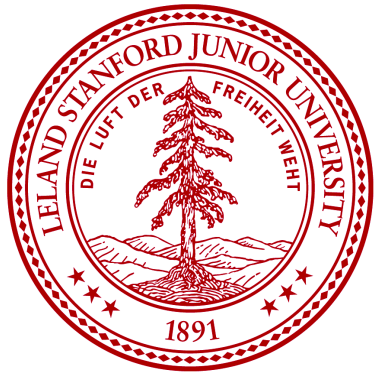


Libra: Divide and Conquer to Verify Forwarding Tables in Huge Networks

James Hongyi Zeng

with Shidong Zhang, Fei Ye, Vimalkumar Jeyakumar, Mickey Ju

Junda Liu, Nick McKeown, Amin Vahdat

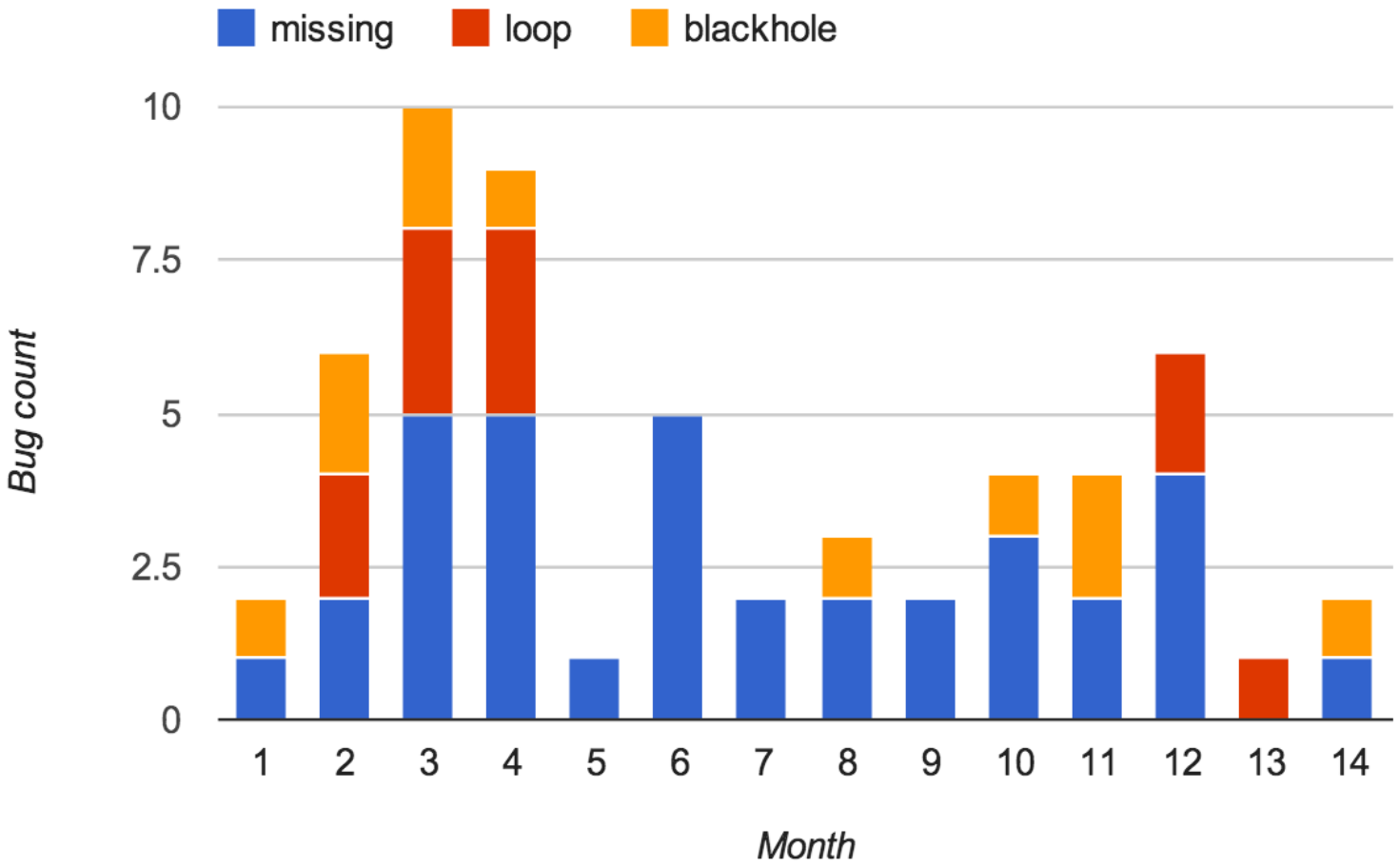
NSDI, April 2nd, 2014

“My dear friend Copperfield,” said Mr. Micawber, “accidents will occur in the best-regulated families.”

-- Charles Dickens


Best Regulated Families: Data Centers
- Homogeneous machines/switches
- Pure IP forwarding
- Few security concerns
- No “silly” human users
- Full control to all devices
Google | google.com/datacenters

Data Plane Tickets in a Google Data Center



Why getting networks right is hard?

Complex interaction

- Between multiple protocols on a switch.
- Between state on different switches.

Multiple uncoordinated writers of state.

