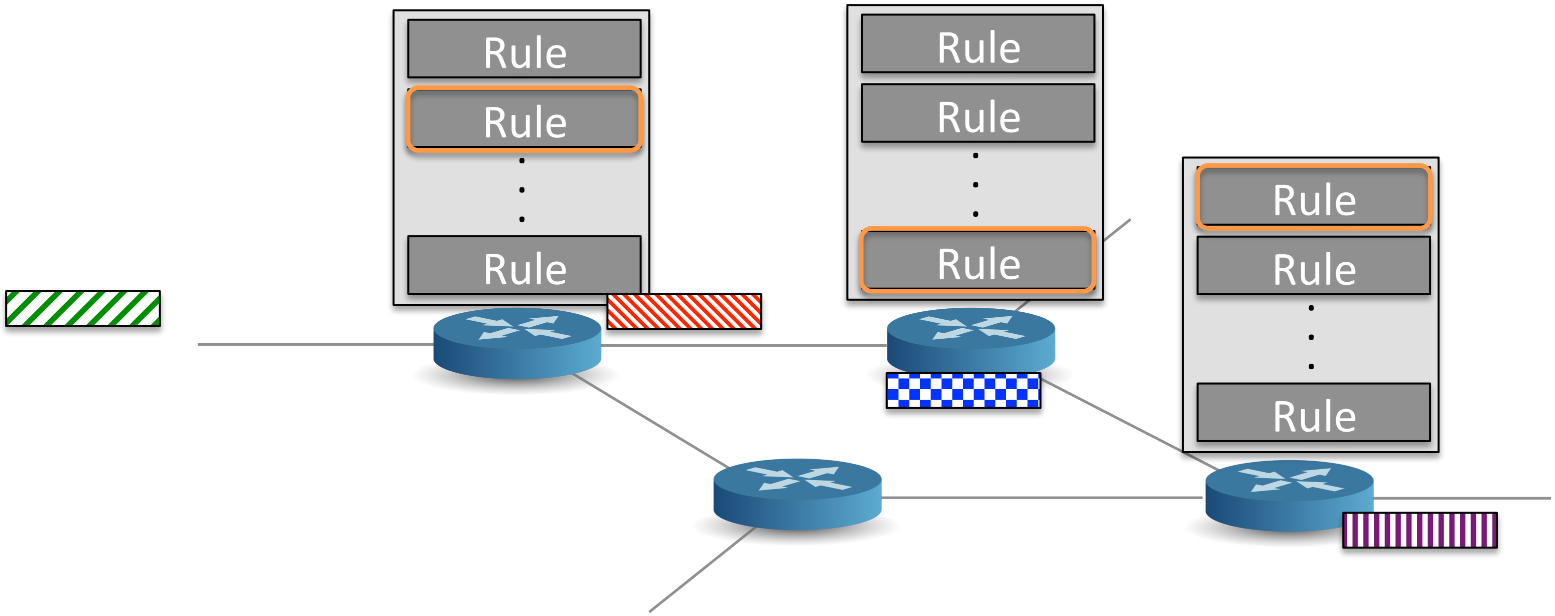
Real Time Network Policy Checking Using Header Space Analysis

Peyman Kazemian
with
Michael Chang, Hongyi Zeng, George Varghese,
Nick McKeown, Scott Whyte

NSDI 2013 – Lombard, IL

Network debugging is hard!

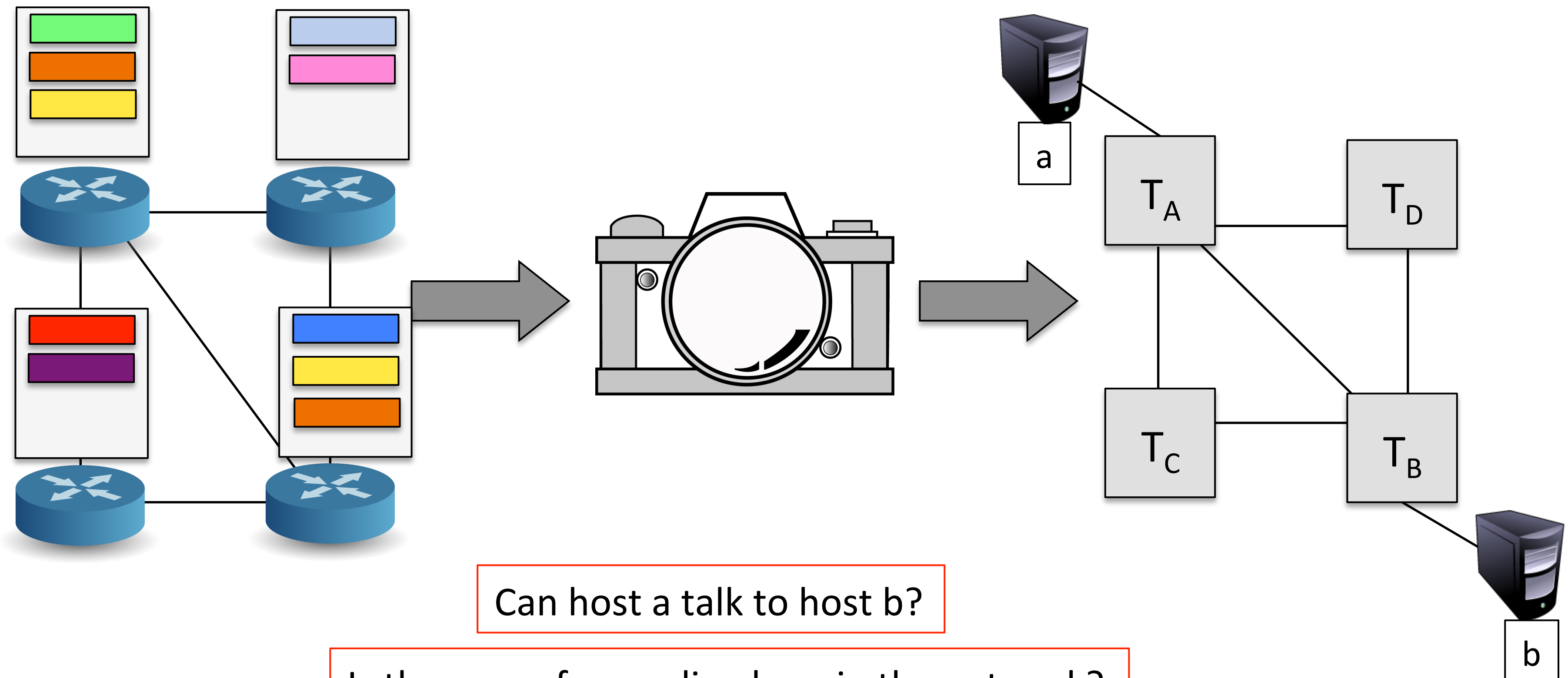
- Forwarding state is hard to analyze!



Network debugging is hard!

- Forwarding state is hard to analyze!
  1. Distributed across multiple tables and boxes.
  2. Written to network by multiple independent writers (different protocols, network admins)
  3. Presented in different formats by vendors.
  4. Not directly observable or controllable.
- Not constructed in a way that lend itself well to checking and verification.

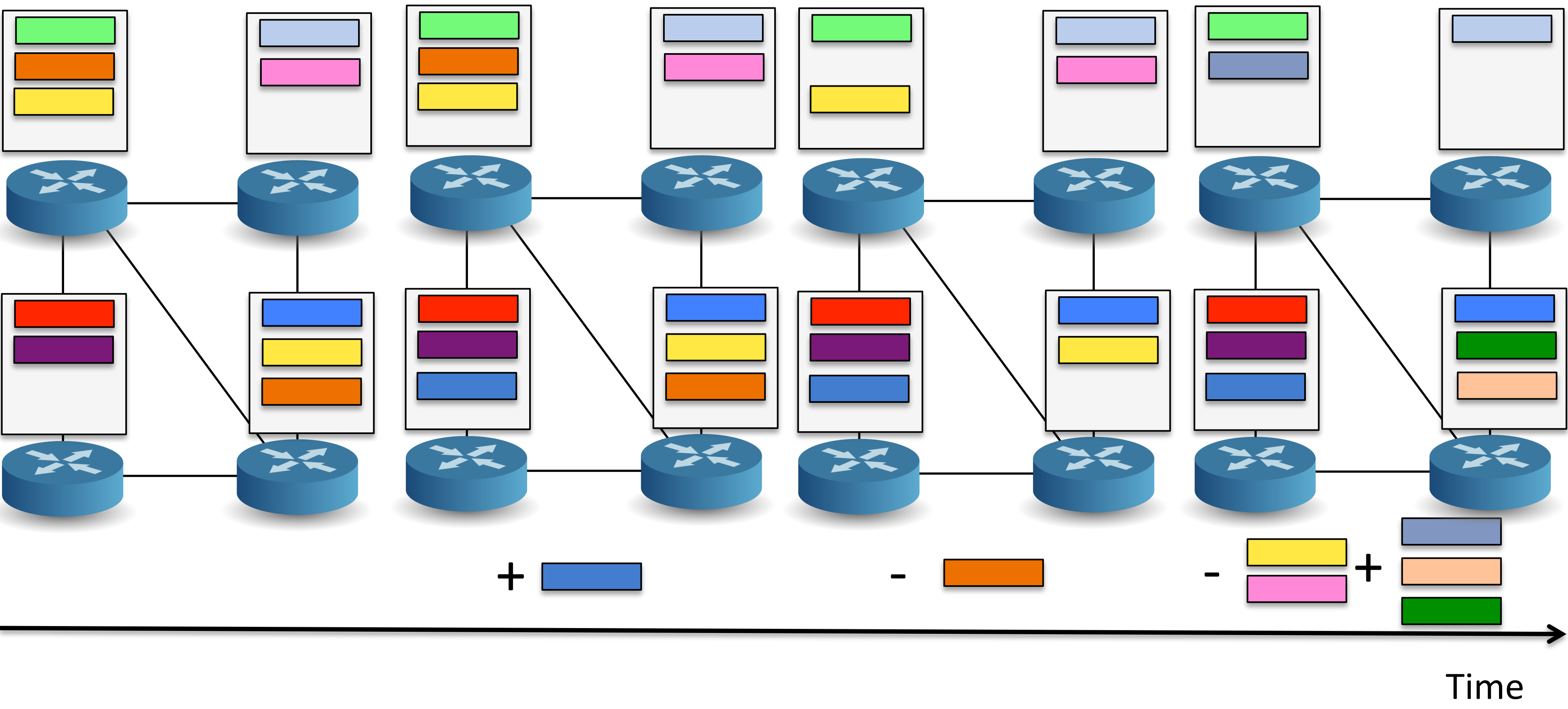
Header Space Analysis: Snapshot-based Checking



