Internet clouds are (also) unpredictable!

A study of **flow rerouting** and **latency variations** across the largest Cloud network worldwide

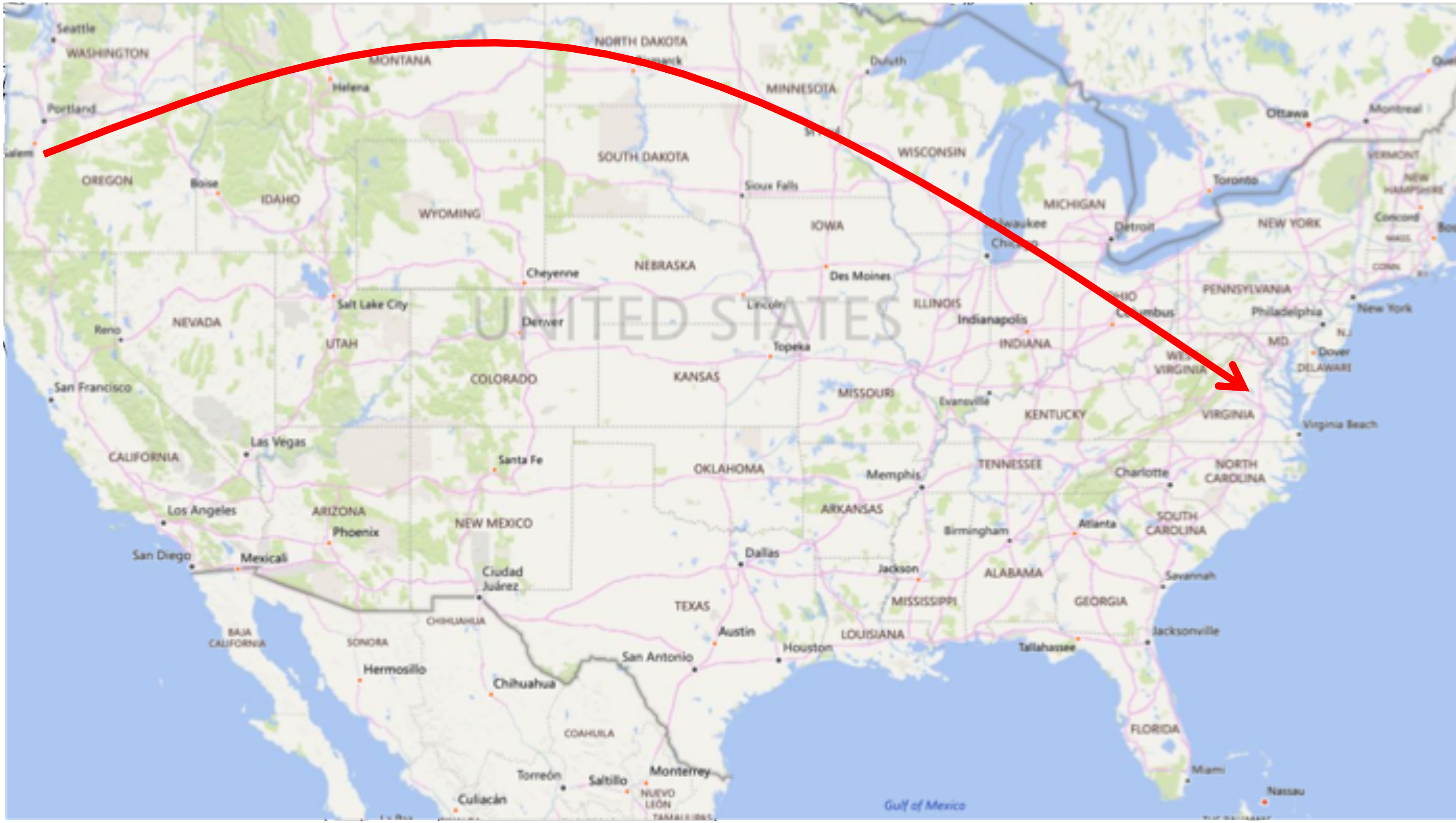
Marco Chiesa

KTH Royal Institute of Technology

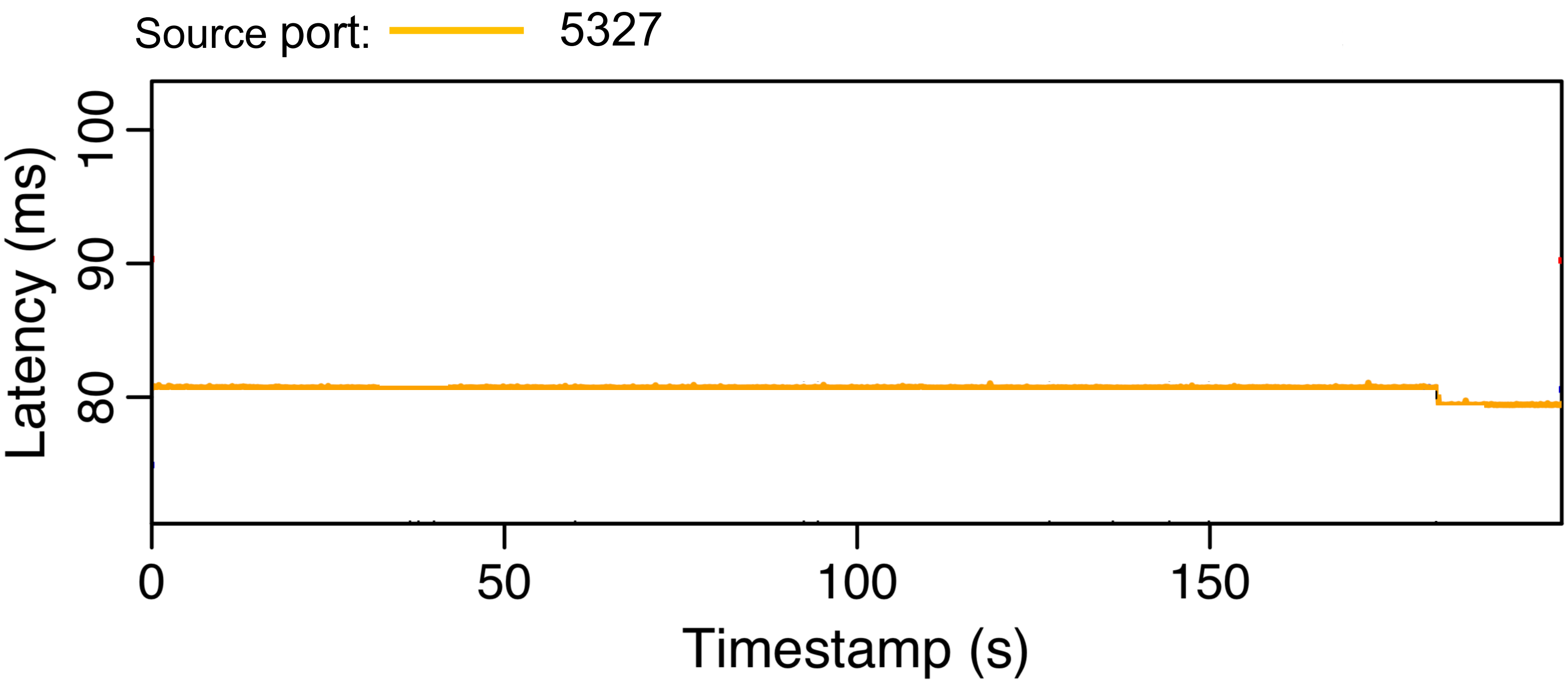
Joint work with: Waleed Reda, Kirill Bogdanov, Alexandros Milolidakis, Gerald Q Maguire Jr, Dejan Kostic

Thanks to the RACI initiative!

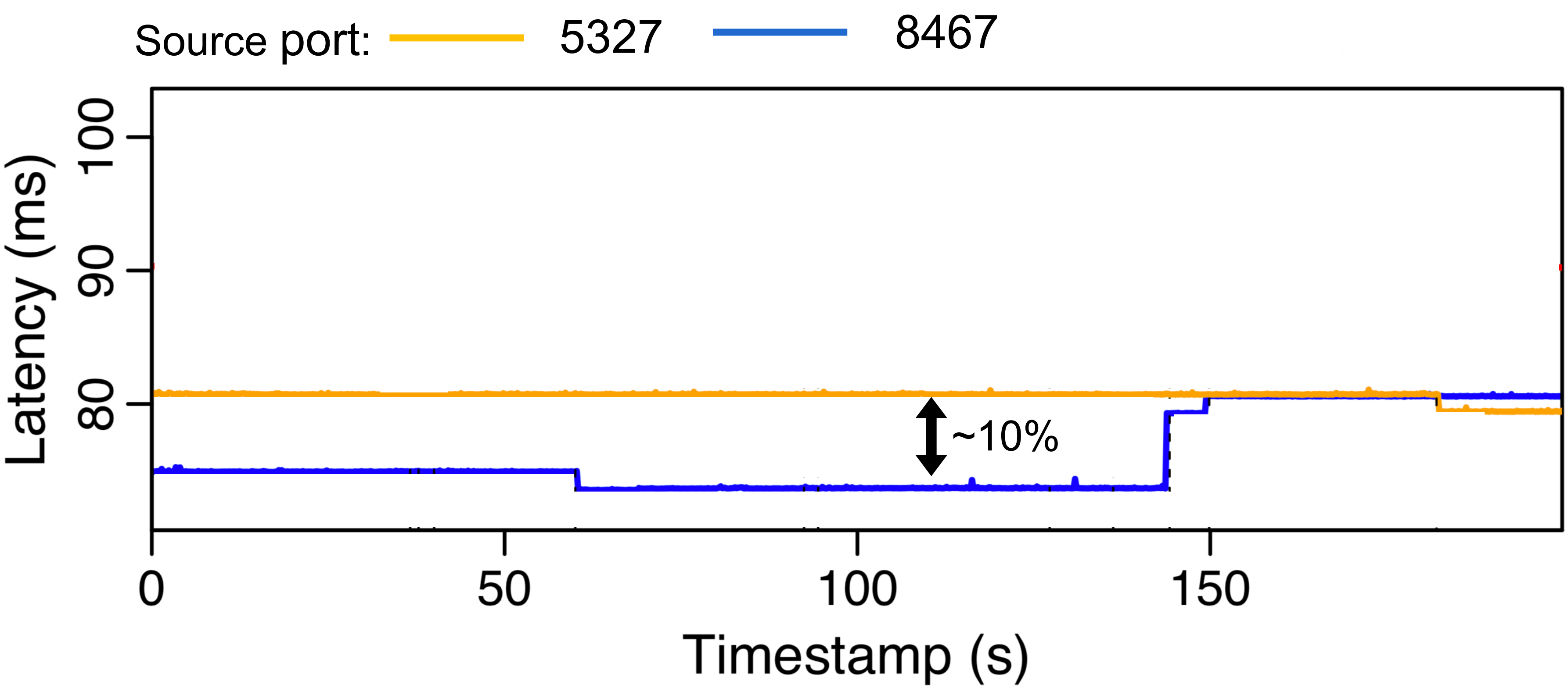
Let’s open a TCP connection between Oregon and Virginia



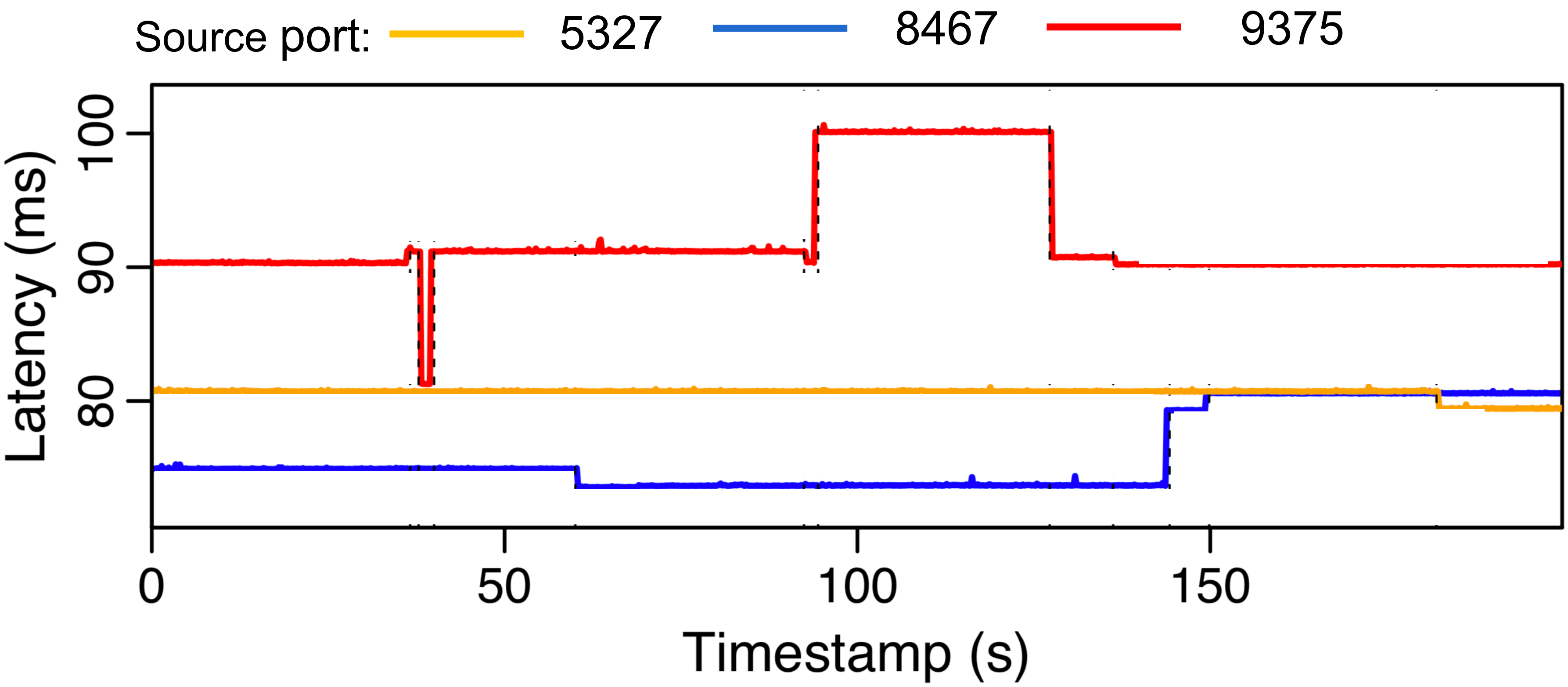
Stable Round Trip Time latency of roughly 80ms