Operators cannot…

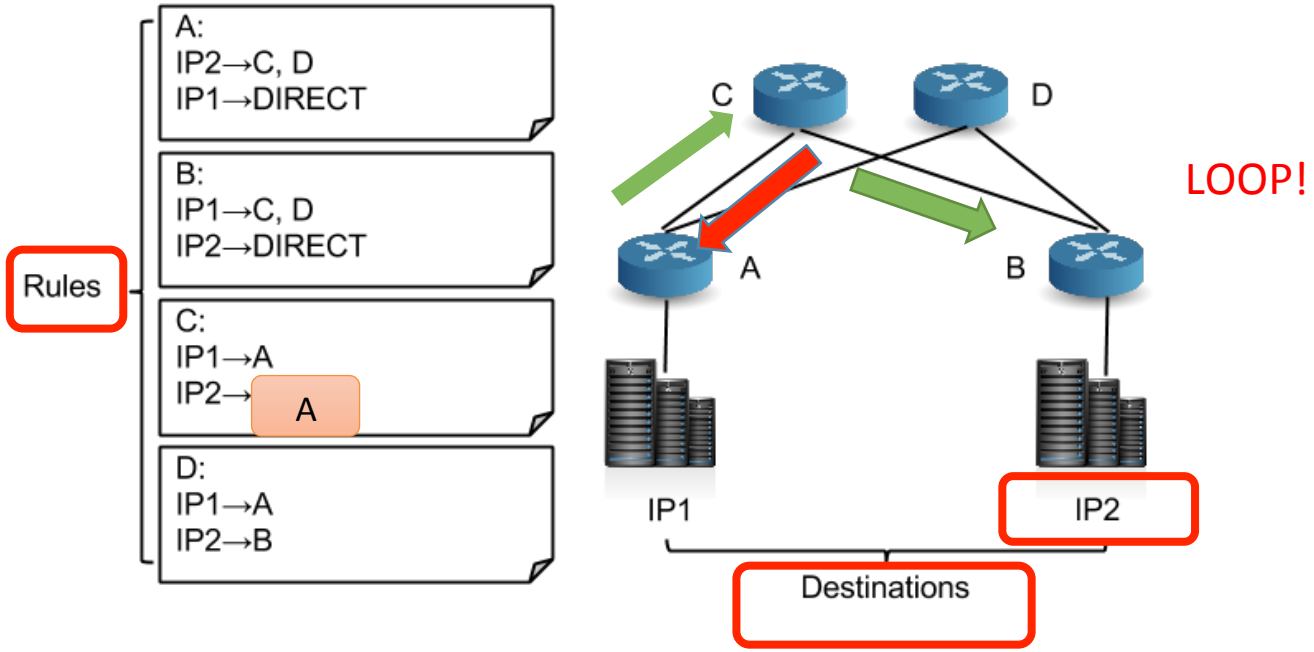
- Observe all state.
- Control all state.

One solution: Static Data Plane Verification

Static Data Plane Verification

Goal: Verifying data plane's logical correctness

- Example failures: loops, blackholes, partitions
- Inputs: destinations, rules, topology
- Independent of control plane