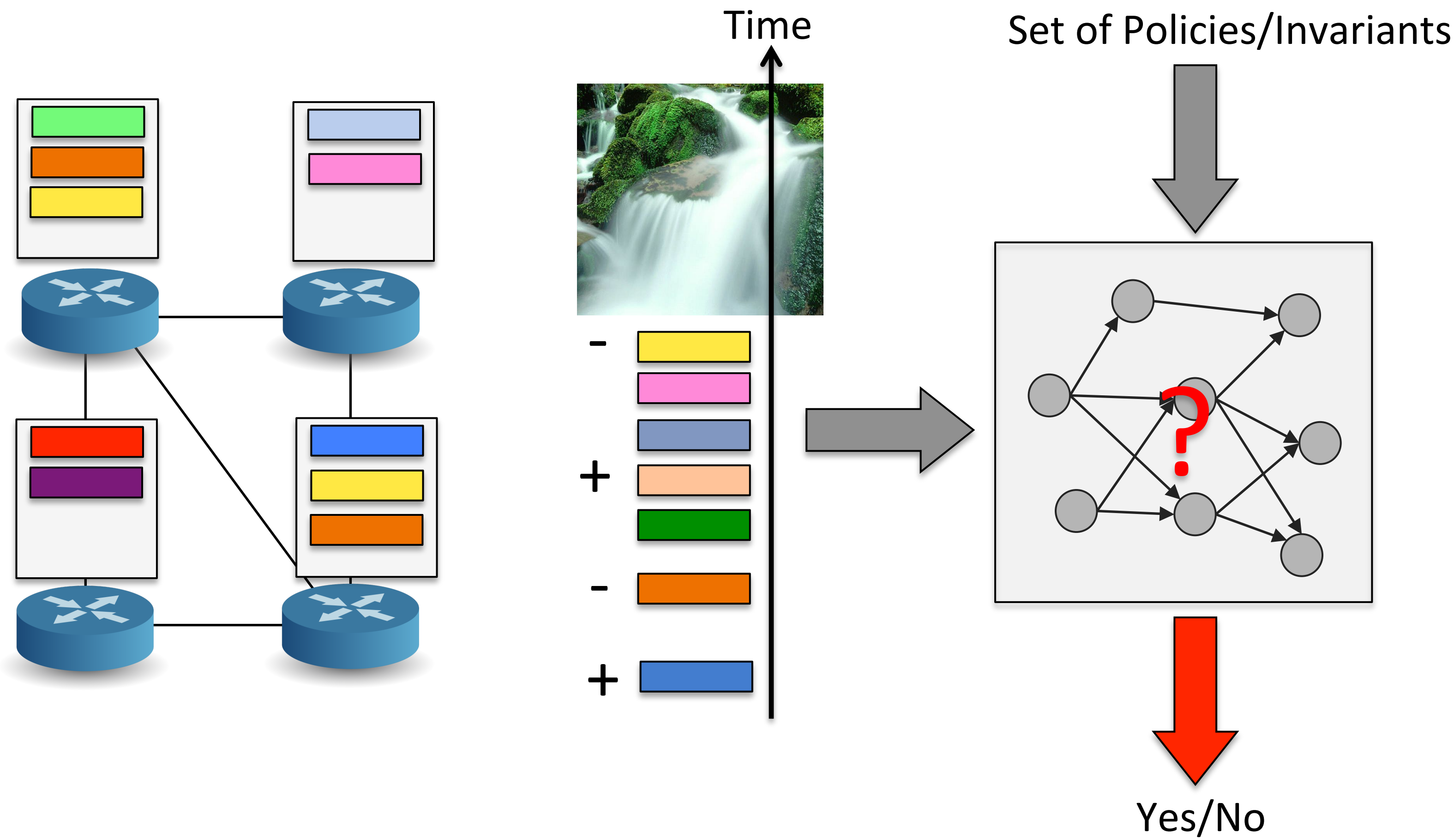
Can host a talk to host b?

Is there any forwarding loop in the network?

Steam-based Checking



Steam-based Checking



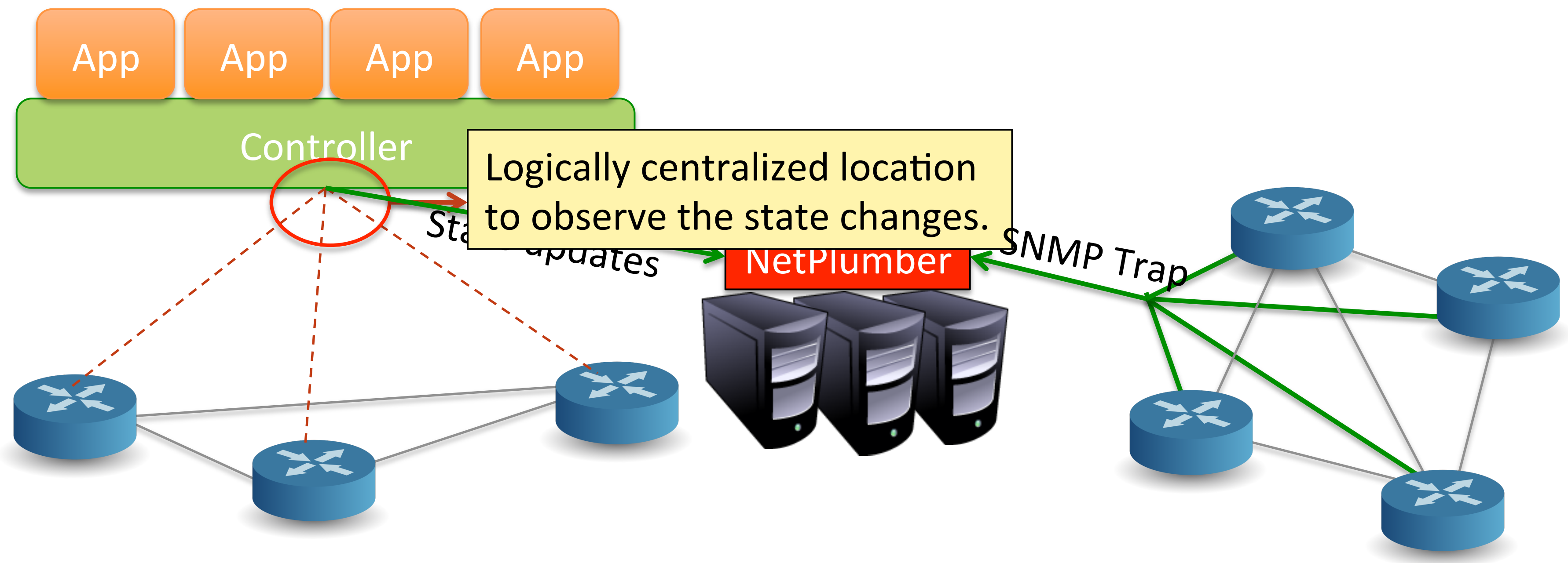
Prevent errors before they hit network.
Report a violation as soon as it happens.

Outline

- Motivations.
- NetPlumber: Real time policy checking tool.
  - How it works?
  - How to check policy?
  - How to parallelize?
- Evaluation on Google WAN.
- Conclusions.

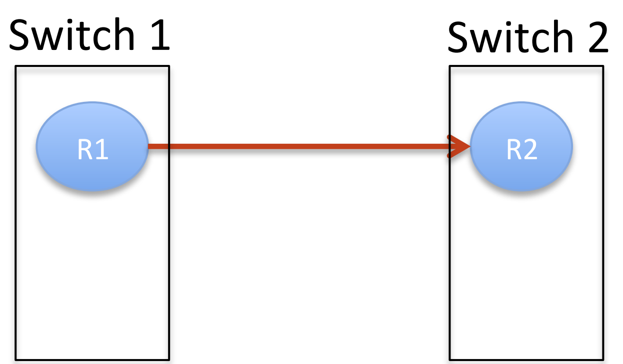
NetPlumber

- The System we build for real time policy checking is called NetPlumber.

