SpeedLight: Synchronized Network Snapshots

Nofel Yaseen, John Sonchack, Vincent Liu

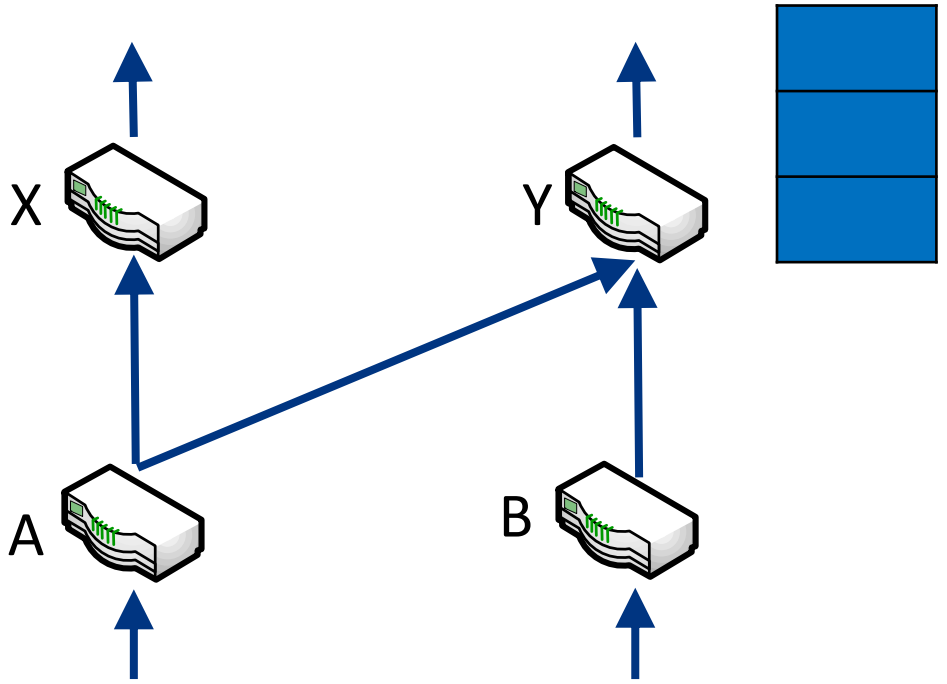
Network Measurements

Network Measurements

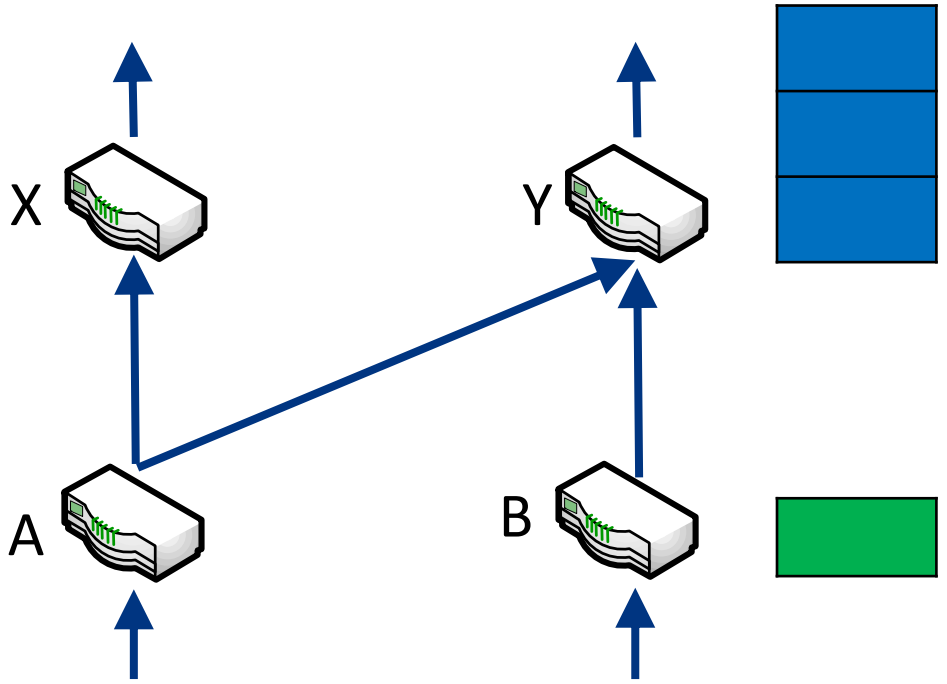
- Measurements are how we understand networks
  - Operators: configuration, management and provisioning
  - Architects: designing new protocols and topologies
  - Researchers: measurement studies and evaluation

- Today's measurement techniques
  - Single device, e.g., counters, sampling
  - Single path or packet, e.g., pings, INT, ECN

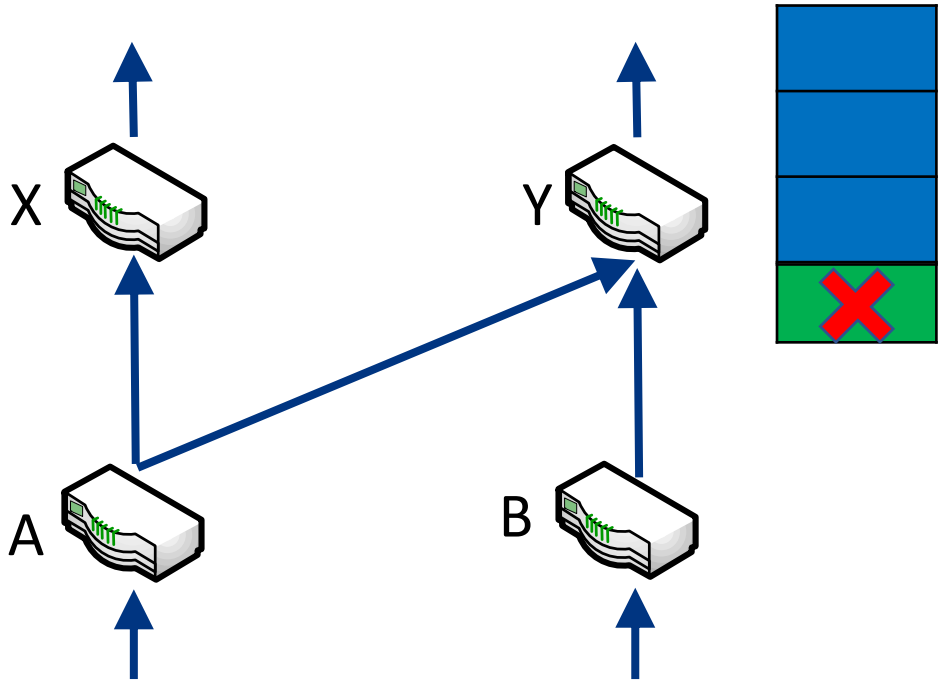
A Case for Consistency



A Case for Consistency

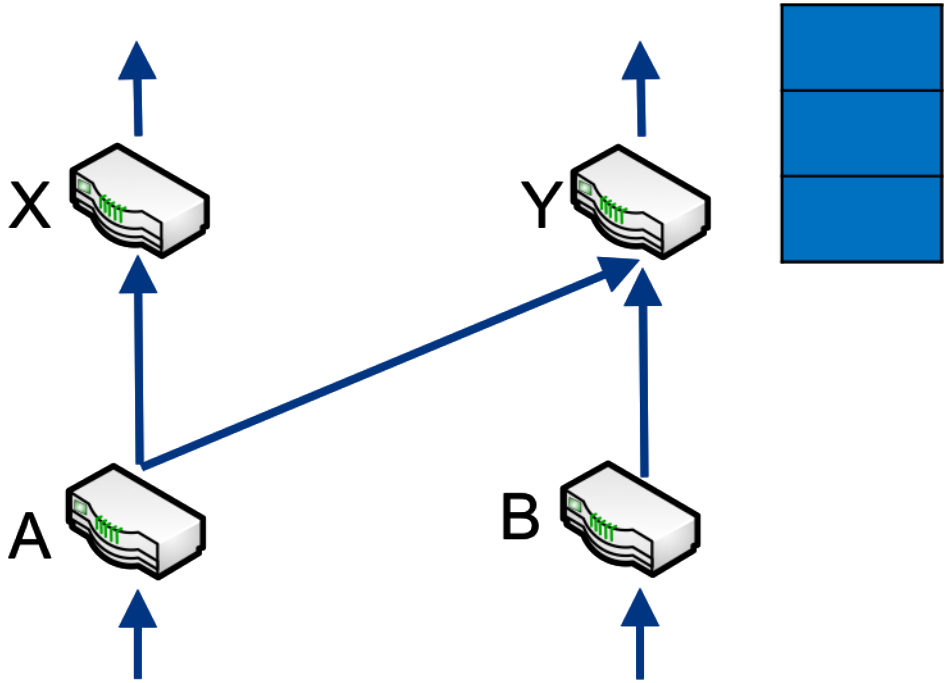


A Case for Consistency

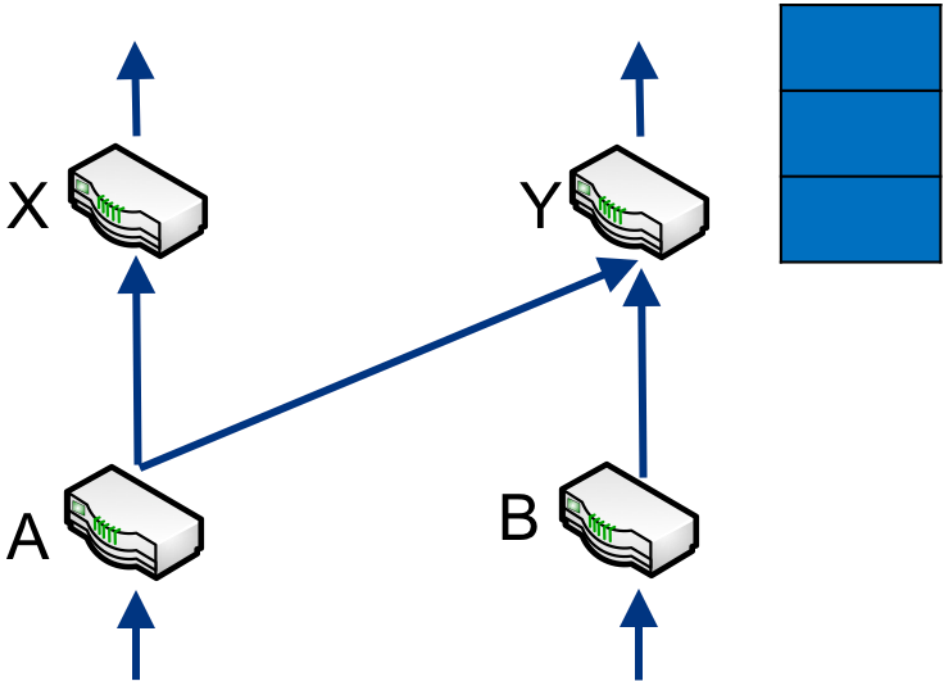


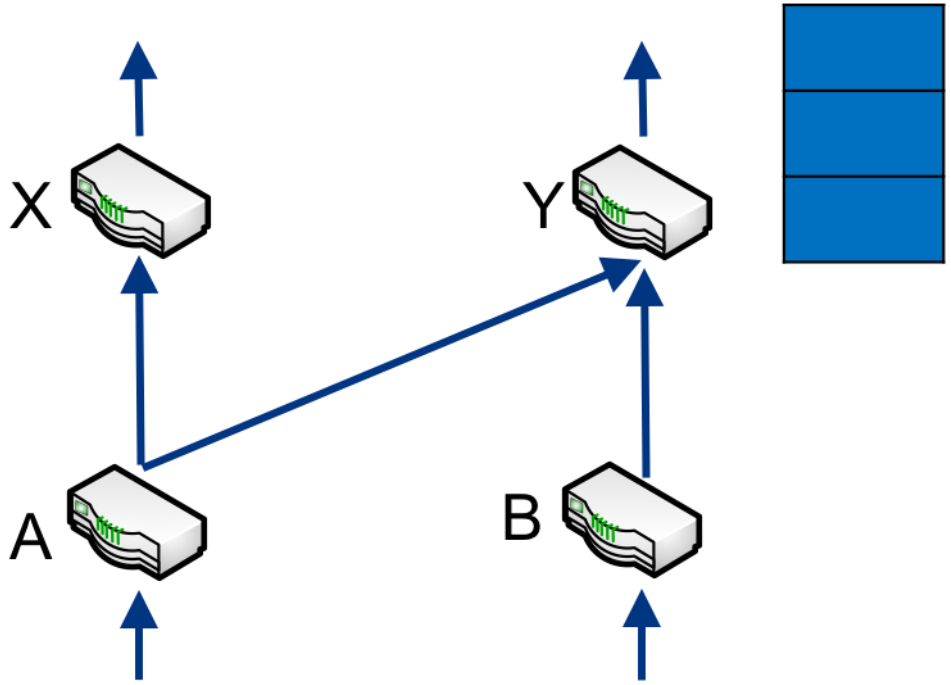
What is the reason for this packet drop?