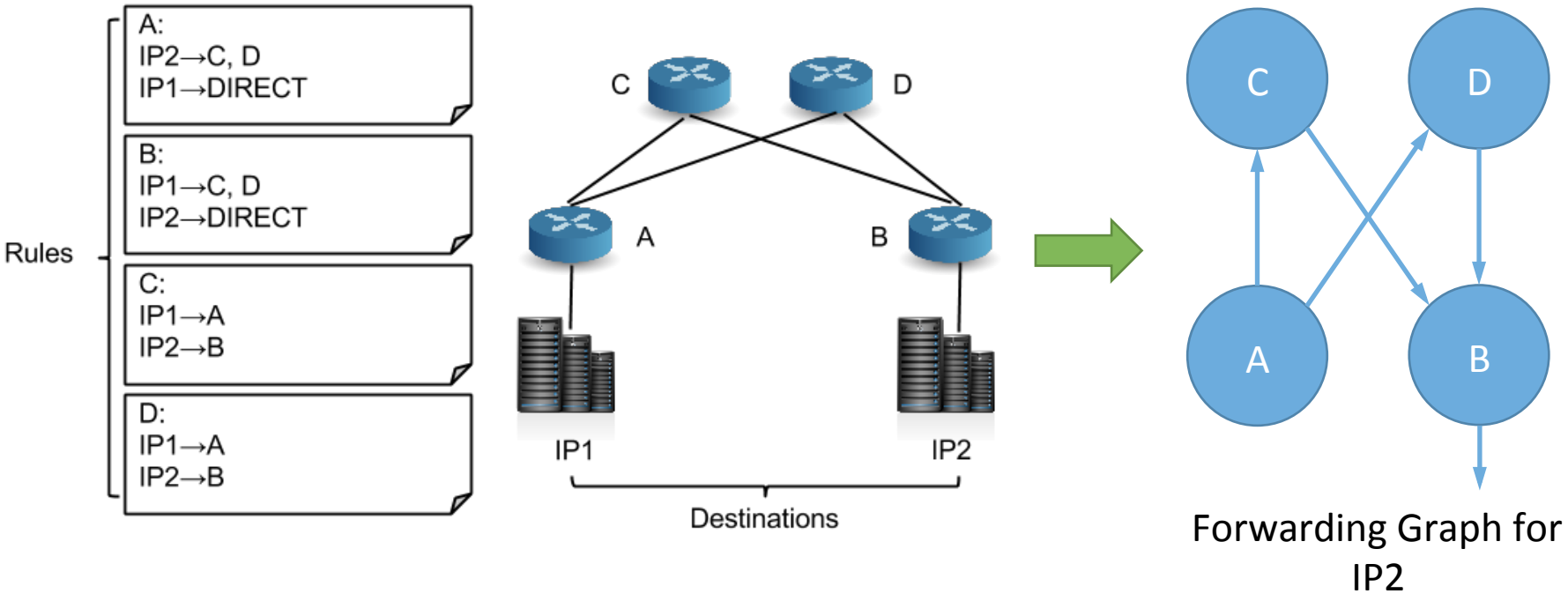


Two Challenges of Data Plane Verification in Data Centers

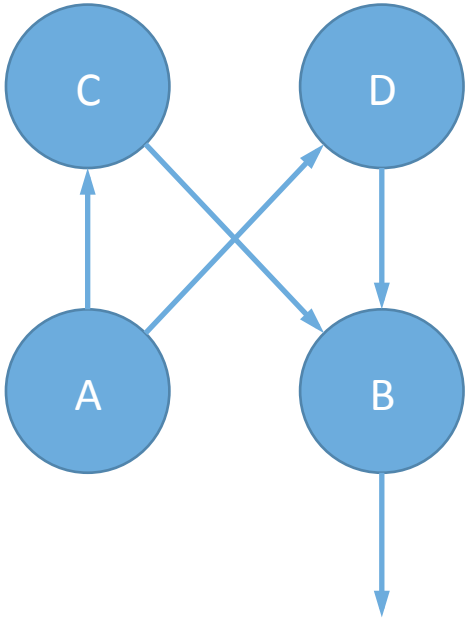
- Constant Rule Changes
  - Link up/down, route recalculation, new policy, etc.
  - Careless snapshot of forwarding tables may result in false positives

- Large Scale
  - 10,000s of switches, 1,000,000s of rules
  - How to finish the verification within a reasonable time?

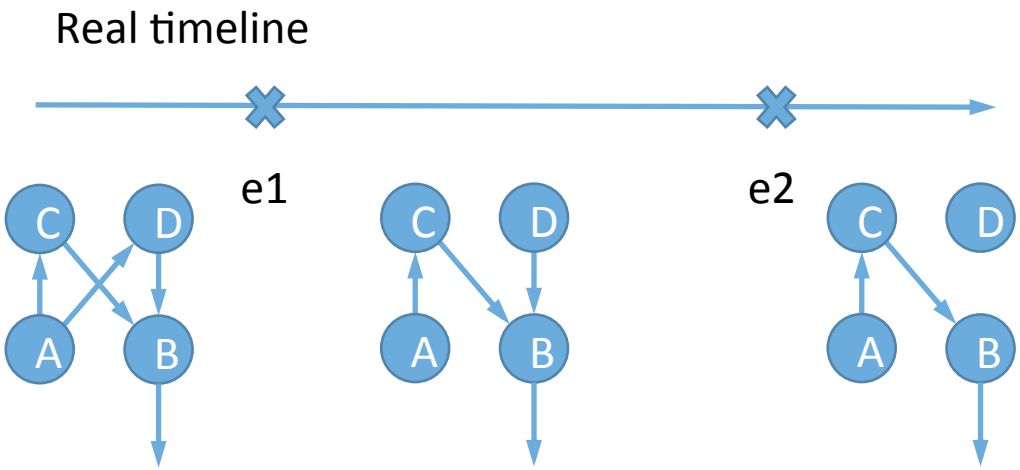
Forwarding Graph



The Snapshot Problem



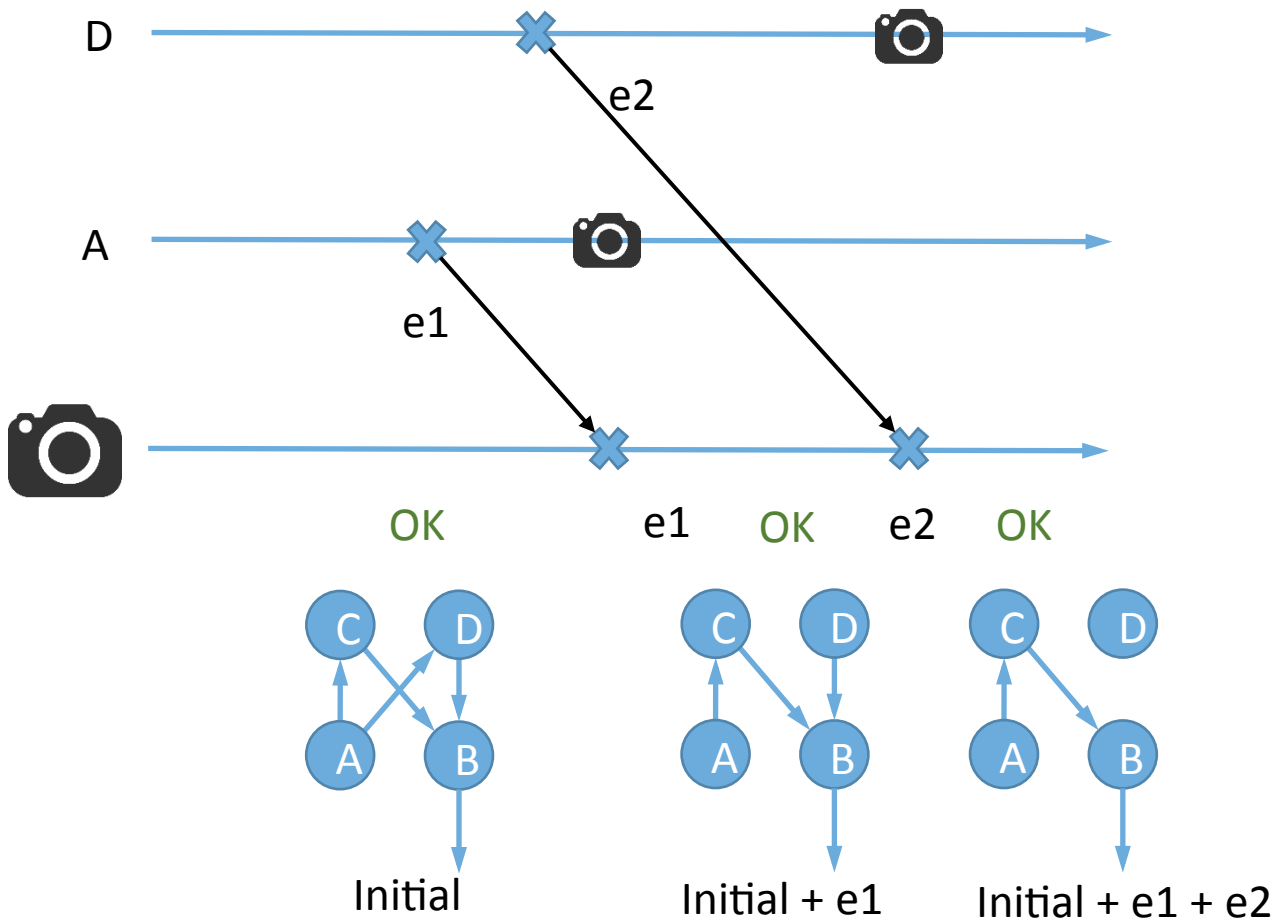
e1) A deletes rule A→D
e2) D deletes rule D→B