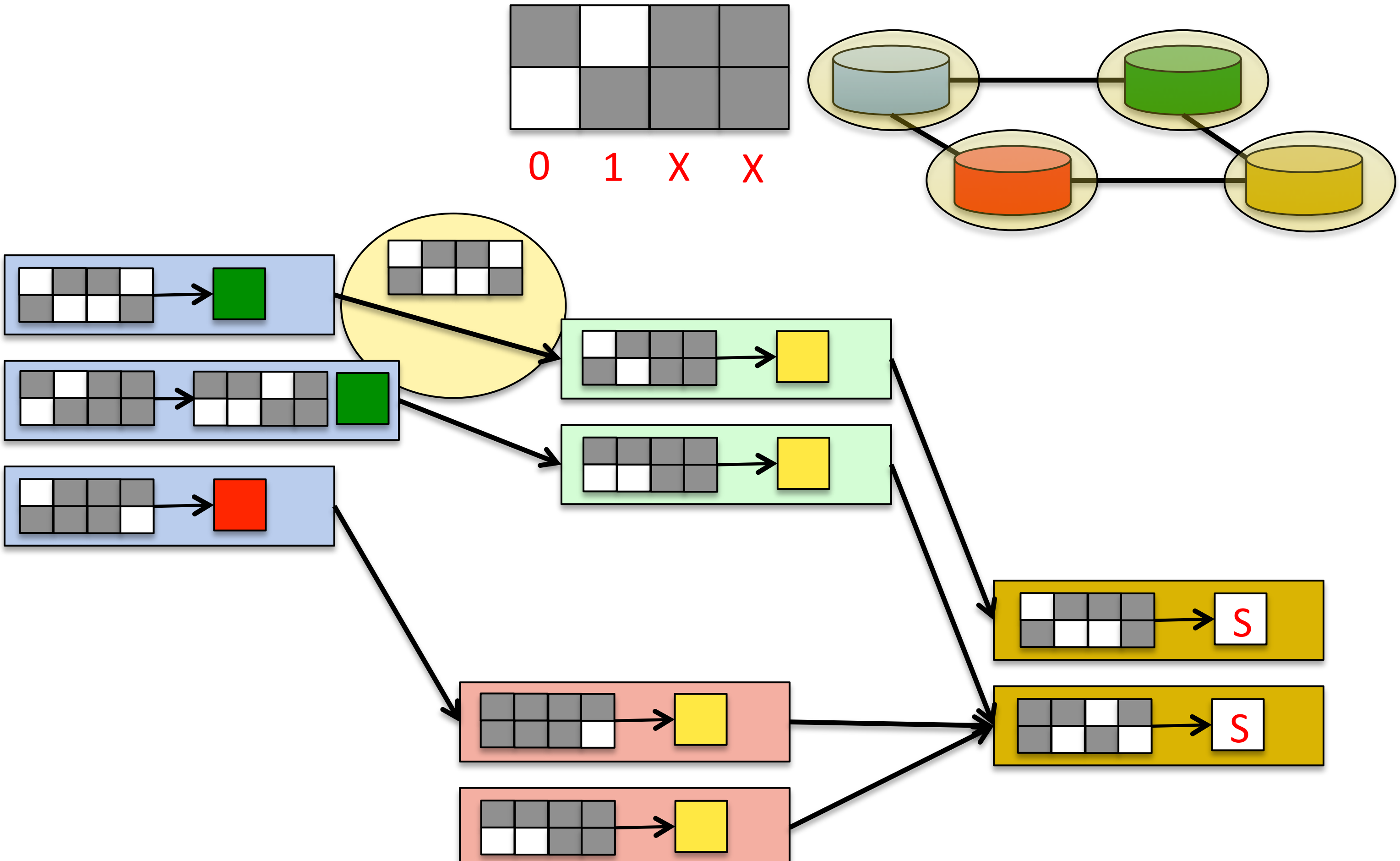


NetPlumber

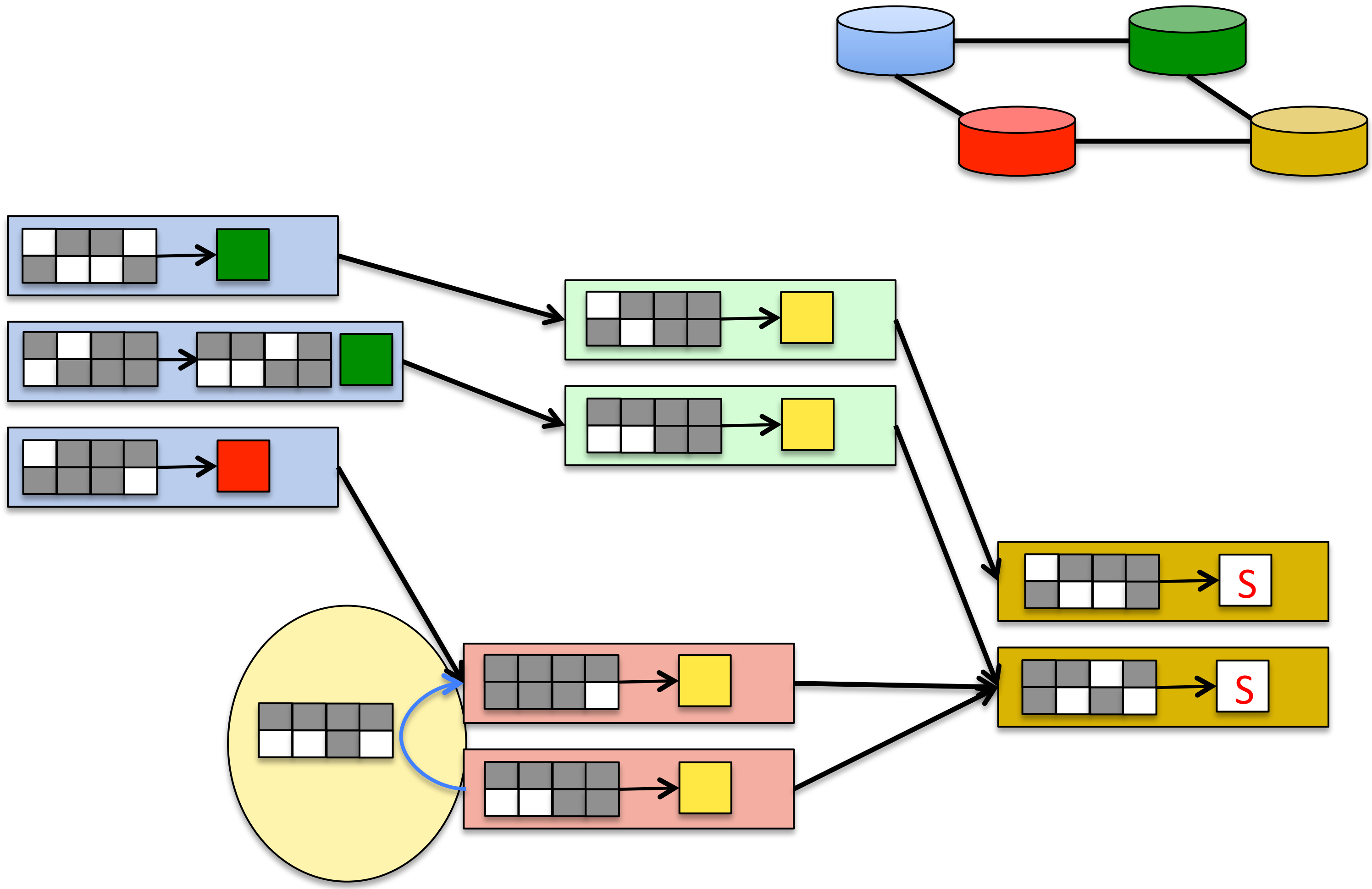
- The System we build for real time policy checking is called NetPlumber.
  - Creates a dependency graph of all forwarding rules in the network and uses it to verify policy.
  - Nodes: forwarding rules in the network.
  - Directed Edges: next hop dependency of rules.



