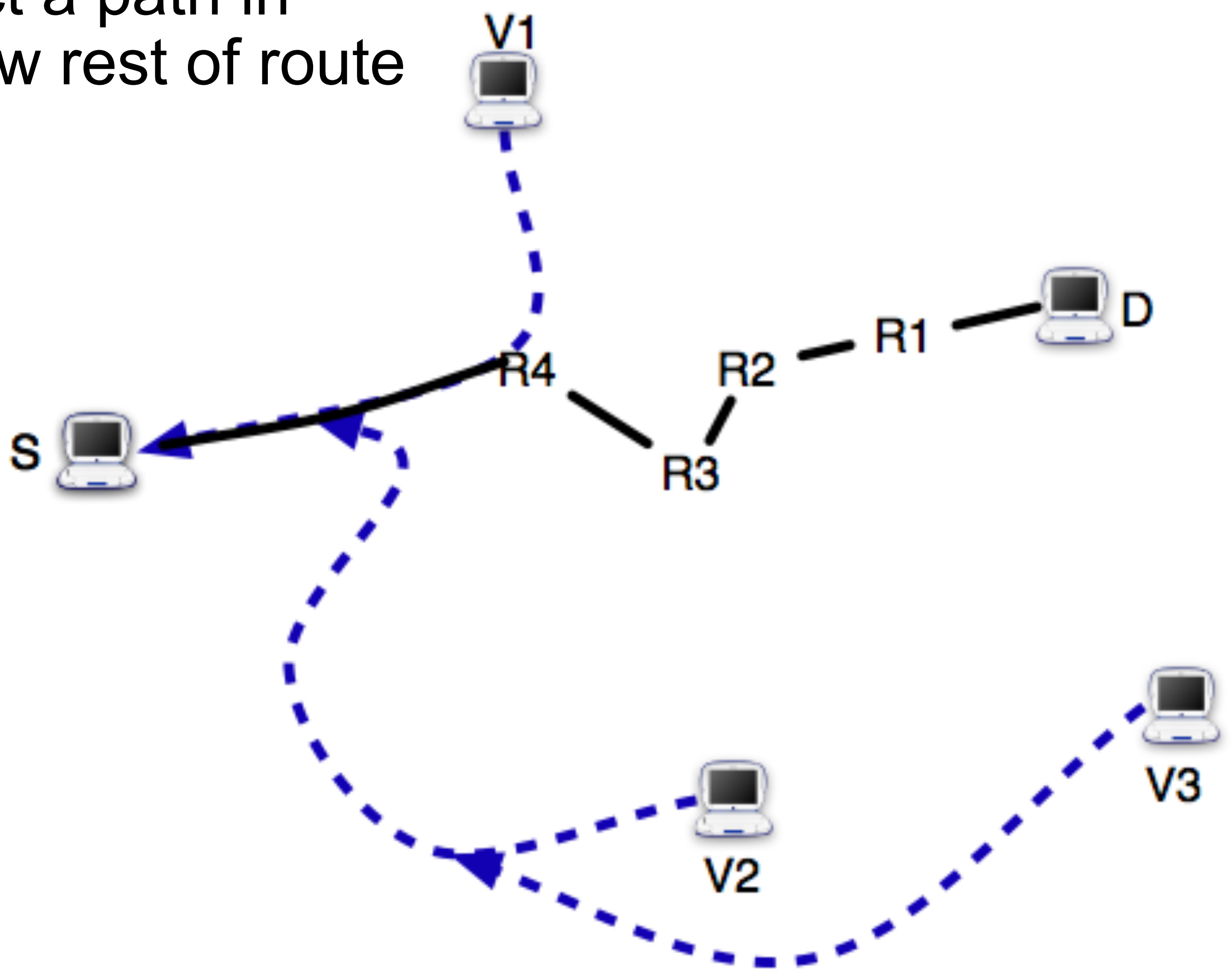
- Known paths provide set of possible next hops to guess
- IP Options work over forward and reverse path