A Case for Consistency

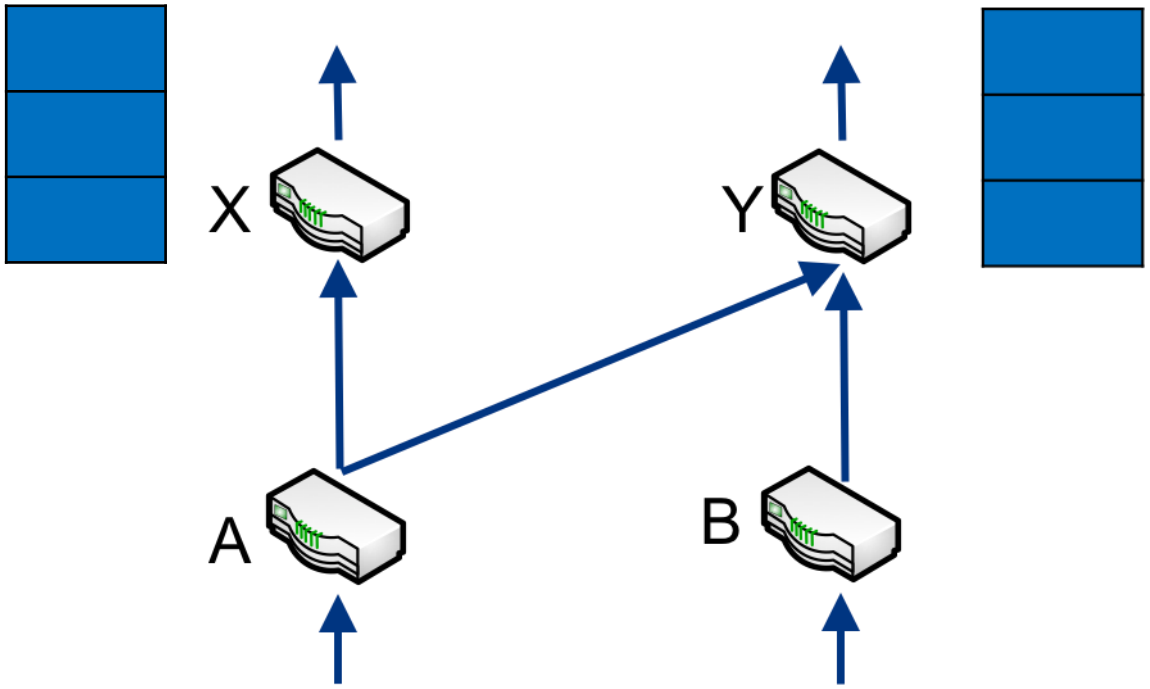


A Case for Consistency

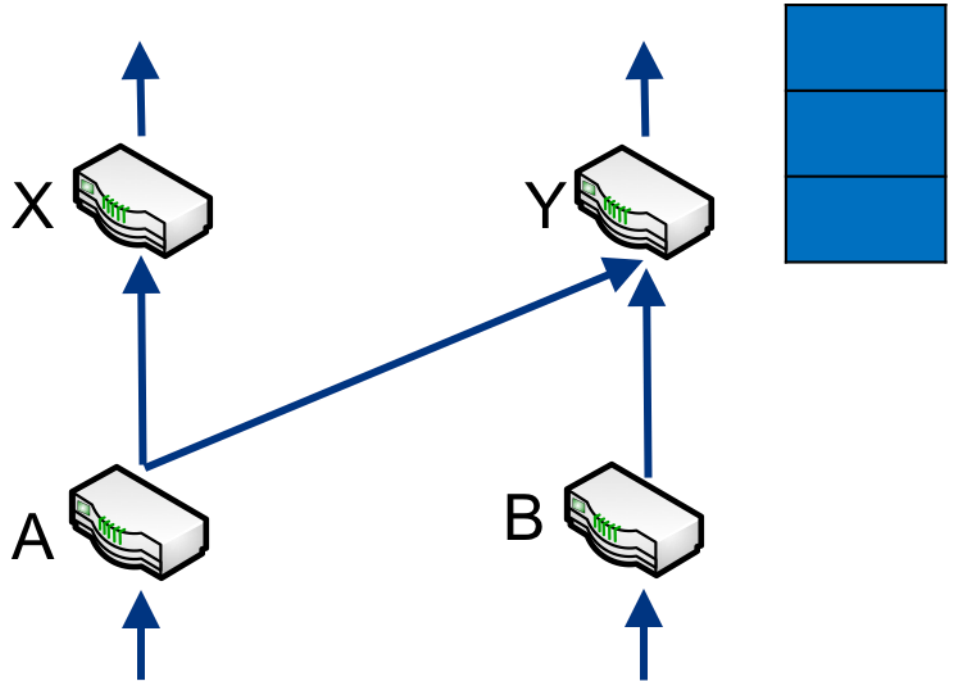




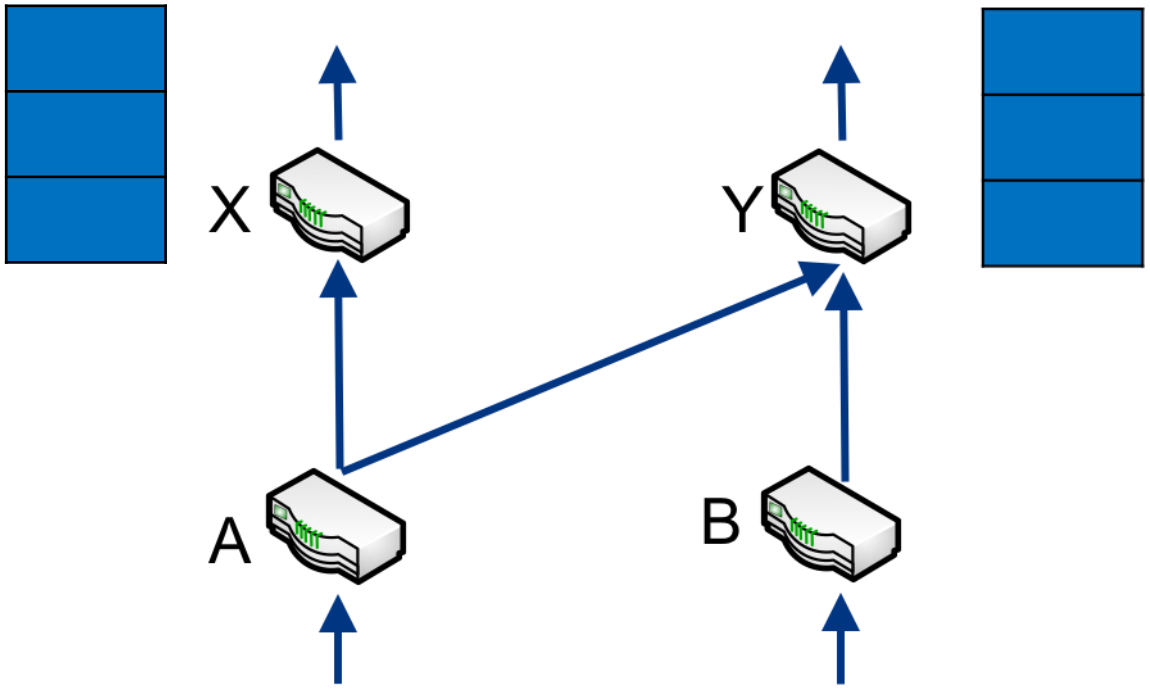
A Case for Consistency



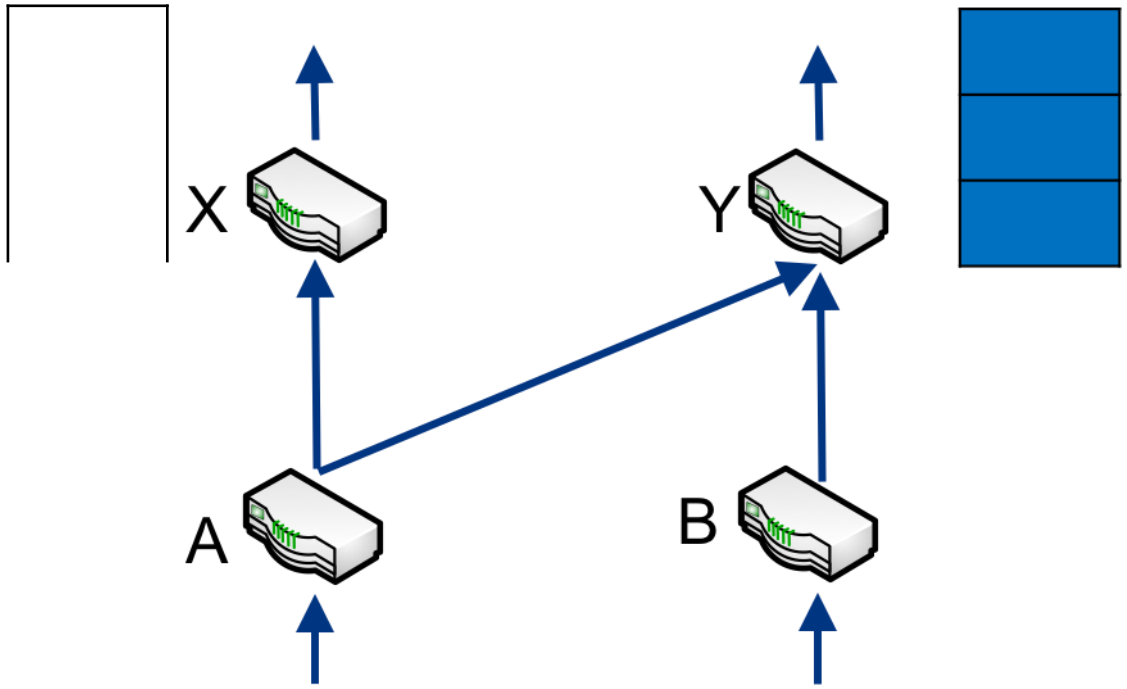
Congestion



A Case for Consistency

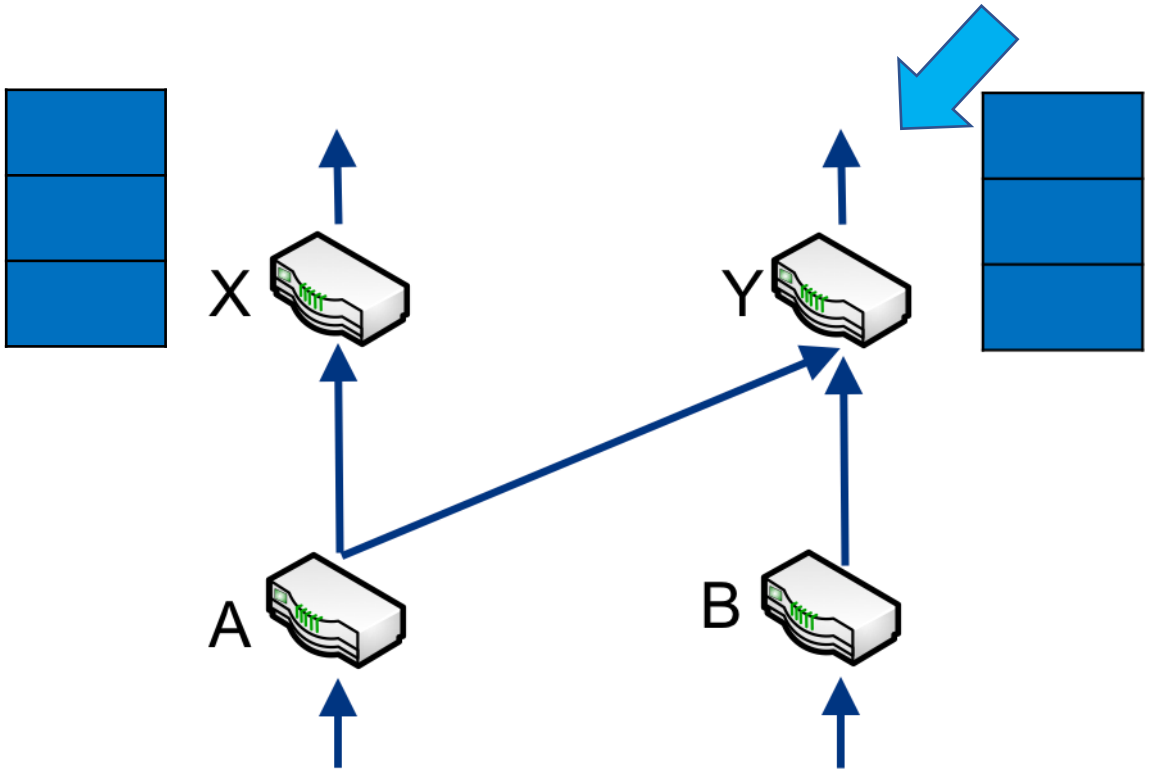


Congestion

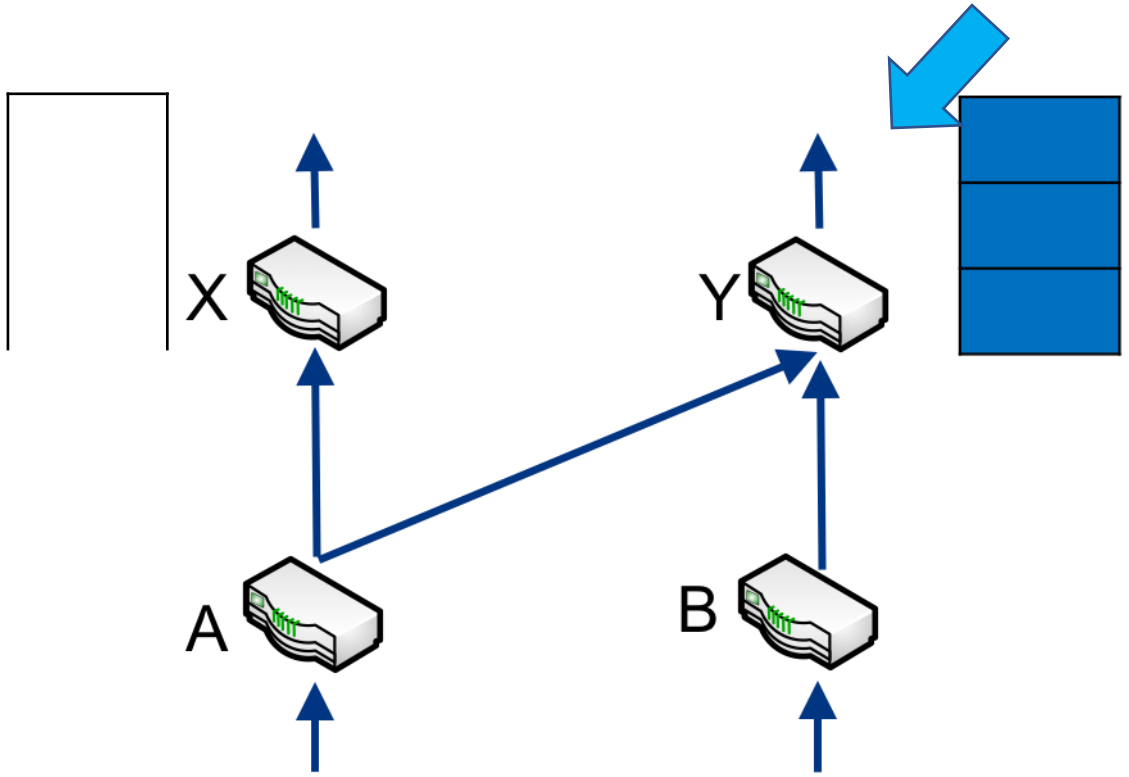


Poor Load Balancing

A Case for Consistency



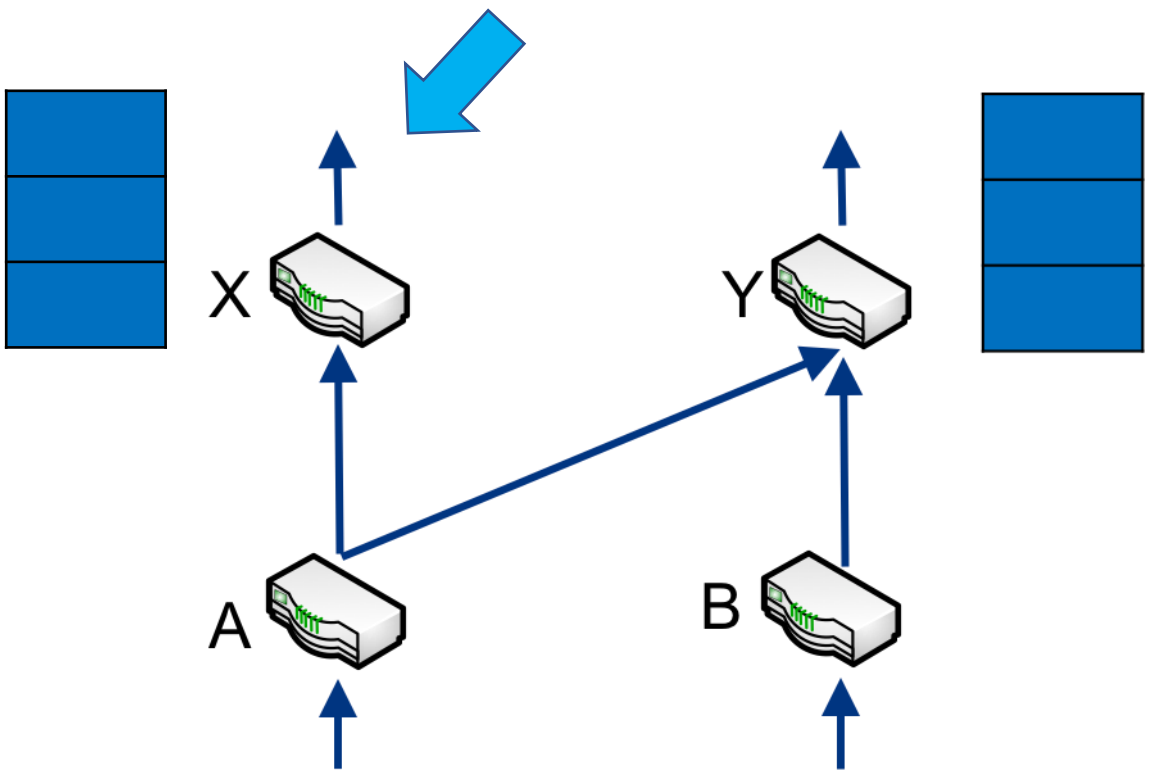
Congestion



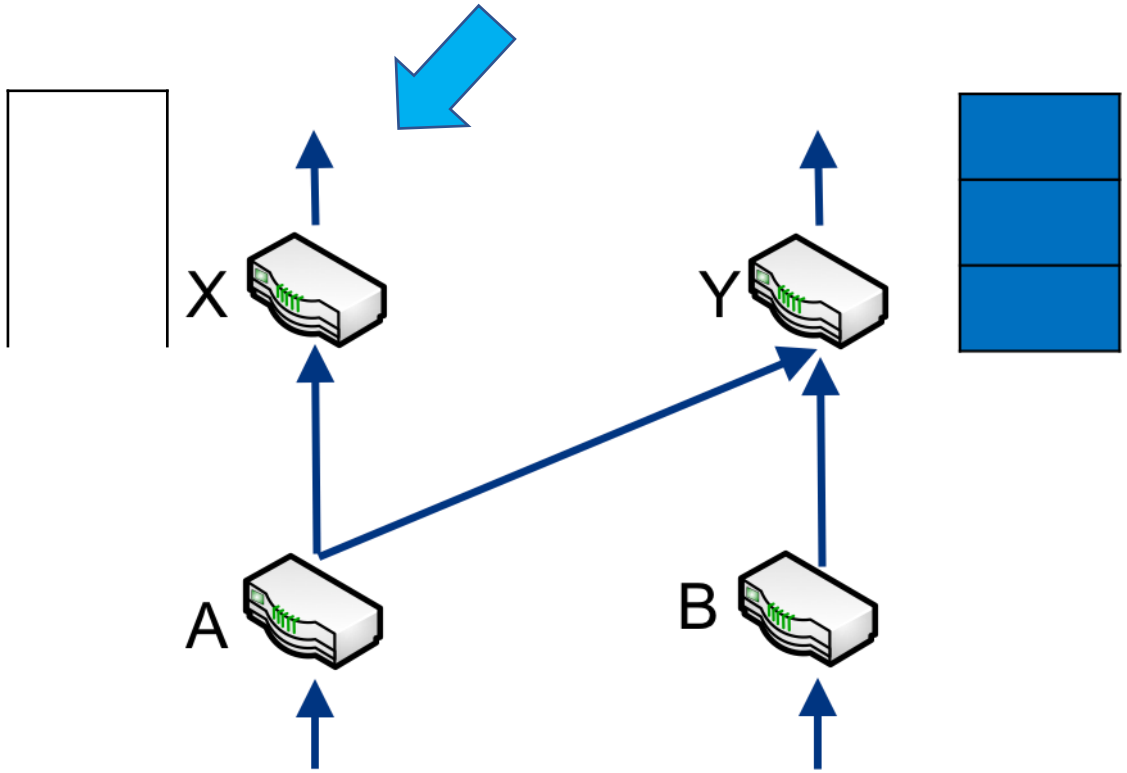
Poor Load Balancing

- **Single Device**: No relationship among measurements across time or devices.

A Case for Consistency



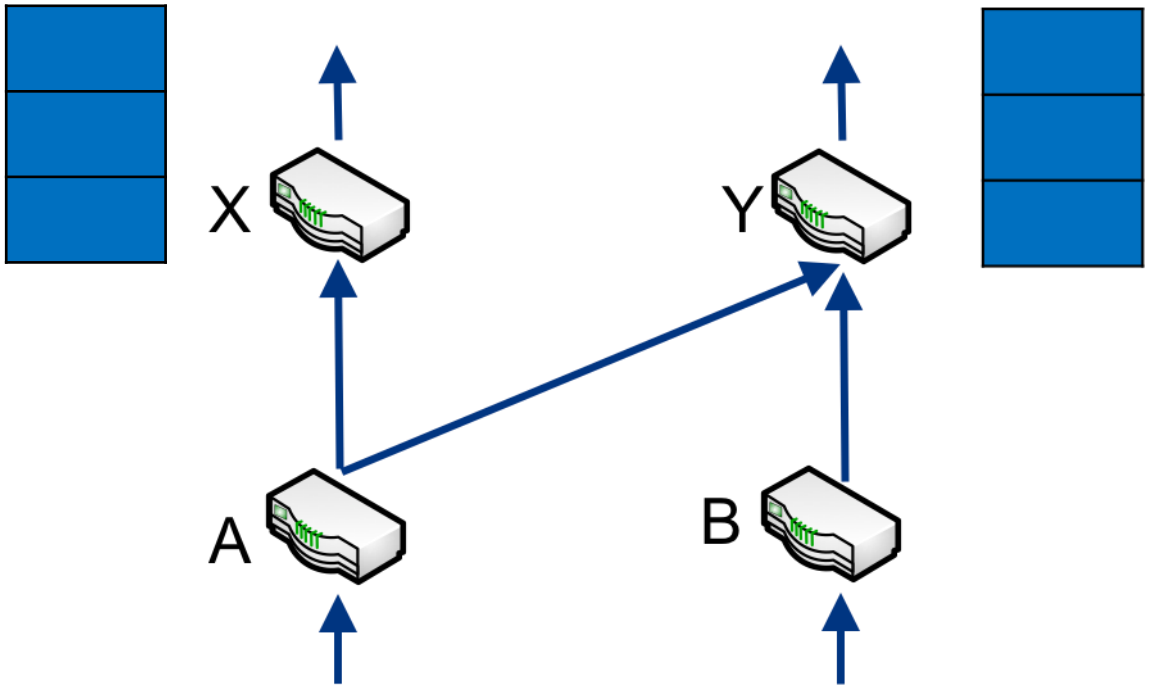
Congestion



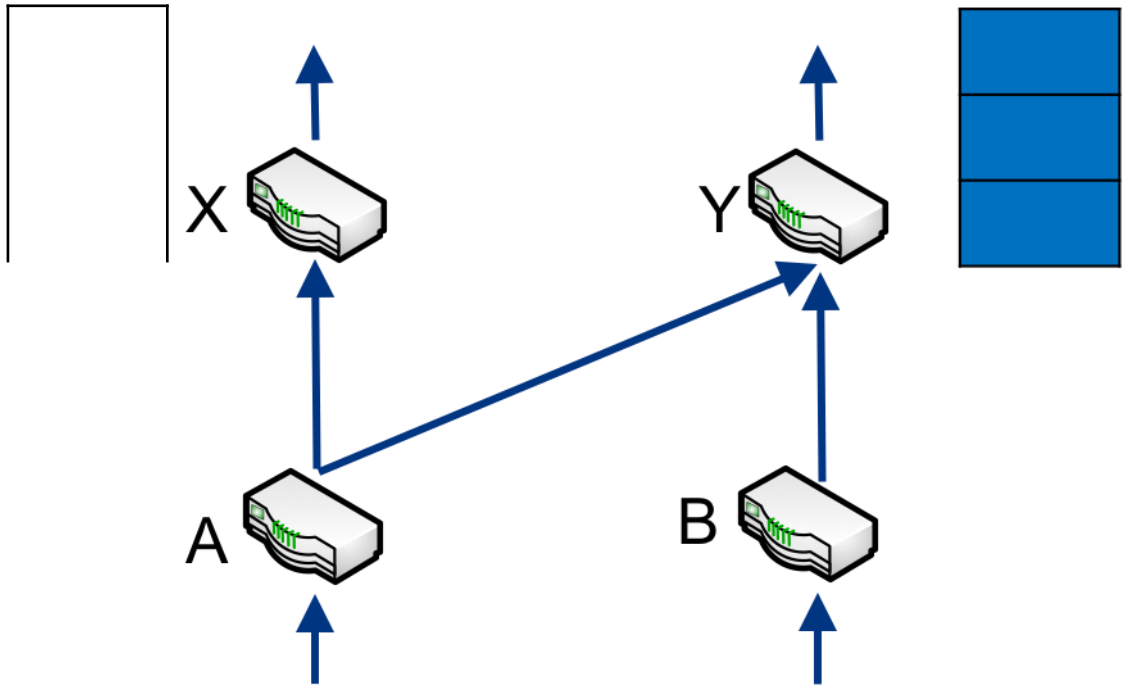
Poor Load Balancing

- **Single Device**: No relationship among measurements across time or devices.