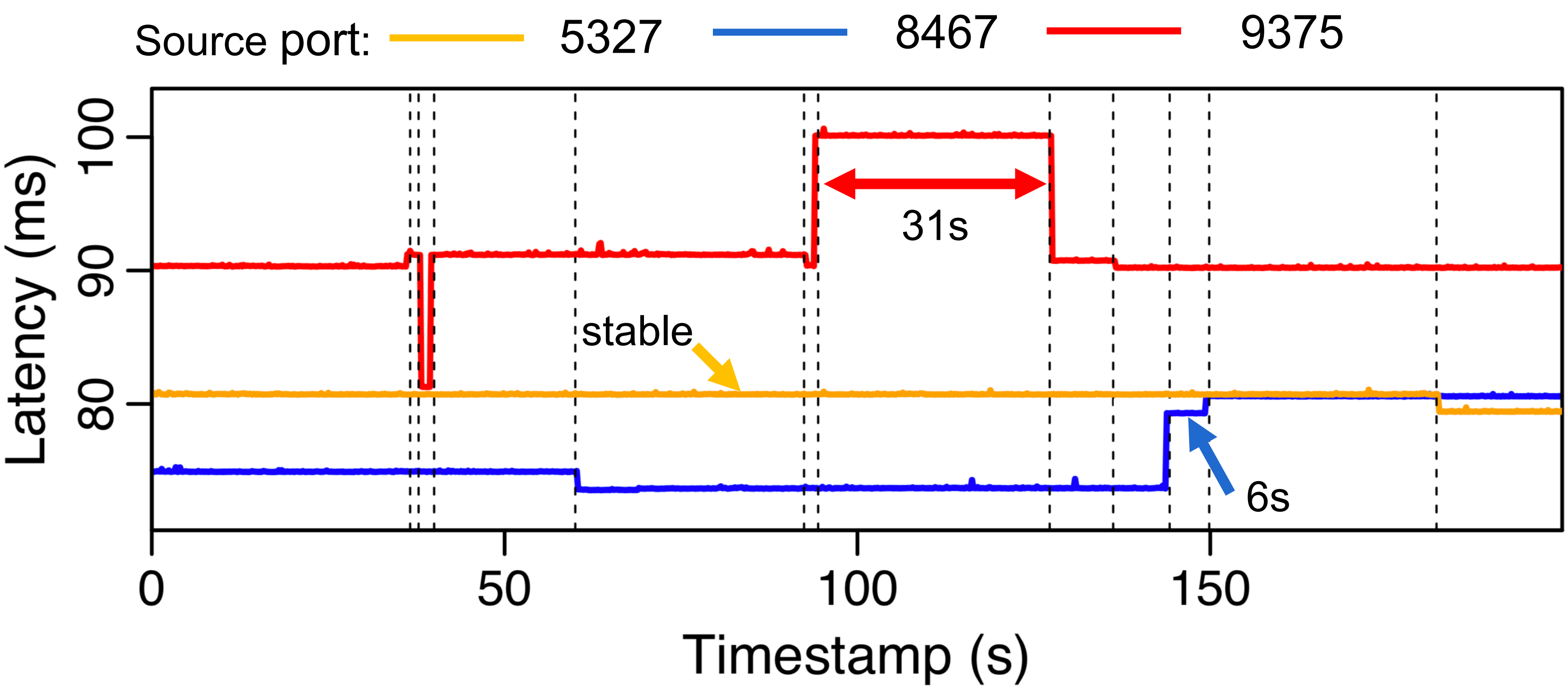
We open a second connection: Up to 10% better latency!



We open a third connection: Unstable and worse latency!

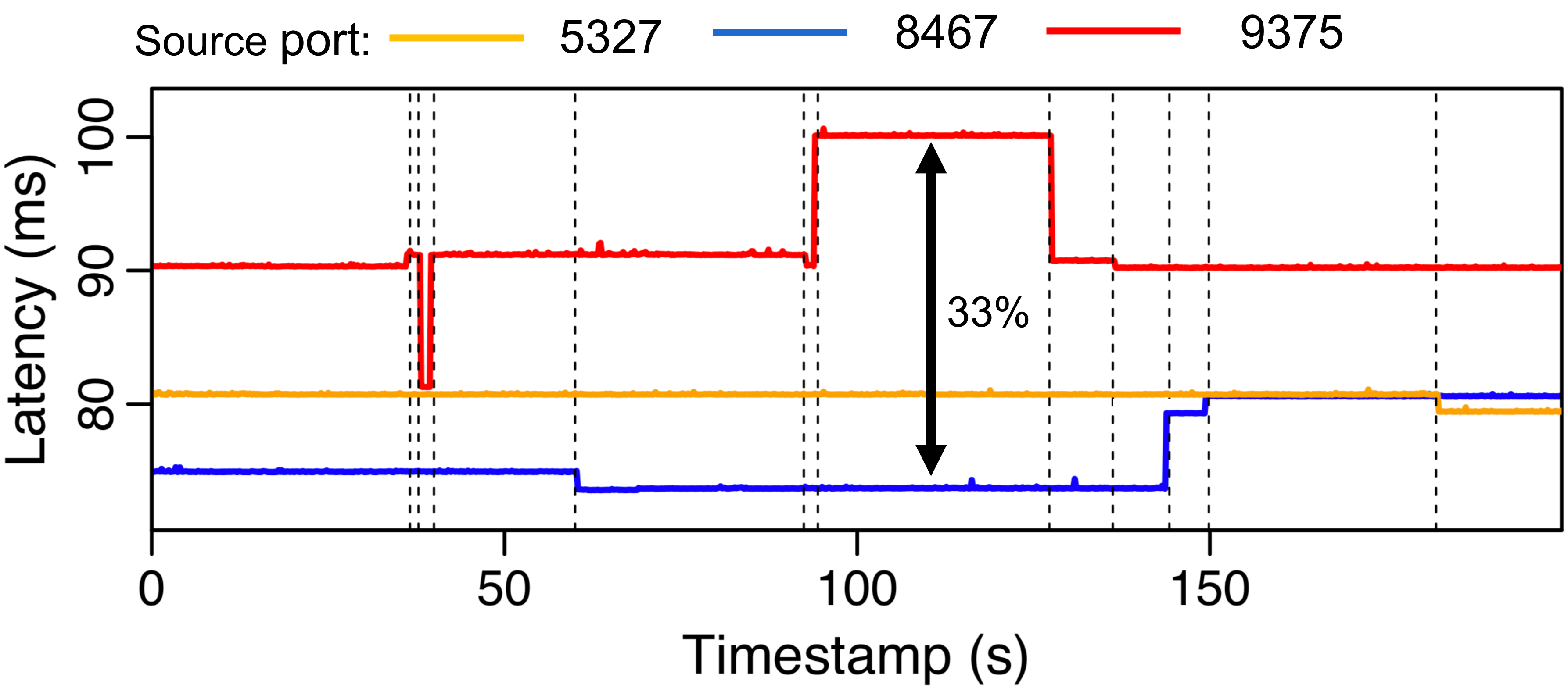


Insight #1: Flows experiences path changes at a second-level time scales

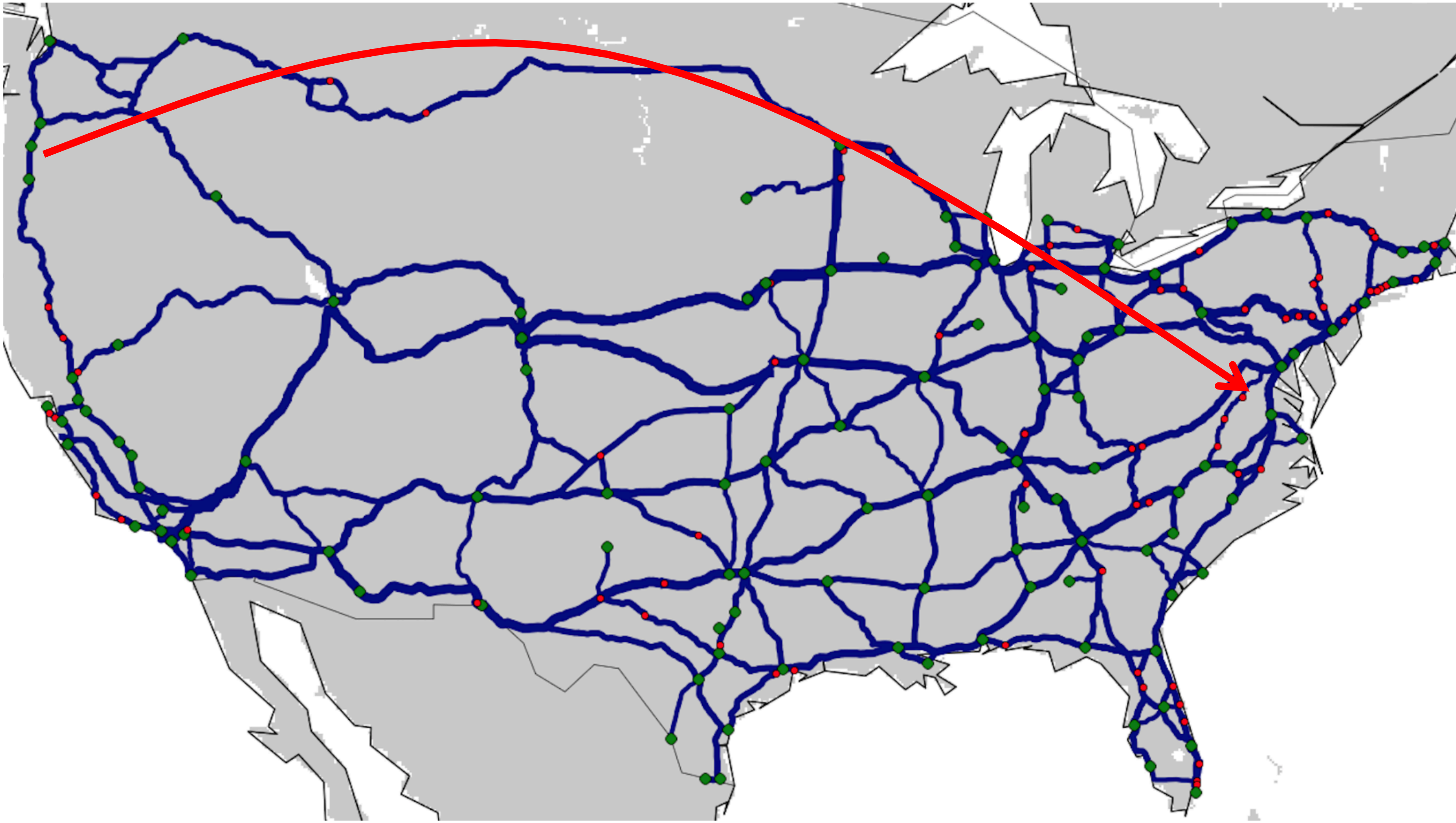


Insight #2:

Flows experience unfair performance



Why do we see so many different latency behaviours?



Traffic Engineering (TE) : balance flows of traffic in a network

Periodic TE re-optimization in cloud networks:

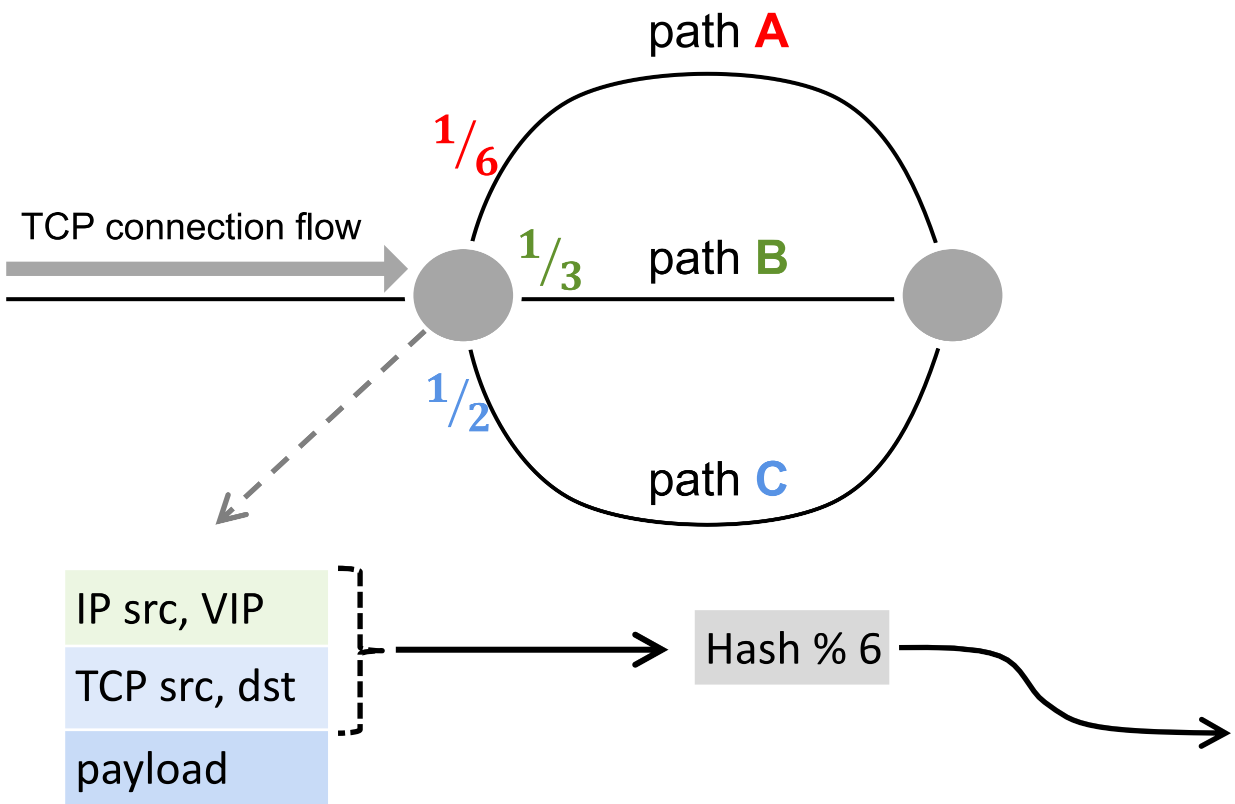
- 2011: MPLS autobandwidth in Microsoft WAN*
- today: (claimed) **SDN-based** TE at Amazon, Google, Microsoft

Two-phase TE:

- **compute** (multiple) **tunnels** among region pairs
- **compute** traffic **splitting ratios** among tunnels
  - typically hash-based implementation -> flow consistency
  - risk of **reshuffling flow packets** upon **splitting ratio update**!