KEY IDEA

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

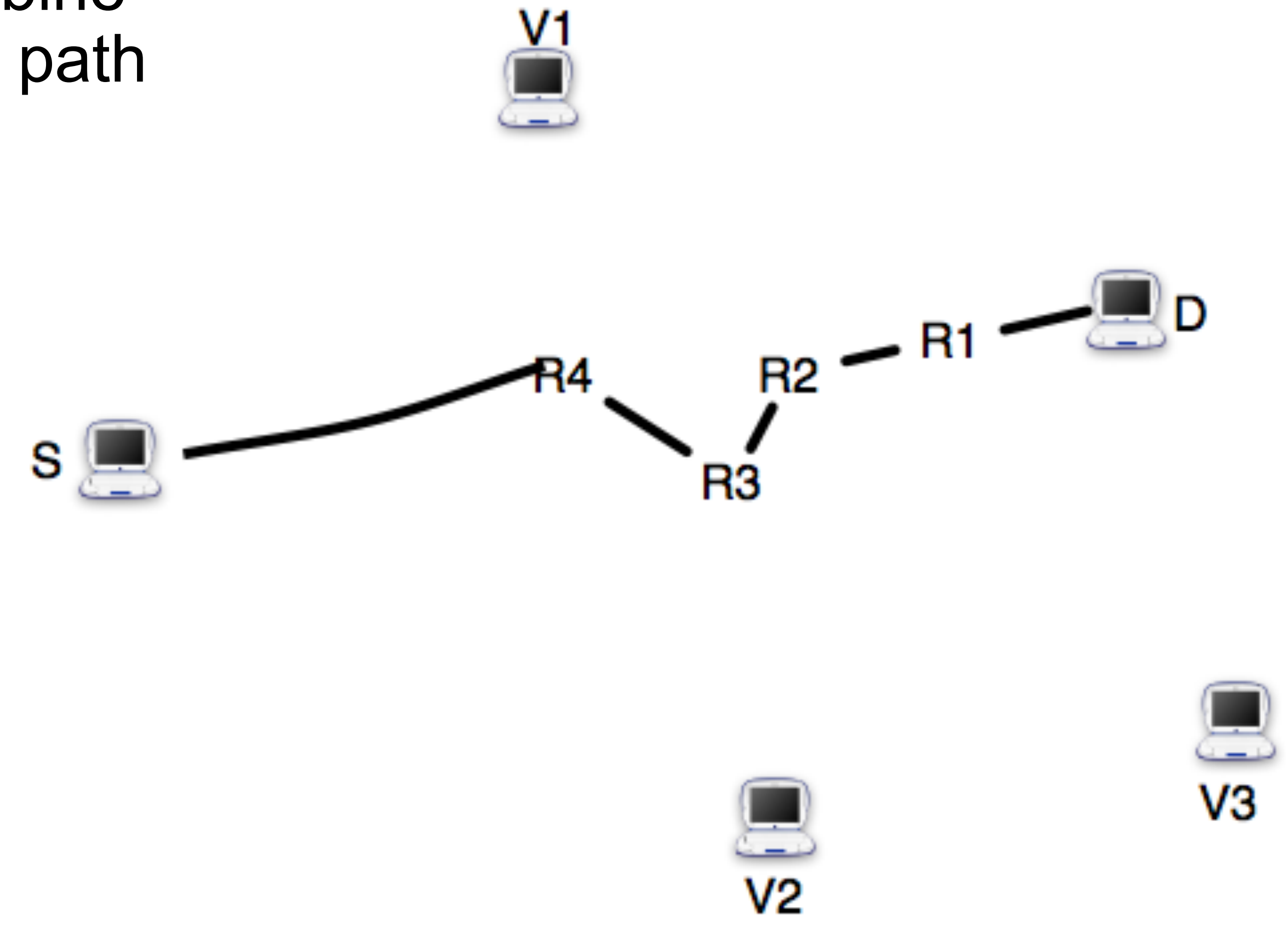
- Once we intersect a path in our atlas, we know rest of route



KEY IDEAS

- 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

- Destination-based routing lets us stitch path hop-by-hop
- Traceroute atlas gives baseline we bootstrap from

Key Ideas

- Works without control of destination
- Multiple vantage points
- Stitch path hop-by-hop
- Traceroute atlas provides:
  - Baseline paths
  - Adjacencies
- IP Options work over forward and reverse path
- Spoofing lets us use vantage point in best position

See paper for 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 spoofing vantage points (VPs)

- Current:
  - VPs: PlanetLab / Measurement Lab
    - ~90 sites did not filter spoofing
  - Sources: Closed system of PlanetLab sources, demo at http://revtr.cs.washington.edu
- Future plans:
  - VPs: Recruit participants to improve coverage
  - Sources: Open system to outside sources

Evaluation

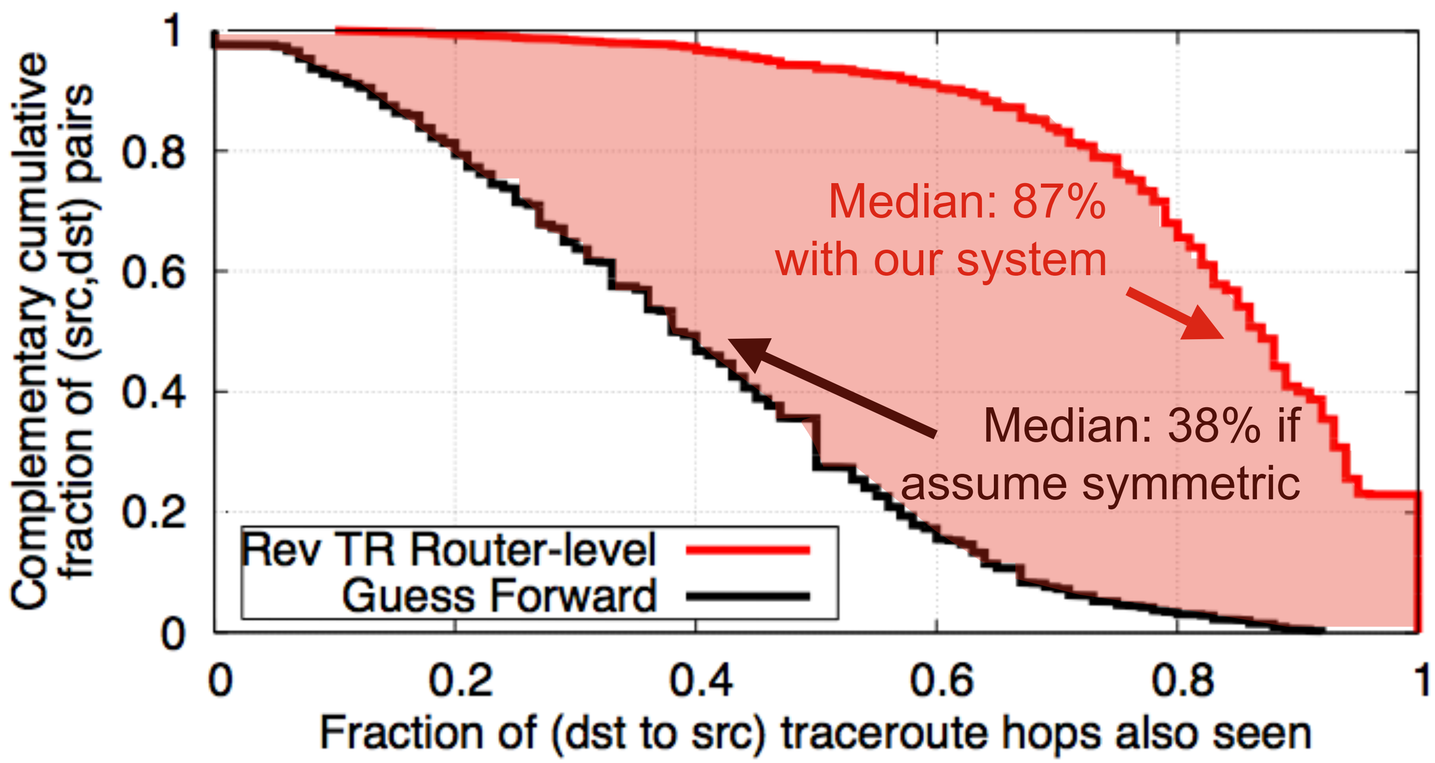
See paper for:

- **Coverage**: How often are our techniques able to measure reverse hops?
- **Overhead**: How much time and how many packets does a reverse traceroute require?

Next:

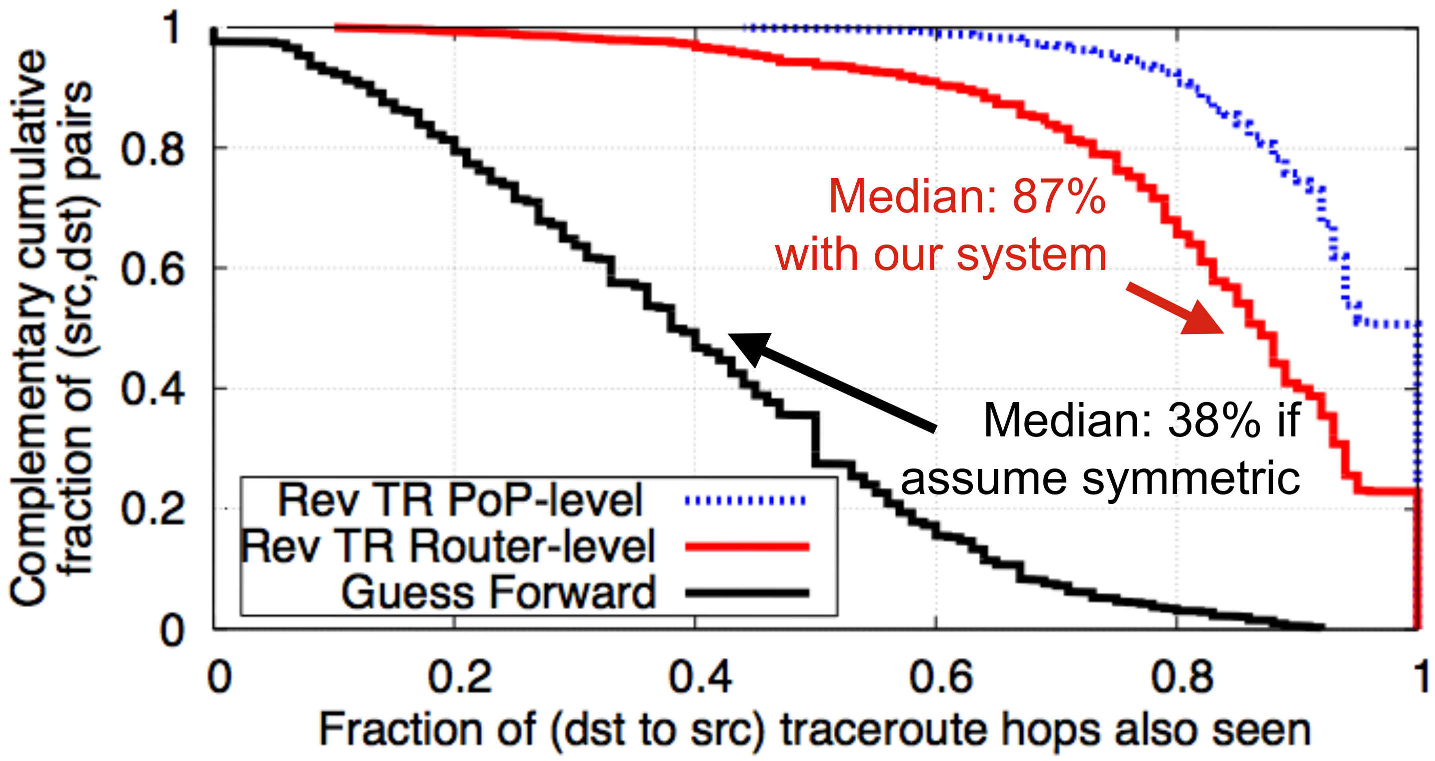
- **Accuracy**: Does it yield the same path as if you could issue a traceroute from destination?
  - 2200 PlanetLab to PlanetLab paths
  - Allows comparison to direct traceroute on “reverse” path

Does it give the same path as traceroute?