A Case for Consistency



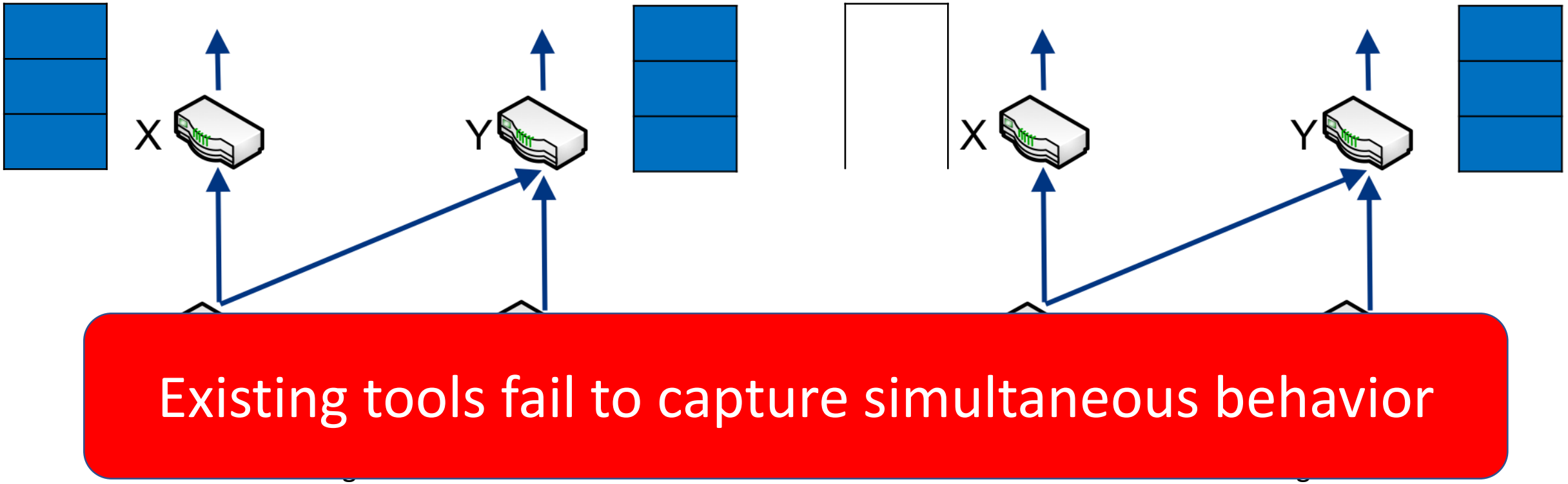
Congestion



Poor Load Balancing

- **Single Device**: No relationship among measurements across time or devices.
- **Single Path or Packet:** No relationship among measurements across paths or packets.

A Case for Consistency



- **Single Device**: No relationship among measurements across time or devices.
- **Single Path or Packet:** No relationship among measurements across paths or packets.

Our Goal

Our Goal

A set of data-plane measurements that capture the state of the network at ~(single point in time)

Truly simultaneous behavior is not possible

- Causal consistency, i.e., the set should make sense
- Near synchrony, i.e., it should be as close as possible to an actual state (<RTT)

Speedlight

A set of data-plane measurements that capture the state of the network at ~(single point in time)

Speedlight

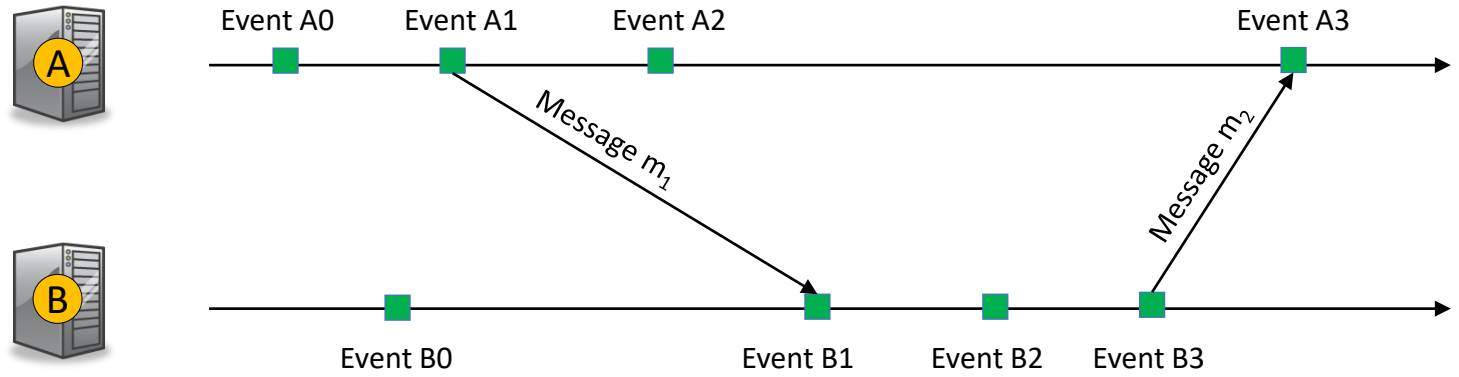
A set of data-plane measurements that capture the state of the network at ~(single point in time)

- A P4-based system for Synchronized Network Snapshot
  - Implemented on Wedge100BF
  - Can capture network-wide state of any value accessible in the data plane
  - Amenable to partial deployment
  - <100µs synchronization, even for large networks

Outline

- Chandy - Lamport Algorithm.
- Challenges of taking Synchronized Network Snapshots.
- Protocol
- Prototype Implementation
- Evaluation

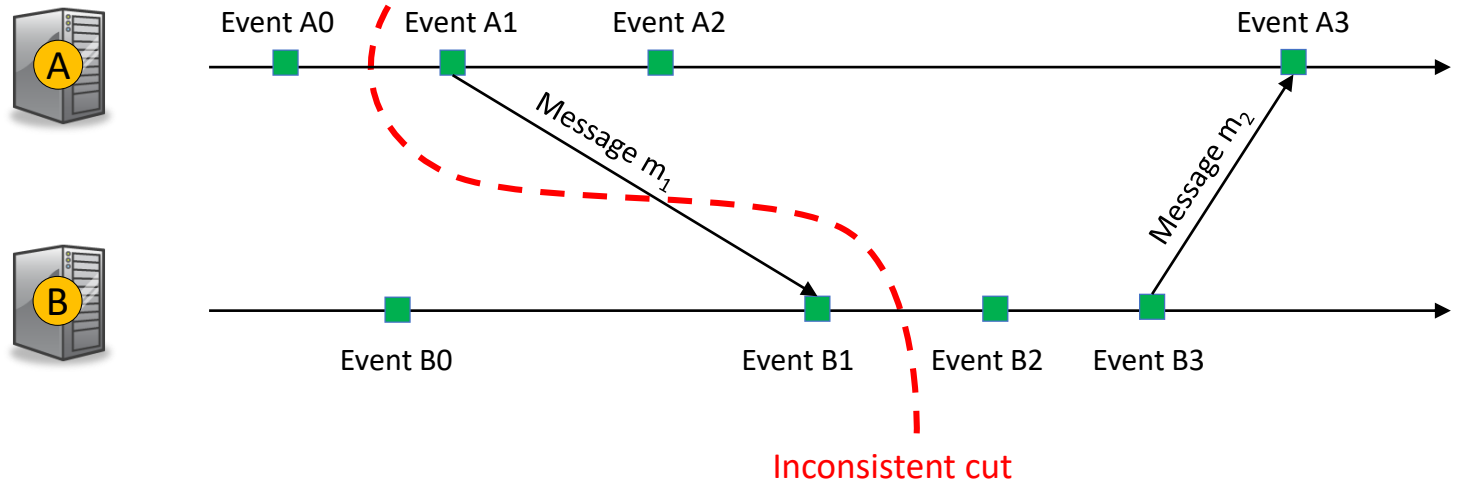
Global Network View



- Partition the network into *pre*- and *post*-snapshot
  - *e* is pre-snapshot ⇒ all events that caused *e* are pre-snapshot
  - E.g., receive and send of a message

Figure adapted from Linh T. X. Phan

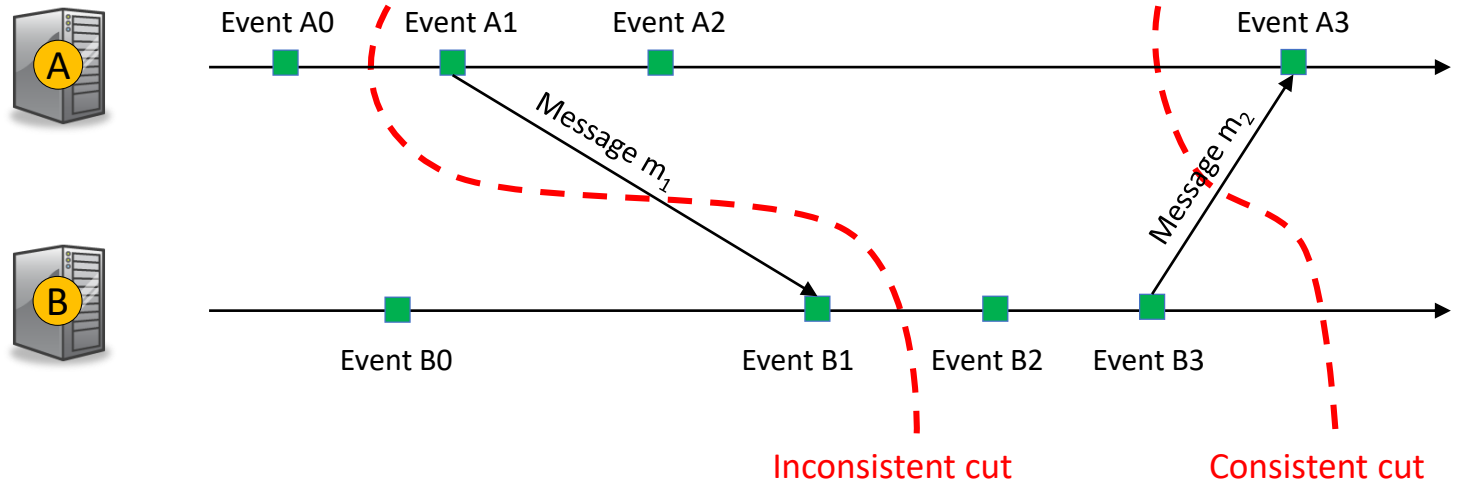
Global Network View



- Partition the network into *pre*- and *post*-snapshot
  - *e* is pre-snapshot ⇒ all events that caused *e* are pre-snapshot
  - E.g., receive and send of a message

Figure adapted from Linh T. X. Phan

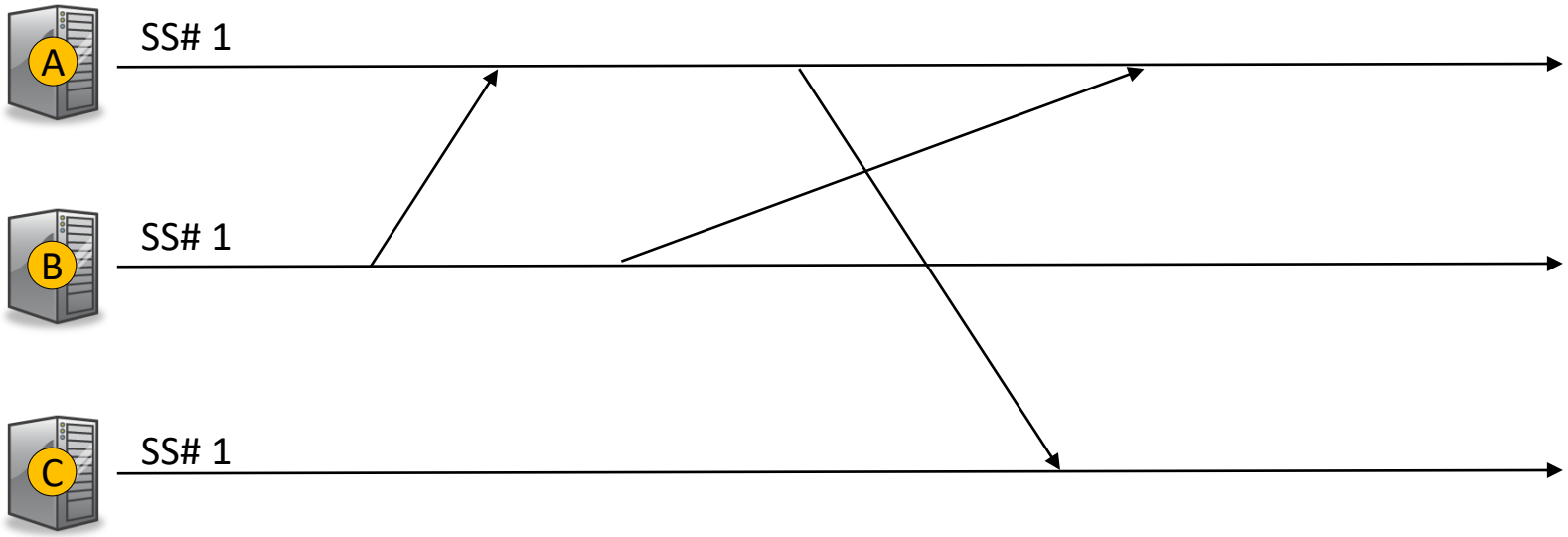
Global Network View



- Partition the network into *pre*- and *post*-snapshot
  - *e* is pre-snapshot ⇒ all events that caused *e* are pre-snapshot
  - E.g., receive and send of a message

Figure adapted from Linh T. X. Phan

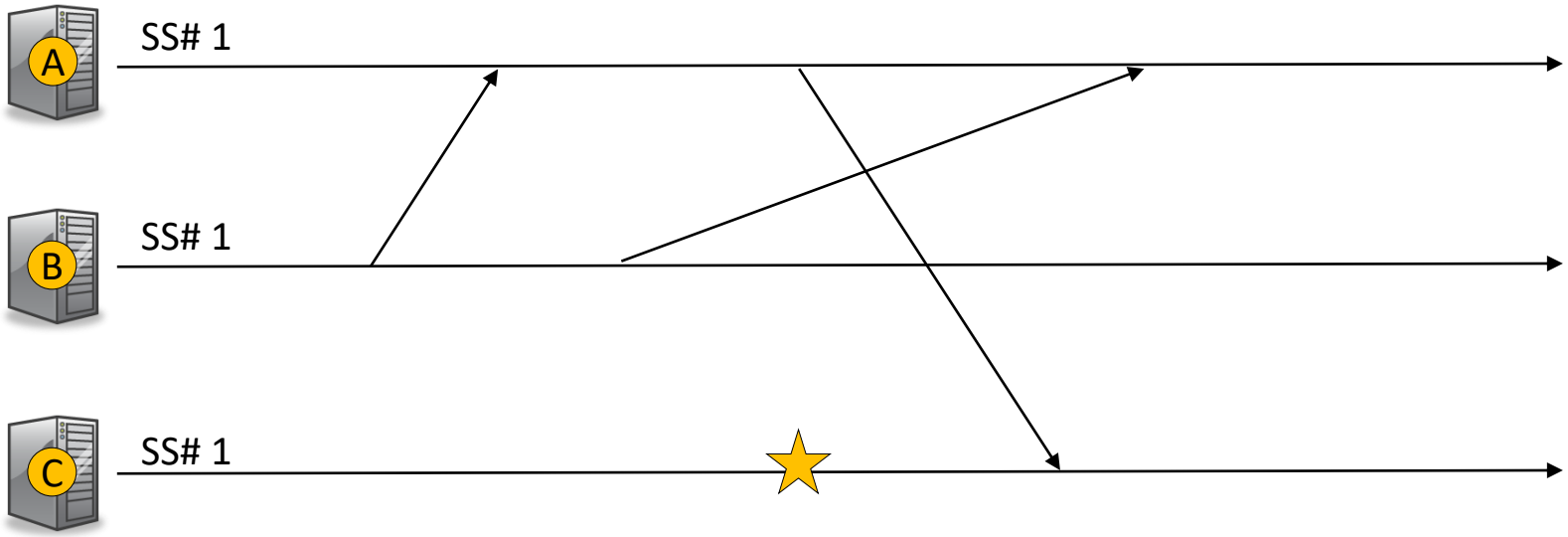
Chandy–Lamport (CL) Snapshots



- Messages carry the current SS#
- On seeing a message with a new SS# for the first time
  - Node takes a local checkpoint
  - Node attaches the new SS# to all subsequent messages
- On seeing a message with an old SS#
  - Message was in-flight. Update channel state.

Figure adapted from Linh T. X. Phan

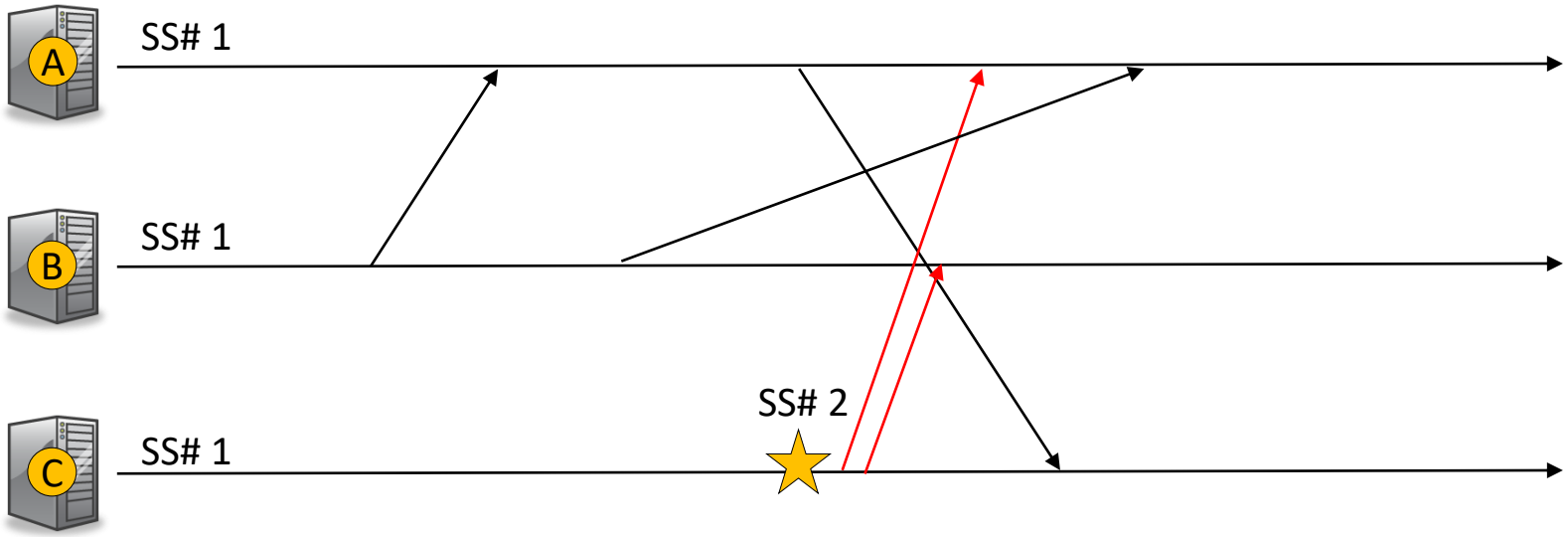
Chandy–Lamport (CL) Snapshots



- Messages carry the current SS#
- On seeing a message with a new SS# for the first time
  - Node takes a local checkpoint
  - Node attaches the new SS# to all subsequent messages
- On seeing a message with an old SS#
  - Message was in-flight. Update channel state.