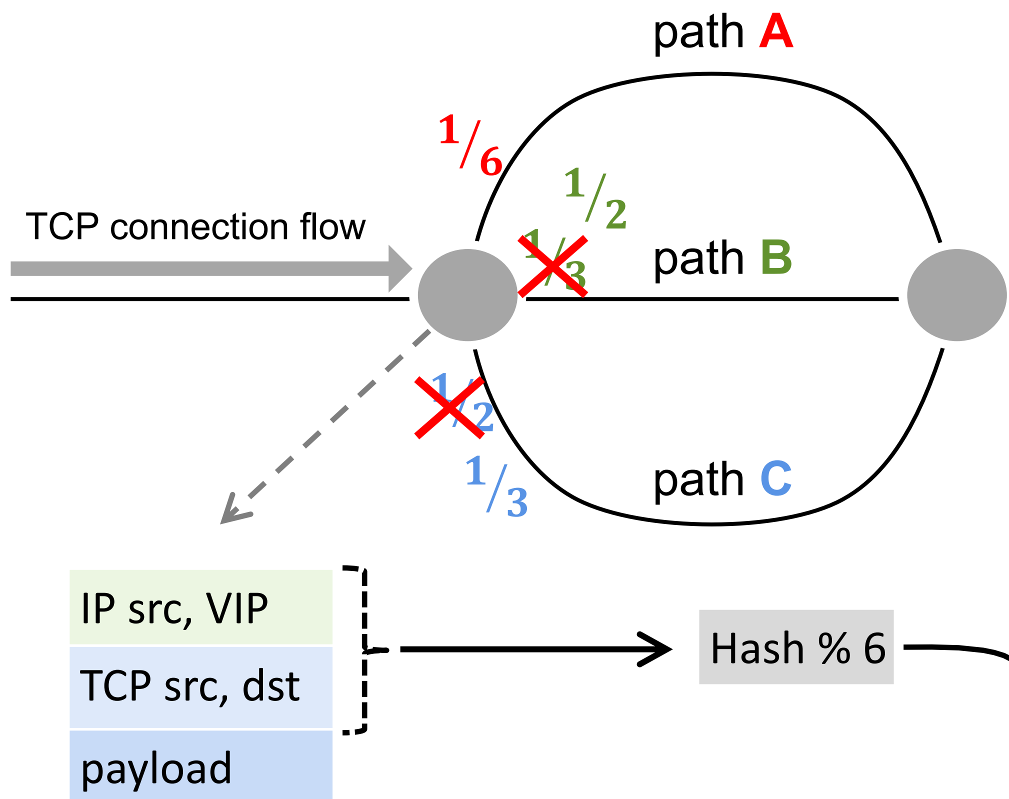
** A. Pathak et al "Latency Inflation with MPLS-based Traffic Engineering" IMC 2011*

Stateless hash-based traffic splitting



index	nexthop
0	A
1	B
2	B
3	C
4	C
5	C

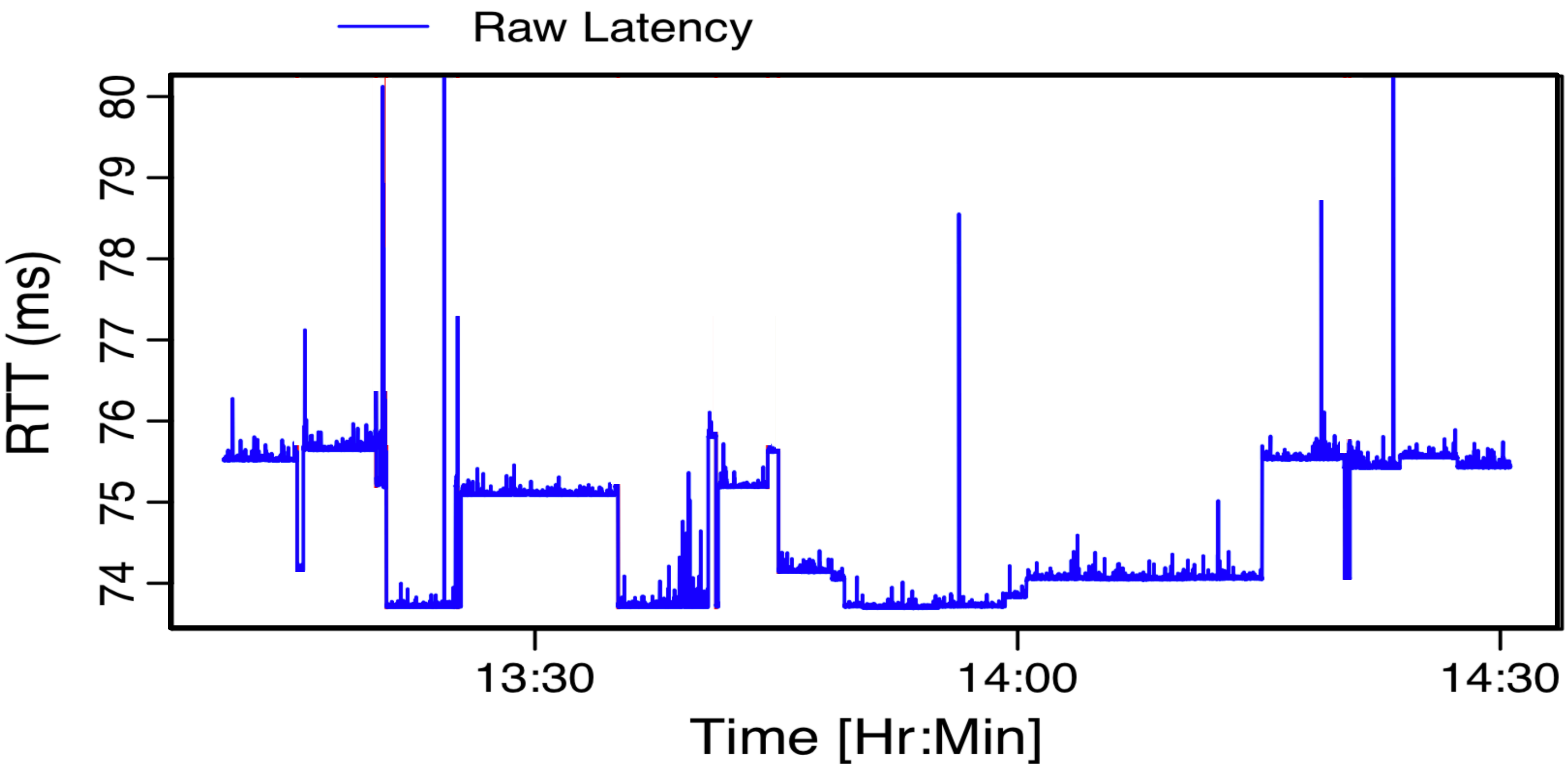
Stateless hash-based traffic splitting: packets belonging to the same connection may be **rerouted** when traffic splitting ratios are modified



index	nexthop
0	A
1	B
2	B
3	~~C~~ → B
4	C
5	C

Let’s open a new connection and dissect it:

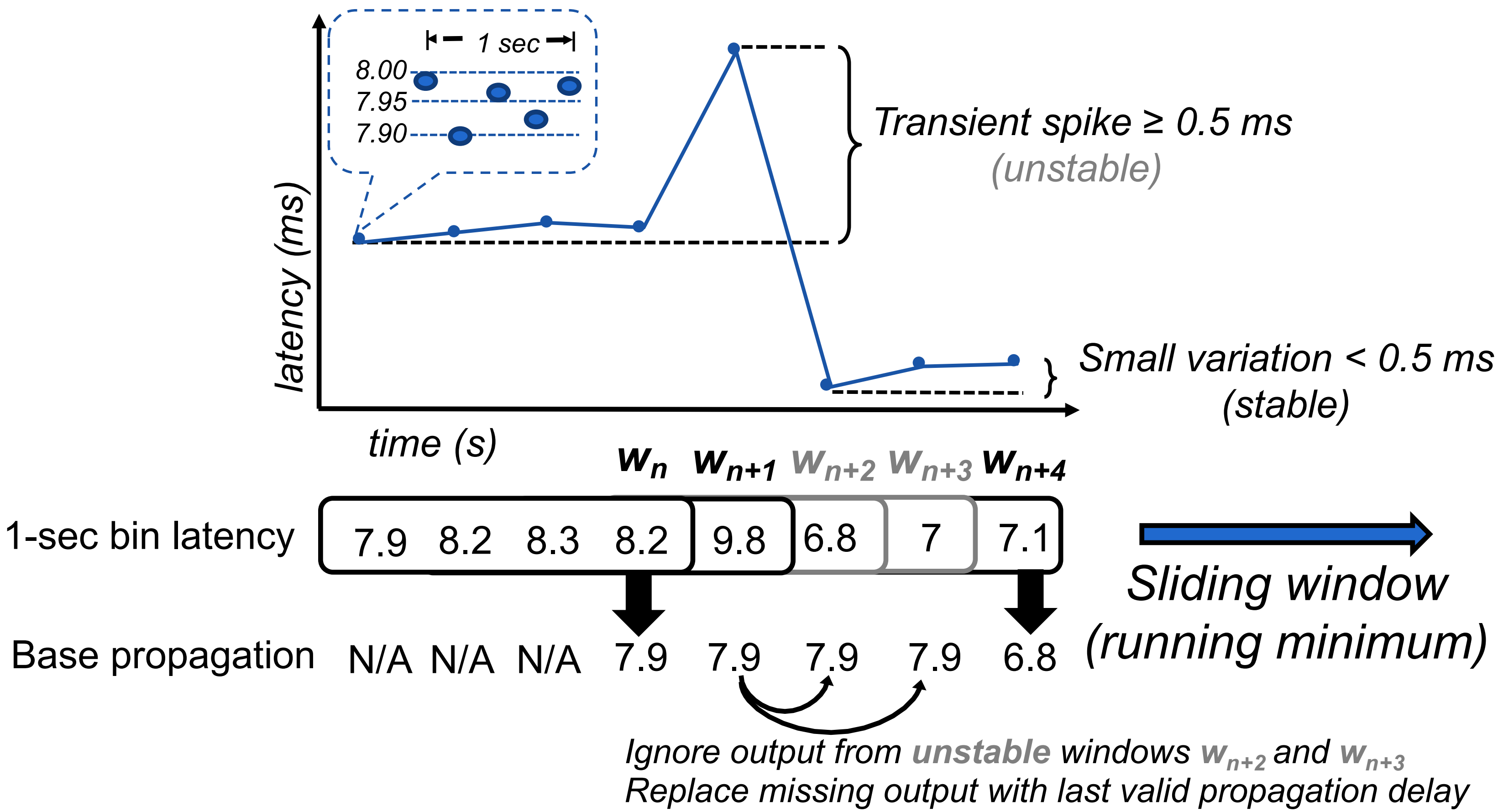
latency = base propagation latency + congestion



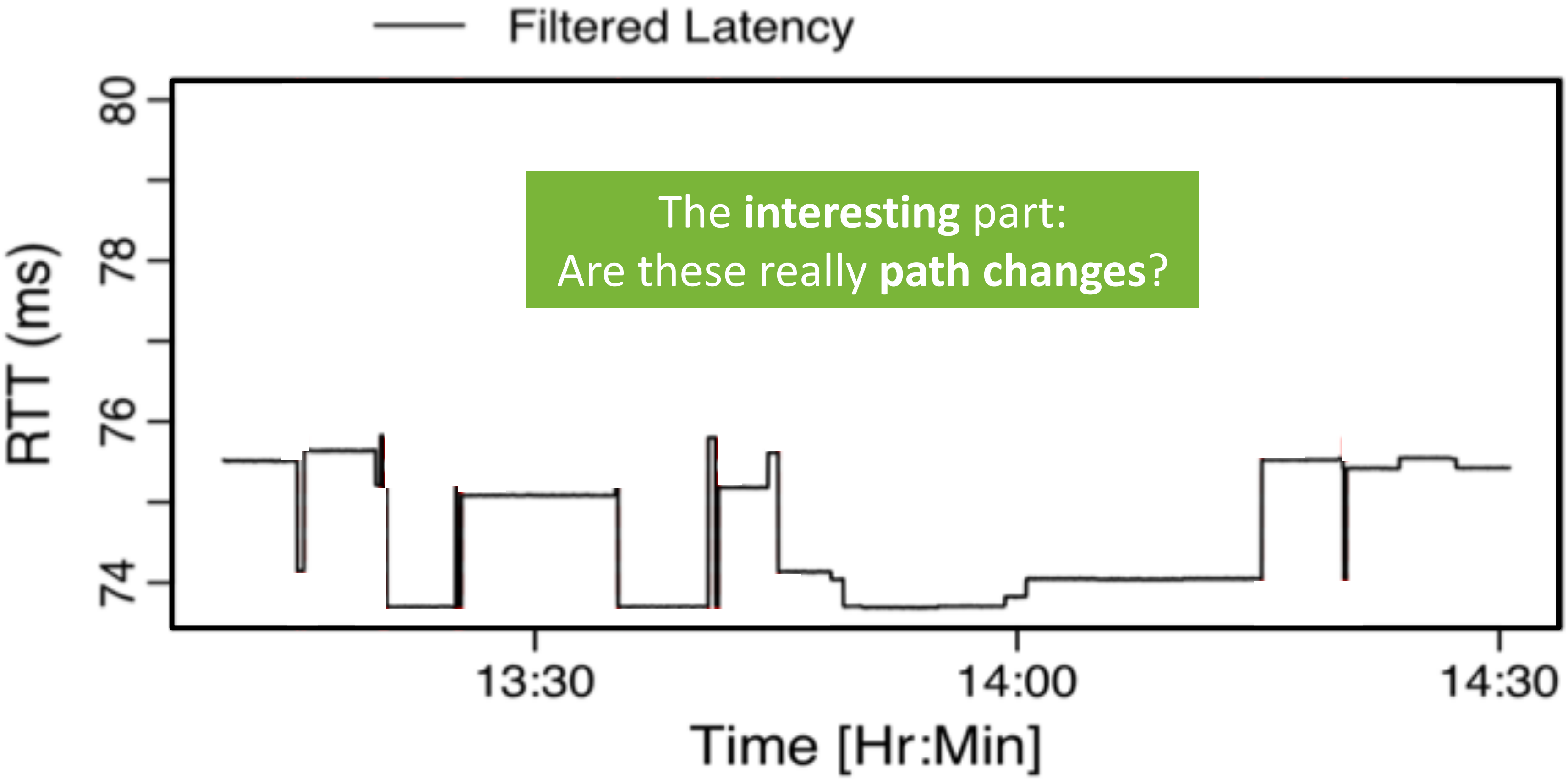
The technical yet necessary part (bear with me): Extracting base propagation latency