- We identify most hops seen by traceroute
- Hard to know if 2 IPs actually are the same router

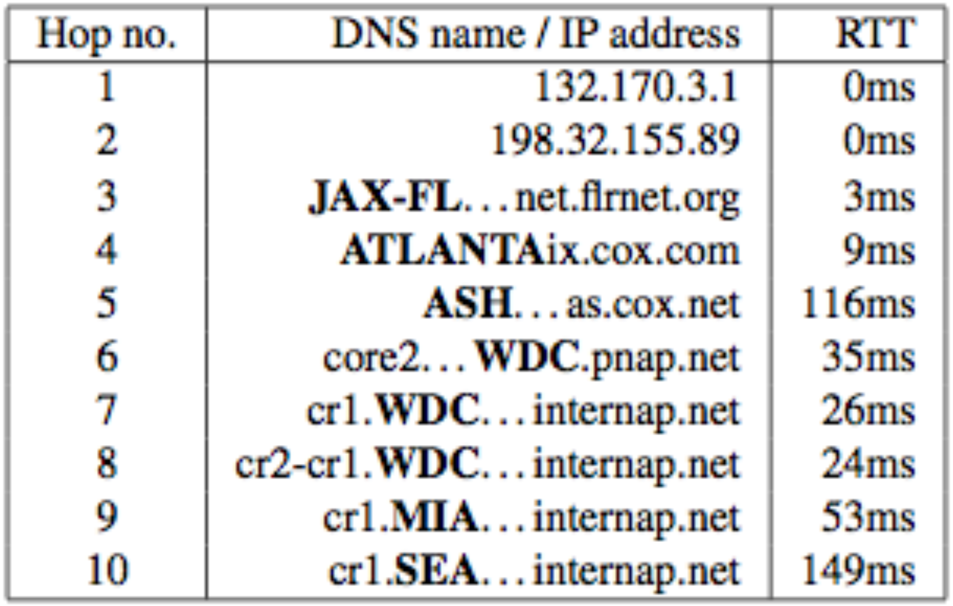
Does it give the same path as traceroute?



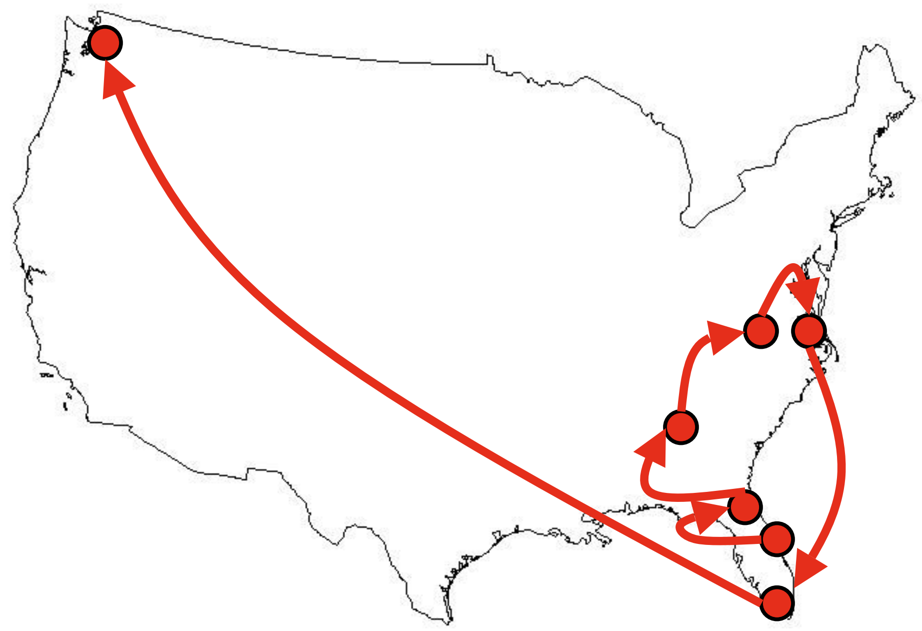
- We identify most hops seen by traceroute
- Hard to know if 2 IPs actually are the same router
  - If we consider PoPs instead, median=100% accurate