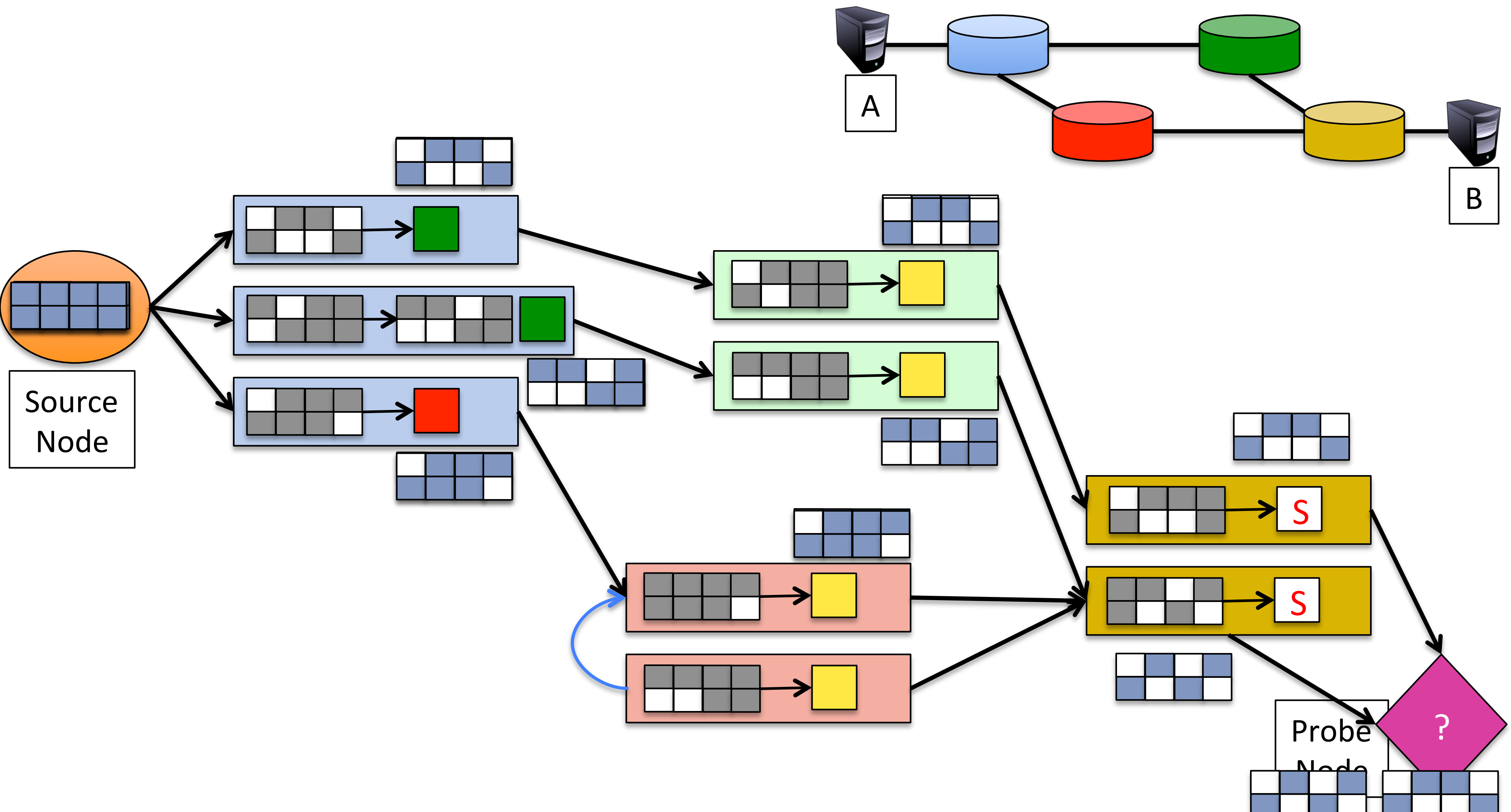
NetPlumber – Nodes and Edges



NetPlumber – Intra table dependency

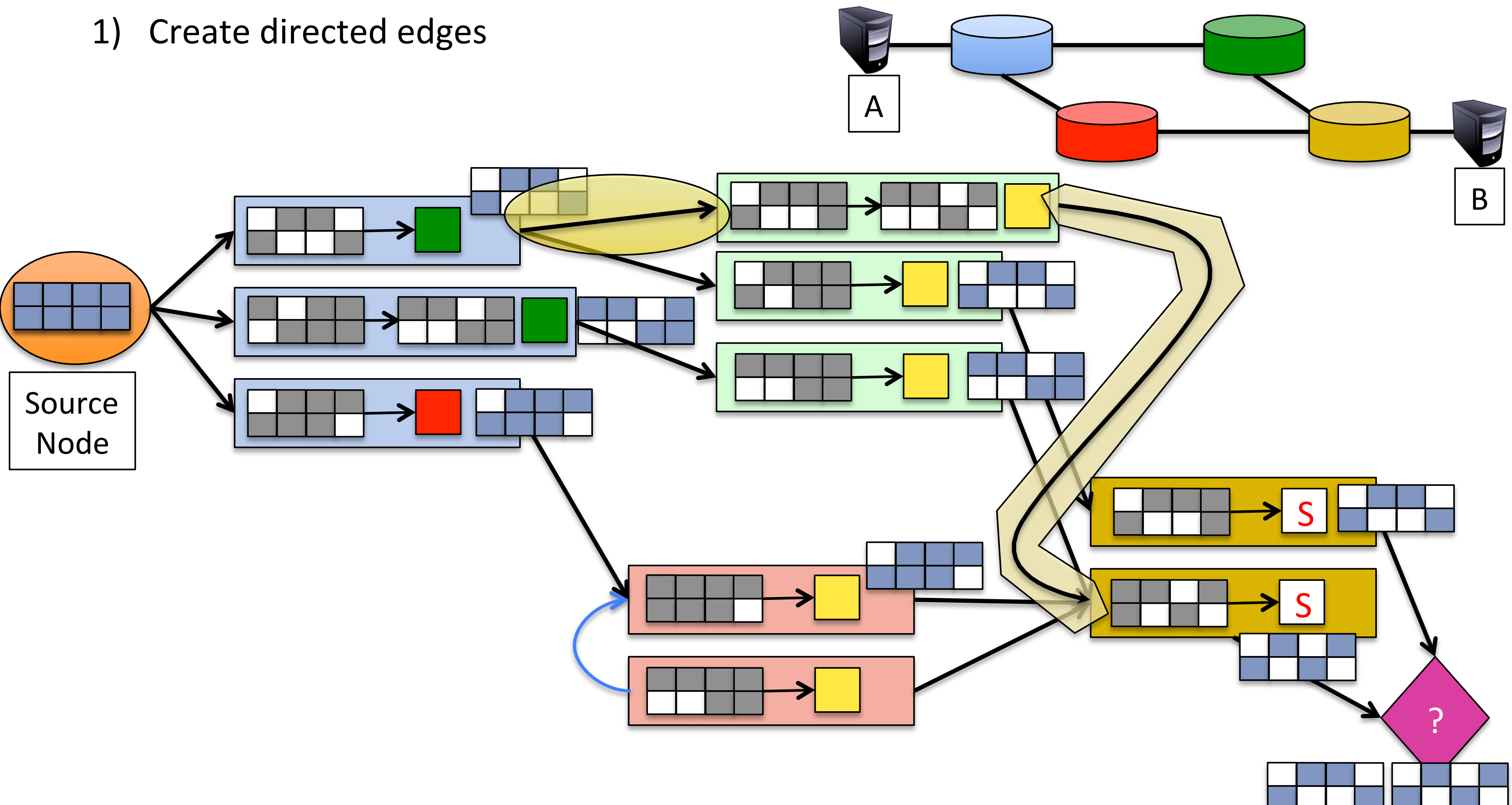


NetPlumber – Computing Reachability



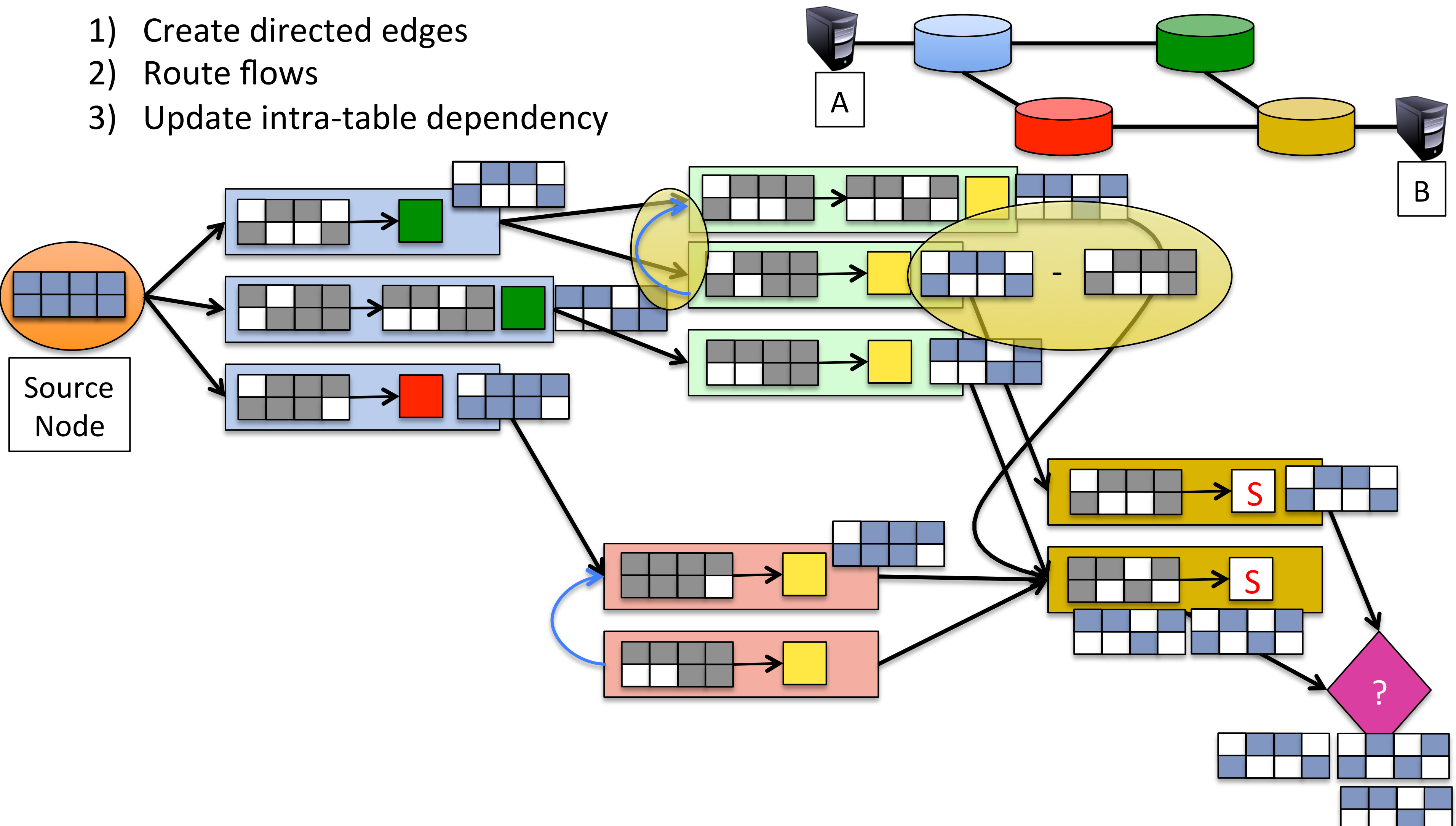