The Snapshot Problem

e1) A deletes rule A→D

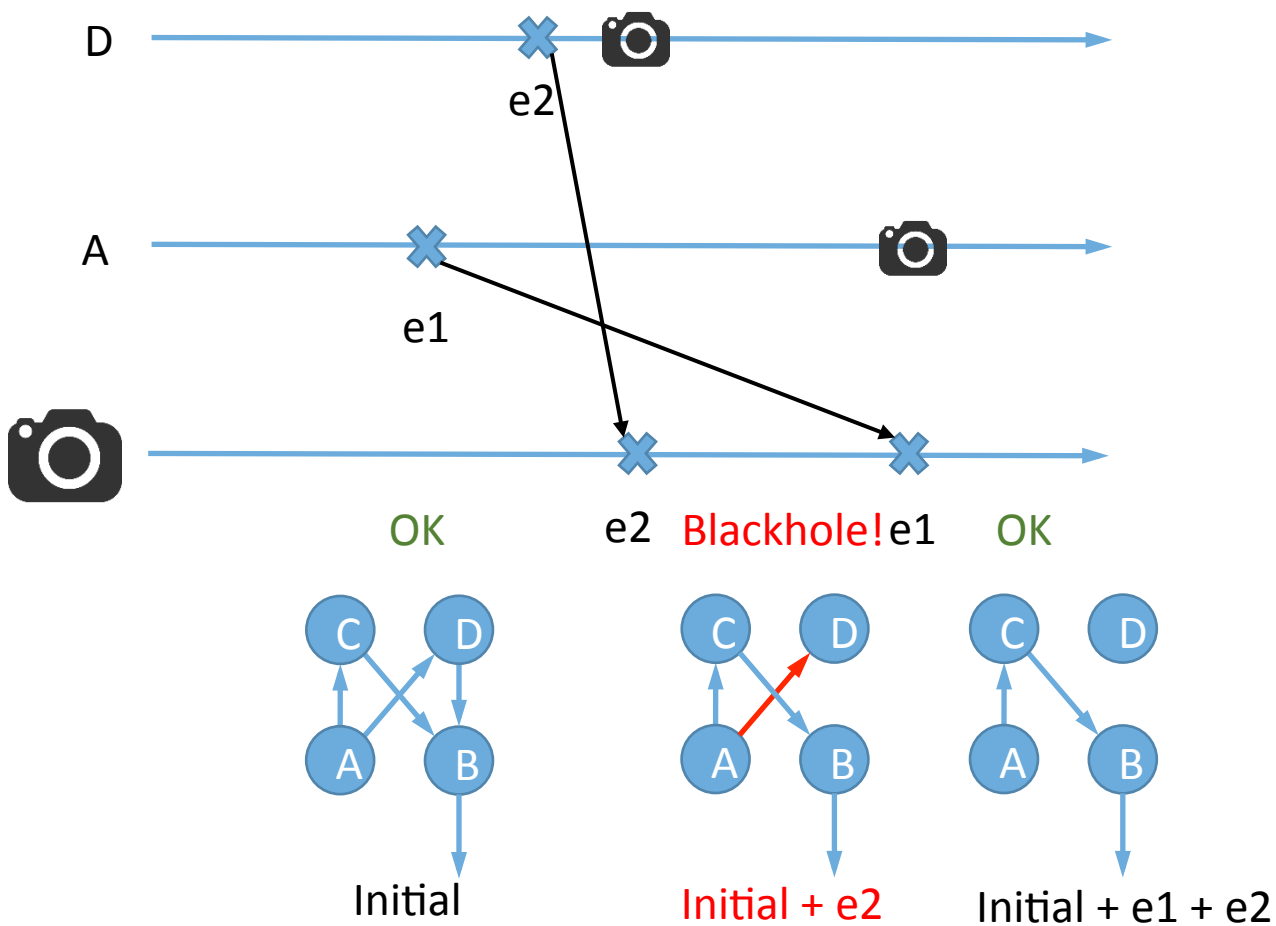
e2) D deletes rule D→B



The Snapshot Problem

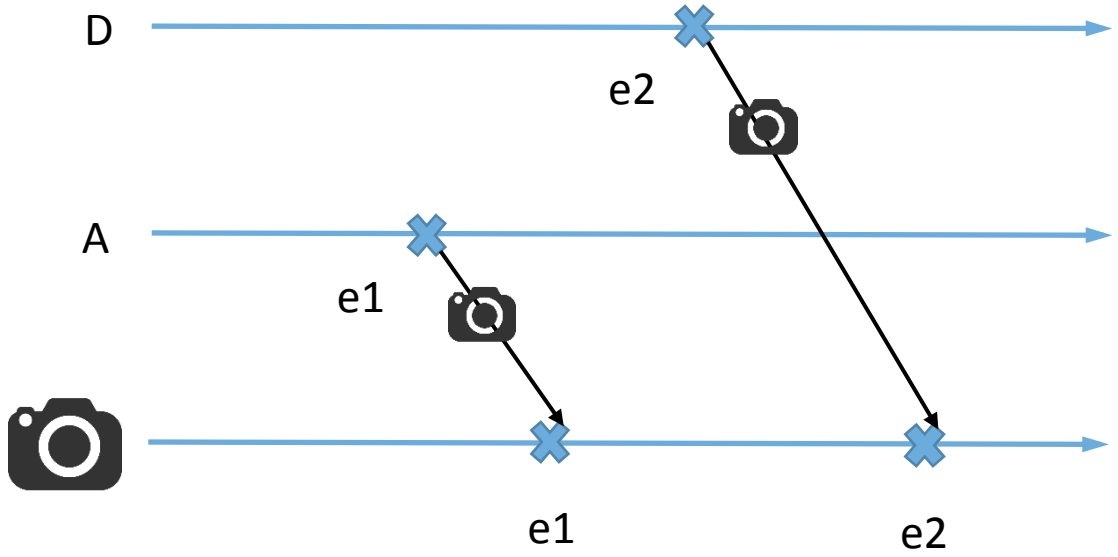
e1) A deletes rule A→D

e2) D deletes rule D→B



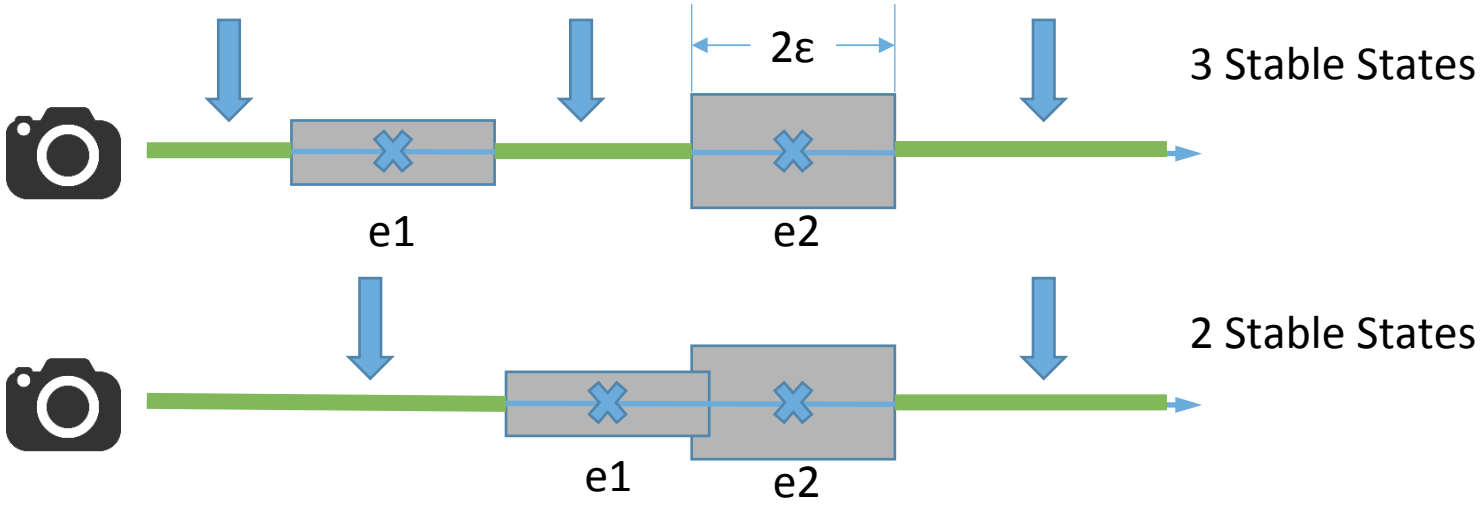
The Stable Snapshot

- Problem: Cannot take a global snapshot simultaneously
  - Otherwise, we will know either (initial), (initial+e1), or (initial +e1+e2) but never (initial+e2)
  - But this is a physical limitation..
- Solution: Record events (rule updates), not states (forwarding tables)



The Stable Snapshot

- Problem: Lack of true Global Clock for timestamps
- Solution: Leverage imperfect global clock – NTP
  - NTP has precision of ε = 1-100ms: Every event is bounded by 2ε
  - Only consider a snapshot when it's stable



Will networks ever be stable?

Example: Bursty rule updates in a production Google DC

- 28,445 events in 24 hours
- 95% of events happen within 400ms of each other