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



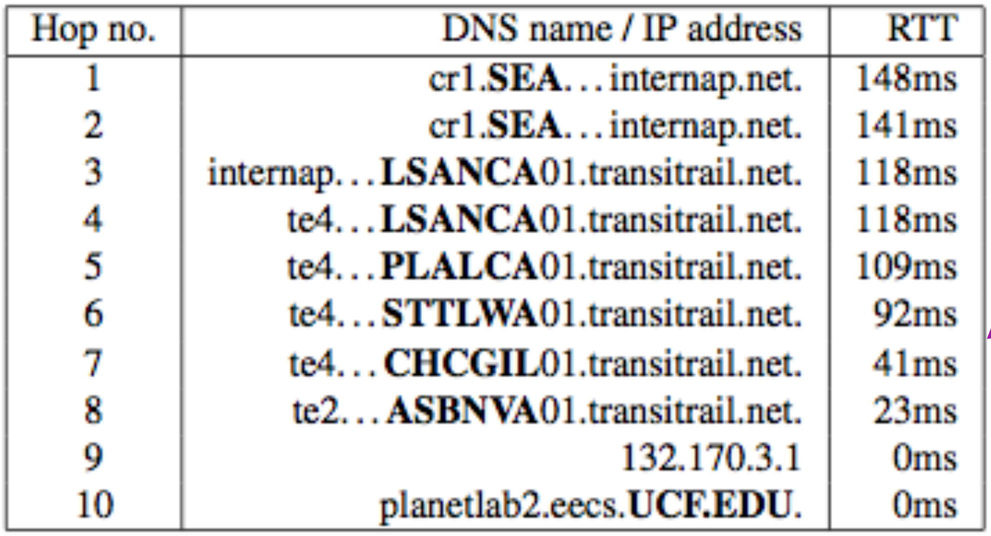
Hop no.	DNS name / IP address	RTT
1	132.170.3.1	0ms
2	198.32.155.89	0ms
3	**JAX-FL**...net.flrnet.org	3ms
4	**ATLANTA**ix.cox.com	9ms
5	**ASH**...as.cox.net	116ms
6	core2...**WDC**.pnap.net	35ms
7	cr1.**WDC**...internap.net	26ms
8	cr2-cr1.**WDC**...internap.net	24ms
9	cr1.**MIA**...internap.net	53ms
10	cr1.**SEA**...internap.net	149ms



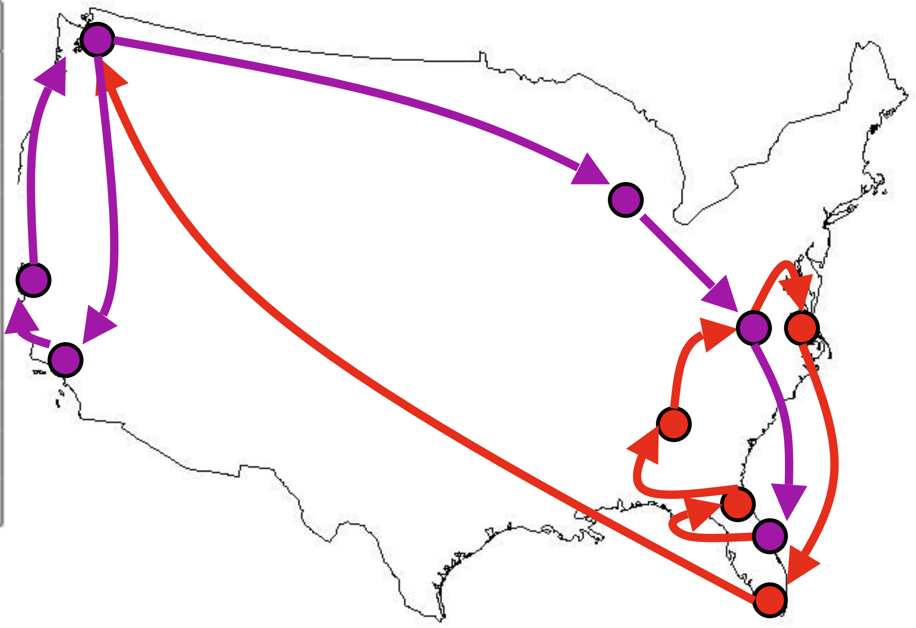
- Indirectness: FL→DC→FL
  But does not explain huge latency jump from 9 to 10

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

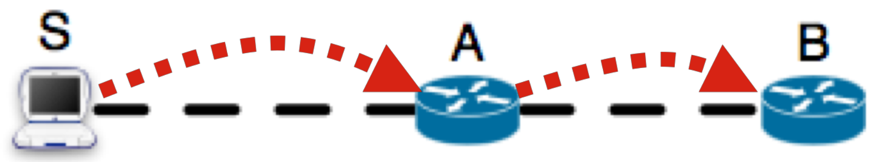


Hop no.	DNS name / IP address	RTT
1	cr1.**SEA**...internap.net.	148ms
2	cr1.**SEA**...internap.net.	141ms
3	internap...**LSANCA01**.transitrail.net.	118ms
4	te4...**LSANCA01**.transitrail.net.	118ms
5	te4...**PLALCA01**.transitrail.net.	109ms
6	te4...**STTLWA01**.transitrail.net.	92ms
7	te4...**CHCGIL01**.transitrail.net.	41ms
8	te2...**ASBNVA01**.transitrail.net.	23ms
9	132.170.3.1	0ms
10	planetlab2.eecs.**UCF.EDU**.	0ms