Rolling-minimum sliding window:

- stable if 3 out of 4 samples are within 0.5ms
- update base propagation latency only if stable AND >0.5ms change

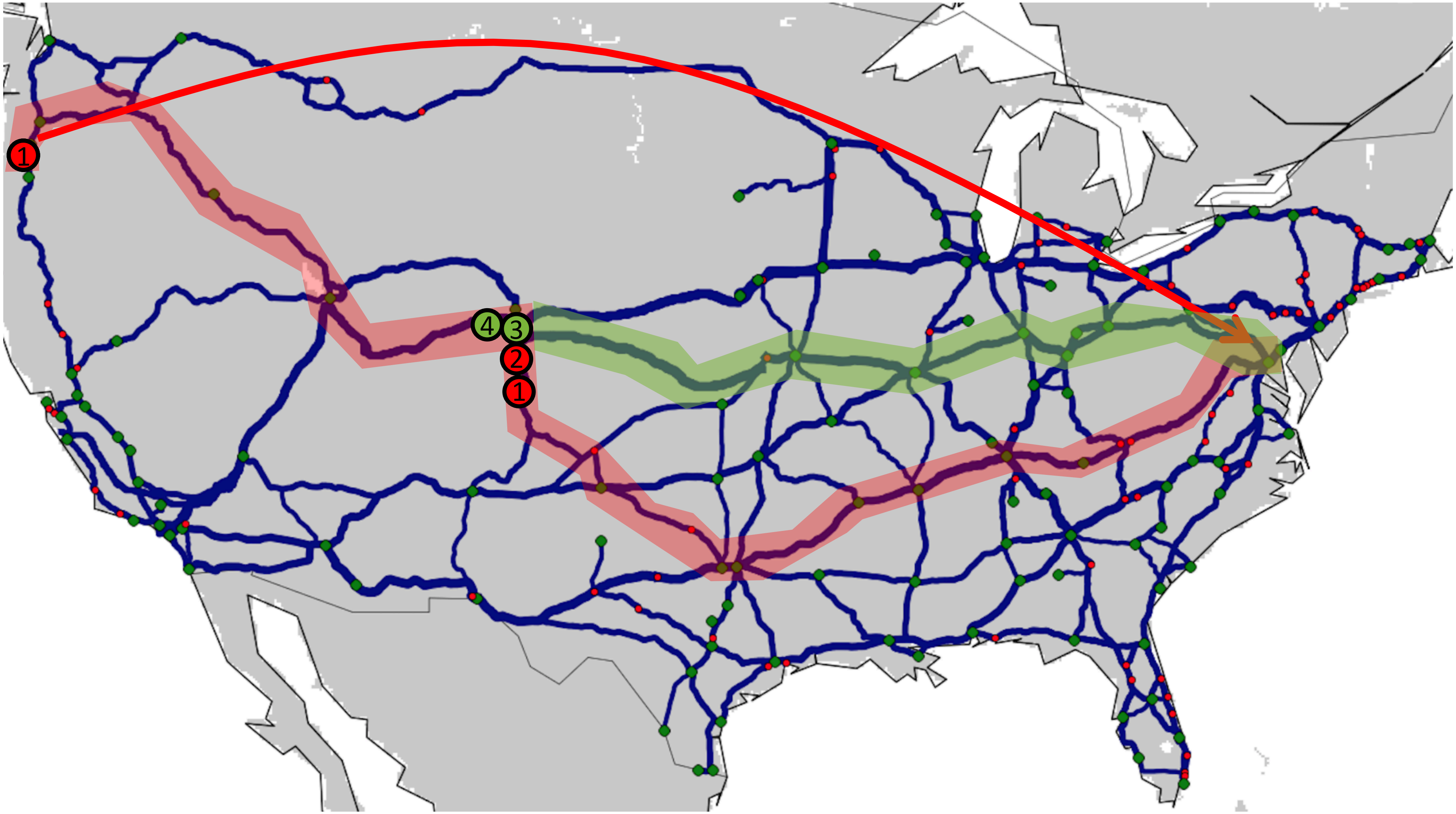


The filtered base propagation latency



The **interesting** part:
Are these really **path changes**?

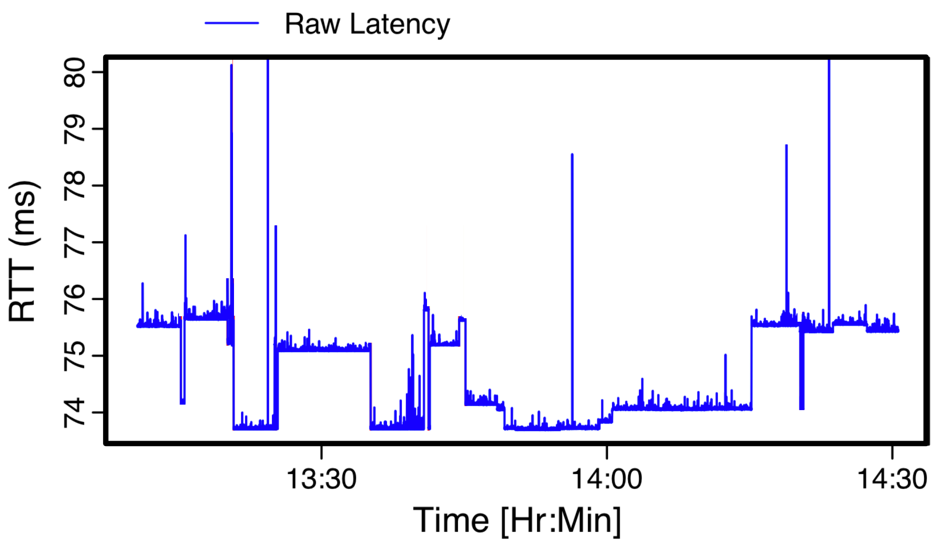
Key intuition: a path change from a high to a low latency path causes packet reordering

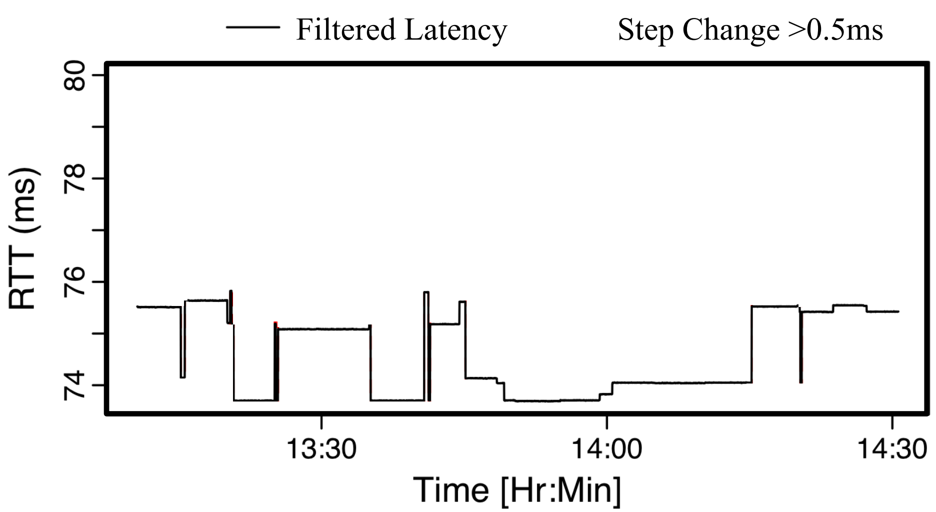


The path change detector is accurate and conservative

We **correlate** packet **reordering** events with propagation **latency**:

- zero false positives, *i.e.*, a latency decrease >0.5ms is a path change
- limited false negatives, *i.e.*, path changes within 0.5ms -> conservative approach
- henceforth, latency means base propagation latency

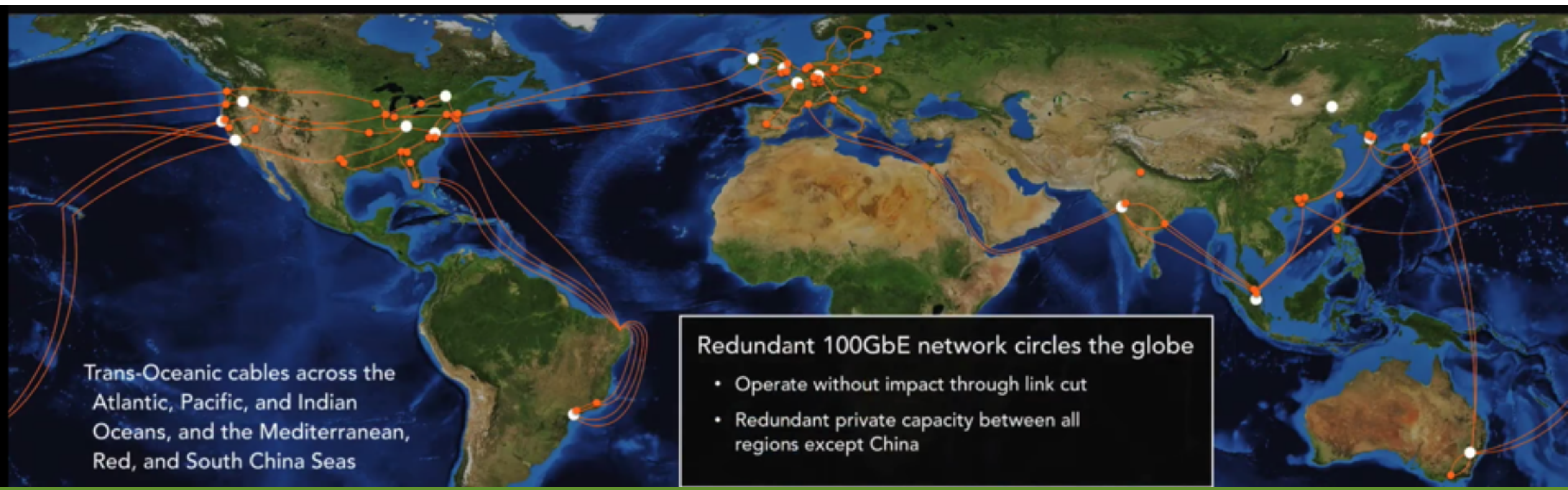




We can now **detect path changes**

We set up an **extensive measurement** study of AWS

Config.	Macro-scale	Micro-scale
# of DC Pairs	120	4
# of Flows	512	512
Probing Rate	10 probes every 30s	5 probes/s
Flow Generation	Dynamic (every 30s)	Static
Duration	2 days	1 week
Ping Mechanism	Raw Sockets	TCP Ping

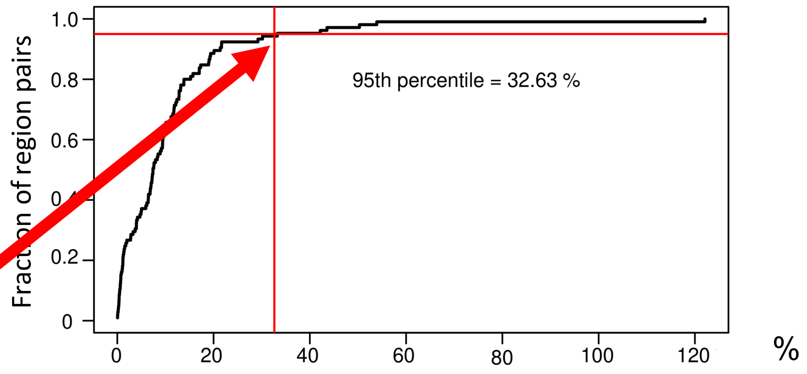


Picture credits: AWS re:Invent 2016: Tuesday Night Live with James Hamilton

Some AWS regions have **high latency variations** across different flows (i.e., different source ports)

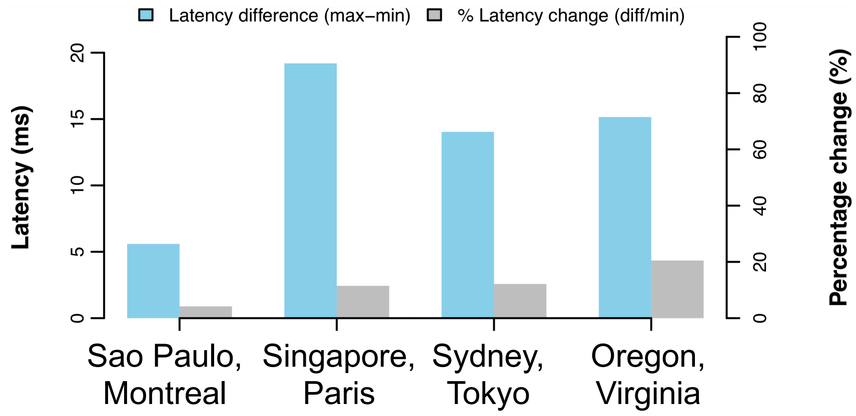
We looked at all the 120 region pairs in AWS:

- **Latency inflation**: max-latency/min-latency
- >**32% latency inflation for 5% of region pairs**

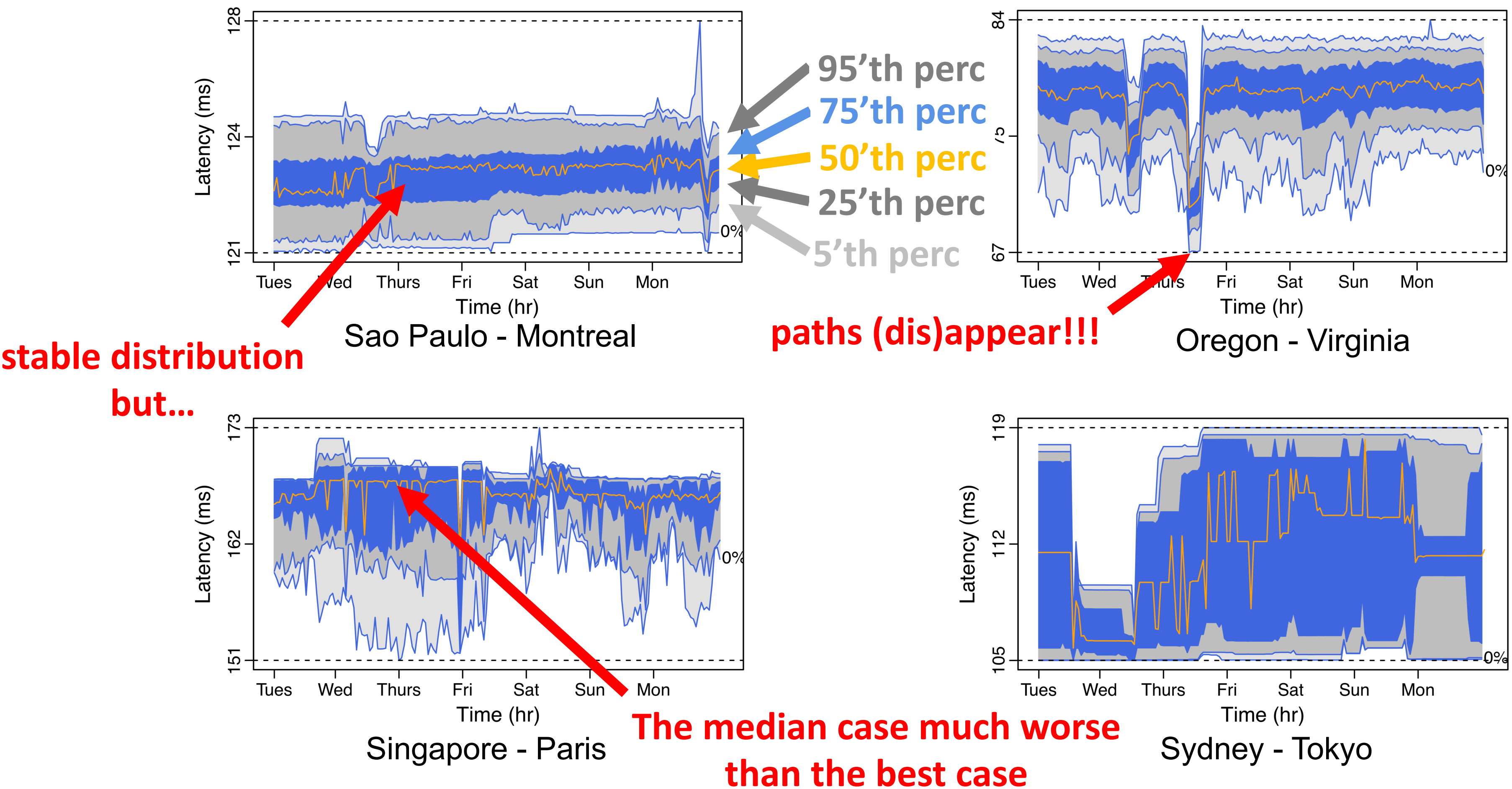


We zoomed into four AWS region pairs for the micro-scale experiments

- neither the worst nor the best pairs



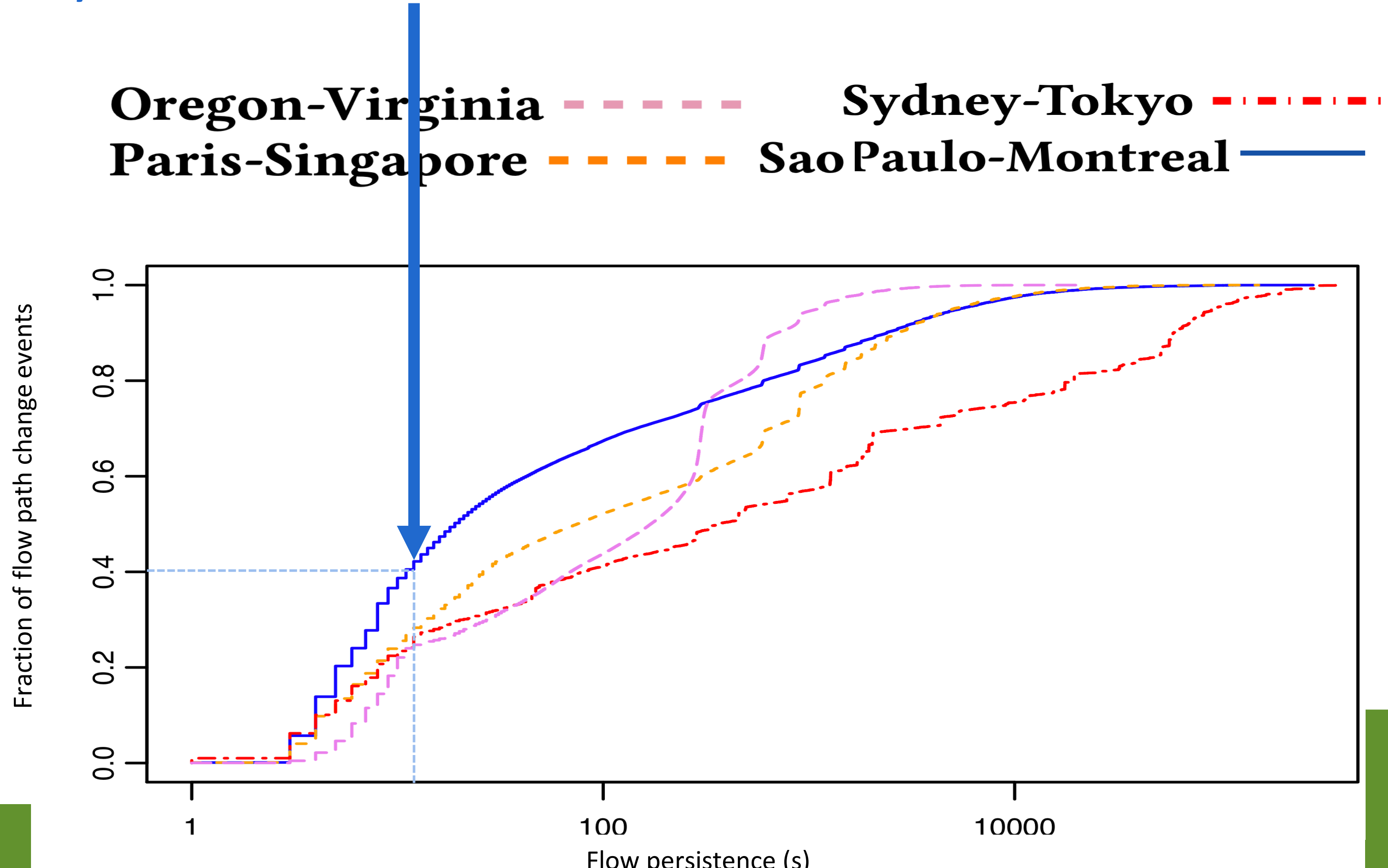
The flow-latency spectrum over time for four different pairs of regions



Stable distribution but… are the flows routed always on the same path? Not necessarily!

Consider all events when a flow changes path:

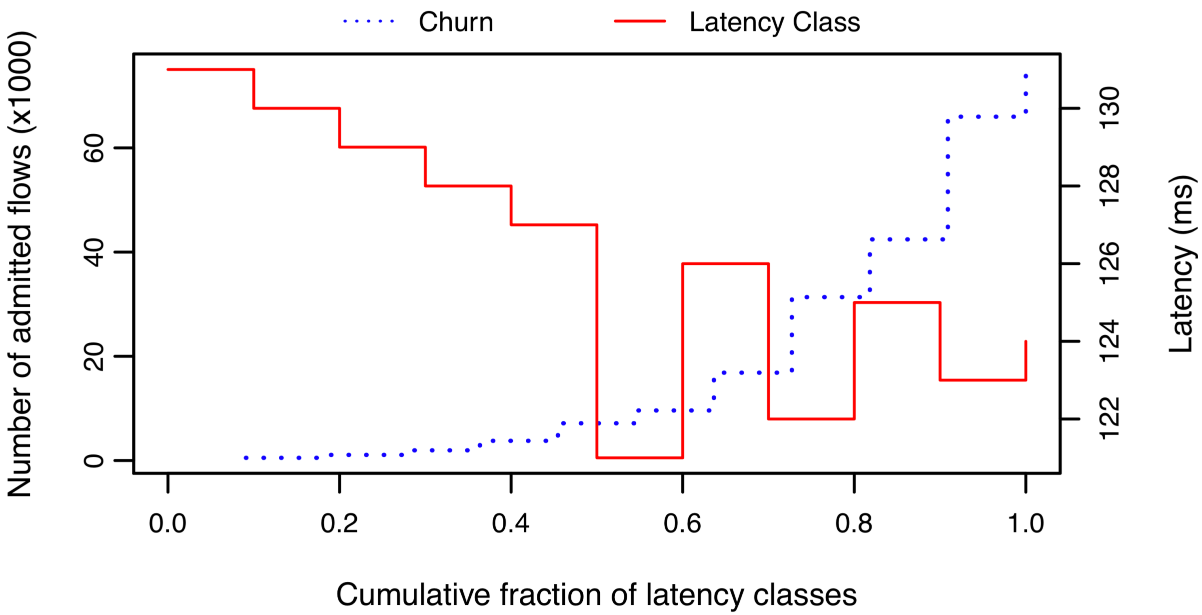
- **40% of the cases, the flow moves away within 10 seconds**



Flows routes on low-latency paths are more likely to experience a path change

We counted the number of flows moved away from a path per flow latency class (Oregon-Virginia):

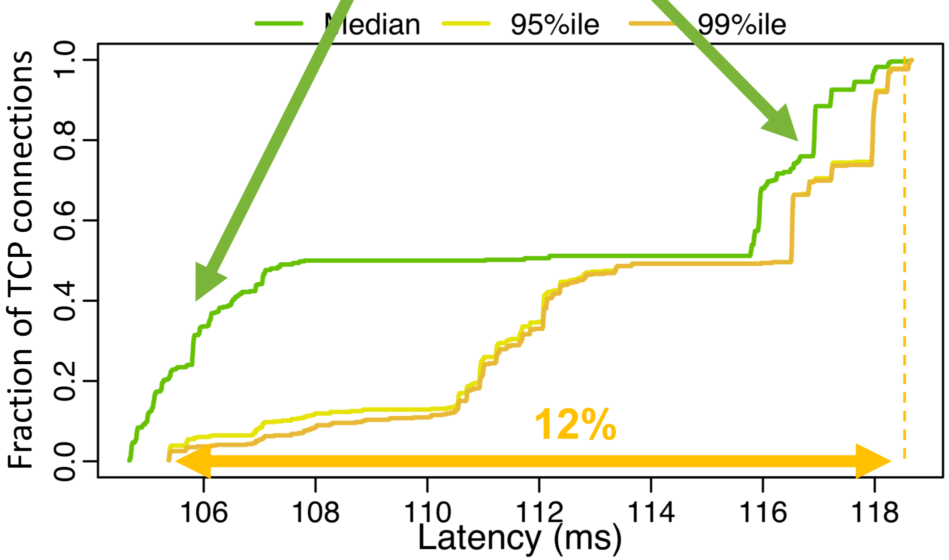
- number of moved flows is **inversely proportional** to flow latency
- likely constrained **shortest-path-based TE**



Measuring the level of ”unfairness” among flows

We measure the flow latency of each flow in 20-sec interval bins

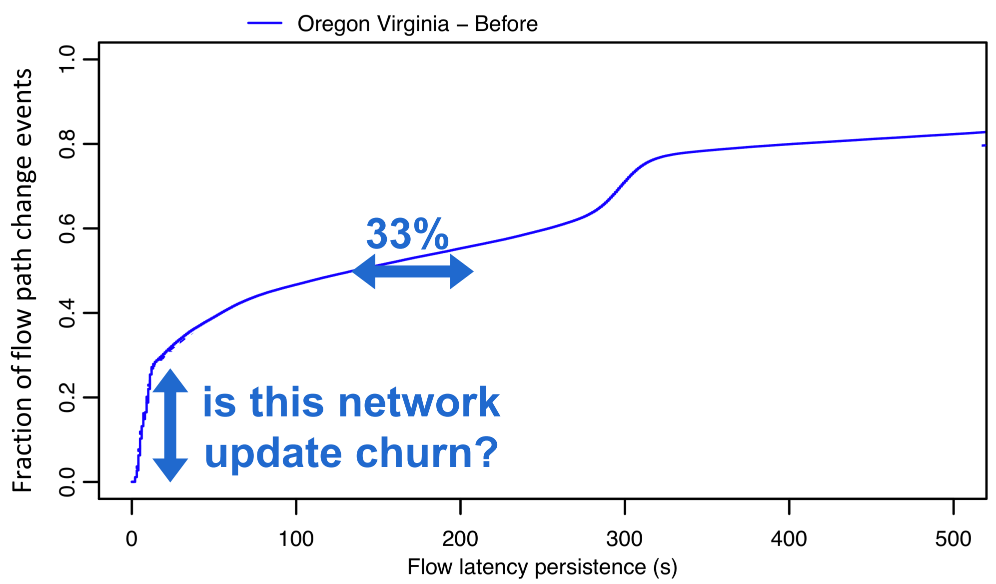
- we plot the 50’th, 95’th, 99’th latency percentiles
- **bimodal behaviour at 50’th**
- **12% difference for all percentiles**