Figure adapted from Linh T. X. Phan

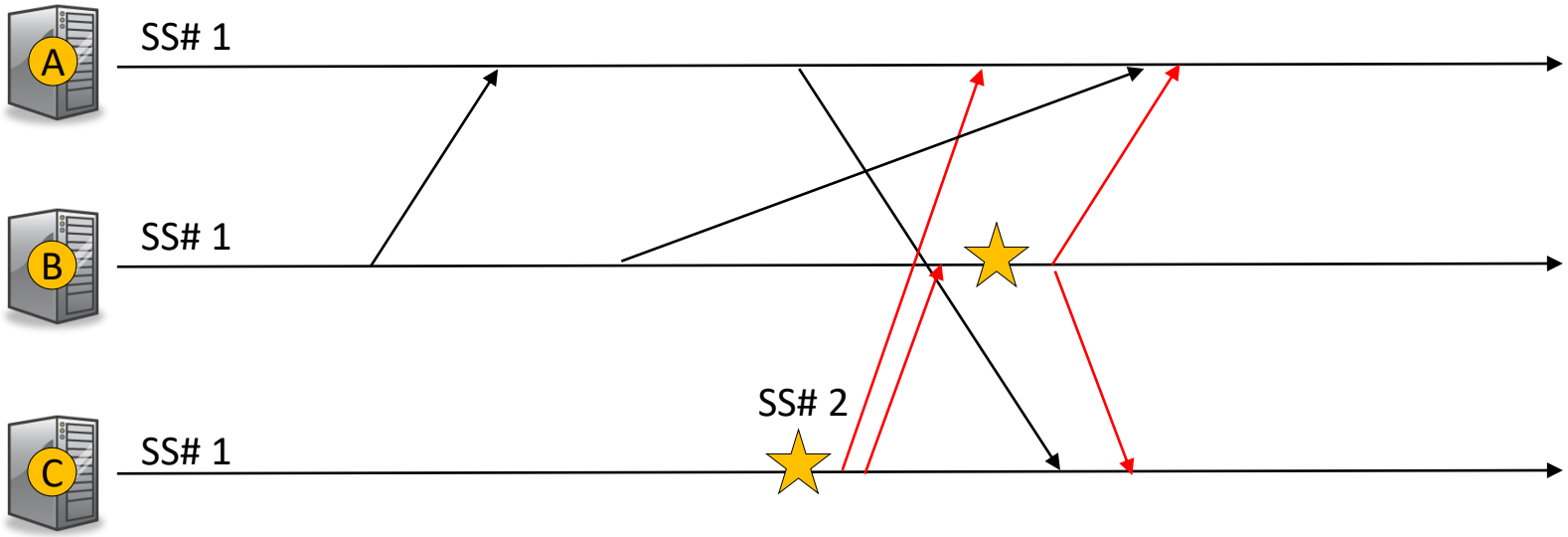
Chandy–Lamport (CL) Snapshots



- Messages carry the current SS#
- On seeing a message with a new SS# for the first time
  - Node takes a local checkpoint
  - Node attaches the new SS# to all subsequent messages
- On seeing a message with an old SS#
  - Message was in-flight. Update channel state.

Figure adapted from Linh T. X. Phan

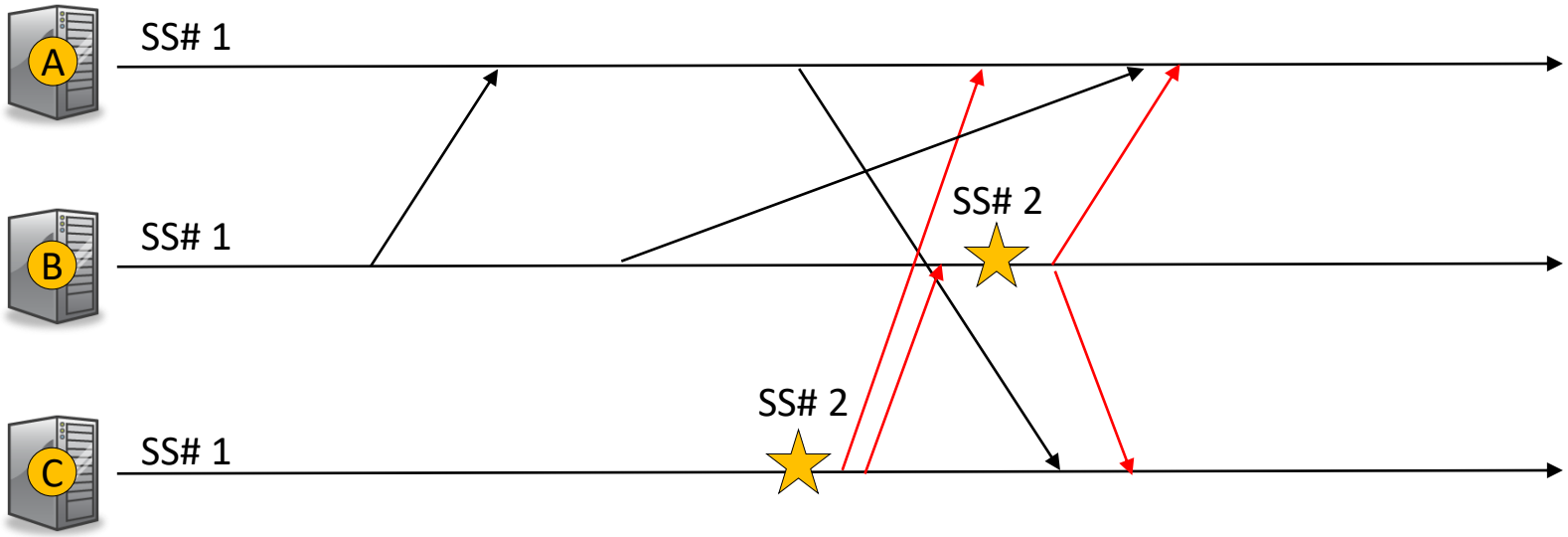
Chandy–Lamport (CL) Snapshots



- Messages carry the current SS#
- On seeing a message with a new SS# for the first time
  - Node takes a local checkpoint
  - Node attaches the new SS# to all subsequent messages
- On seeing a message with an old SS#
  - Message was in-flight. Update channel state.

Figure adapted from Linh T. X. Phan

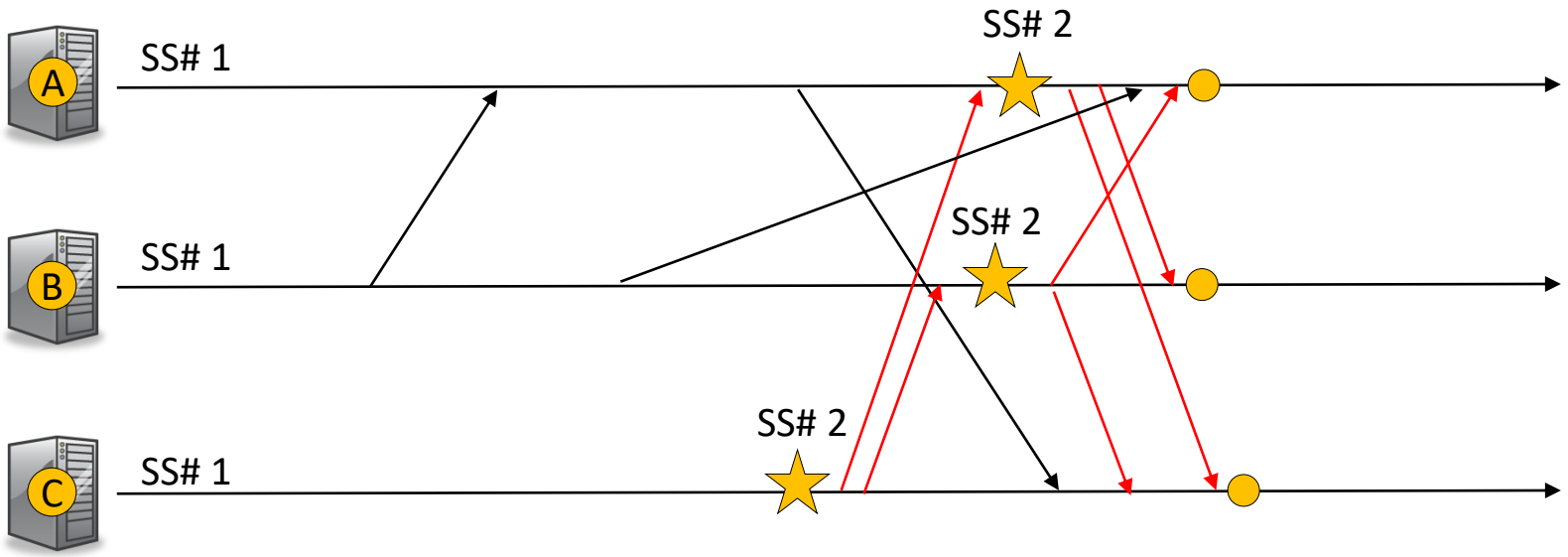
Chandy–Lamport (CL) Snapshots



- Messages carry the current SS#
- On seeing a message with a new SS# for the first time
  - Node takes a local checkpoint
  - Node attaches the new SS# to all subsequent messages
- On seeing a message with an old SS#
  - Message was in-flight. Update channel state.

Figure adapted from Linh T. X. Phan

Chandy–Lamport (CL) Snapshots



- Messages carry the current SS#
- On seeing a message with a new SS# for the first time
  - Node takes a local checkpoint
  - Node attaches the new SS# to all subsequent messages
- On seeing a message with an old SS#
  - Message was in-flight. Update channel state.

Figure adapted from Linh T. X. Phan

Challenges for Synchronized Network Snapshots

Challenges for Synchronized Network Snapshots

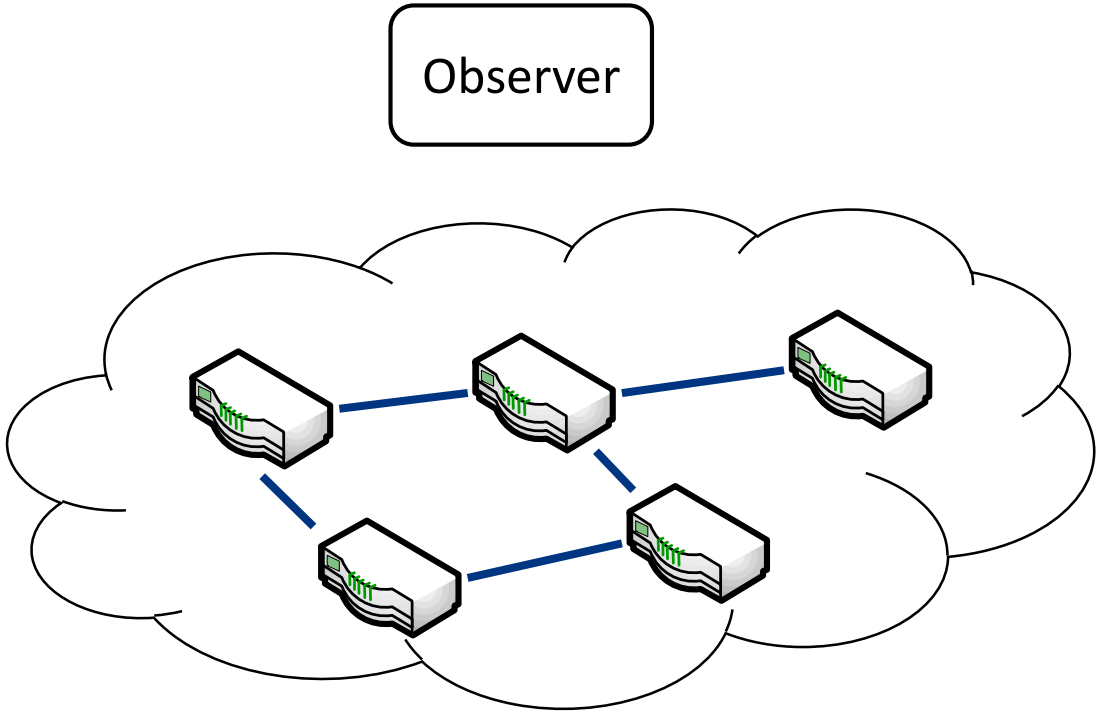
1. CL provides no guarantee of synchrony
   - We want something that's close to an actual state

2. CL assumes single-threaded nodes, FIFO channels
   - Modern networks are highly parallel – breaks consistency

3. CL assumes general purpose CPUs
   - Switch data planes are extremely limited
   - Switch CPUs are no better than remote hosts (wrt consistency)

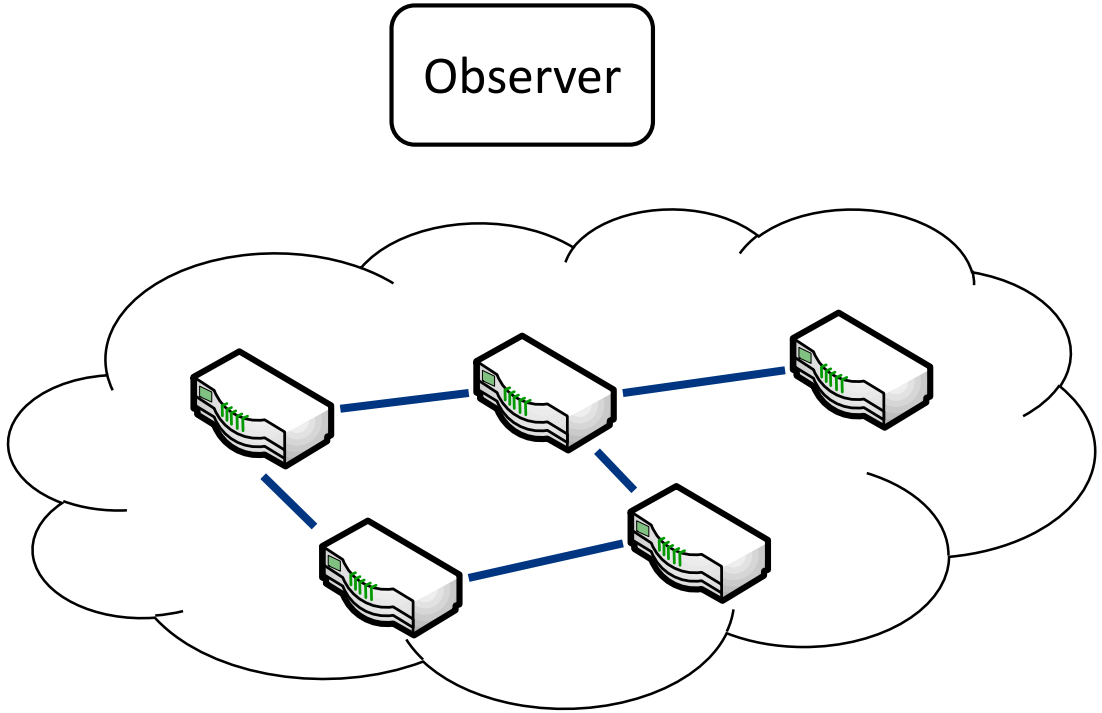
Ensuring Synchrony

Challenge 1: Chandy- Lamport provides no guarantee of synchrony



Ensuring Synchrony

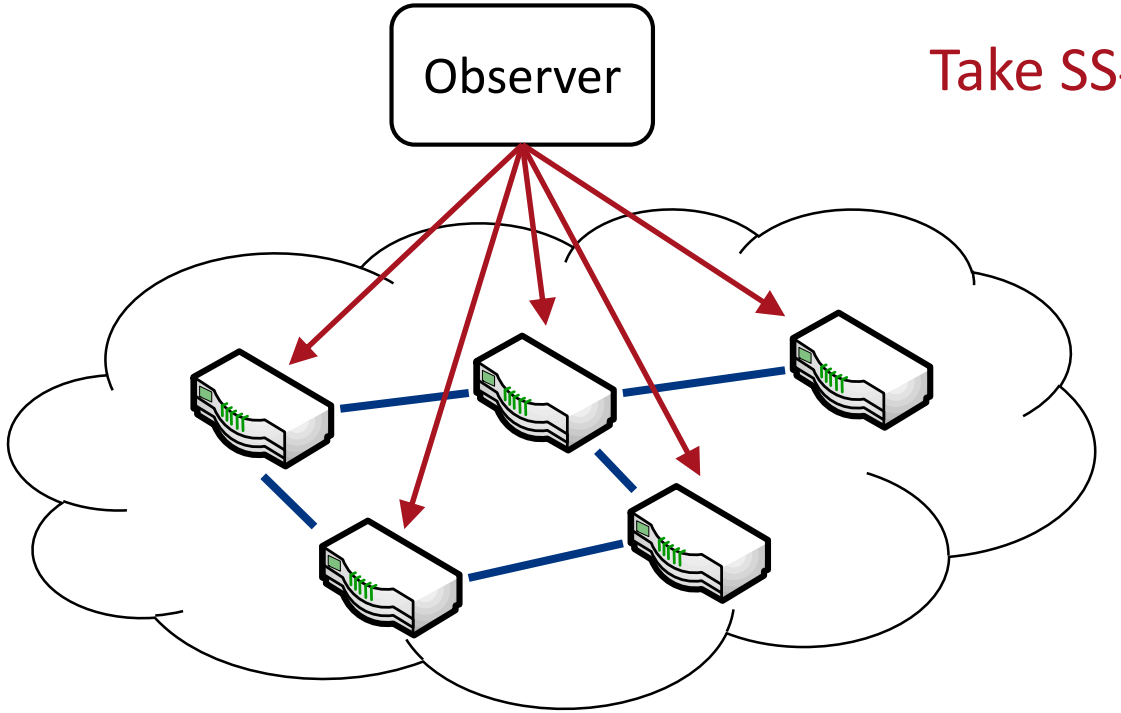
Challenge 1: Chandy- Lamport provides no guarantee of synchrony



- Router CPUs are synchronized via PTP

Ensuring Synchrony

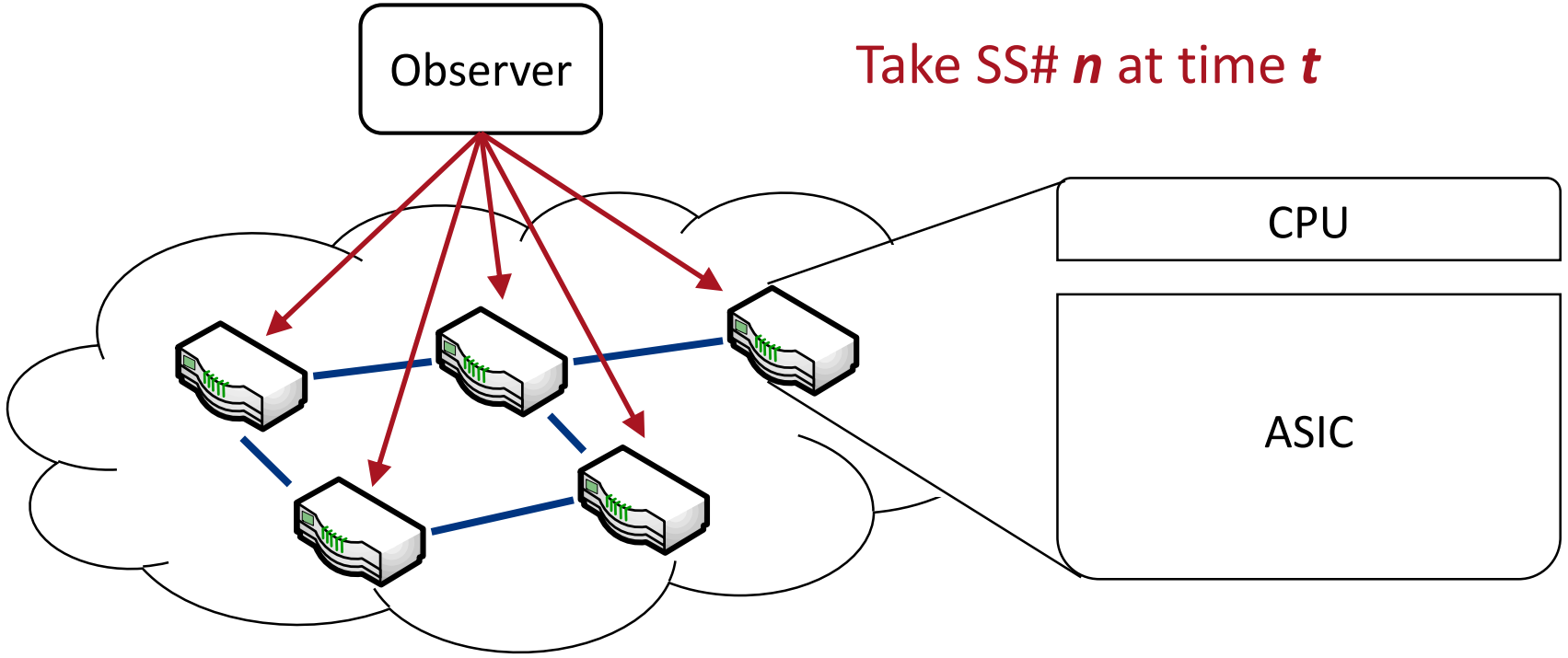
Challenge 1: Chandy- Lamport provides no guarantee of synchrony



- Router CPUs are synchronized via PTP
- User/Observer schedules a snapshot at *every* router