NetPlumber – Computing Reachability

1) Create directed edges

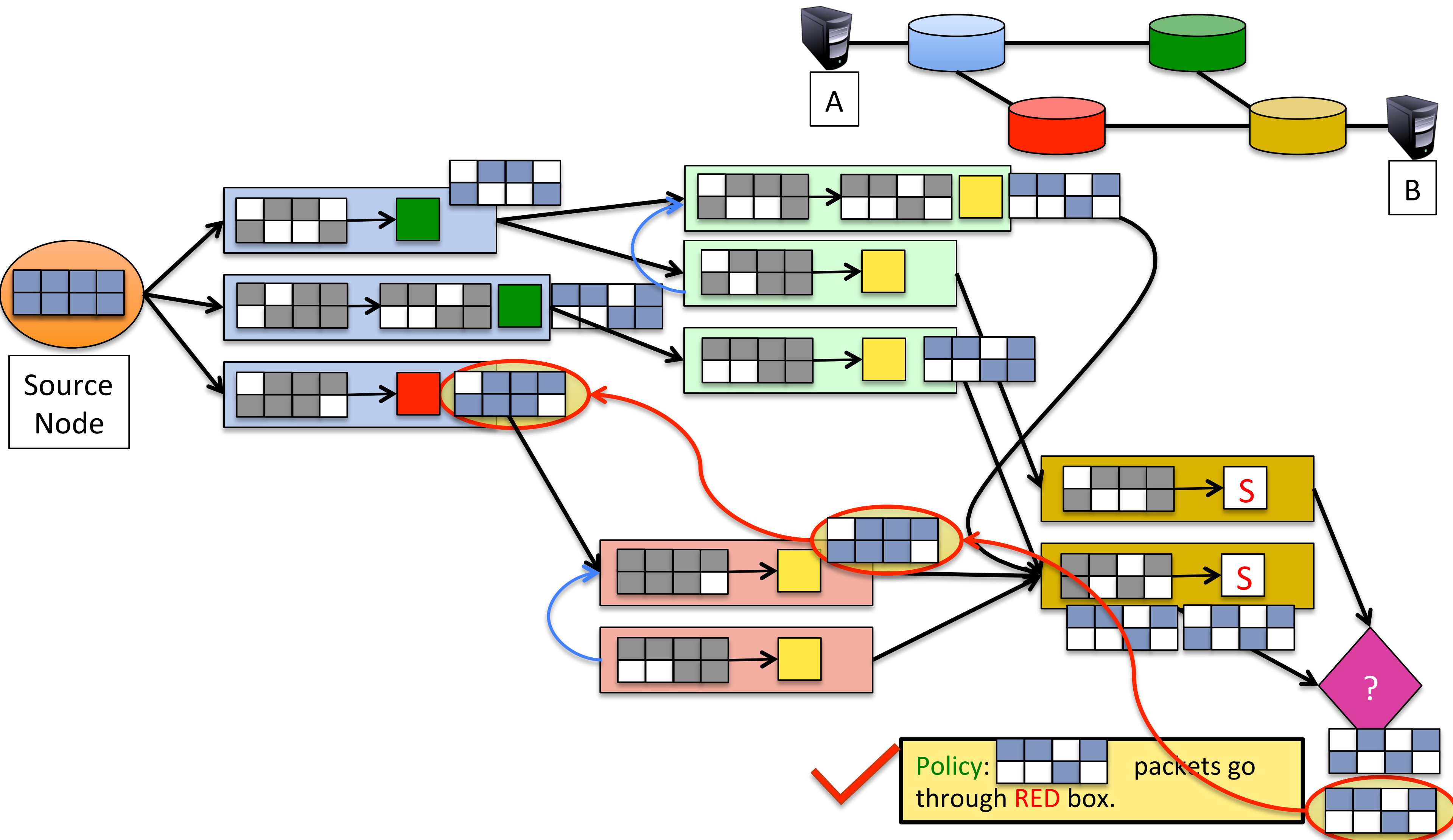


NetPlumber – Computing Reachability

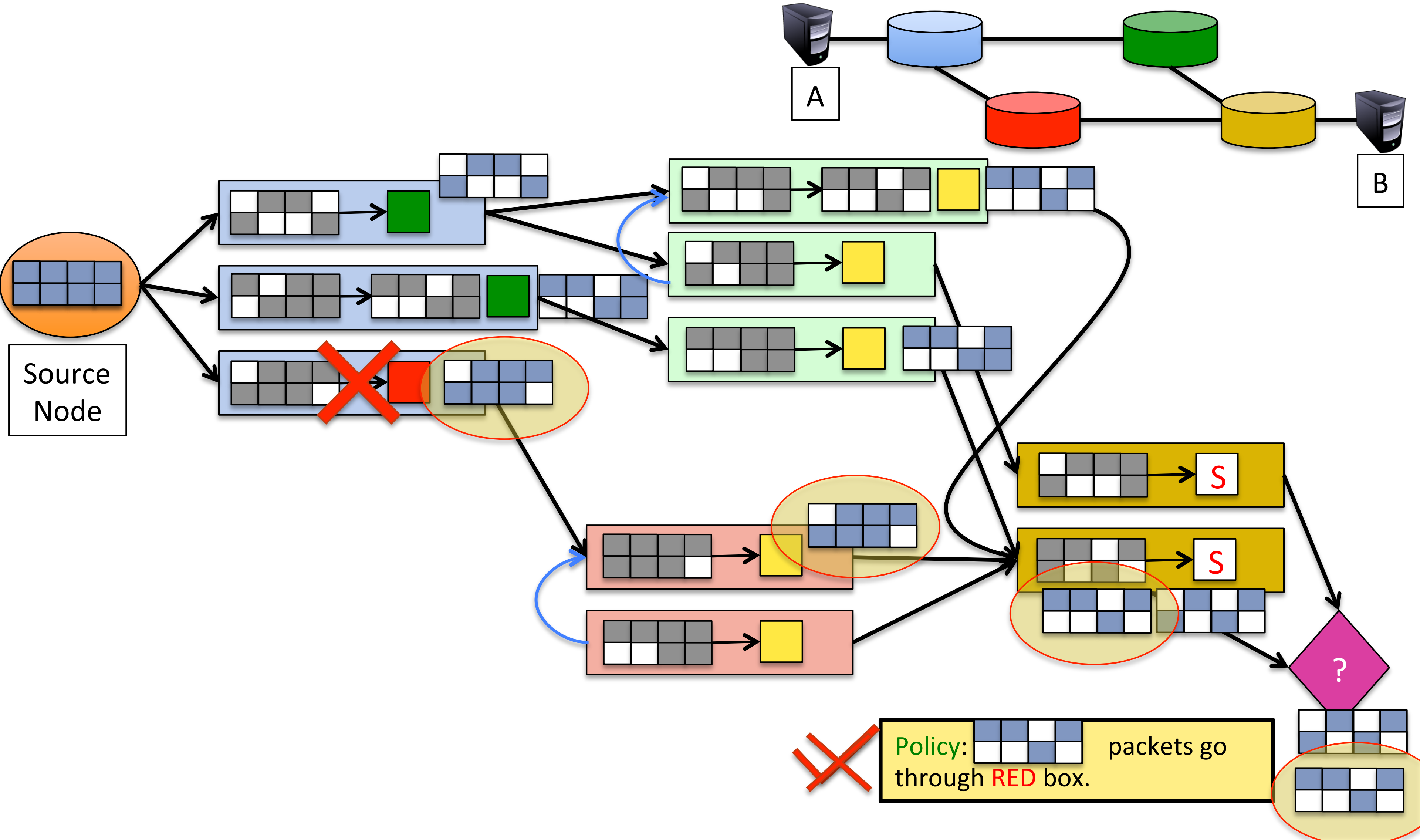
1) Create directed edges
2) Route flows
3) Update intra-table dependency