- Indirectness: WA→LA→WA
  Bad reverse path causes inflated round-trip delay

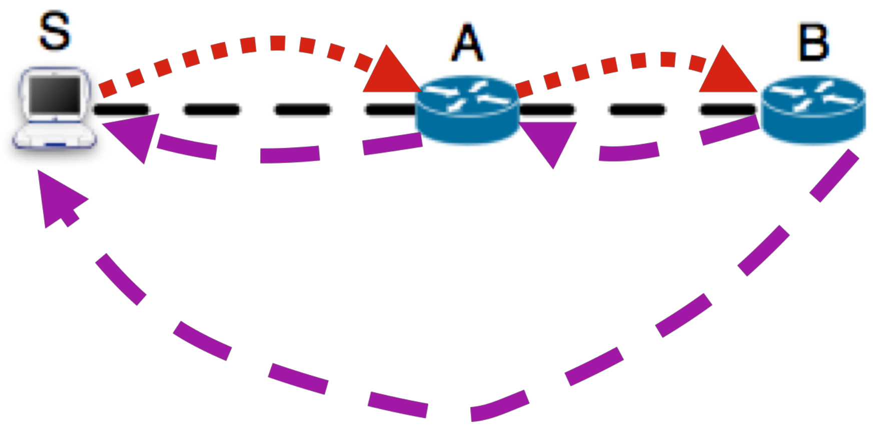
Measuring Link Latency



- Many applications want link latencies
  - IP geolocation, ISP performance, performance prediction, …
- Traditional approach is to assume symmetry:

  Delay(A,B) = (RTT(S,B) – RTT(S,A)) / 2
- Asymmetry skews link latency inferred with traceroute

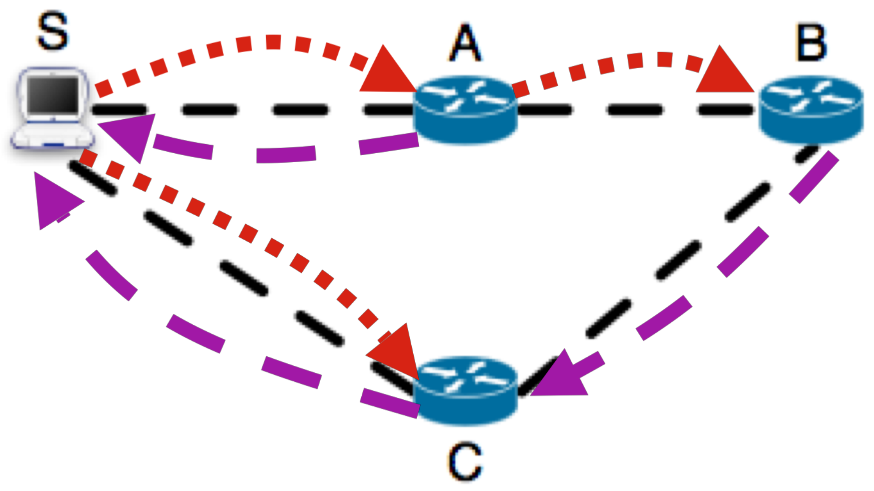
Measuring Link Latency



- Many applications want link latencies
  - IP geolocation, ISP performance, performance prediction, …
- Traditional approach is to assume symmetry:

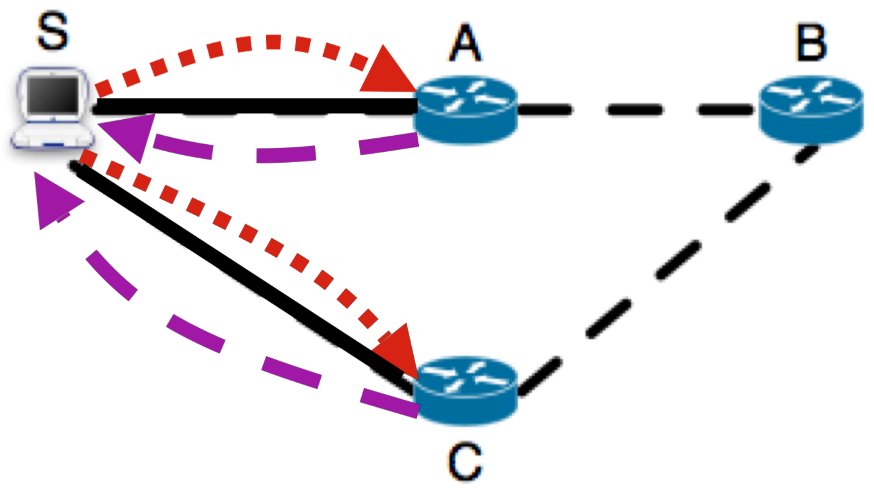
  Delay(A,B) = (RTT(S,B) – RTT(S,A)) / 2
- Asymmetry skews link latency inferred with traceroute

Reverse Traceroute Detects Symmetry



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

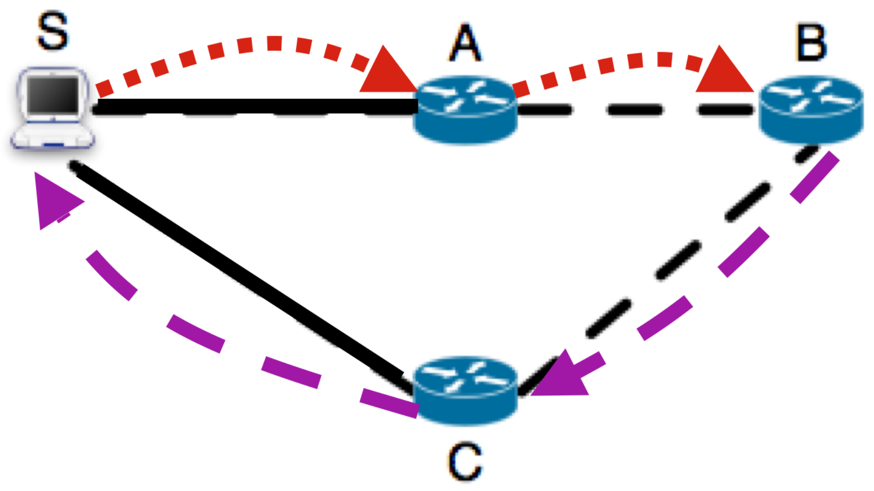
Reverse Traceroute Detects Symmetry



<u>Solved</u>
(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**)