Ensuring Synchrony

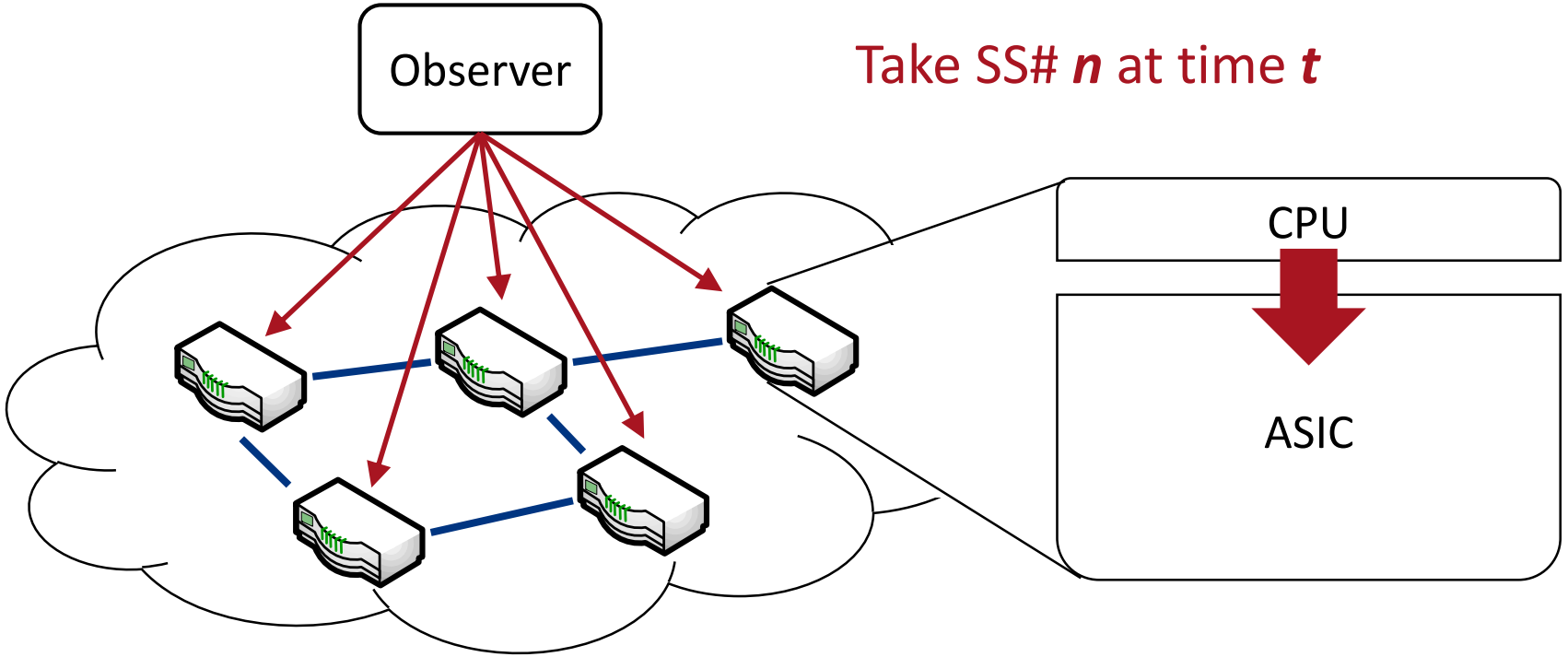
Challenge 1: Chandy- Lamport provides no guarantee of synchrony



- Router CPUs are synchronized via PTP
- User/Observer schedules a snapshot at *every* router

Ensuring Synchrony

Challenge 1: Chandy- Lamport provides no guarantee of synchrony



- Router CPUs are synchronized via PTP
- User/Observer schedules a snapshot at *every* router

Ensuring Consistency

Challenge 2: CL assumes single-threaded nodes, FIFO channels

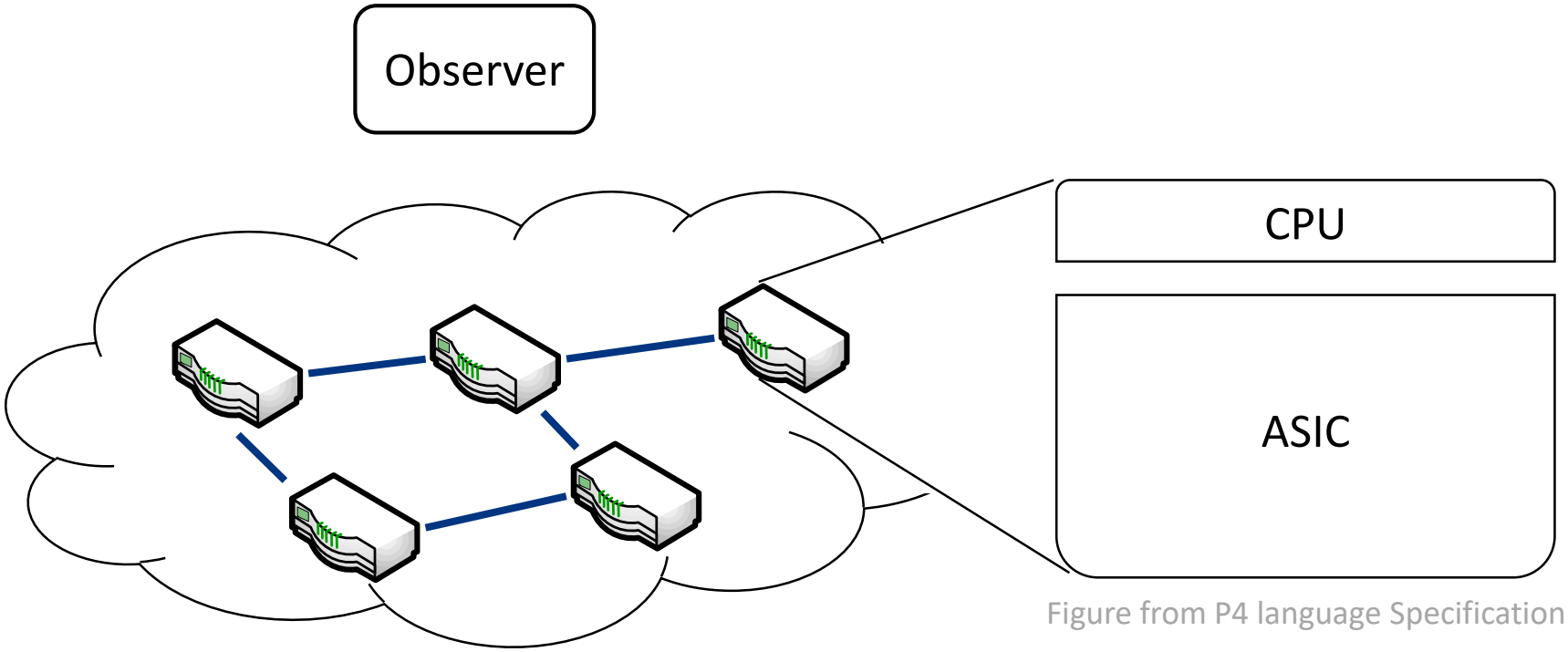
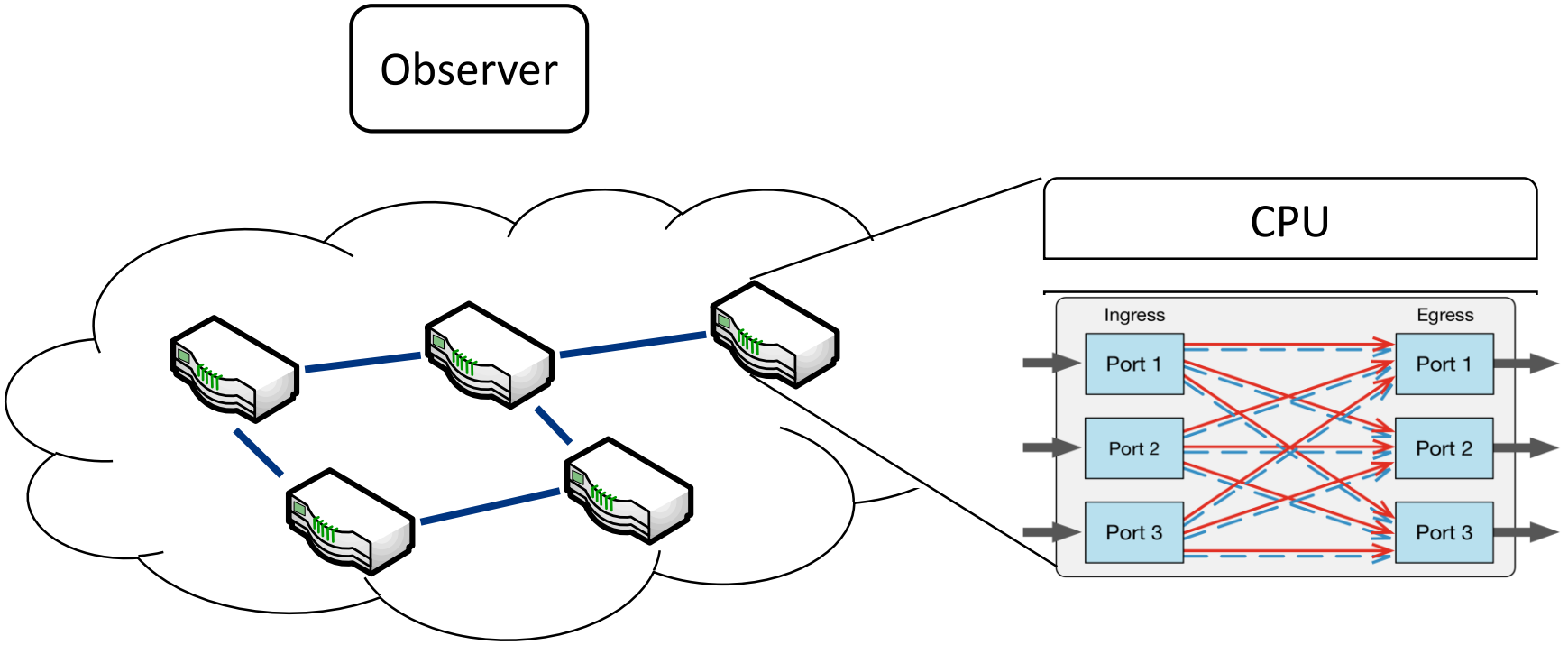


Figure from P4 language Specification

Ensuring Consistency

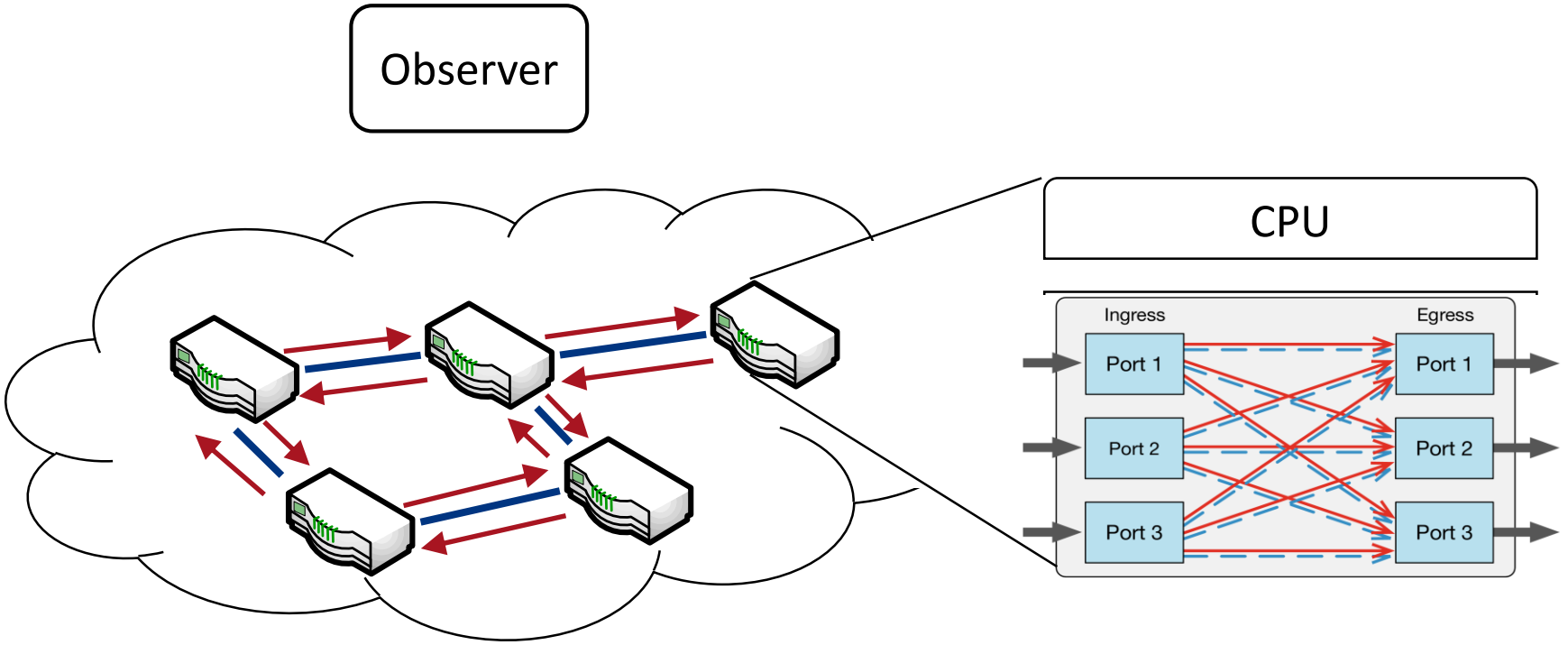
Challenge 2: CL assumes single-threaded nodes, FIFO channels



- Data plane snapshot on the level of individual processing units and priority channels

Ensuring Consistency

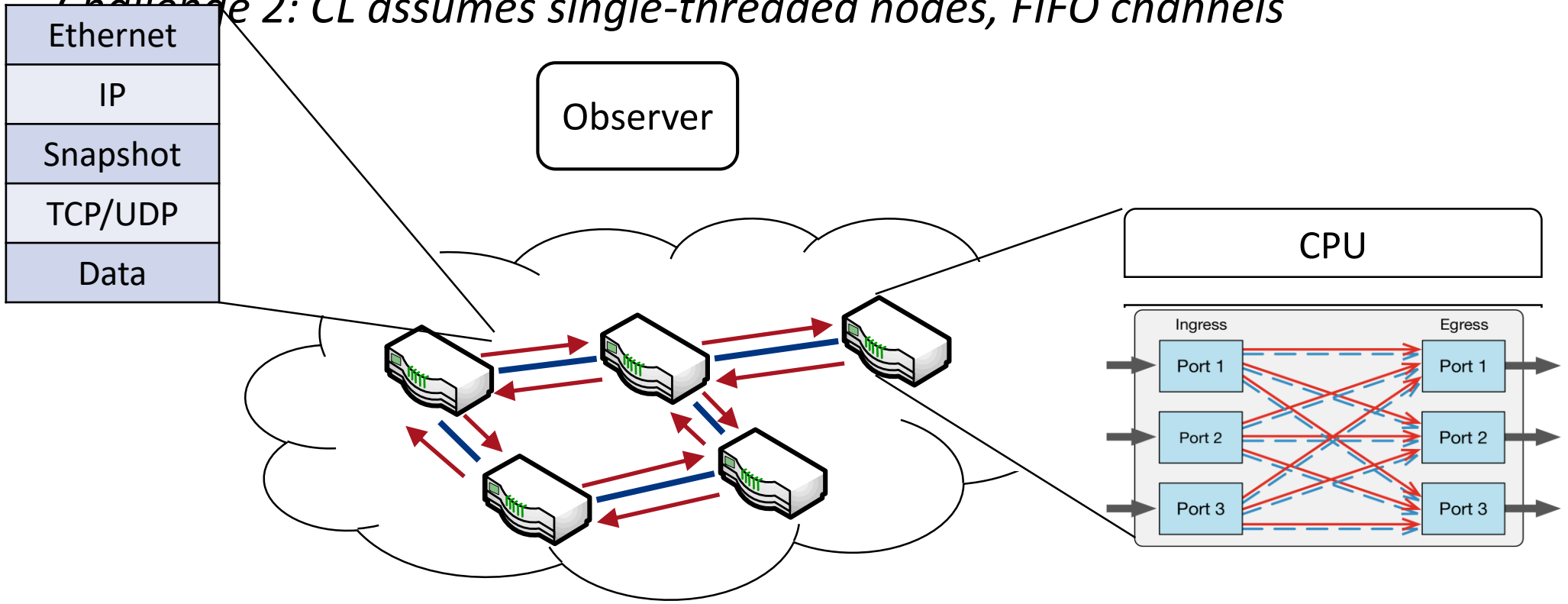
Challenge 2: CL assumes single-threaded nodes, FIFO channels



- Data plane snapshot on the level of individual processing units and priority channels
- Snapshot propagates even if CPU invocation is delayed

Ensuring Consistency

Challenge 2: CL assumes single-threaded nodes, FIFO channels



- Data plane snapshot on the level of individual processing units and priority channels
- Snapshot propagates even if CPU invocation is delayed

Compensate for Data-plane Limitations

Challenge 3: CL assumes general purpose CPUs

Compensate for Data-plane Limitations

Challenge 3: CL assumes general purpose CPUs

- Programmable ASICs are limited
  - Limited programming model, registers and accesses

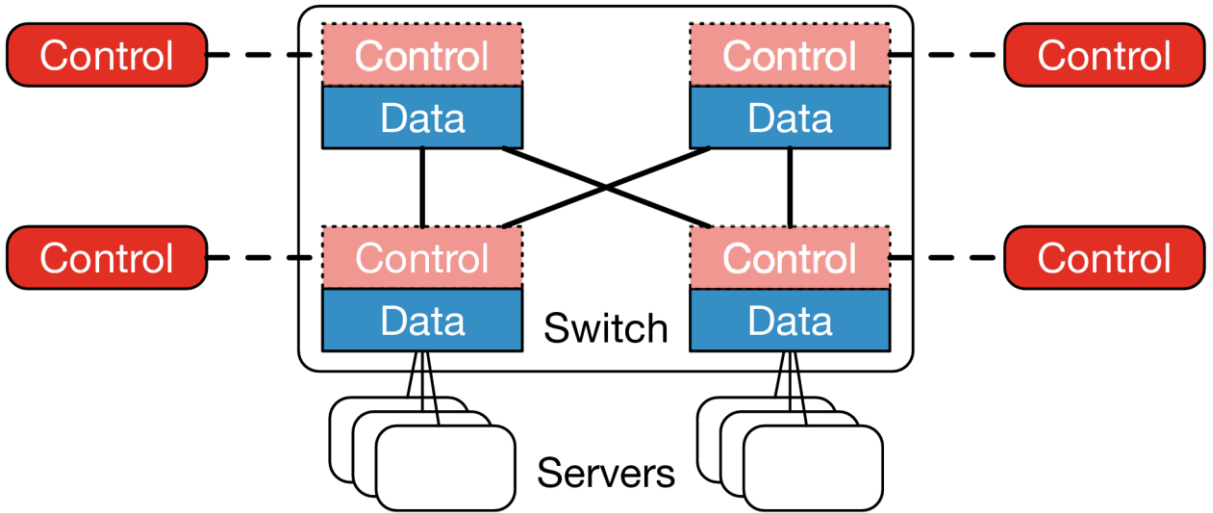
- Control plane compensates, for example:
  - Detects snapshot completion
    - Notifications
    - Extract from RAM
  - Lack of traffic
    - Liveness
    - Skipped snapshots

Implementation and Evaluation

- Implemented on a Barefoot Wedge100BF-32X
  - Control plane: ~2000 lines of Python
  - Data plane: ~1000 lines of P4 (per variant)
- Evaluation
  - How synchronized is Speedlight?
  - What is the overhead?
  - How does its results compare against current mechanism?