Measuring instabilities during ”low” or ”high” season

Large cloud networks tune the “aggressiveness” of their TE mechanisms according to the users’ load

We performed again the same measurements during “low” season, expecting to see less path changes

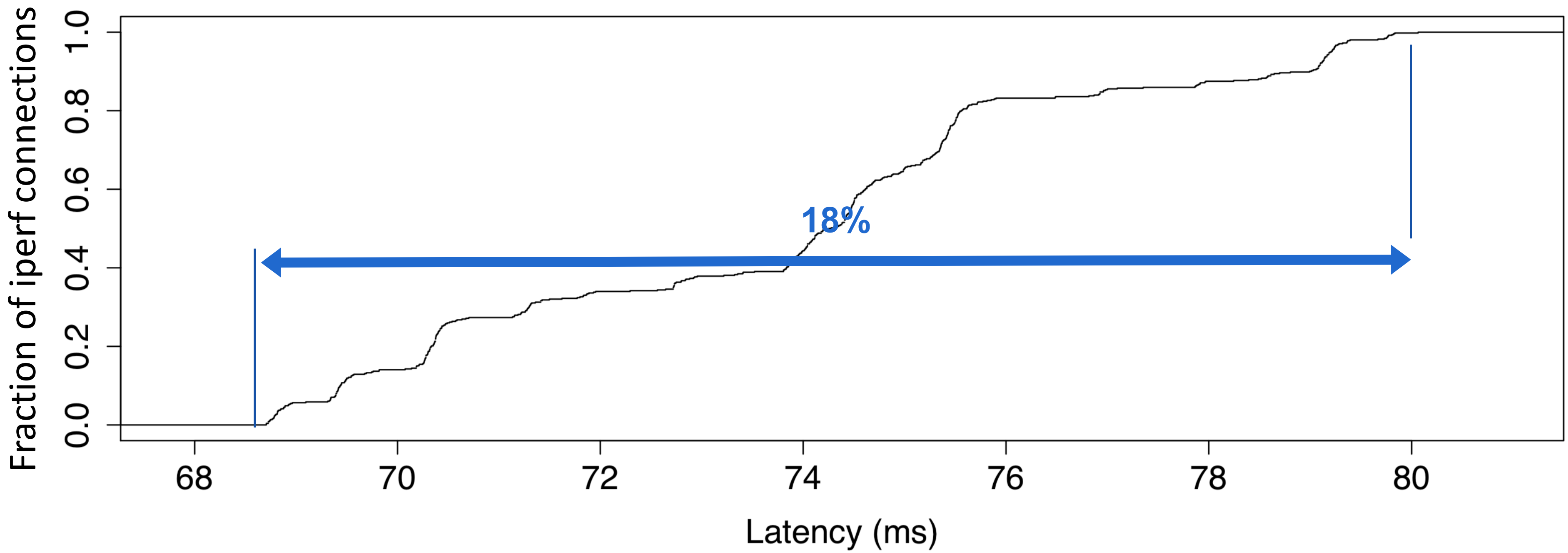


Moving 1K of data can take 18% more time when the flow is mapped to a high latency path

We generated a large number of iperf tests from Oregon (US west coast) to Virginia (US east coast)

We moved 1K of data in each iperf test

We did not observe any packet retransmission



Conclusions and takeaway

We measured the largest worldwide cloud backbone network (AWS):

- **Insight #1**: **very reactive TE**, especially across some regions
- **Insight #2**: flows experience **unfair** treatment
  - both in Round Trip Time and in the number of path changes

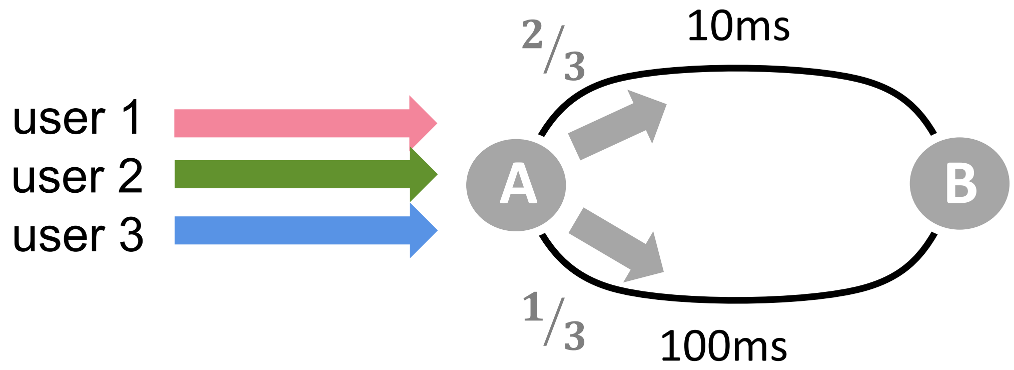
Far-reaching implications on the cloud ←→ tenants interaction:

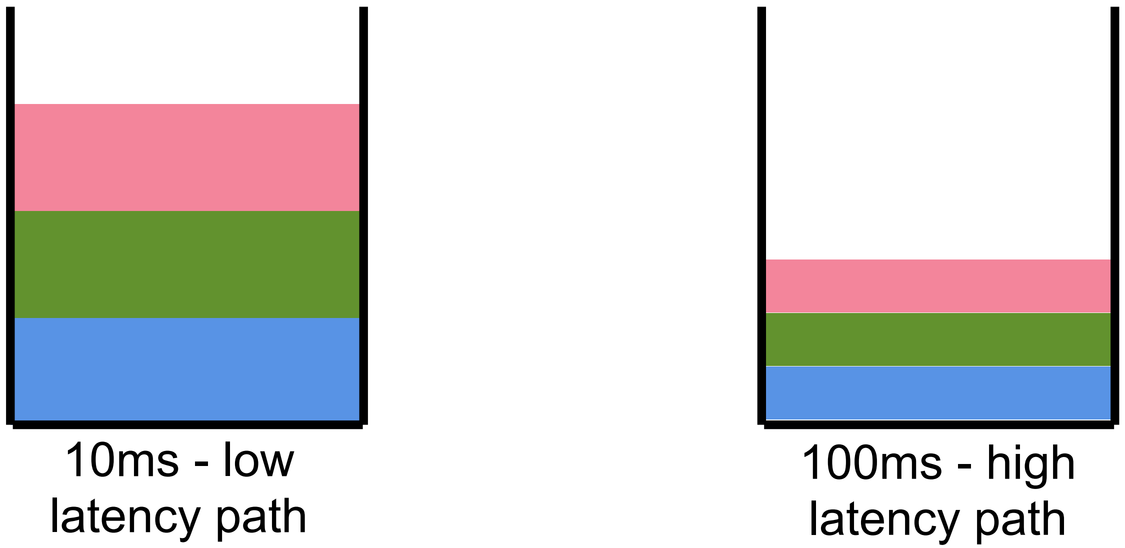
- **congestion control:** TCP Cubic suffers from packet reordering
- **latency**: low-latency geo-distributed emerging applications require **low** and **deterministic** latencies
- **selfish-routing**: application developers can force multiplexing of traffic on shortest paths, ultimately defeating cloud TE operation

Thank you!

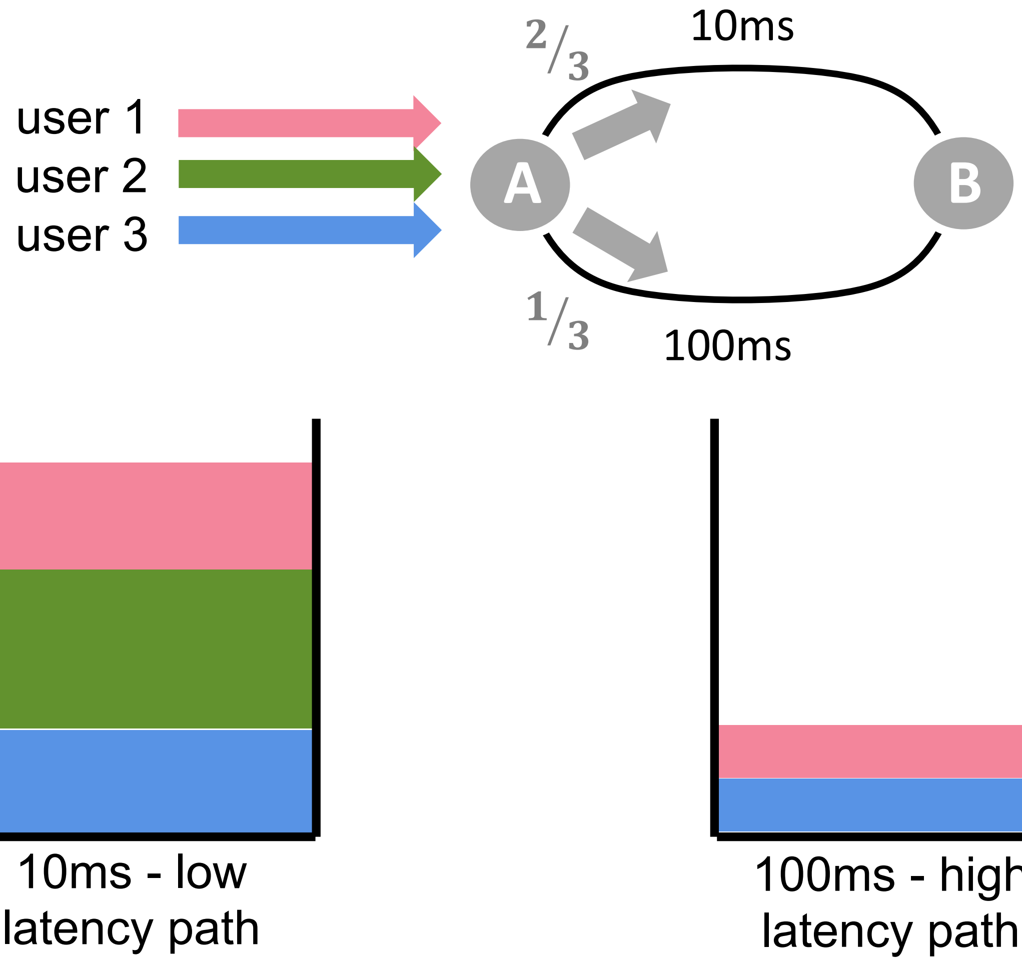
Marco Chiesa < mchiesa@kth.se >

Selfish routing: giving control to the cloud tenants may lead to an unhealthy situation