NetPlumber – Checking Policy

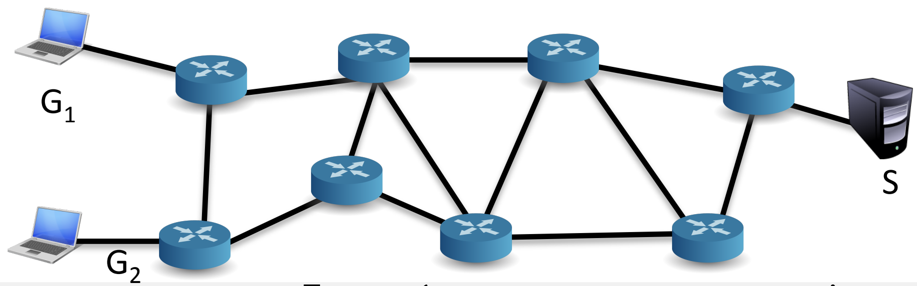


NetPlumber – Checking Policy

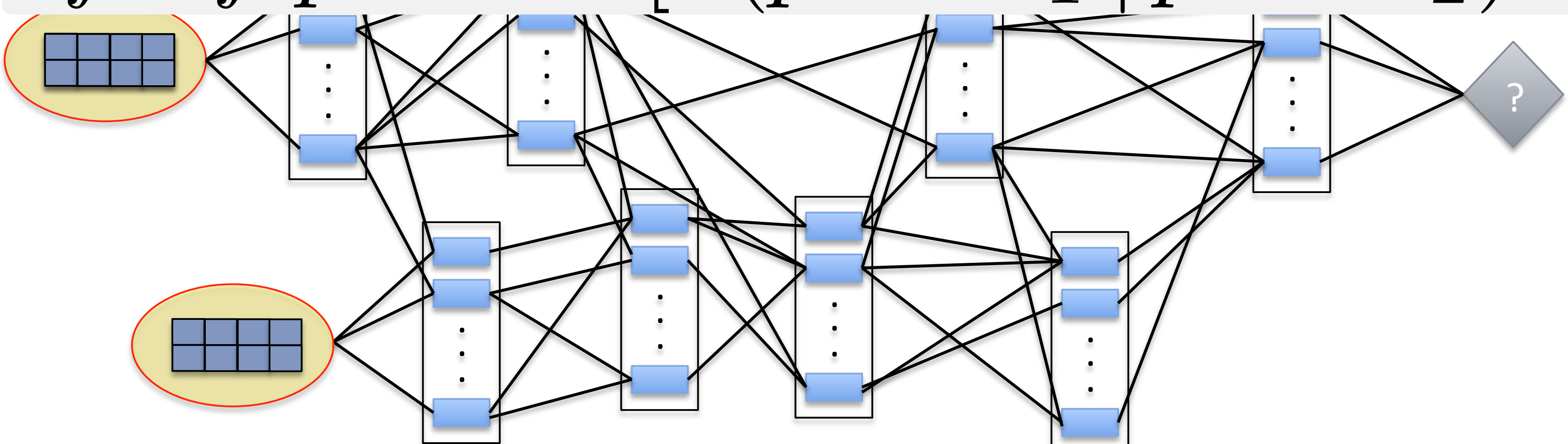


Checking Policy with NetPlumber

Policy: Guests can not access Server S.

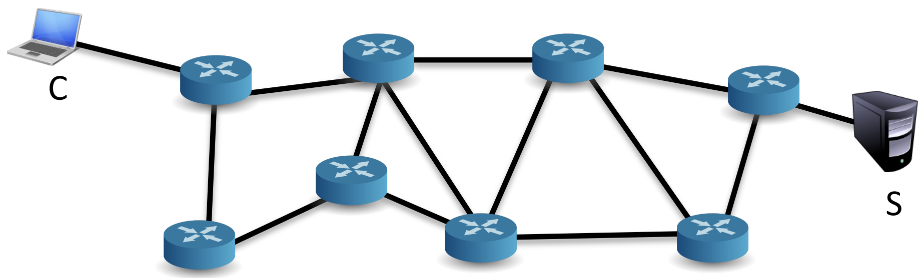


$$\forall f : f.path \sim ![\,\hat{}\,(p = G_1 \mid p = G_2).*]$$

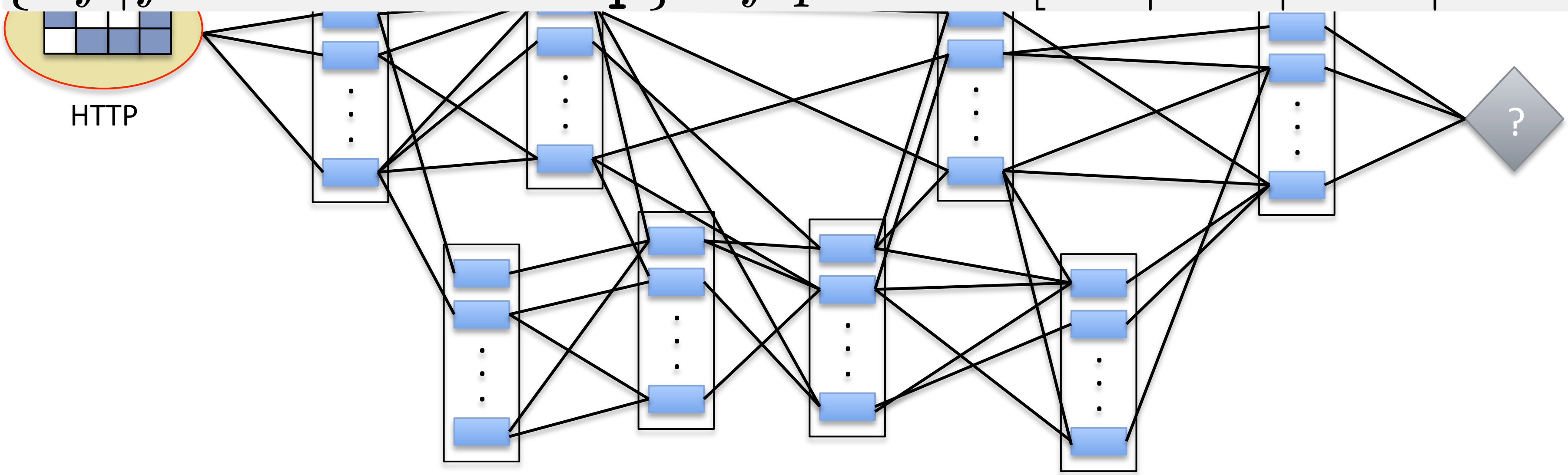


Checking Policy with NetPlumber

Policy: http traffic from client C to server S doesn't go through more than 4 hops.

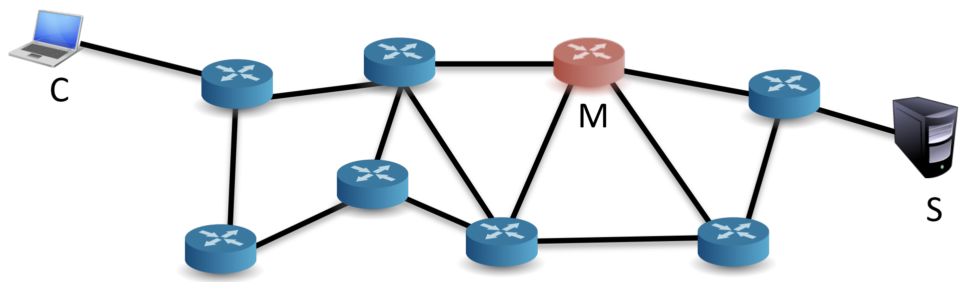


$$\{\forall f | f.header \sim \texttt{http}\} : f.path \sim [\hat{}.\$|\hat{}..\$|\hat{}...\$|\hat{}....\$]$$

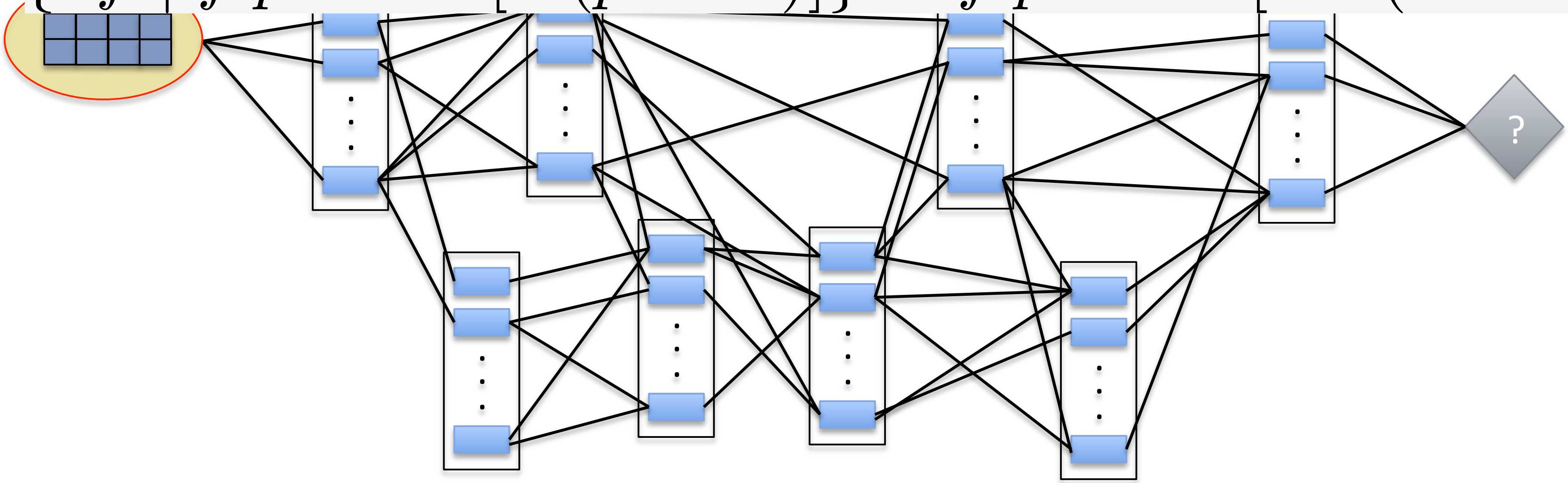


Checking Policy with NetPlumber

Policy: traffic from client C to server S should go through middle box M.