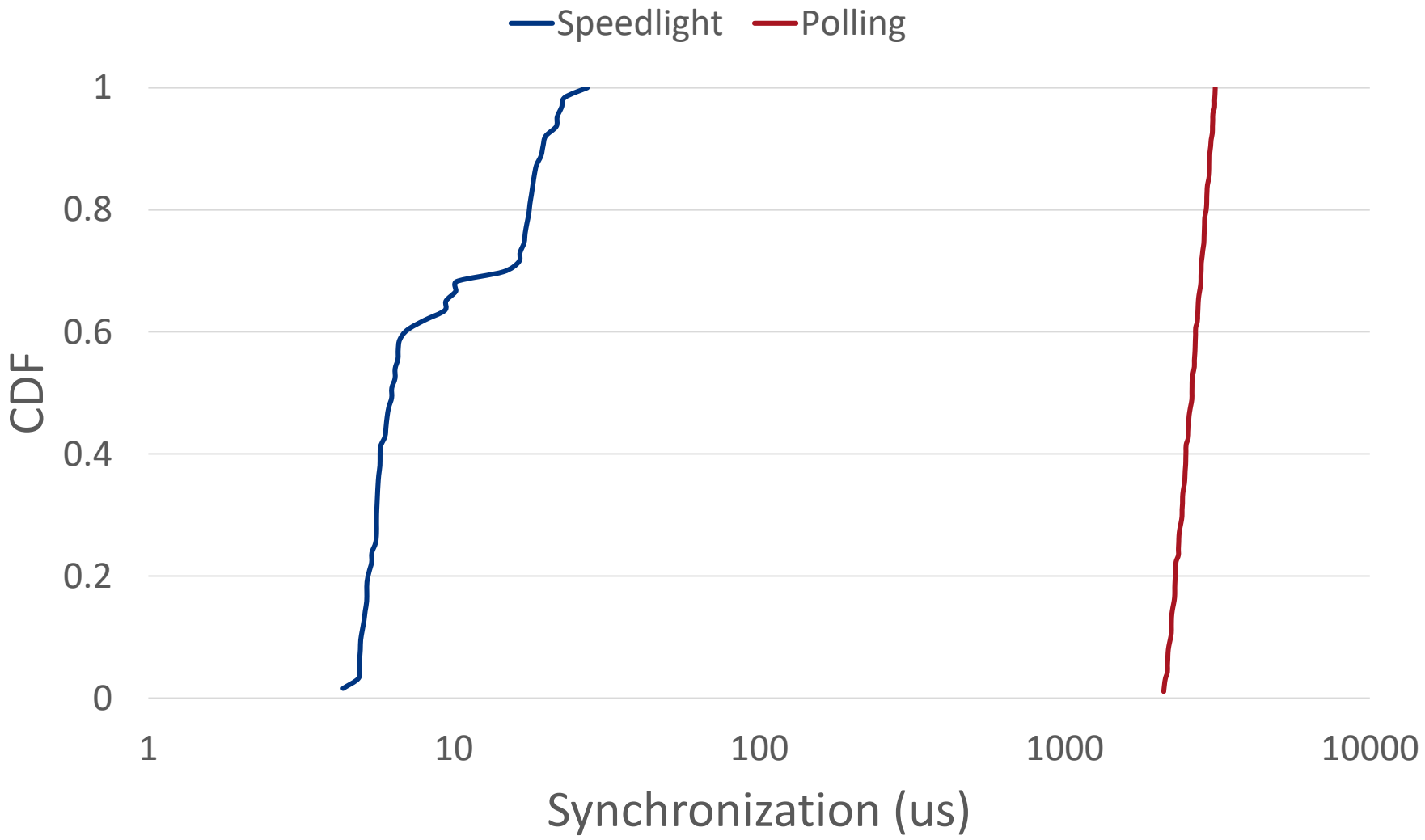
Implementation and Evaluation

- Implemented on a Barefoot Wedge100BF-32X
  - Control plane: ~2000 lines of Python
  - Data plane: ~1000 lines of P4 (per variant)
- Evaluation
  - How synchronized is Speedlight?
  - What is the overhead?
  - How does its results compare against current mechanism?

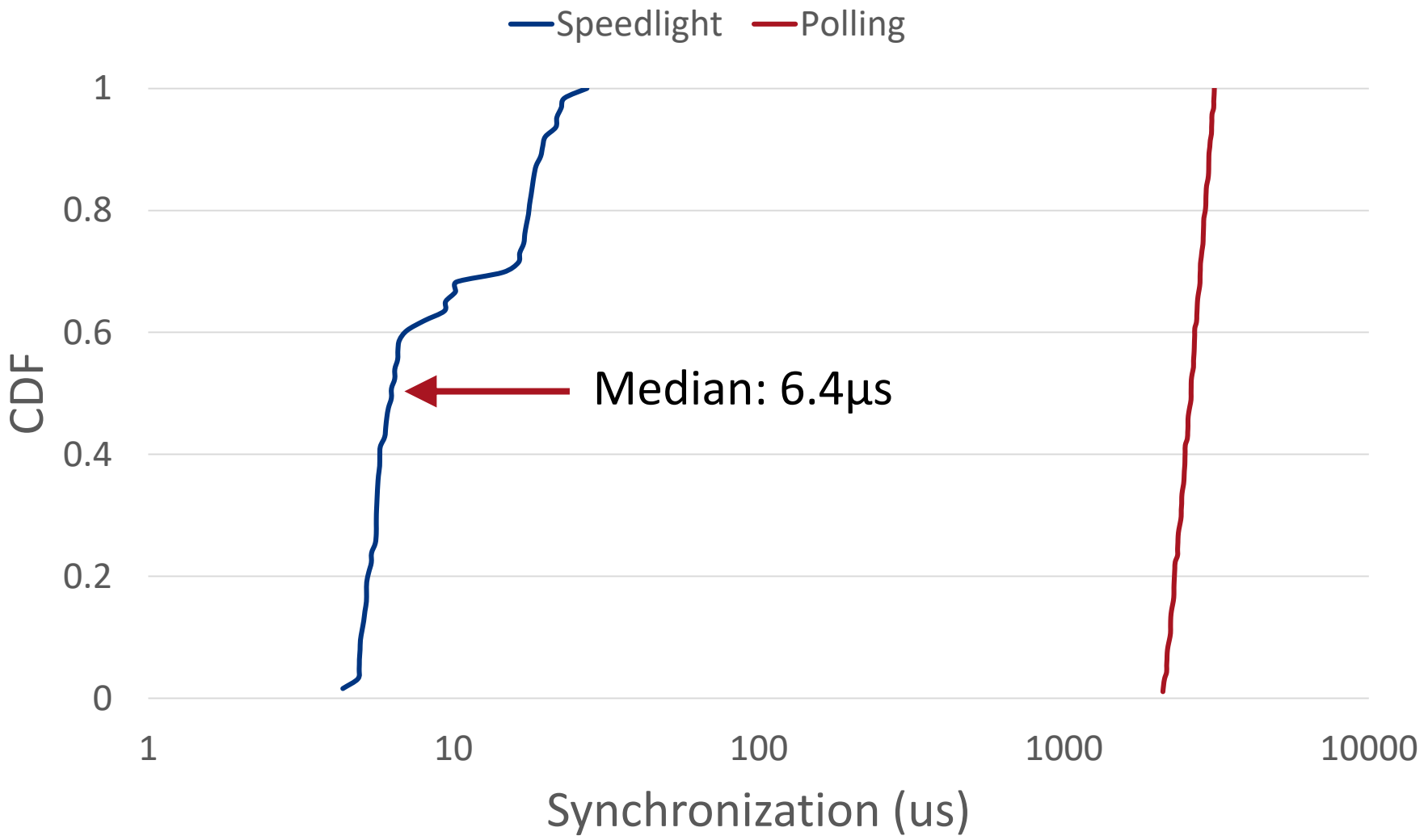


How Synchronized is Speedlight?

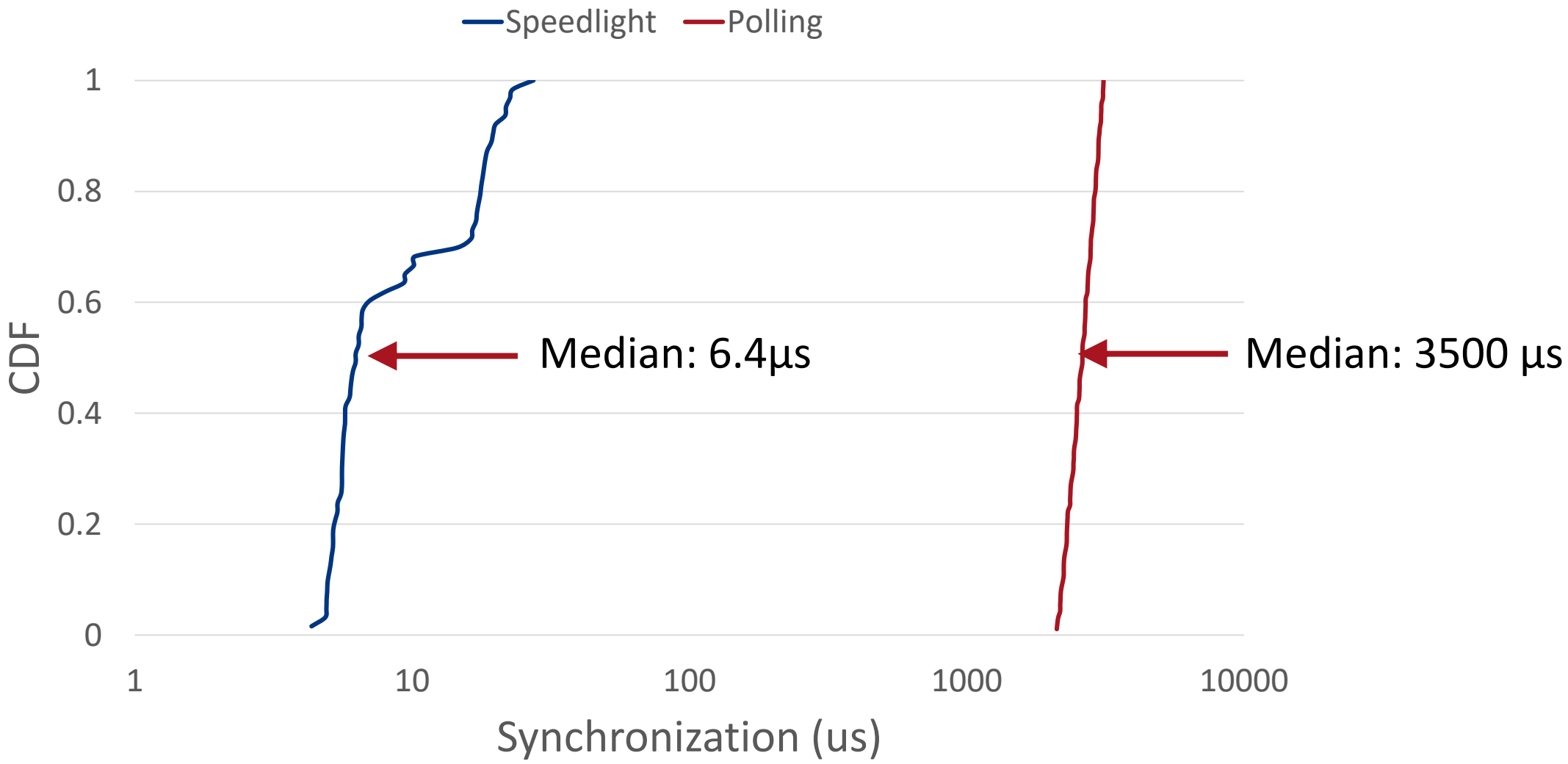
How Synchronized is Speedlight?



How Synchronized is Speedlight?

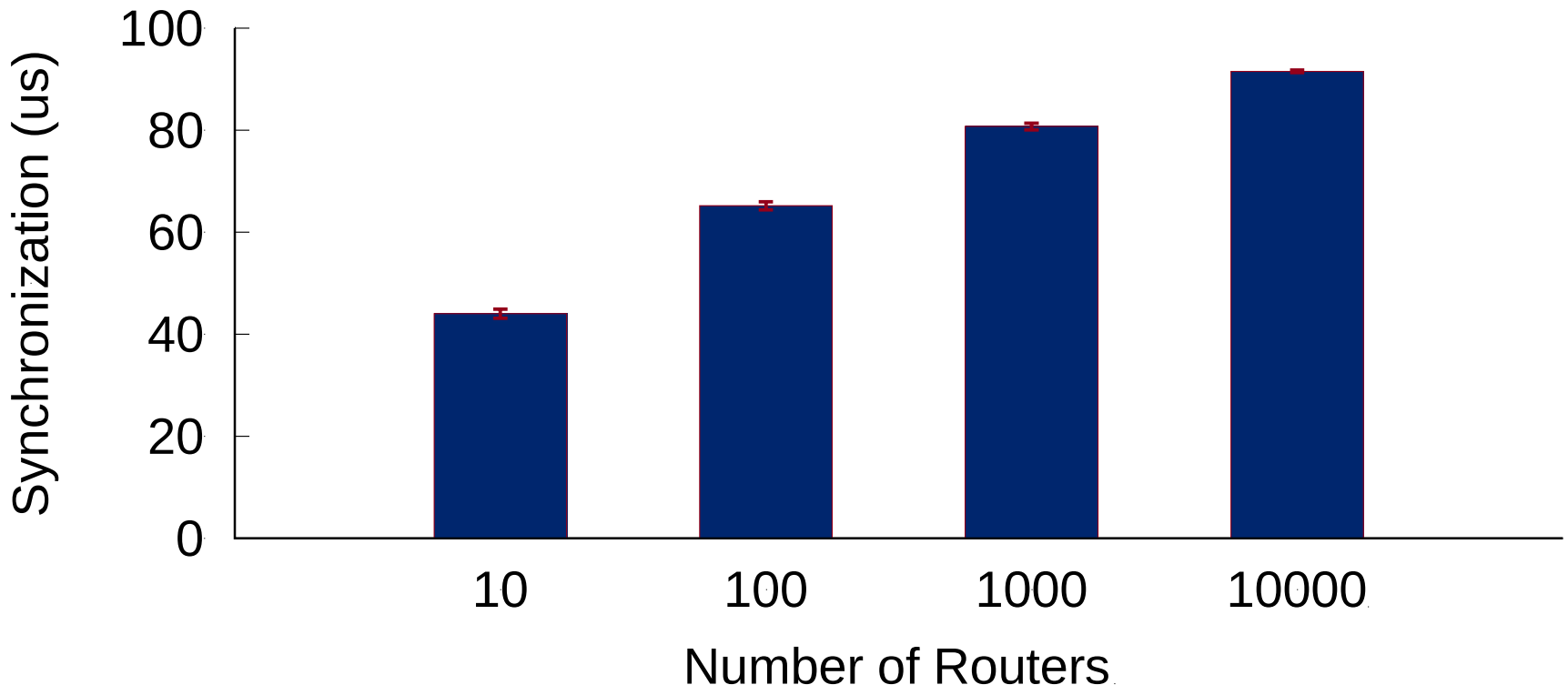


How Synchronized is Speedlight?



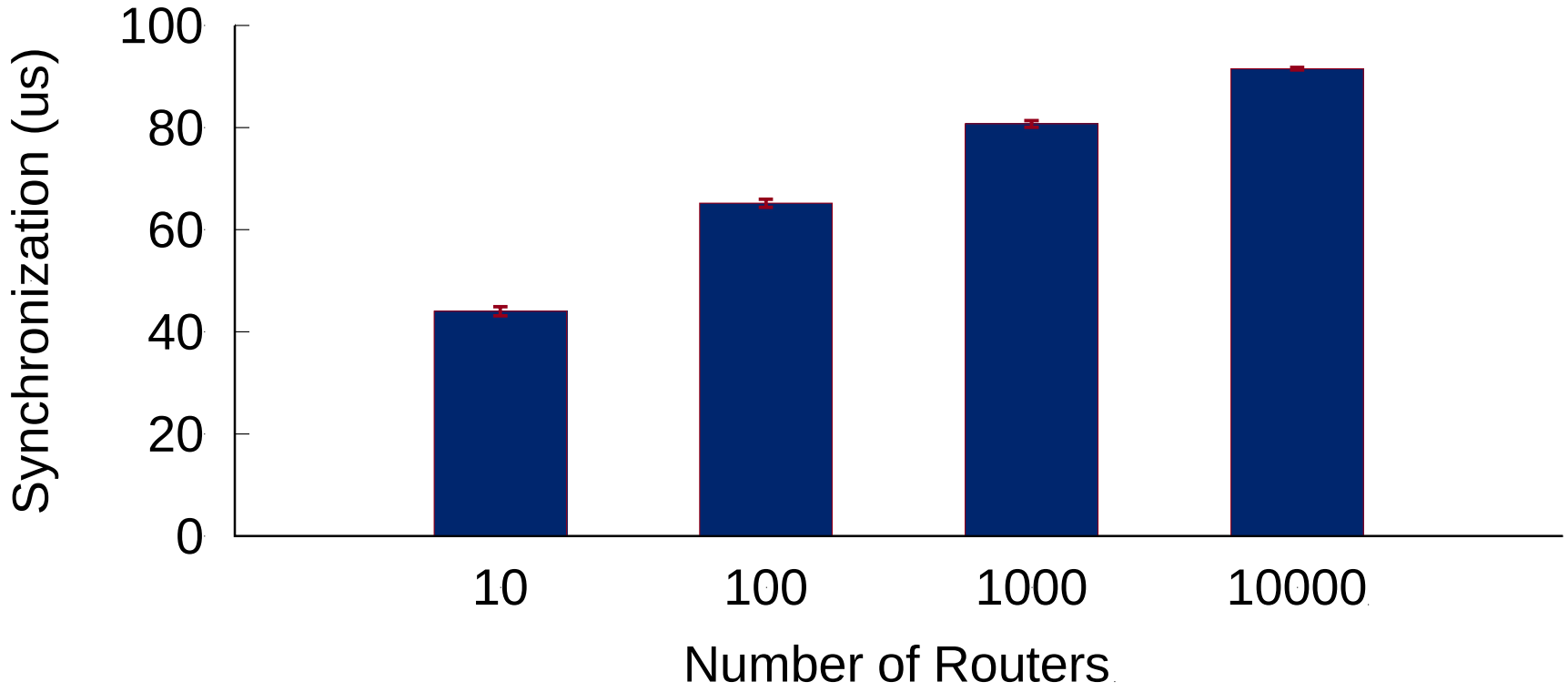
How Does Synchronization Scale?

How Does Synchronization Scale?



- Average synchronization in simulated network of 64-port routers

How Does Synchronization Scale?



- Average synchronization in simulated network of 64-port routers
- Number of routers only increases probability of hitting tail, not length of the tail

What’s the Overhead?

What’s the Overhead?

- No delays
- Network Overhead: 8 bytes per Packet

What's the Overhead?