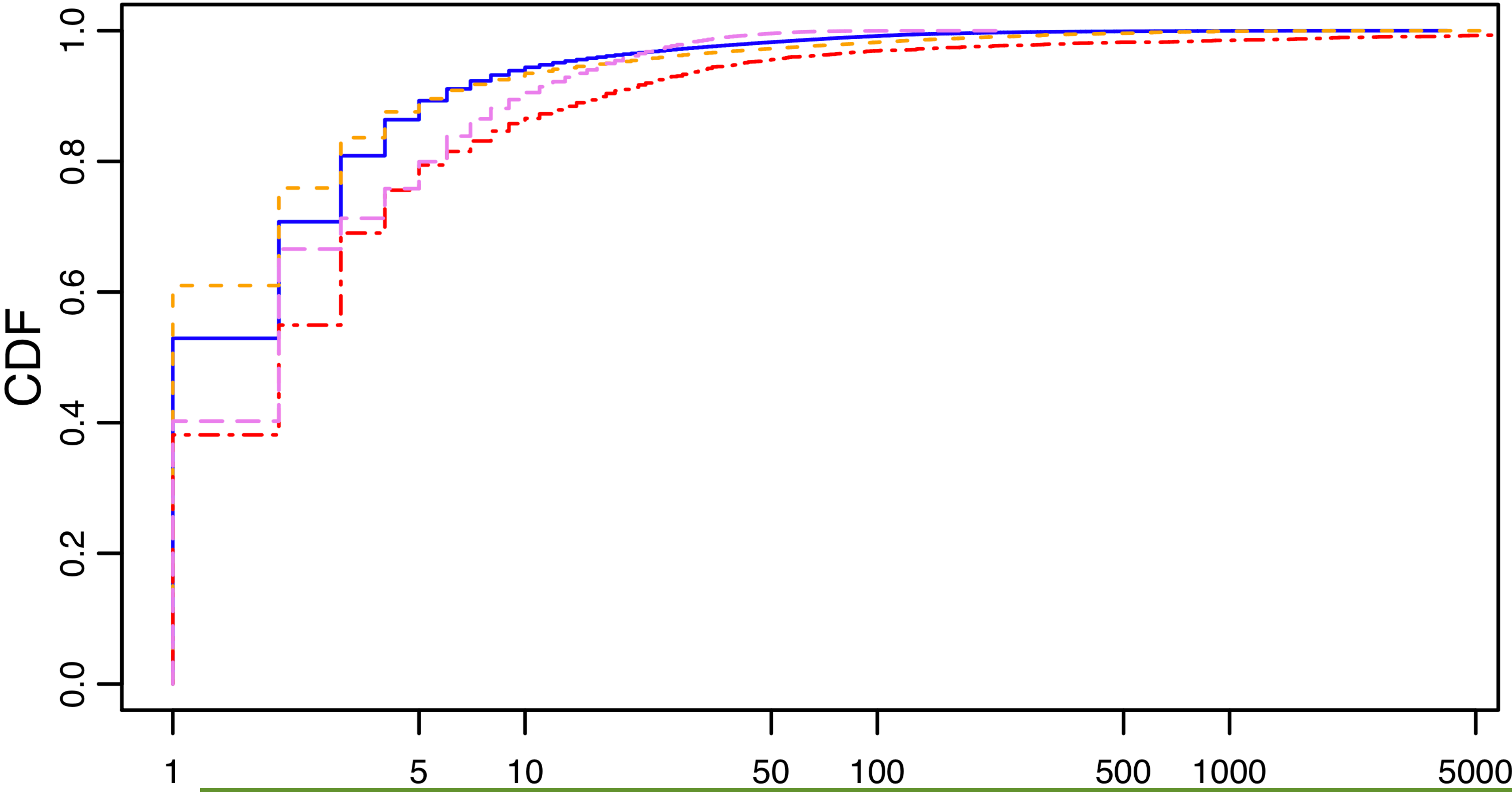




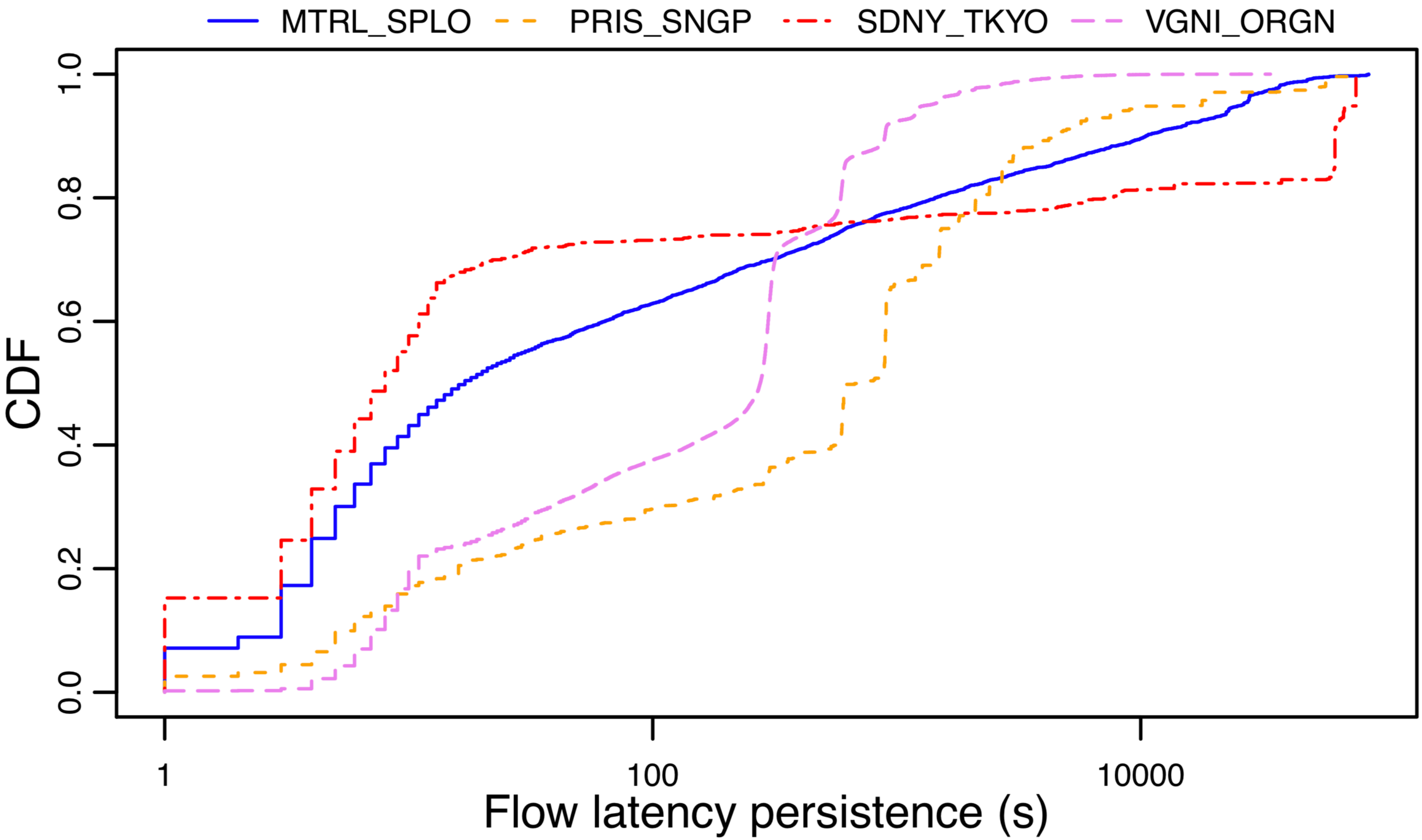
Selfish routing: giving control to the cloud tenants may lead to an unhealthy situation