- 99.9% of time the network is stable ("silent")
  - Assuming ε = 100ms

How about transient states?

- We can report "potential" problems
- Network operators may not care anyway

Two Challenges of Data Plane Verification in Data Centers

Constant Data Plane Changes

- Link up/down, route recalculation, new policy, etc.
- Bad snapshot may result in false positives
- Solution: Stable Snapshots

Large Scale

- 10,000s of switches, 1,000,000s of rules
- How to finish the verification within a reasonable time?

Scalable Verification

Naïve Static Checker:

- Load all rules in memory
- Pick one destination, pick one ingress port, *simulate*
- Try the next one

Data Center Networks are immense

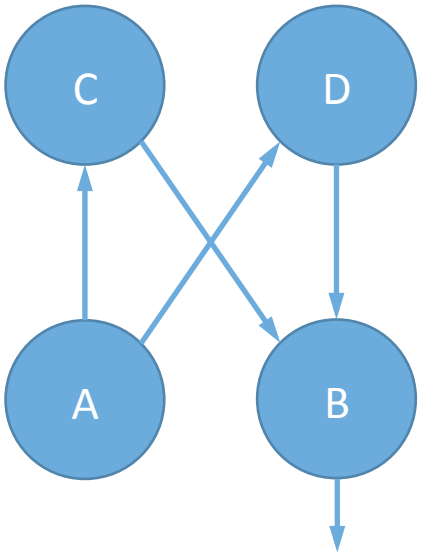
- 10,000 switches x 1,000 rules = 10M rules
- Thousands of destinations

Verification Complexity grows as $O(N^2)$

- N times switches/rules
- N times destinations

Parallelism! – But, how to partition?