- No delays
- Network Overhead: 8 bytes per Packet

Computational Resources	
Stateless ALUs	24
Stateful ALUs	11

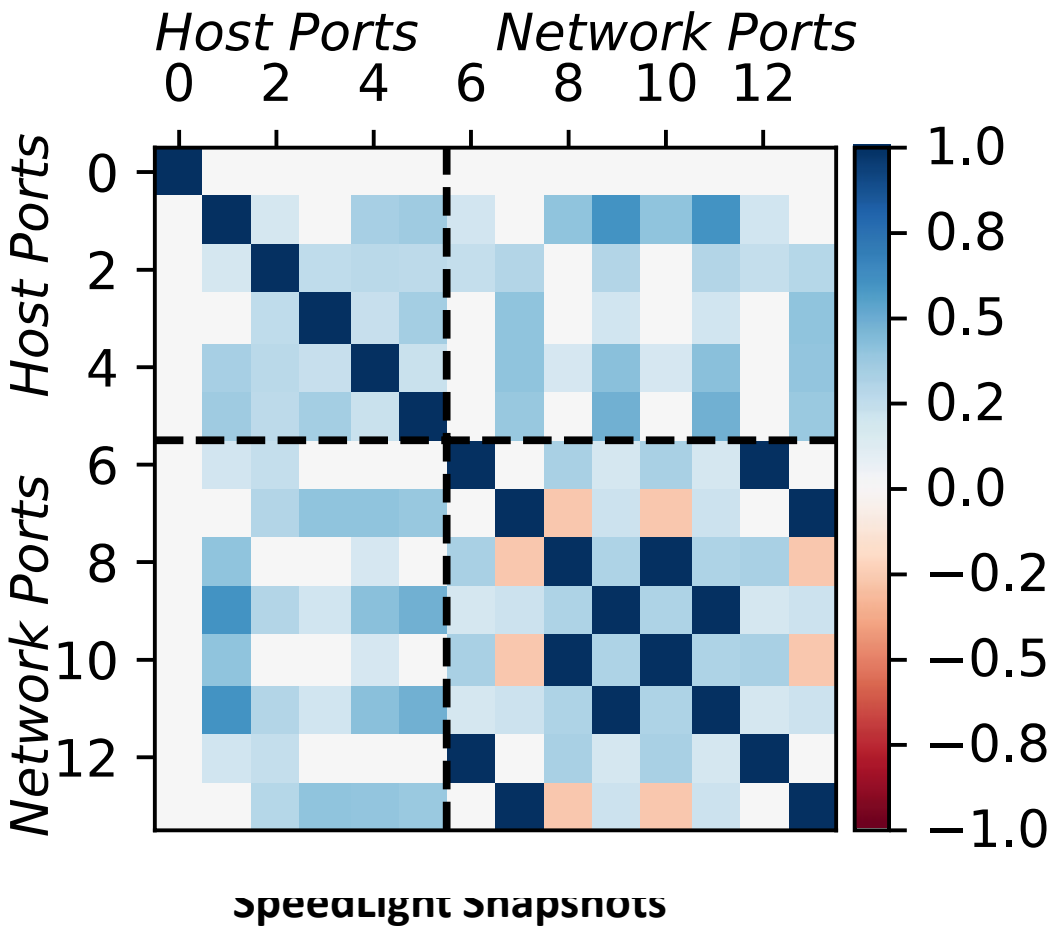
What's the Overhead?

- No delays
- Network Overhead: 8 bytes per Packet

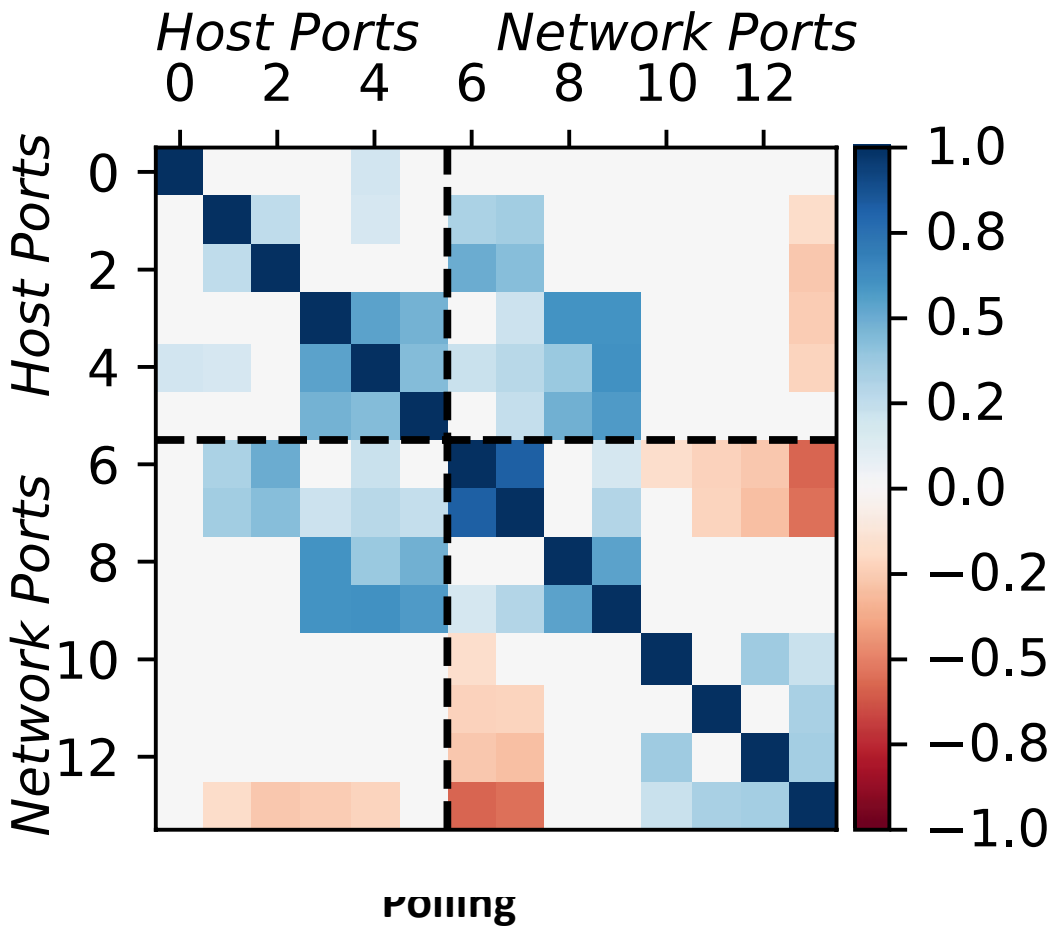
Computational Resources	
Stateless ALUs	24
Stateful ALUs	11

