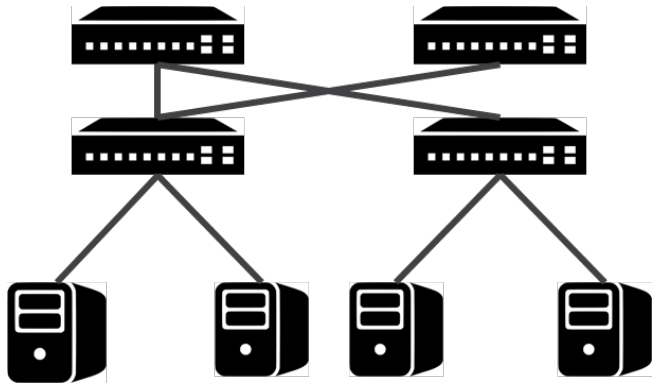
TurboNet: Faithfully Emulating Networks with Programmable Switches

Jiamin Cao (Tsinghua University)
Yu Zhou (Tsinghua University), Ying Liu (Tsinghua University),
Mingwei Xu (Tsinghua University), Yongkai Zhou (UnionPay)

To conduct network experiments



Congestion control
Topologies
Routing
IPv4/IPv6
…

Simulators ?

Emulators ?

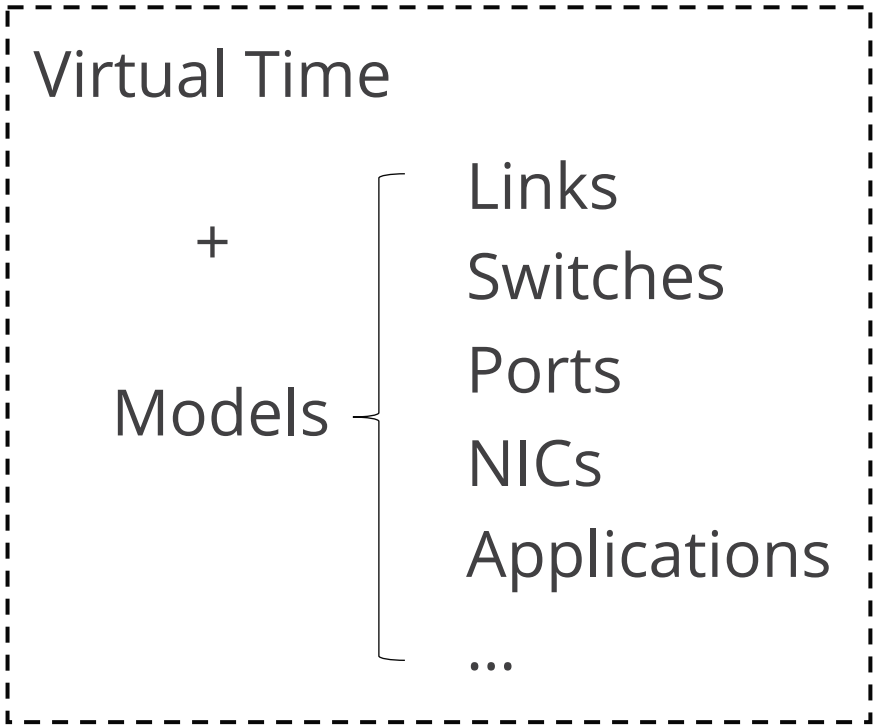
Test-beds ?

Network experiments platforms should satisfy:

- **High Fidelity:** like realistic hardware environment
- **High Flexibility:** topologies, properties (link delay, packet losses, link bandwidth, …)
- **High Scalability:** easily scale to meet increasing network bandwidth and sizes

Simulators

Use CPU to model networks as simulated events



ns-3 NETWORK SIMULATOR

OPNET Making Networks and Applications Perform®

OMNeT++ Discrete Event Simulator

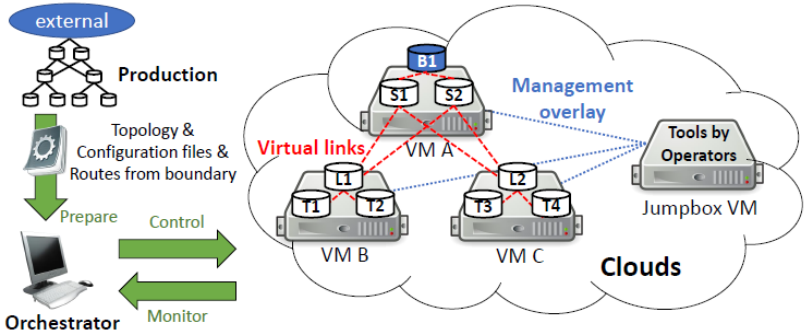
Fidelity Concerns

Emulators

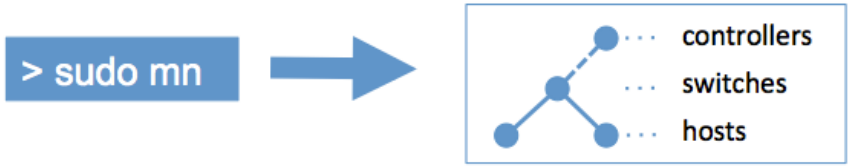
Run the same code on CPU as real platforms with interactive network traffic

Cannot provide accurate performance results for large and high-speed networks

- 10Gbps networks?
- 100Gbps networks?