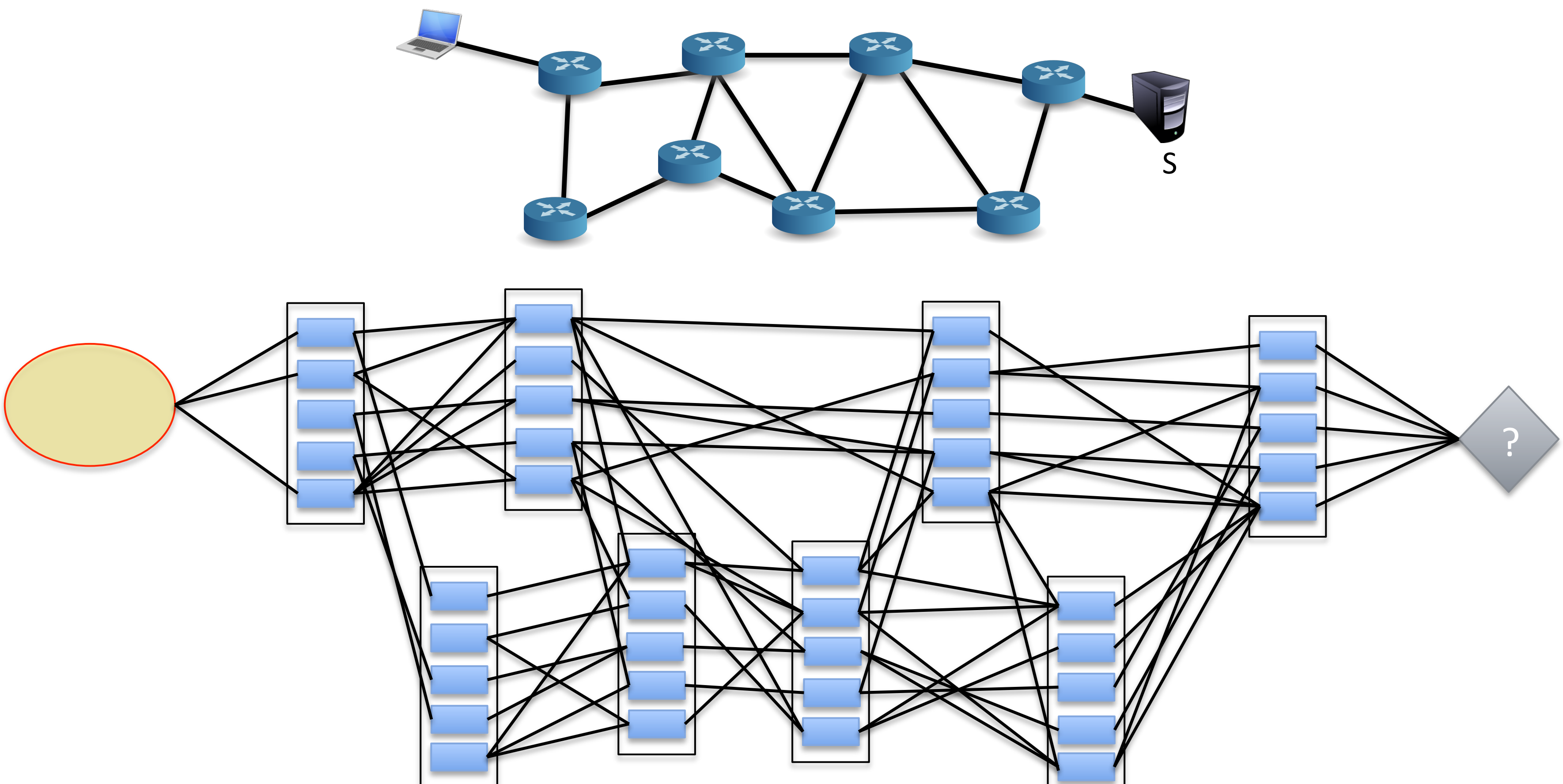
$$\{\forall f \mid f.path \sim [\,\hat{}\,(p = C)]\} \; : \; f.path \sim [\,\hat{}\,.^*(t = M)]$$



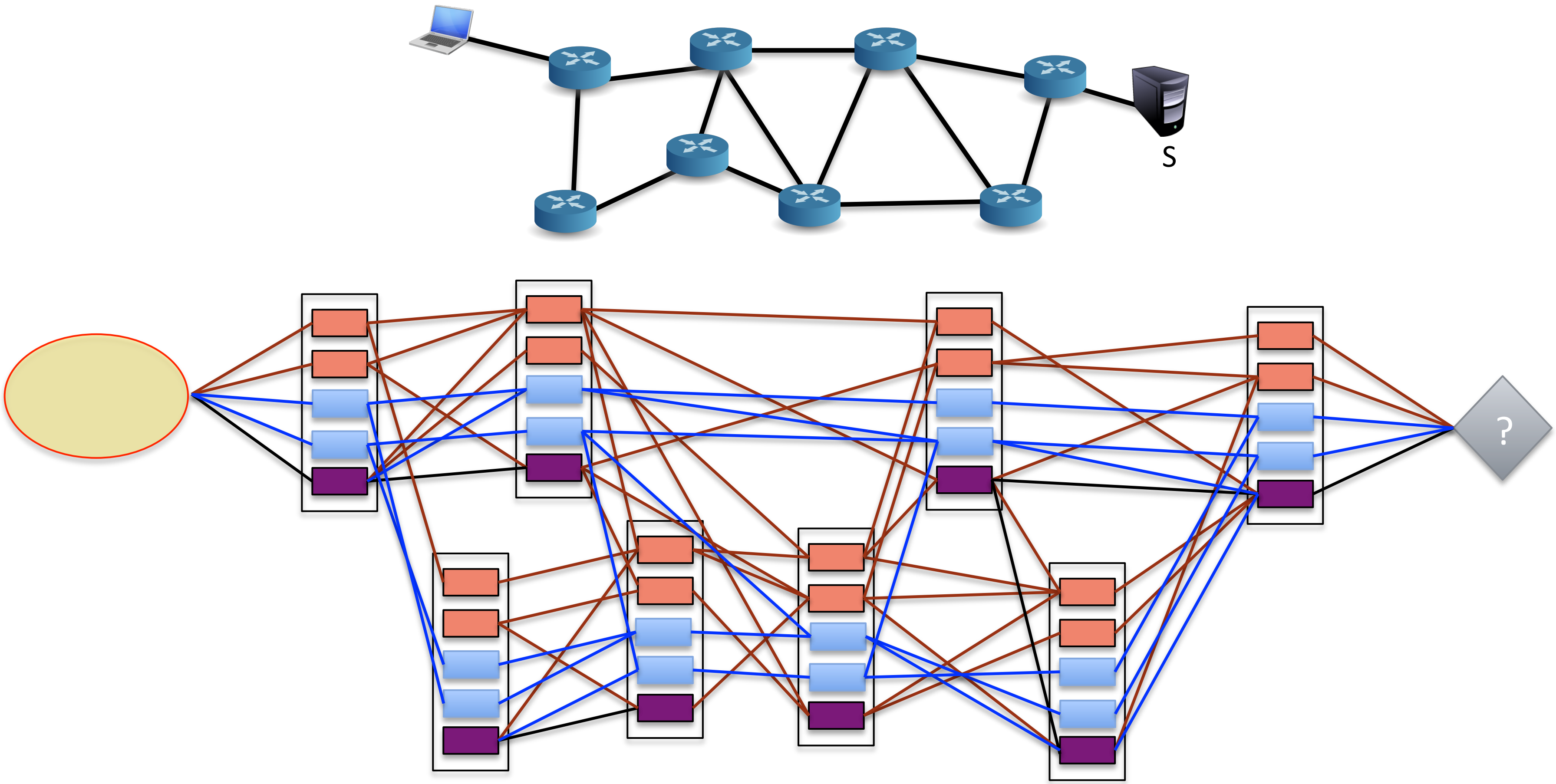
Why the dependency graph helps

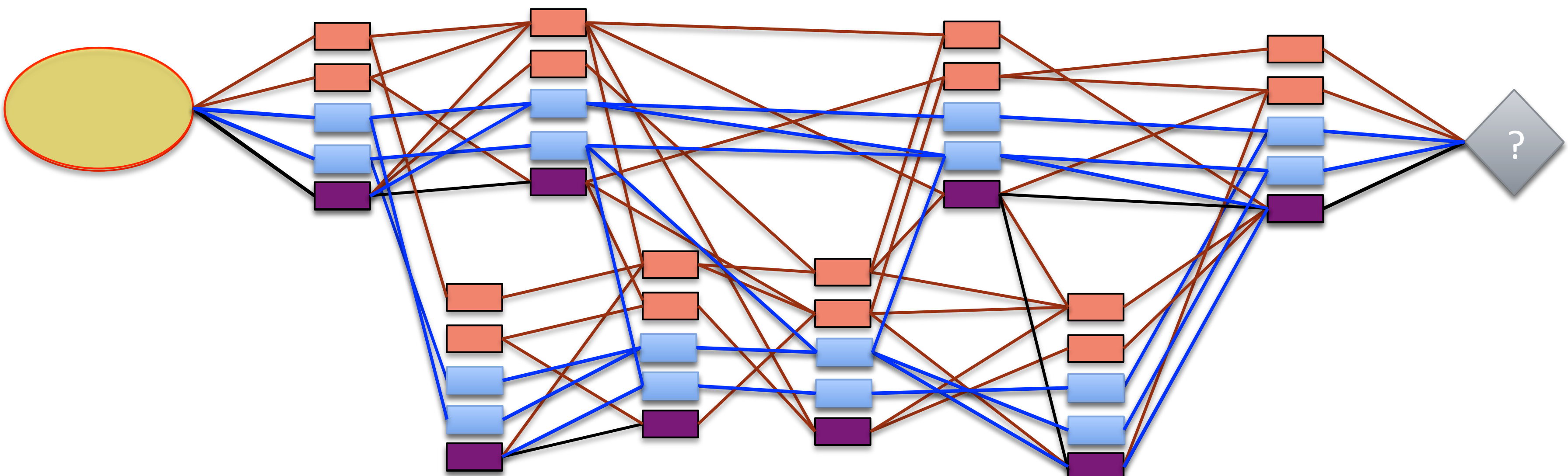
- Incremental update.
  - Only have to trace through dependency sub-graph affected by an update.
- Flexible policy expression.
  - Probe and source nodes are flexible to place and configure.
- Parallelization.
  - Can partition dependency graph into clusters to minimize inter-cluster dependences.