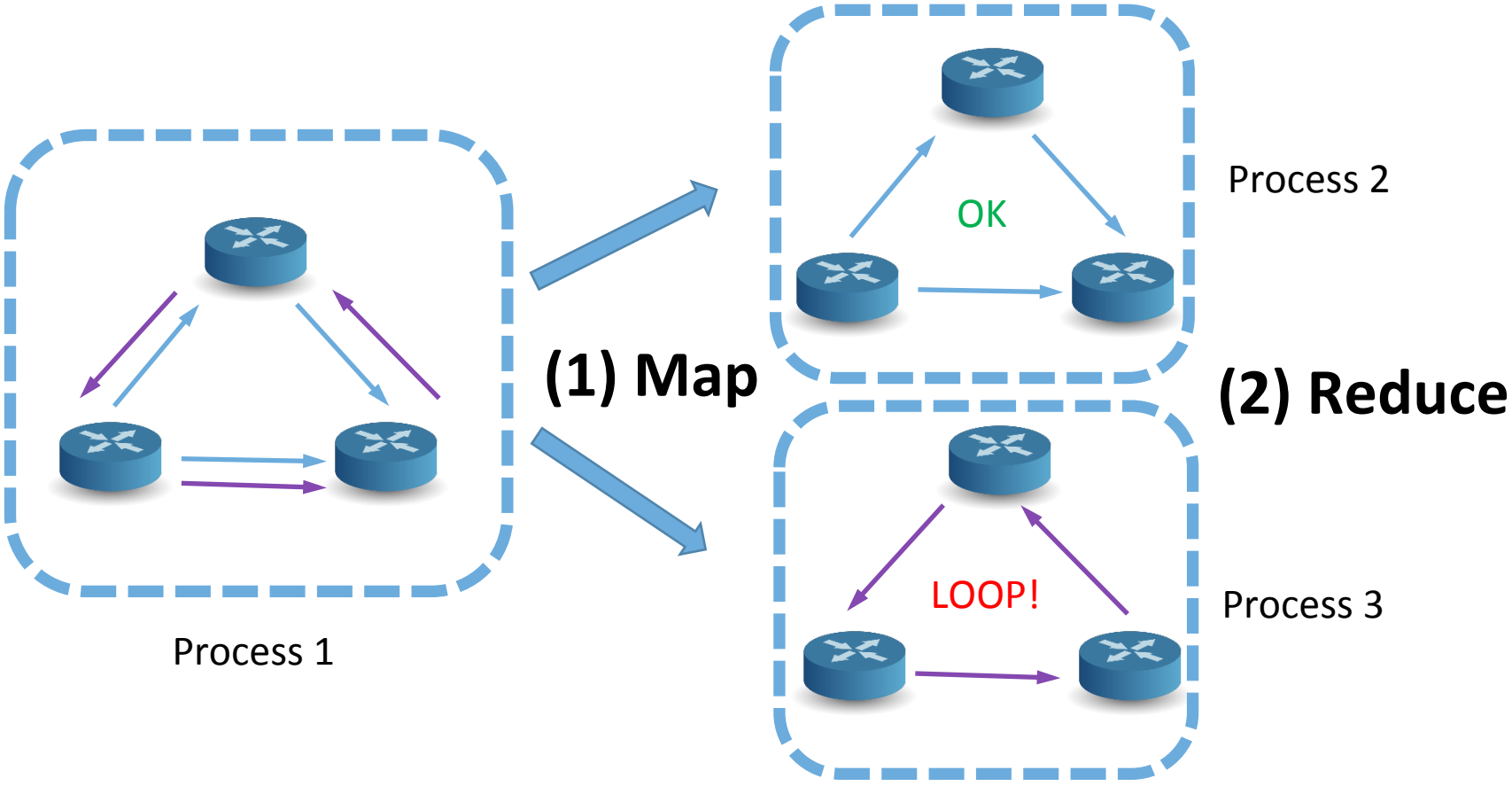
Forwarding Graph



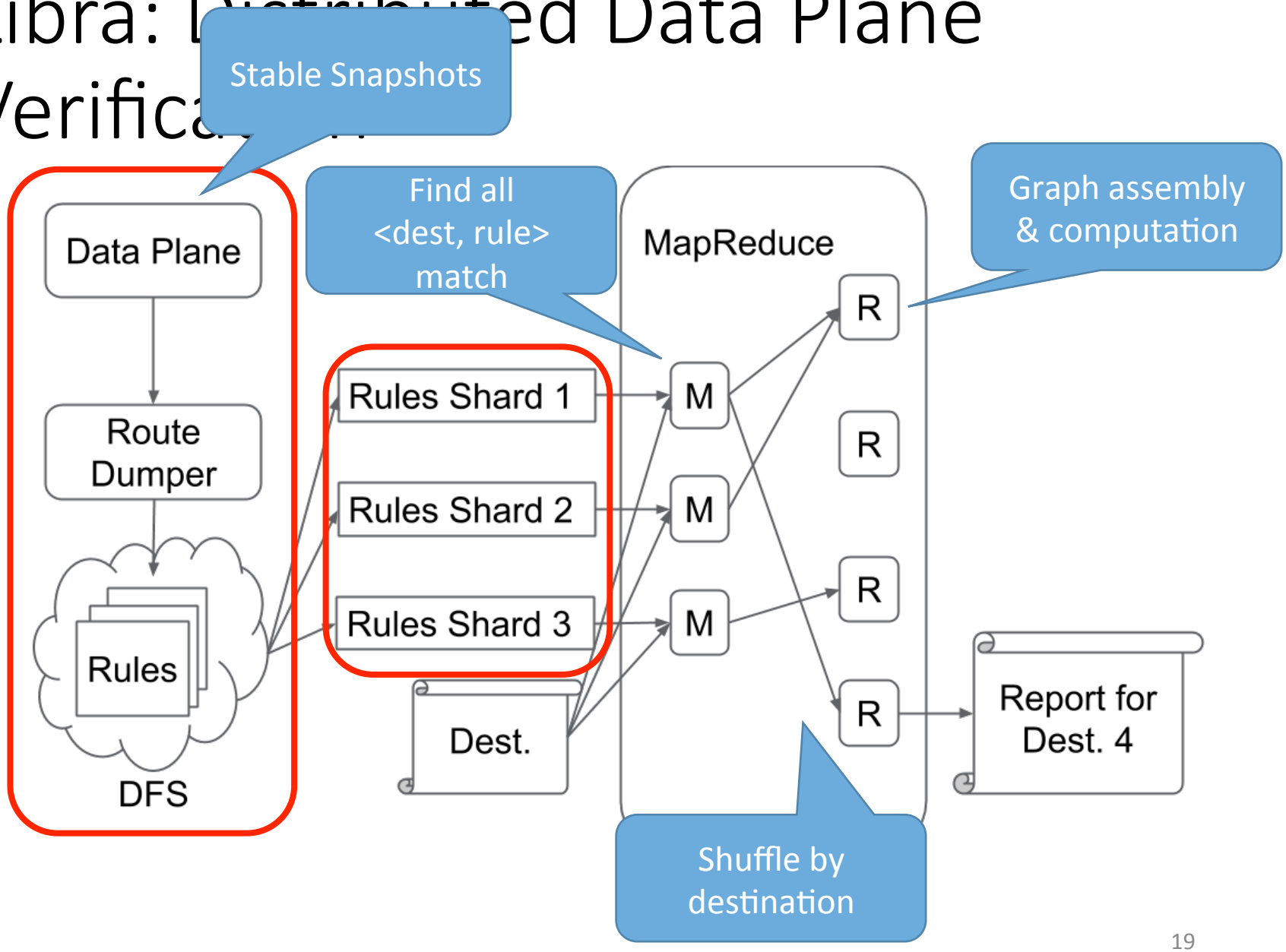
Forwarding Graph for IP2

- Each destination has a forwarding graph
- All data plane abnormalities are graph problems
  - loops, blackholes, partitions
- Forwarding graph size == Physical network size
  - Each physical node is a vertex
- For each graph, only a subset of rules are needed

Divide and Conquer



Libra: Distributed Data Plane Verification



Evaluation: Loop detection

	DCN	DCN-G	INET
Switches	11,260	1,126,001	316
Rules	2,657,422	265,742,626	151,649,486
Destinations	11,136	1,113,600	482,966
Machines	50	20,000*	50
Time/s	57	906	93

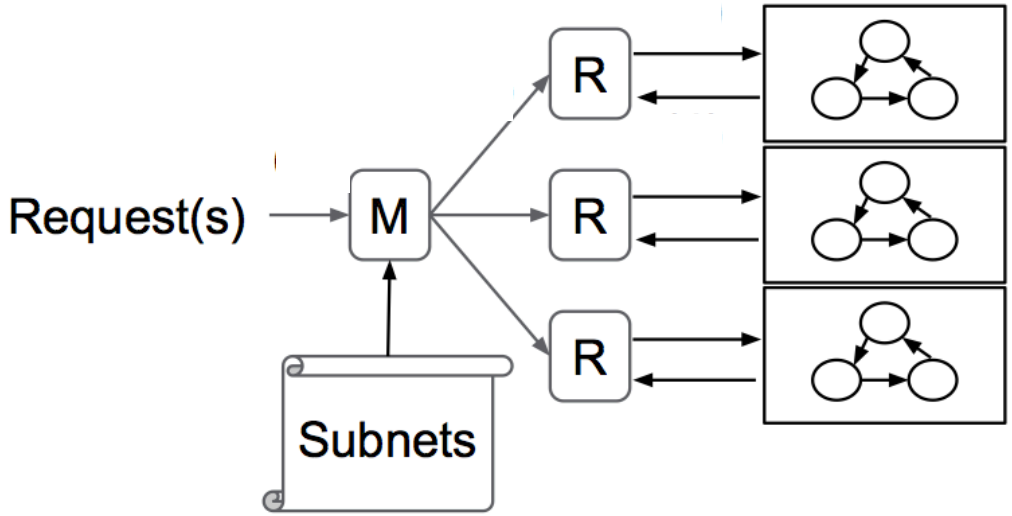
DCN: Google's emulated data center network

DCN-G: 100 DCNs connected in a star topo

INET: 300 Internet routers with full BGP table

*extrapolated

Even Faster: Incremental Updates



	DCN	DCN-G	INET
Map (us)	0.133	0.156	0.158
Reduce (ms)	0.62	1.76	<0.01

Libra: Data Plane Verification at Data Center Scale

- Stable Snapshots
  - Only consider stable states of the data plane
  - Avoid false positives in bad snapshots
- Divide and Conquer
  - Implemented in MapReduce
  - Handles massive scale with parallelism

Thank you!

http://eastzone.github.io/libra/

Acknowledgement: Thanks to our shepherd T.V. Lakshman and anonymous NSDI reviewers for constructive feedbacks.