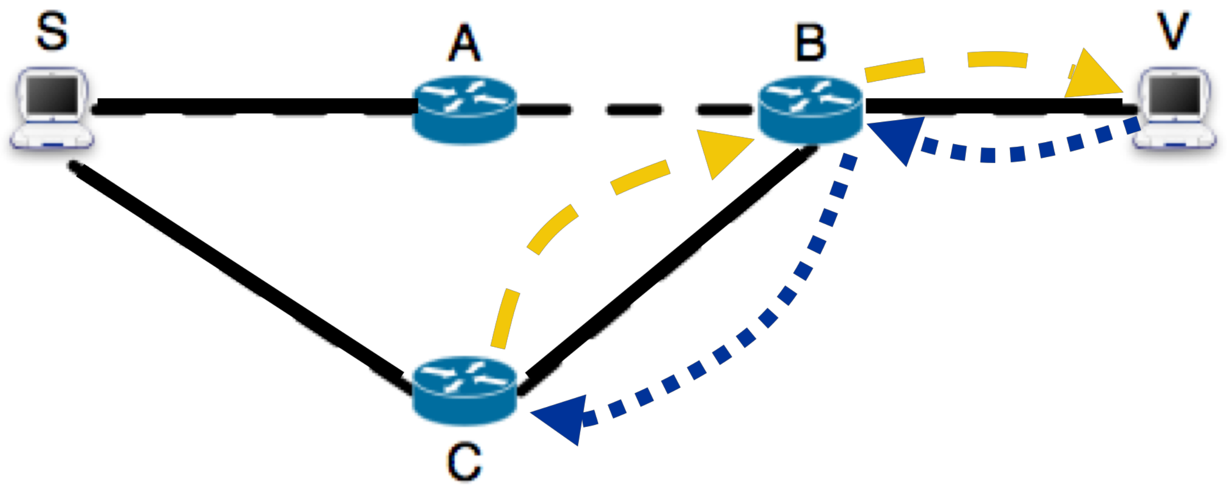
Reverse TR Constrains Link Latencies



<u>Solved</u>
(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

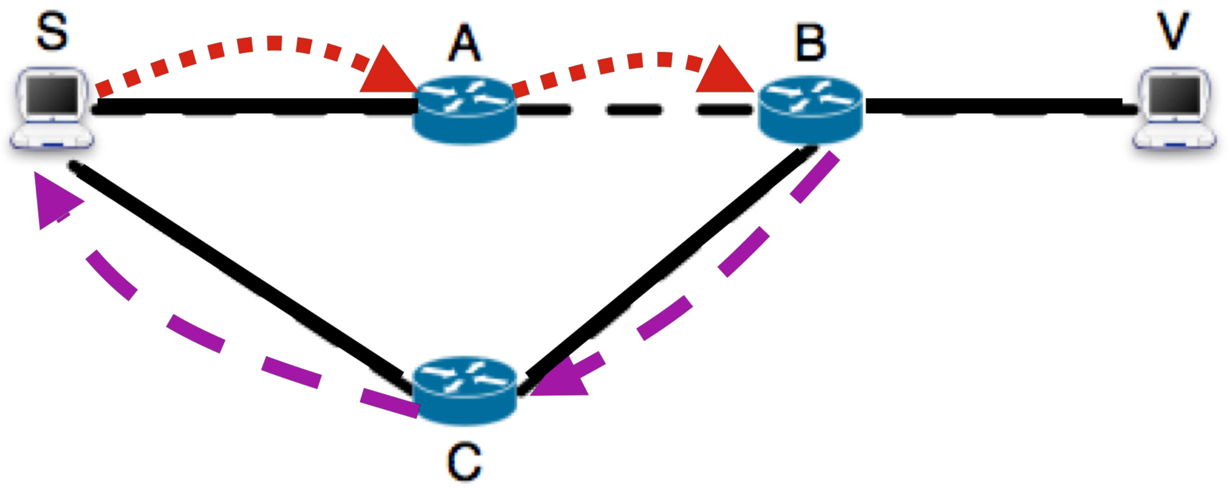
Reverse TR Constrains Link Latencies



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

- Build up system of constraints on link latencies of all intermediate hops
  - Traceroute and reverse traceroute to all hops
  - RTT = Forward links + Reverse links