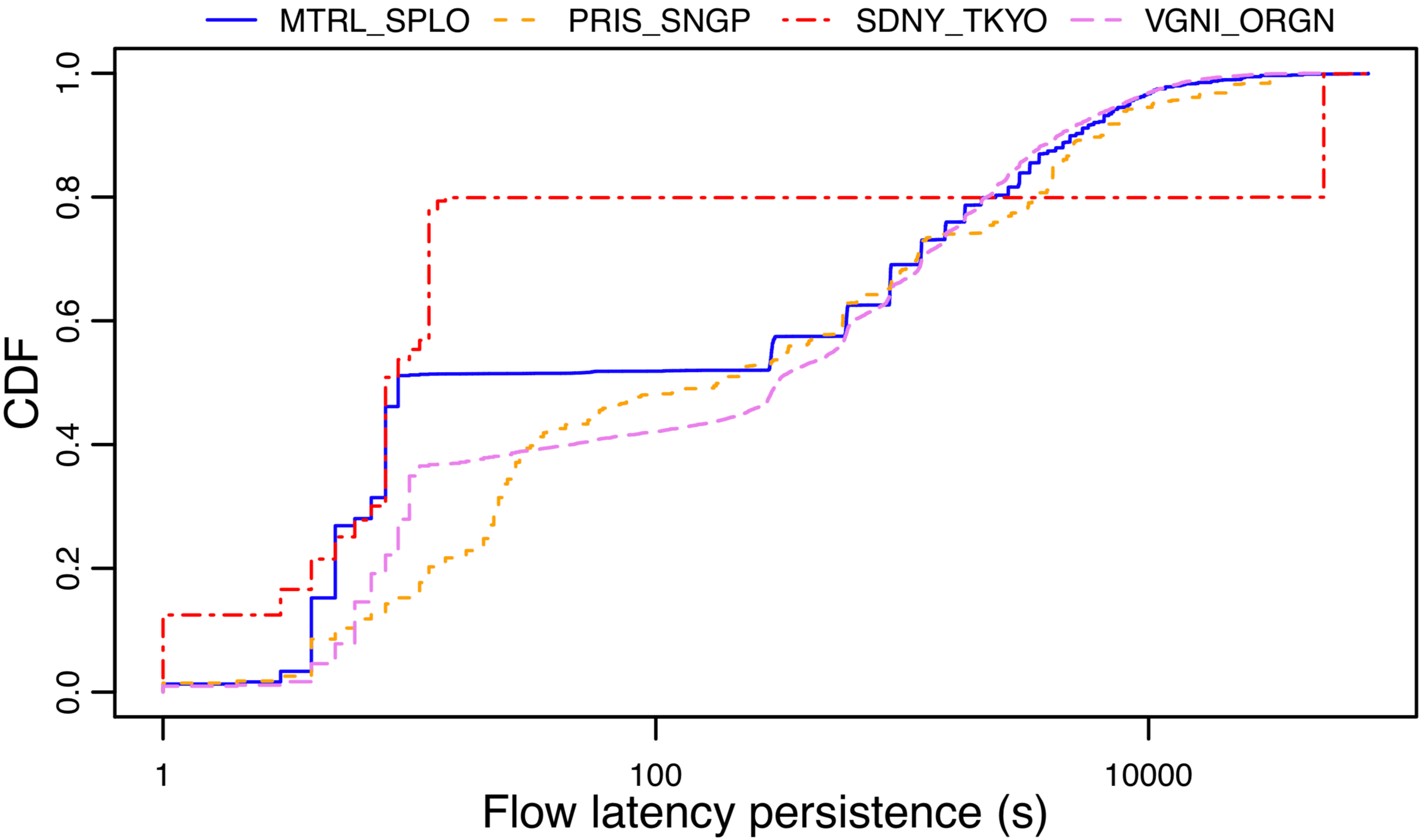
Flow path change interarrival time



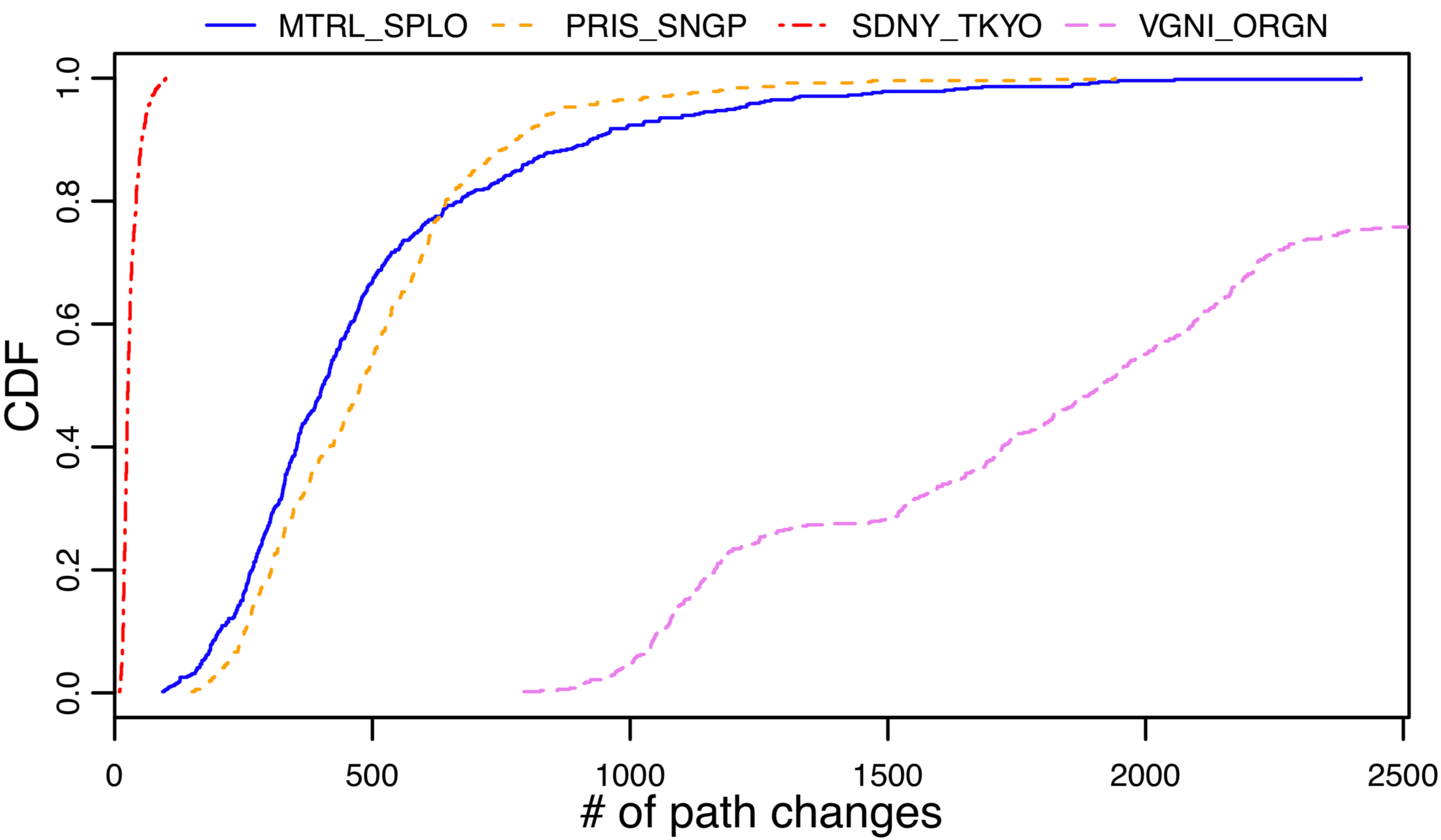
Flow persistence – forward path



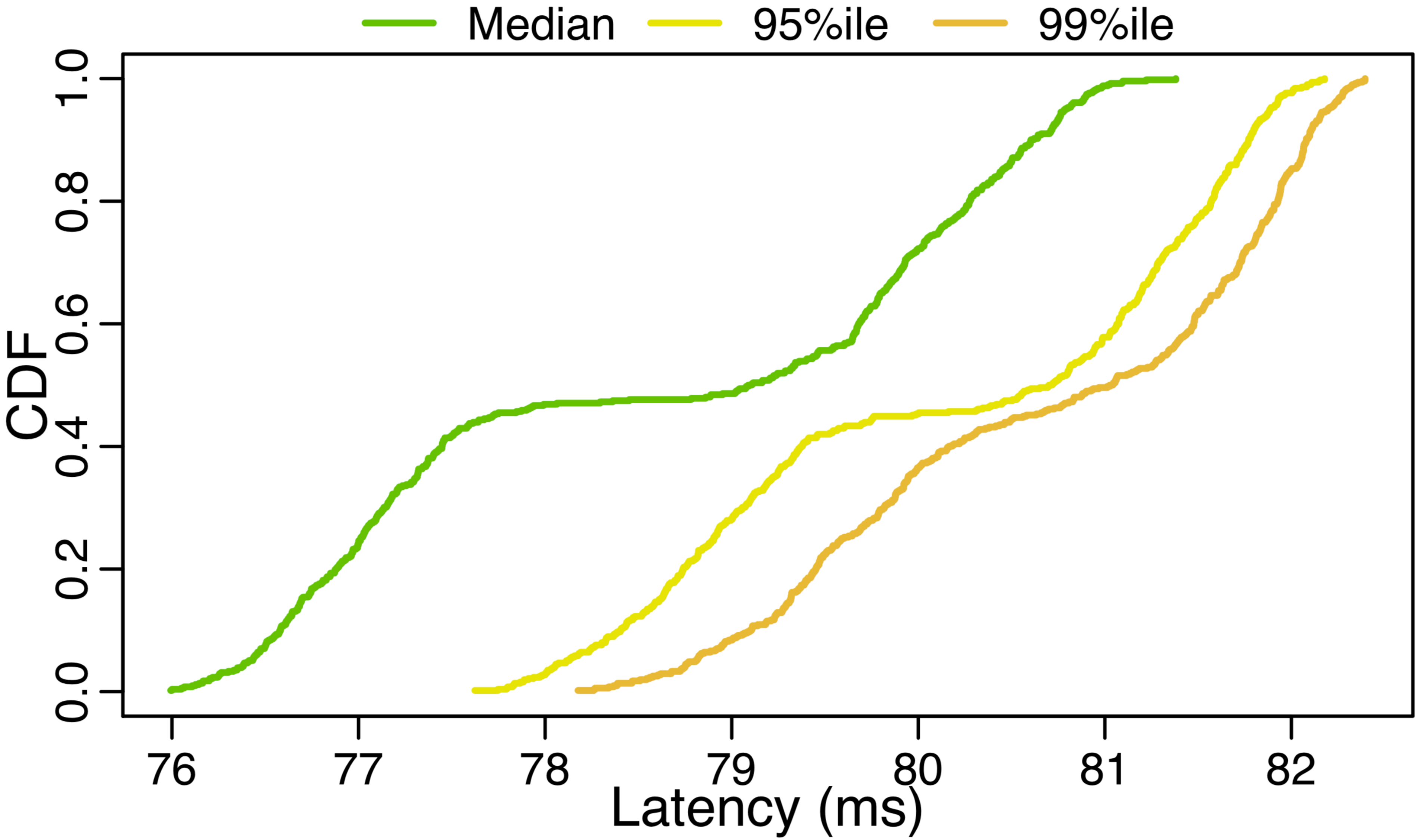
Flow persistence –reverse path



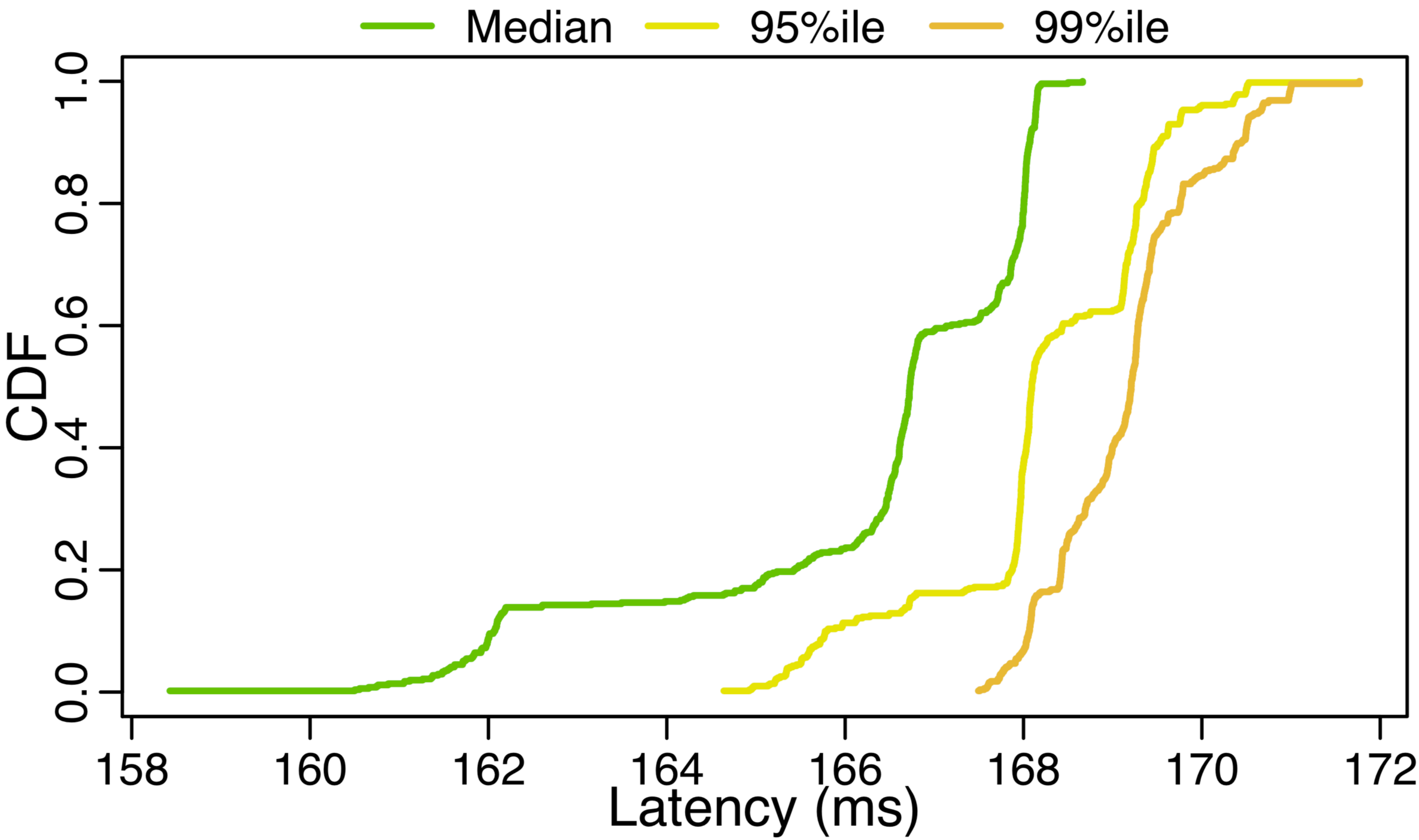
Traffic Engineering (TE) performed to (re)balance flows of traffic in a network



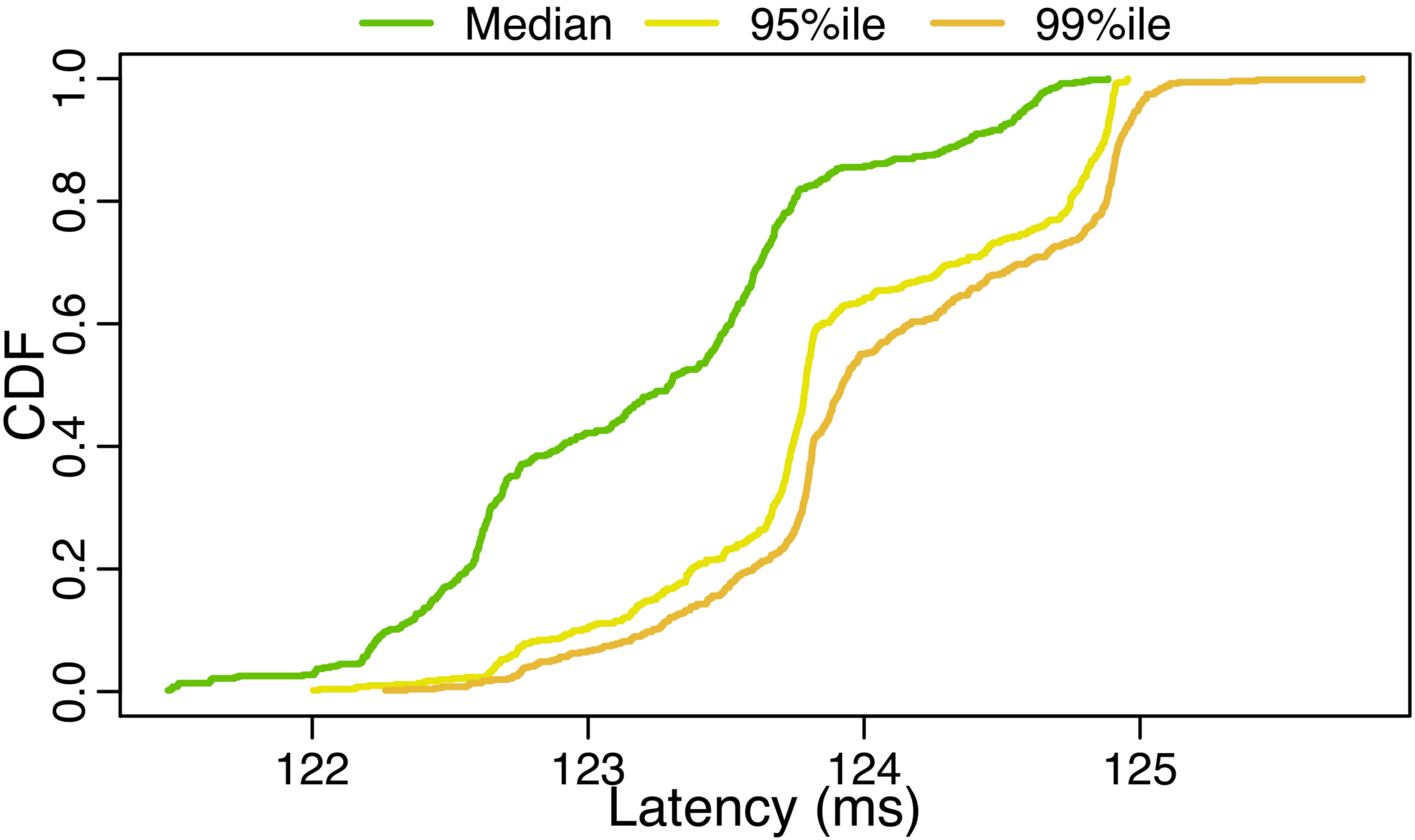
Traffic Engineering (TE) performed to (re)balance flows of traffic in a network



Traffic Engineering (TE) performed to (re)balance flows of traffic in a network



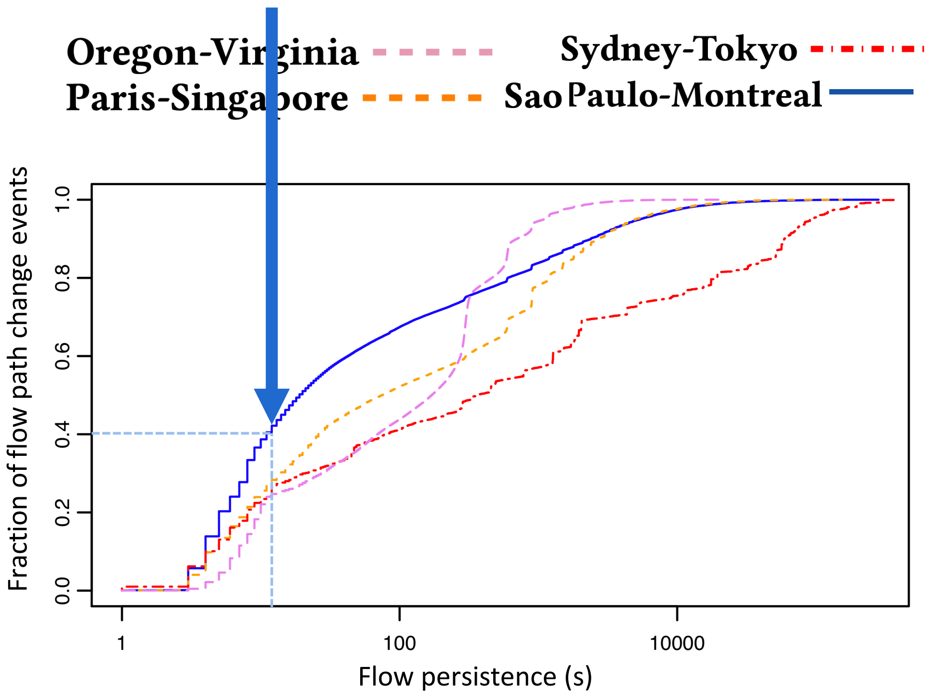
Traffic Engineering (TE) performed to (re)balance flows of traffic in a network



Stable distribution but… are the flows routed always on the same path? Not necessarily!

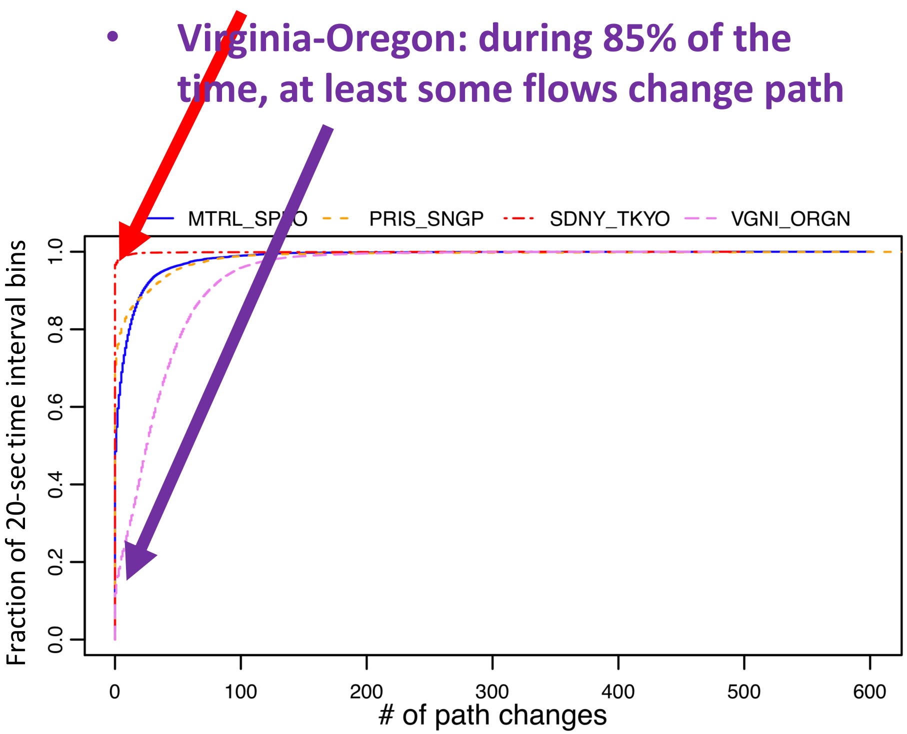
Consider all events when a flow changes path:

- 40% of the cases, the flow moves away within 10 seconds



We count #path-changes each 20 seconds

- Sidney-Tokyo: during 1% of the time, we observe all the path changes
- Virginia-Oregon: during 85% of the time, at least some flows change path



This talk in a nutshell

We measured the largest worldwide cloud backbone network (AWS):

- **Insight #1**: **very reactive TE** performed across datacenters
- **Insight #2**: tenants' flows experience **unfair** treatment
  - both in Round Trip Times and number of path changes

Far-reaching implications on how clouds and tenants interact:

- **congestion control:** TCP Cubic suffers from packet reordering
- **latency**: low-latency geo-distributed emerging applications require **low** and **deterministic** latencies
- **selfish-routing**: application developers can force multiplexing of traffic on low-latency paths, ultimately hindering cloud TE operation