"Distributed" Longest Prefix Matching

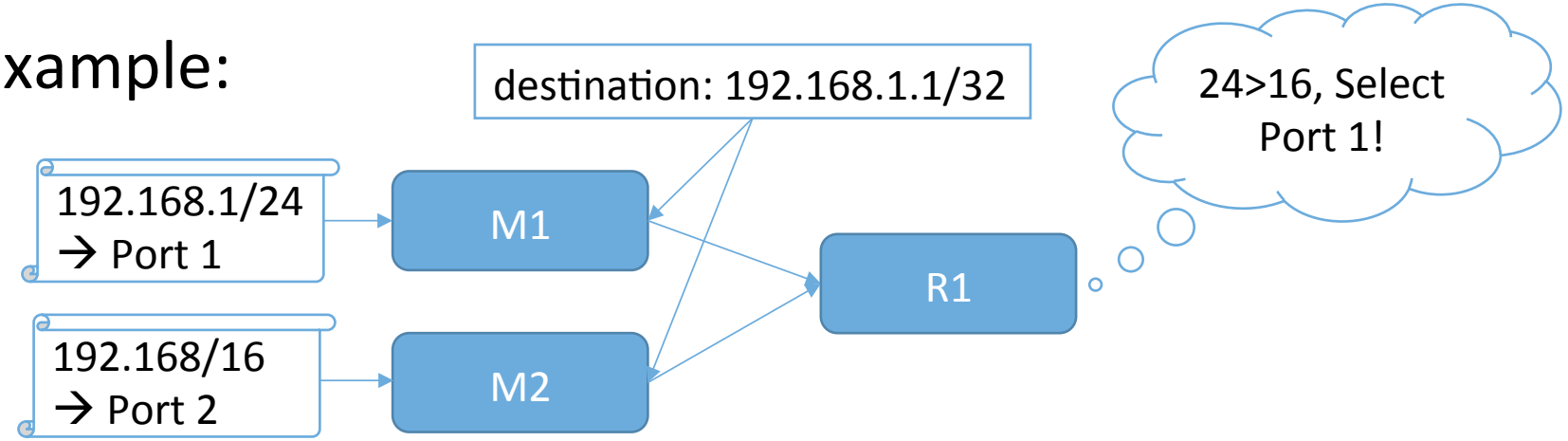
Problem: Relevant rules may be in different shards

- Mappers cannot see each other!

Solution: Defer length comparison to Reduce phase

- Reducer can see everything related to a destination

Example:



Challenge: All-prefix Matching

Problem: In mappers, given a rule, find out *all* matching destinations

- Conventional matching: Given a destination, find out *one* matching rule
- Naïve approach: Linear search, O(*TR*) where *T* = # of destinations, *R* = # of rules

Solution: Use a Trie (prefix Tree)

- Put all destinations in a Trie (once per mapper)
- O(*LR*) where *L*=32 for IPv4
  - Assuming a small number of matching destinations

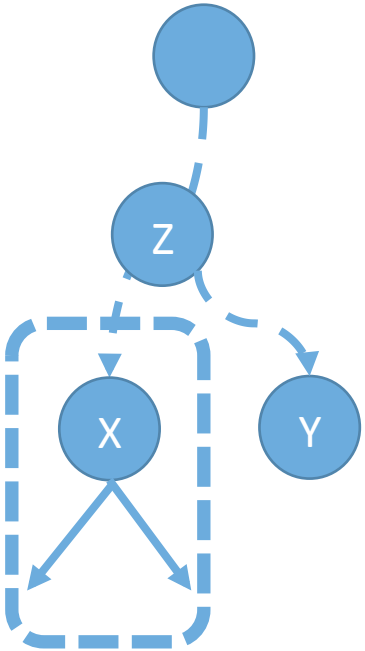
Challenge: All-prefix Matching

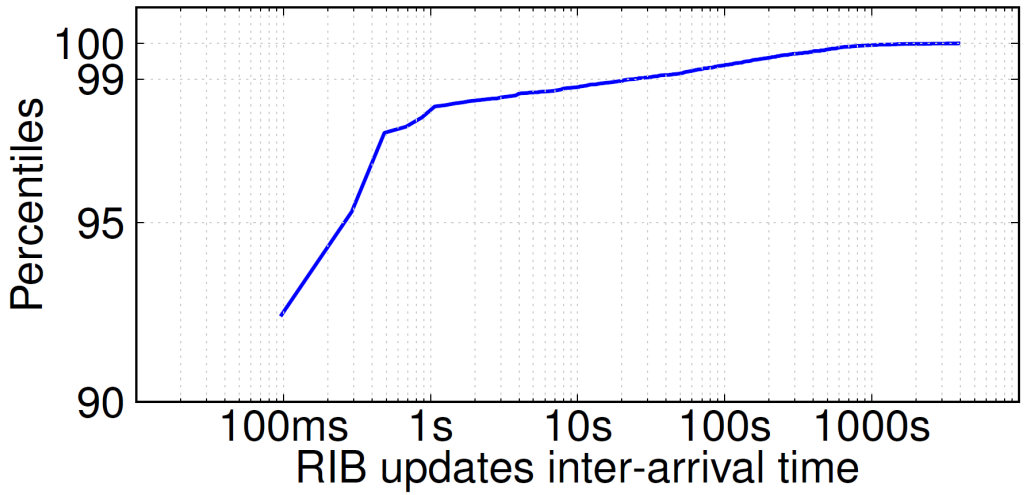
Algorithm:

- Step 1: Find the smallest matching trie node (*X*) that is bigger or equal to the rule (*A*) (lexical order)
- Step 2: Enumerate all *X*'s descendants including *X*.

Proof:

- Assume there is another node *Y* that isn't *X*'s descendant
- *X* and *Y* have common ancestor *Z*, *Z*!=*X* and *Z*!=*Y*
- *Z* matches *A* because both *X* and *Y* match *A*, and
- *Z* is smaller than *X*. Contradiction! QED




100
99
95
90
Percentiles
100ms
1s
10s
100s
1000s
RIB updates inter-arrival time

Who benefits

Network Engineers

- Ensure policy compliance
- Reduce service downtime
- Report specific errors to vendors/Fix bugs

Network Users

- Visibility in networks
- Reduce application diagnosis time
  - Verification OK → Debug application
  - Verification Failed → Engage network engineers