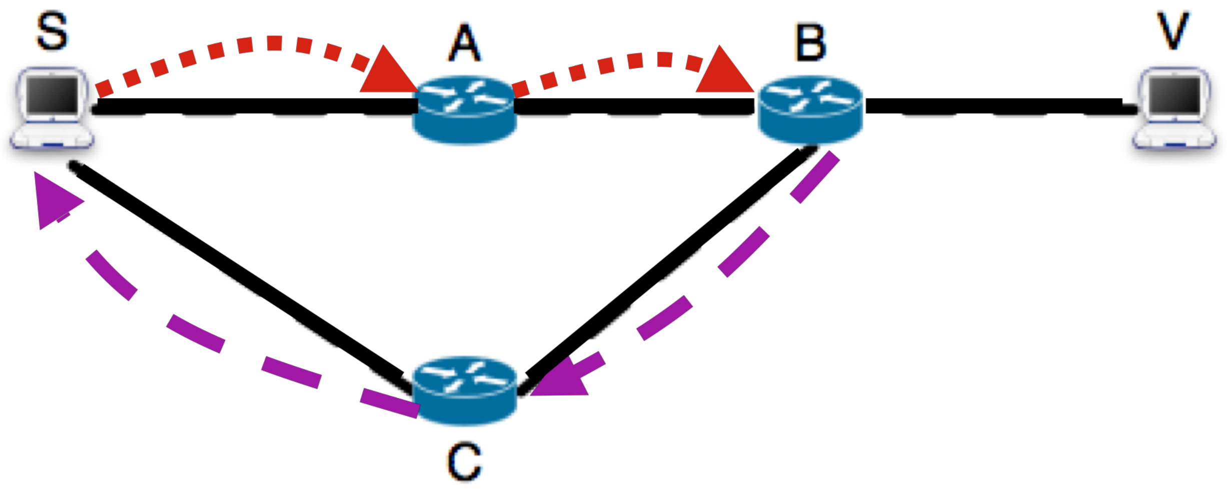
Reverse TR Constrains Link Latencies



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

- Build up system of constraints on link latencies of all intermediate hops
  - Traceroute and reverse traceroute to all hops
  - RTT = Forward links + Reverse links

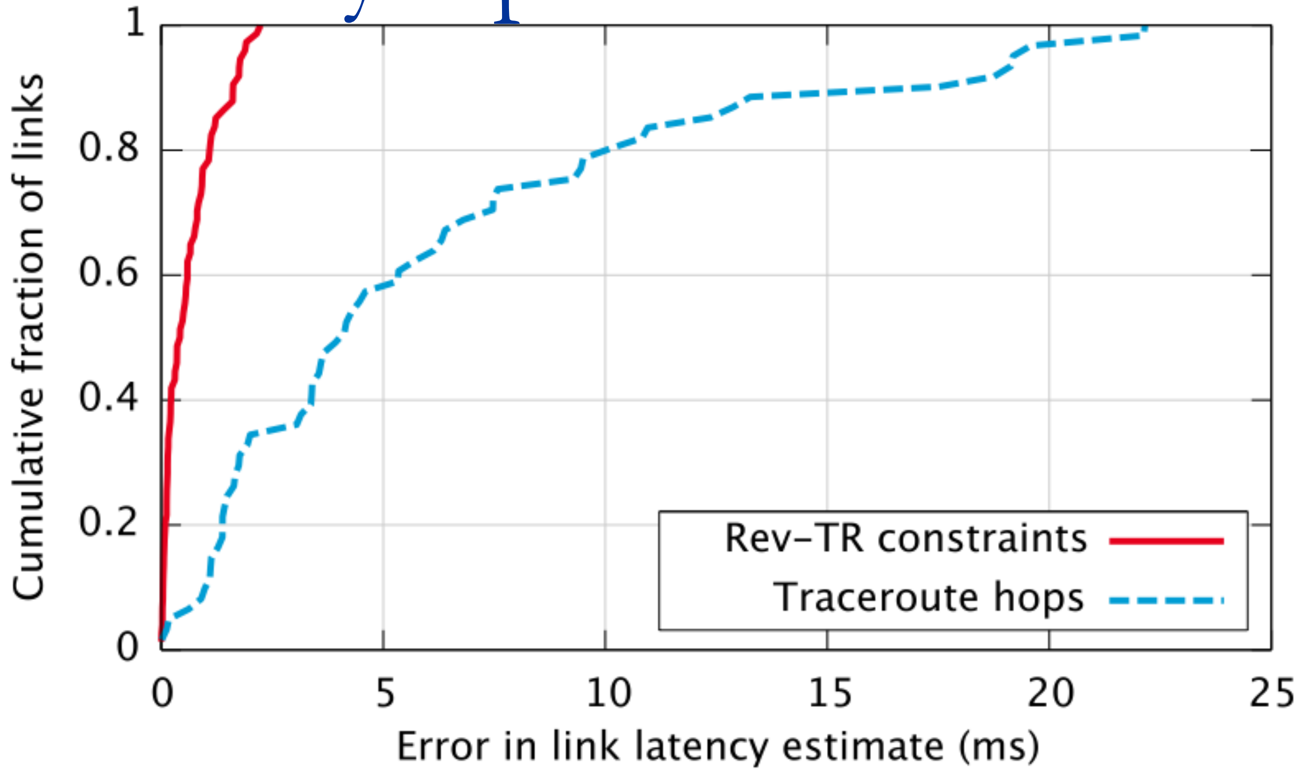
Reverse TR Constrains Link Latencies



<u>Solved</u>
(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

Conclusion

- Traceroute is very useful, but can’t give reverse path
- Our reverse traceroute system addresses limitation, providing complementary information
  - Multiple vantage points build the path incrementally
  - Gives most hops as if you issued traceroute from destination, without requiring you to control it
- Useful in a range of contexts

- Demo at http://revtr.cs.washington.edu
- Plan to open system to outside sources in future