Memory Resources	
SRAM	770 kB
TCAM	244 kB

Use Case: Synchronized Traffic - GraphX

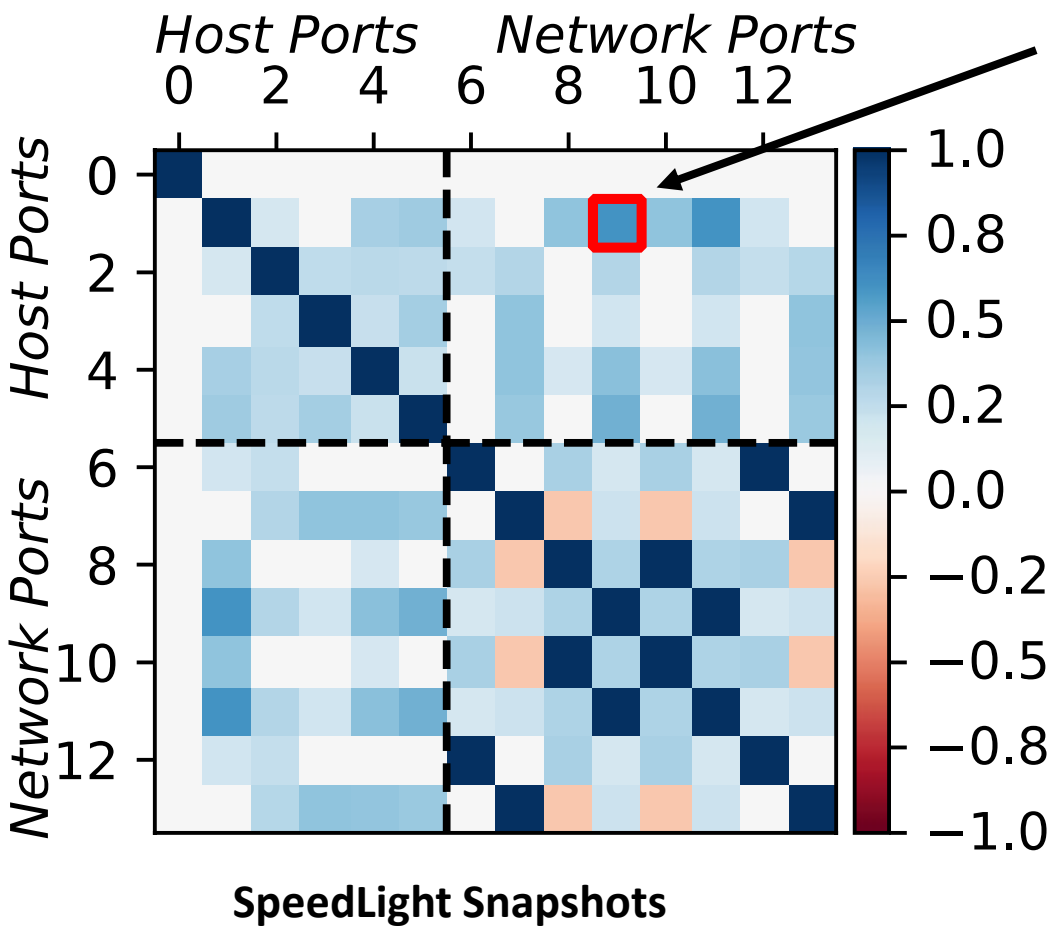


SpeedLight Snapshots

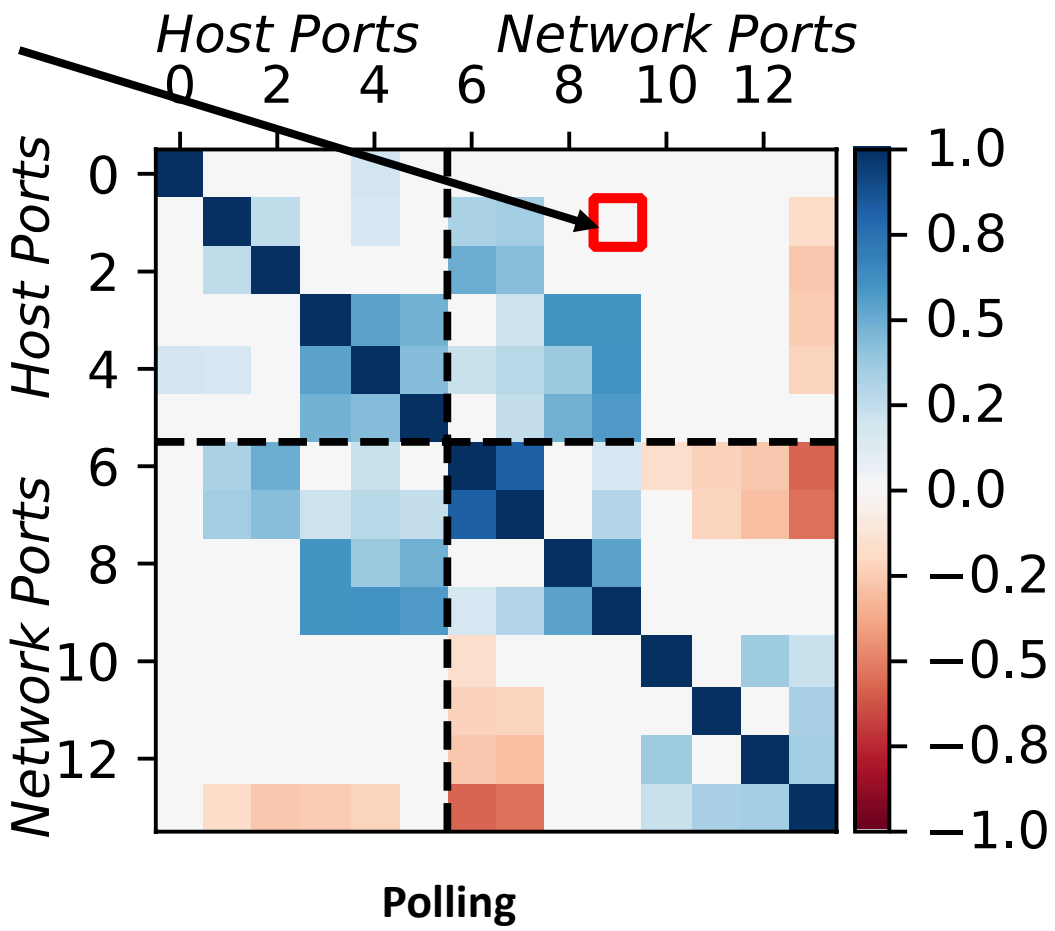


Polling

Use Case: Synchronized Traffic - GraphX

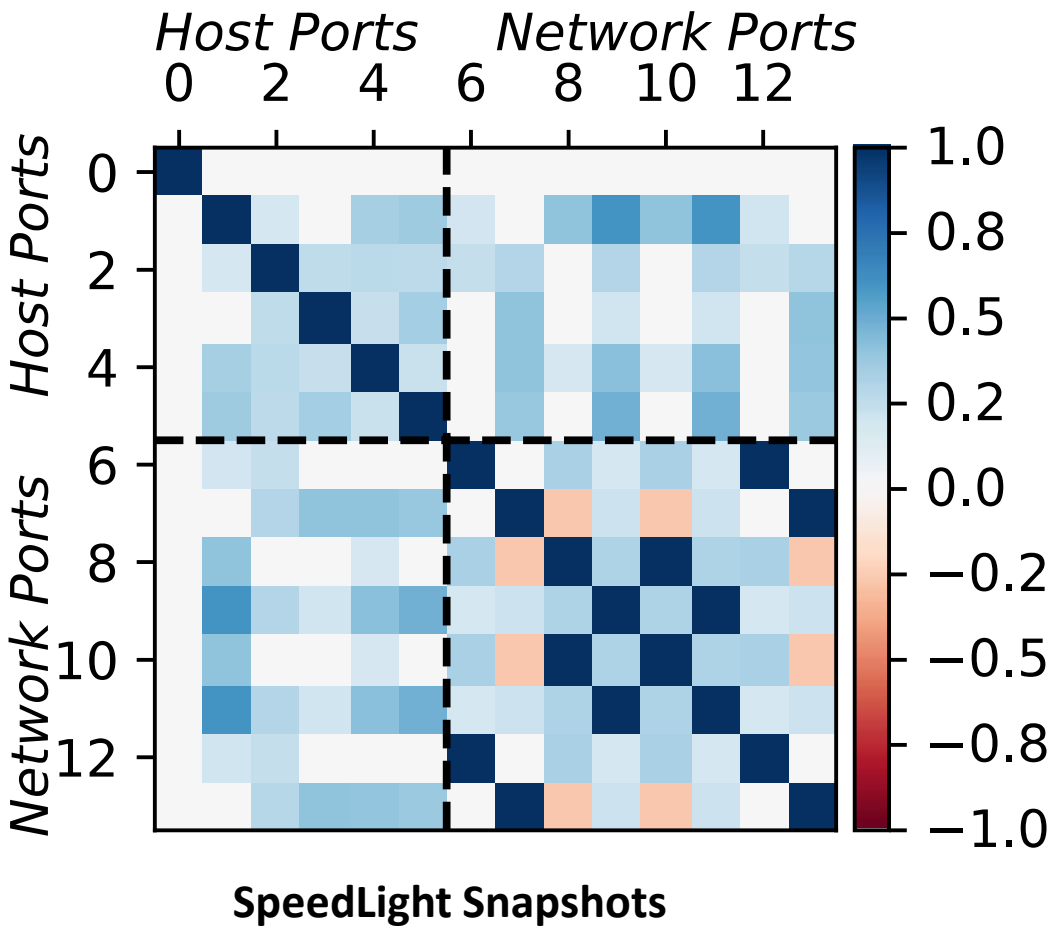


SpeedLight Snapshots

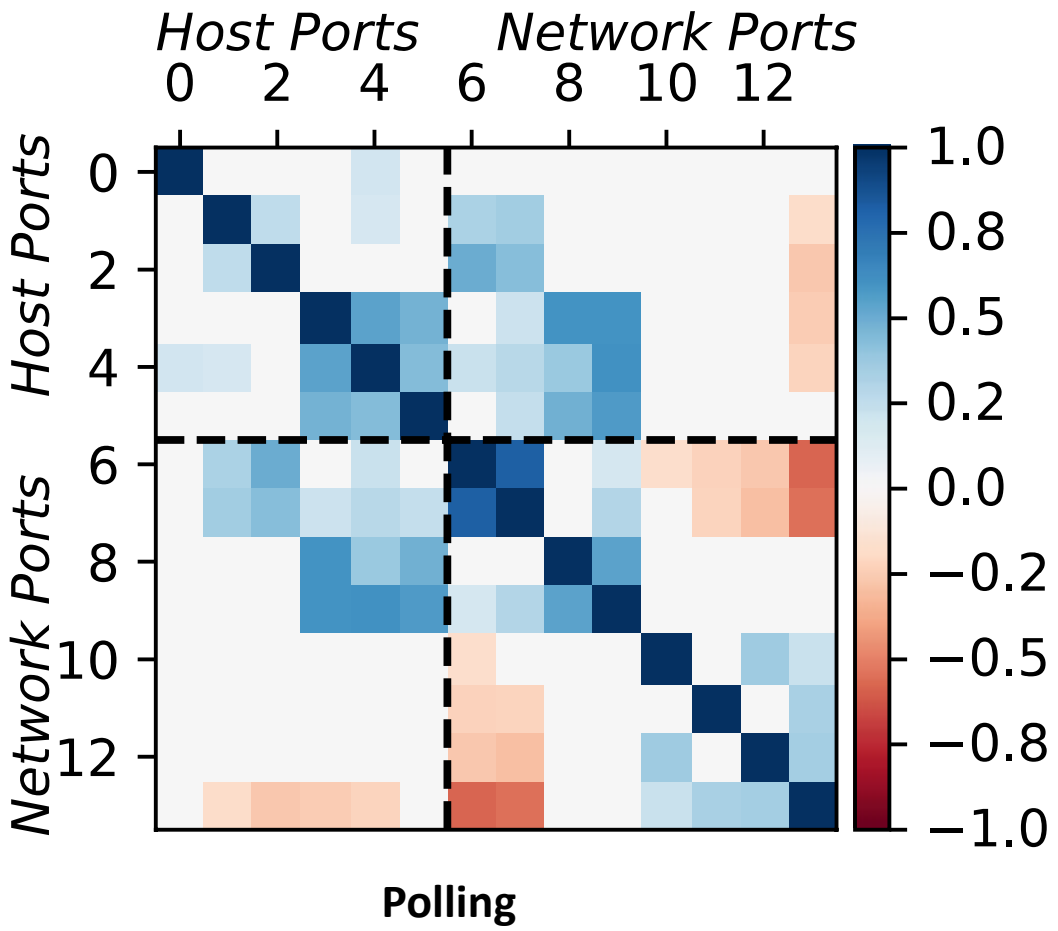


Polling

Use Case: Synchronized Traffic - GraphX

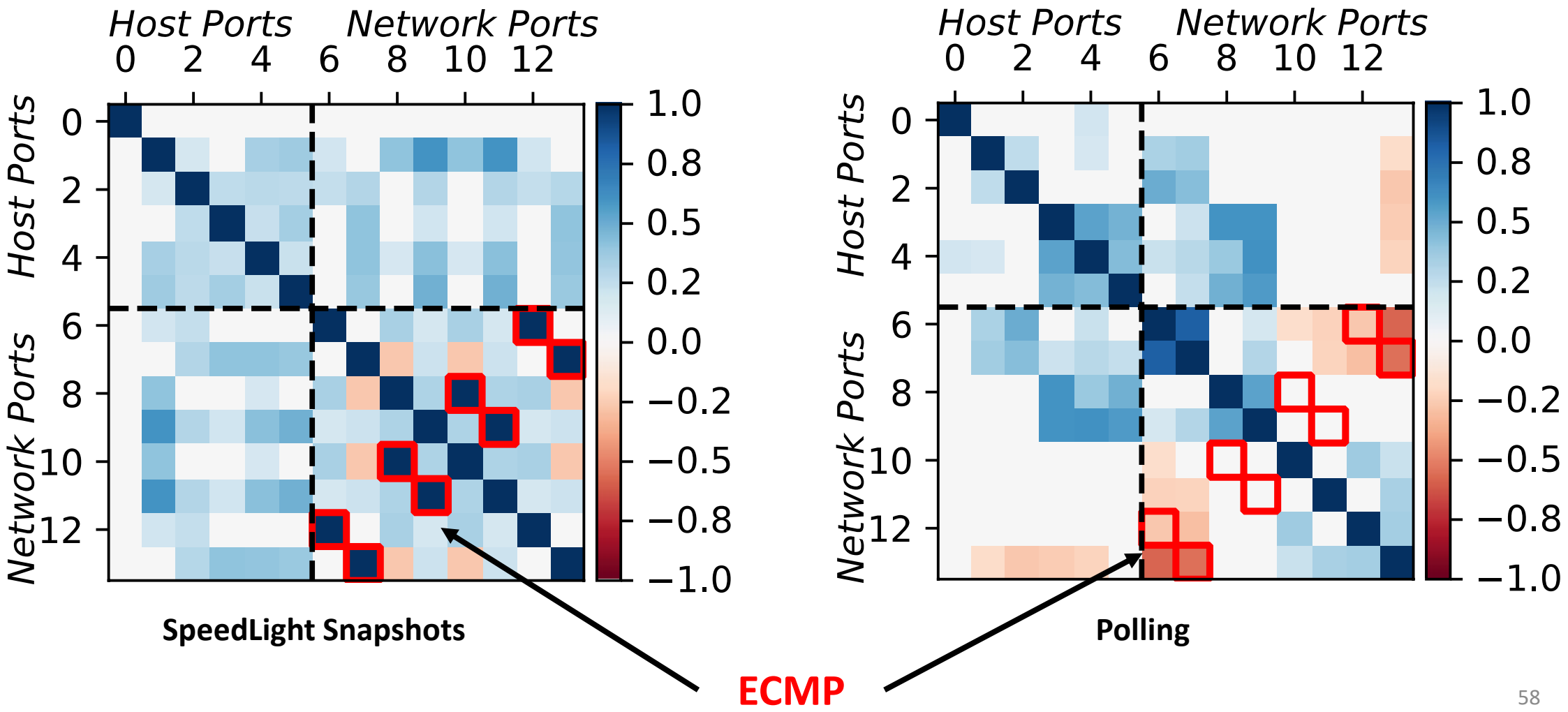


SpeedLight Snapshots



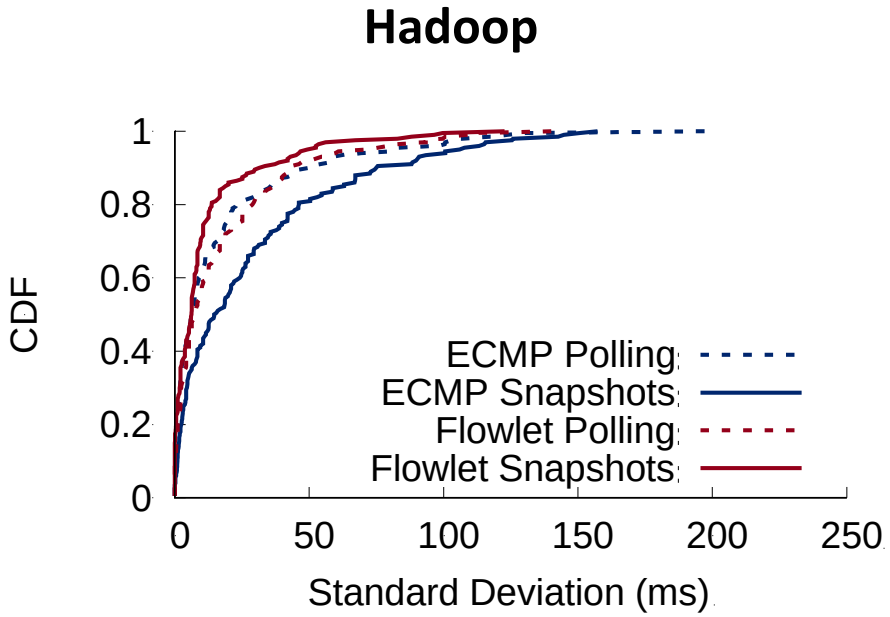
Polling

Use Case: Synchronized Traffic - GraphX

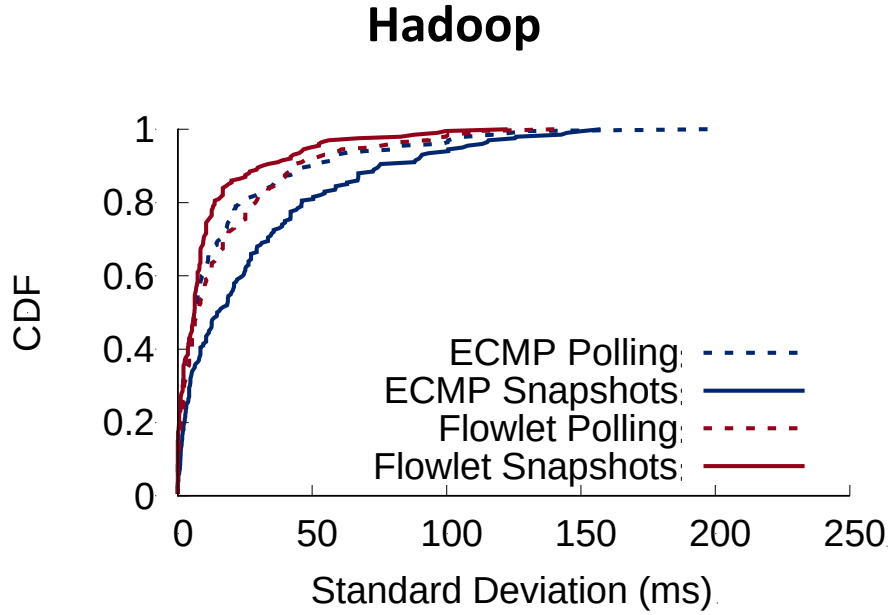


Use Case: Load Balancing

Use Case: Load Balancing

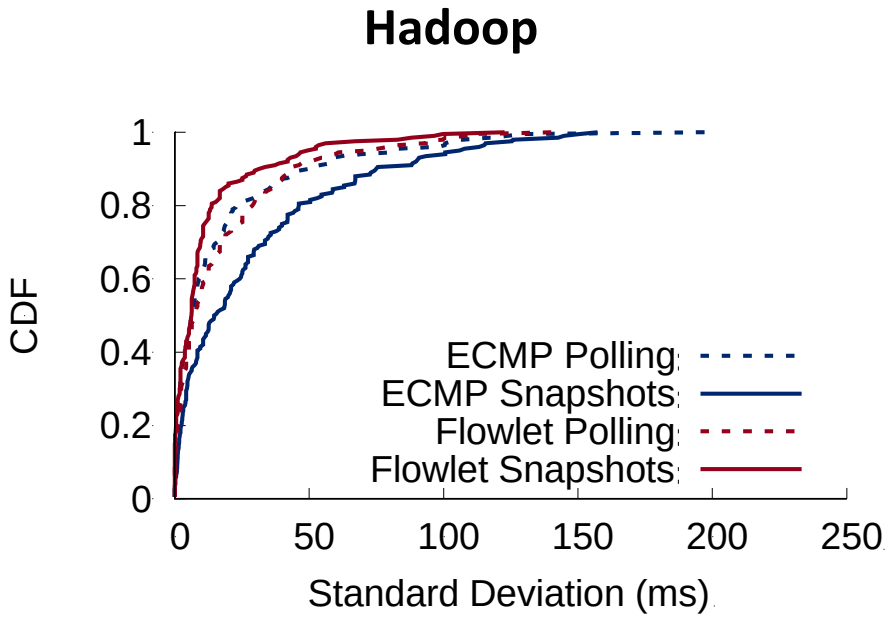


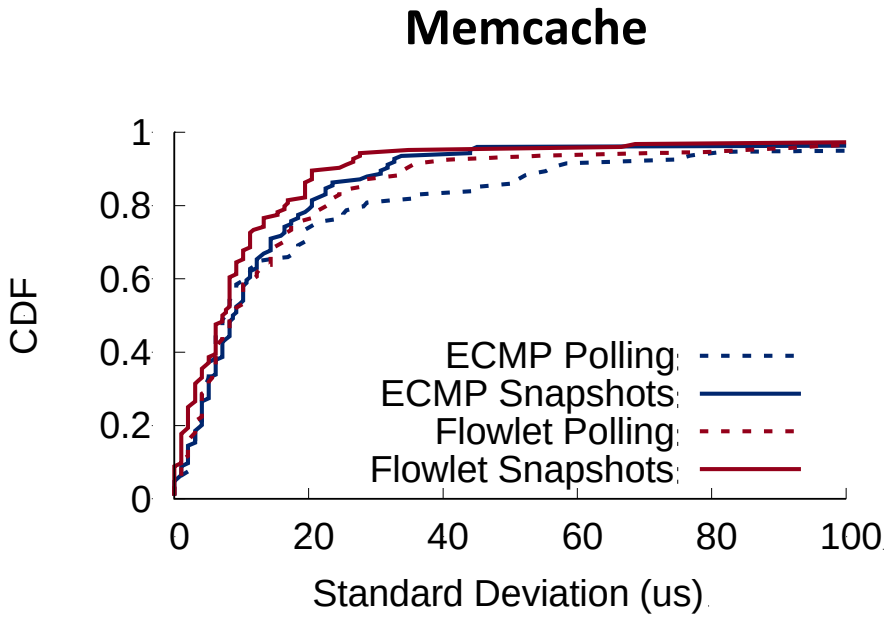
Use Case: Load Balancing



- Polling shows no difference between ECMP and flowlets.
- Reality: flowlets halve 90 pct stddev

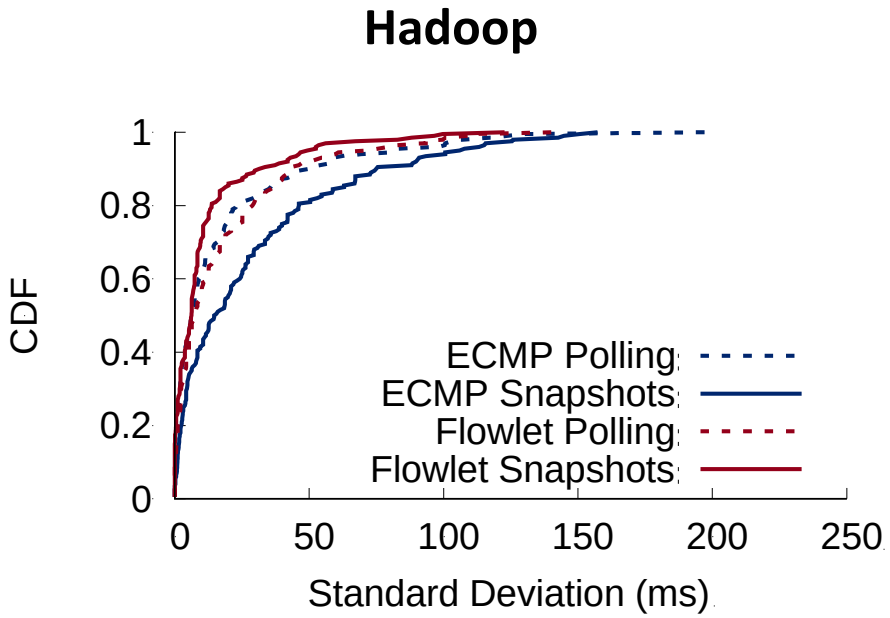
Use Case: Load Balancing

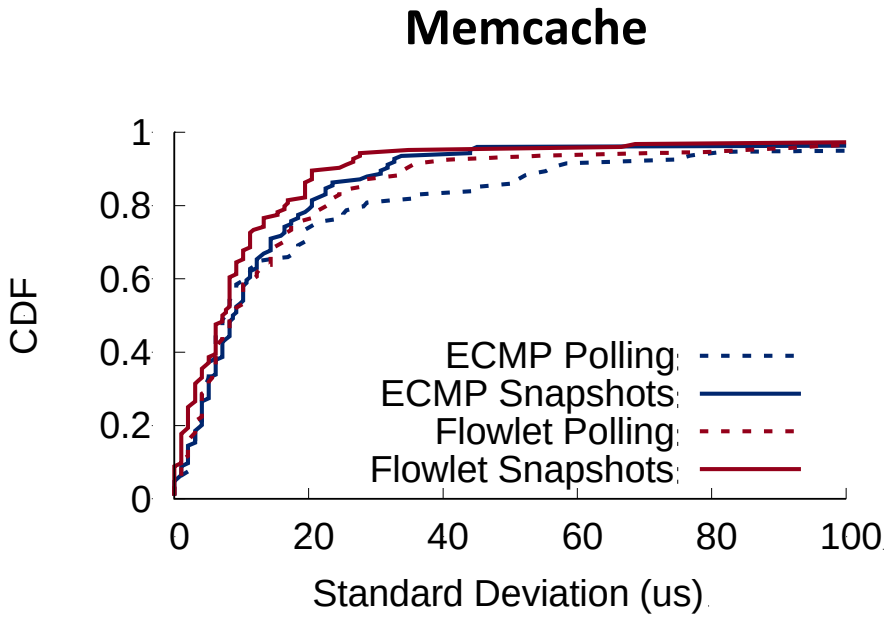




- Polling shows no difference between ECMP and flowlets.
- Reality: flowlets halve 90 pct stddev

Use Case: Load Balancing

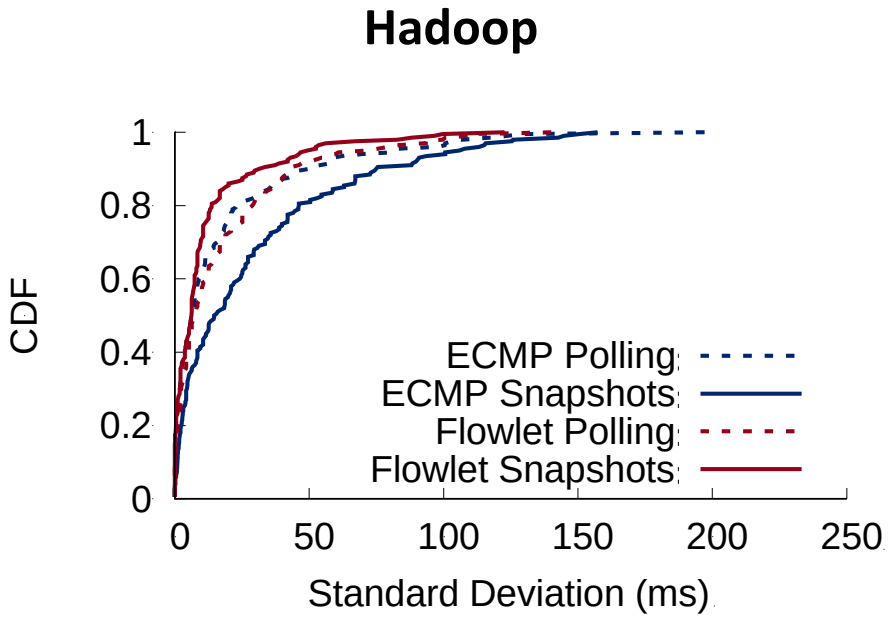


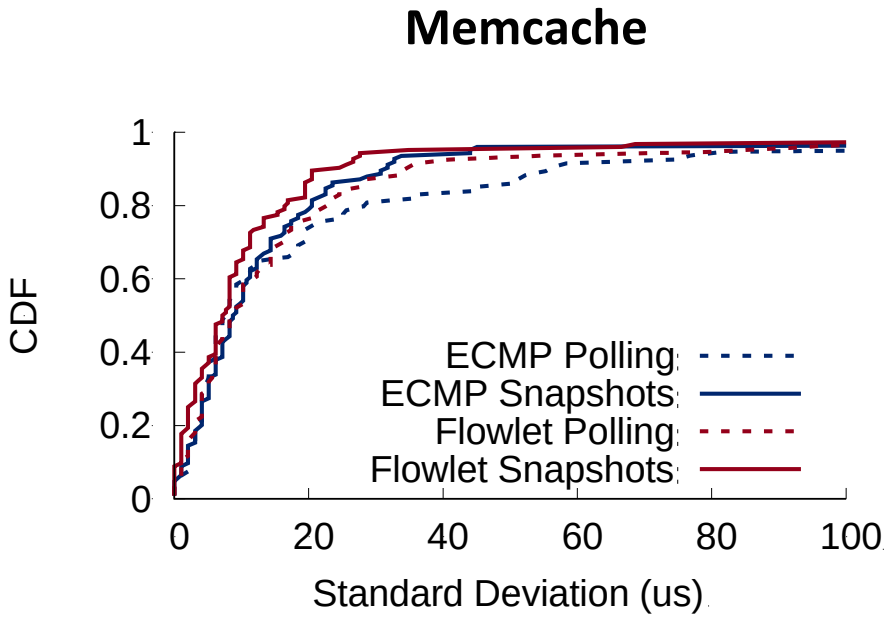


- Polling shows no difference between ECMP and flowlets.
- Reality: flowlets halve 90 pct stddev

- Polling consistently overestimates imbalance

Use Case: Load Balancing





- Polling shows no difference between ECMP and flowlets.
- Reality: flowlets halve 90 pct stddev

- Polling consistently overestimates imbalance

Averaging shows perfect balance in both cases

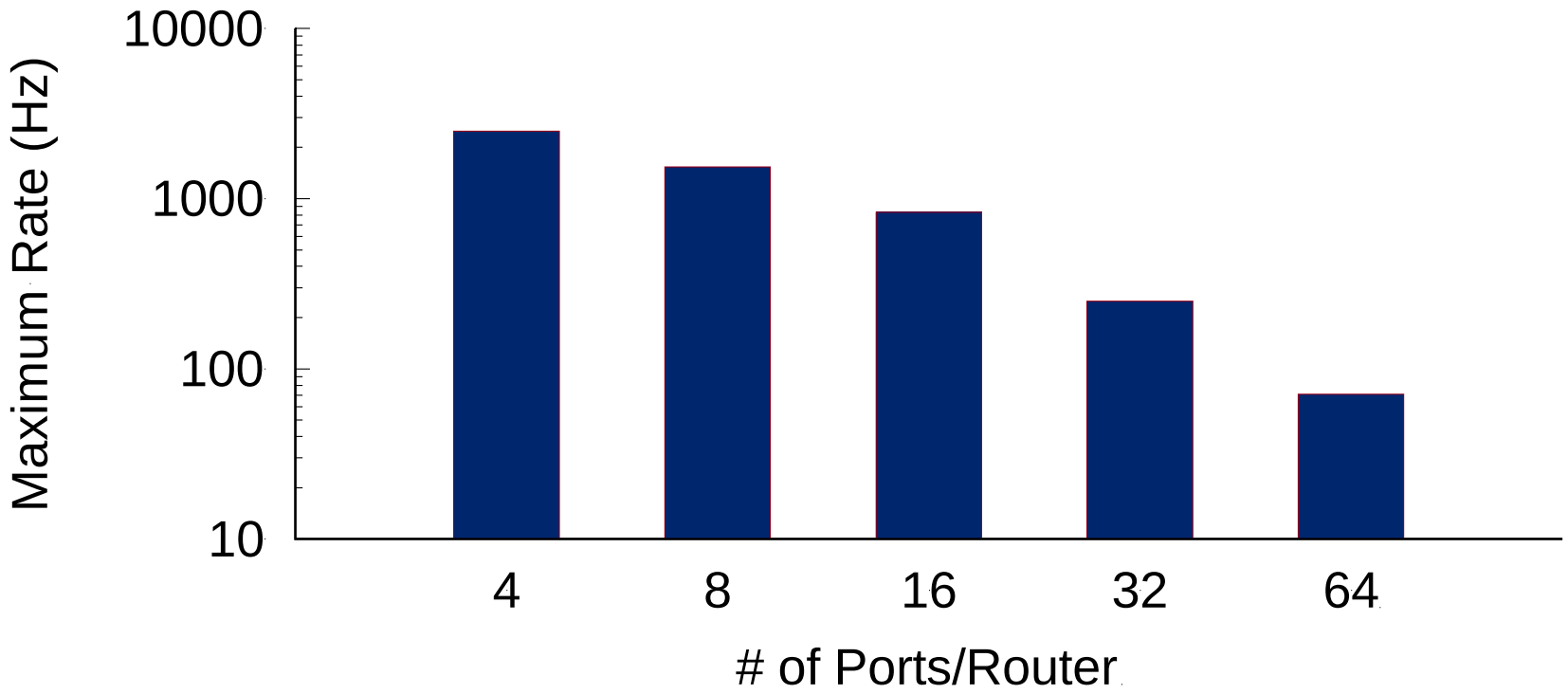
Speedlight Summary

- Unsynchronized measurements can be misleading
- Speedlight: A complete picture of the network
  - Causal consistency
  - Approximate synchrony (<RTT)
  - Wedge100BF-32X implementation
- https://github.com/eniac/Speedlight

THANK YOU

QUESTIONS AND COMMENTS

When We Go Too Fast



- Limited by number of Ports
- Detect Inconsistent/Incomplete Snapshots