Distributed NetPlumber



Dependency Graph Clustering



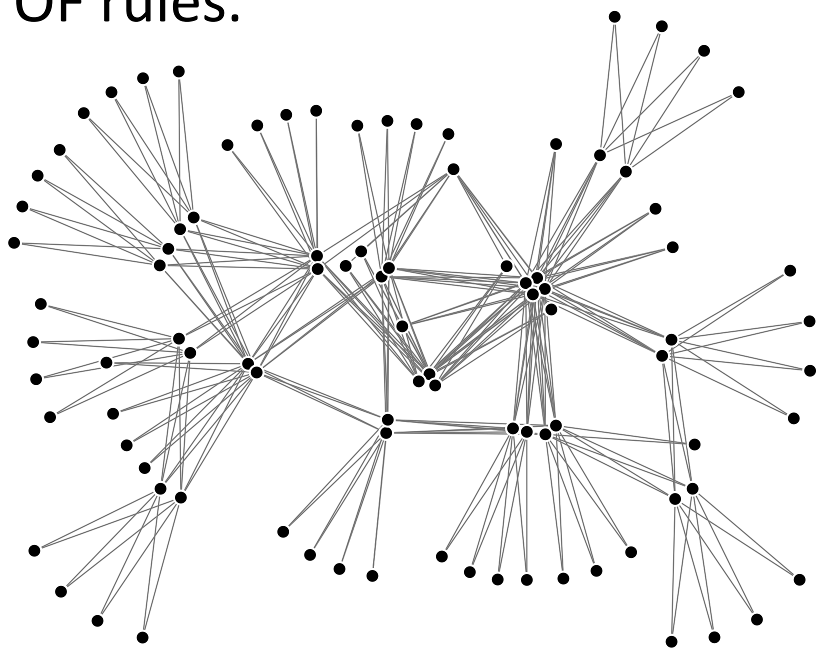


Outline

- Motivations.
- NetPlumber: Real time policy checking tool
  – How it works?
  – How to check policy?
  – How to parallelize?
- Evaluation on Google WAN.
- Conclusions.

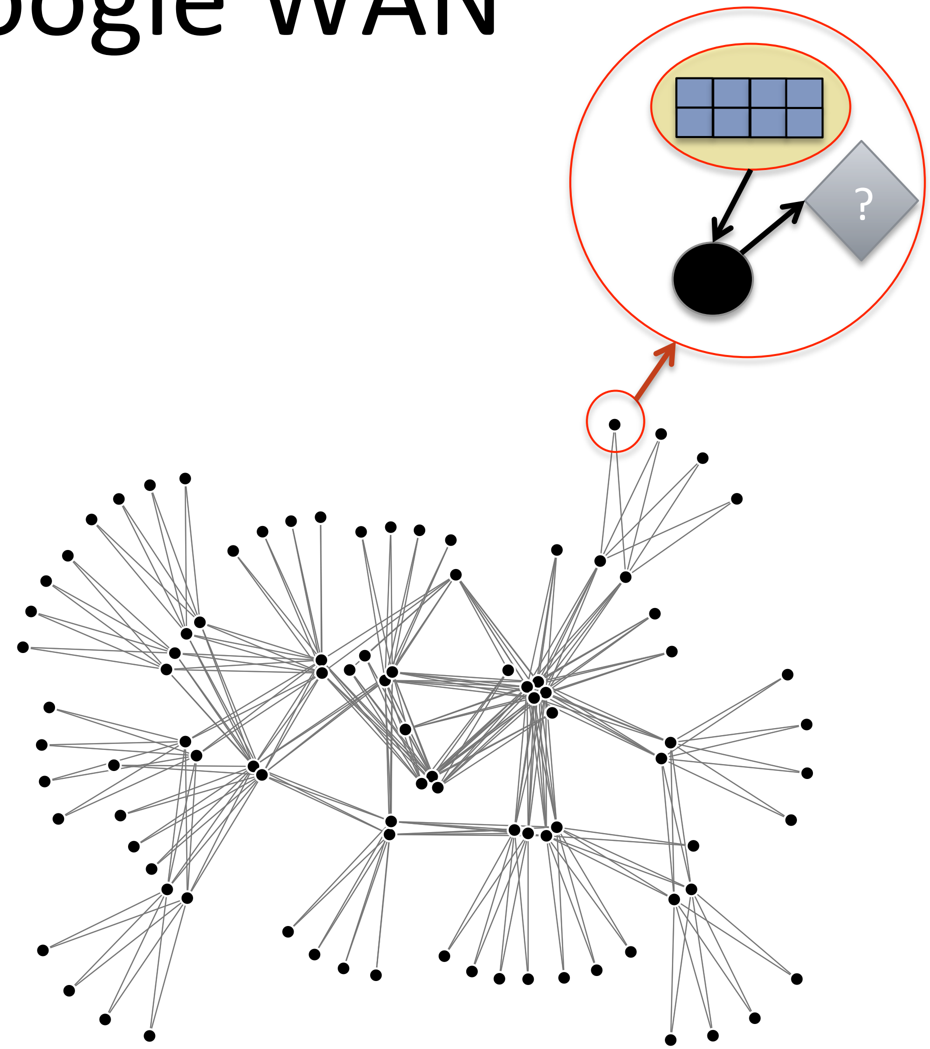
Experiment On Google WAN

- Google Inter-datacenter WAN.
  - Largest deployed SDN, running OpenFlow.
  - ~143,000 OF rules.



Experiment On Google WAN

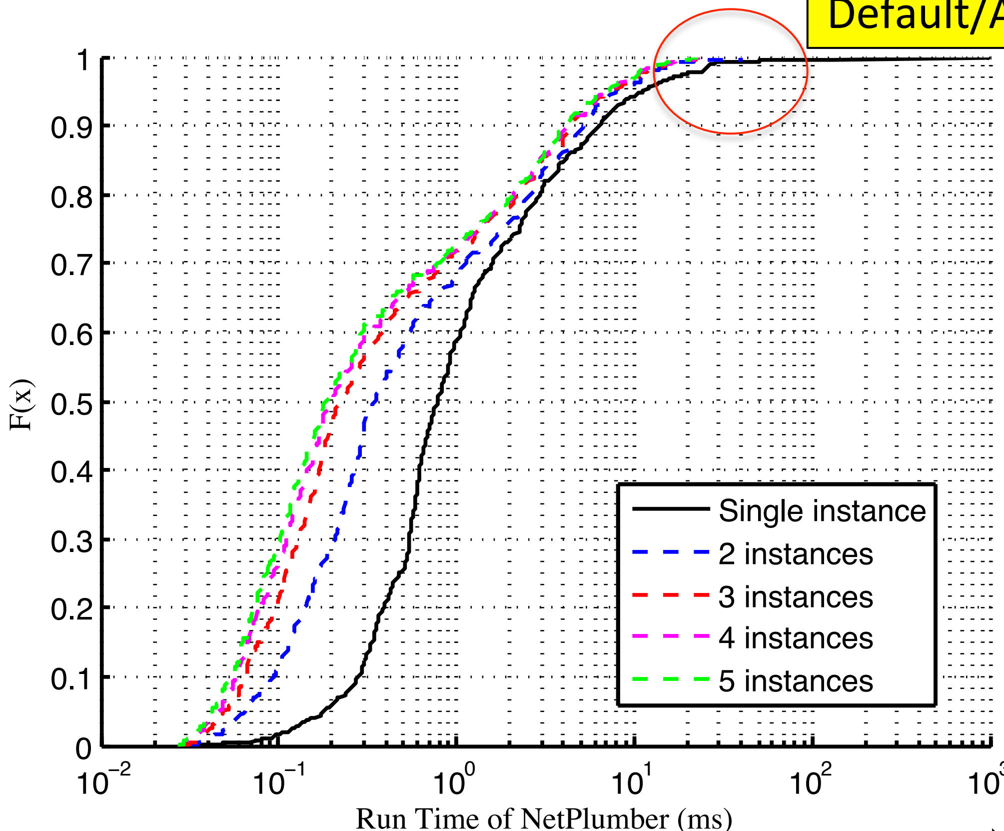
- Policy check: all 52 edge switches can talk to each other.
- More than 2500 pairwise reachability check.
- Used two snapshots taken 6 weeks apart.
- Used the first snapshot to create initial NetPlumber state and used the diff as a sequential update.



Experiment On Google WAN

Default/Aggregate Rules.

Run time with Hassel > 100s



Not much more benefit!

#instances:	1	2	3	4	5	8
median (ms)	0.77	0.35	0.23	0.2	0.185	0.180
mean (ms)	5.74	1.81	1.52	1.44	1.39	1.32

Benchmarking Experiment

- For a single pairwise reachability check.

#Network:	Google		Stanford		Internet 2	
Run Time	mean	median	mean	median	mean	median
Add Rule (ms)	0.28	0.23	0.2	0.065	0.53	0.52
Add Link (ms)	1510	1370	3020	2120	4760	2320

Conclusions

- Designed a protocol-independent system for real time network policy checking.
- Key component: dependency graph of forwarding rule, capturing all flow paths.
  - Incremental update.
  - Flexible policy expressions.
  - Parallelization by clustering.

Thank You!