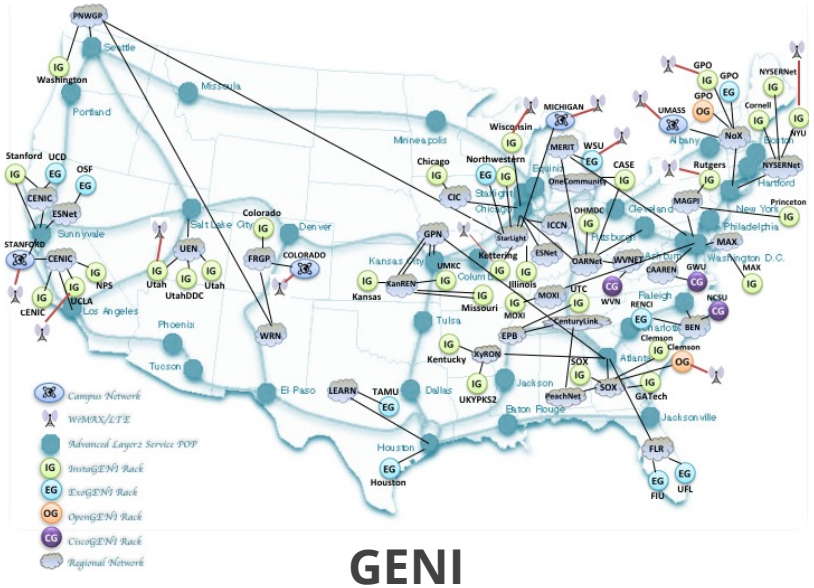
CrystalNet [SOSP 17]



Mininet

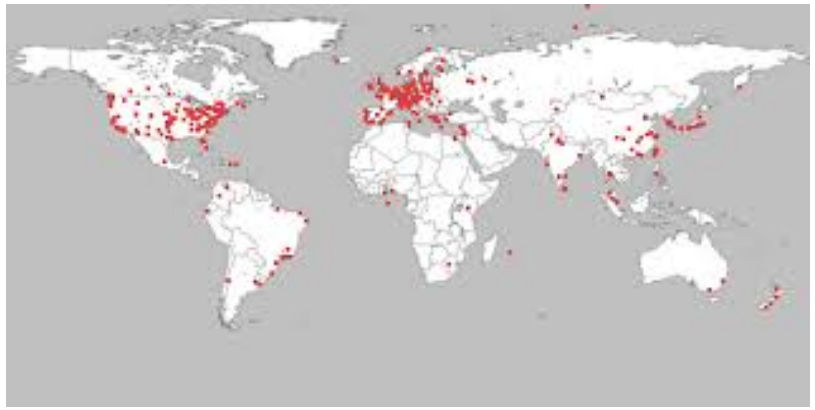
Fidelity & Scalability Concerns

Test-Beds



GENI

CERNET2



PlanetLab

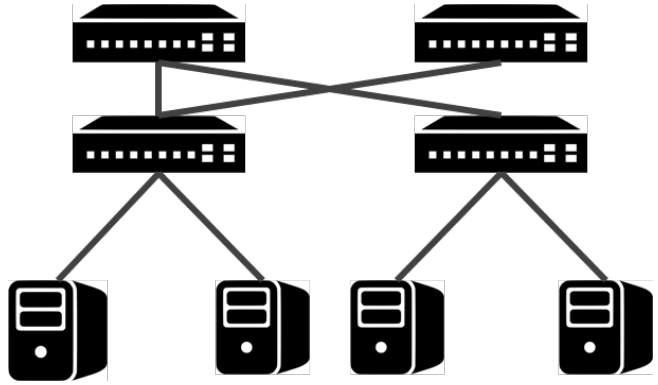
Experiments with real-world settings

☺ Can guarantee fidelity

☹ Lack customizability for topologies, forwarding behaviors, metrics (link delay, link loss, …)

☹ Costly to scale and upgrade ***Scalability & Flexibility Concerns***

To conduct network experiments



Congestion control
Topologies
Routing
IPv6
…

	Simulator	Emulator	Test-Bed
Fidelity	✗	✗	✓
Flexibility	✓	✓	✗
Scale	✓	✗	✗

Our Approach

TurboNet: emulating various networks with one programmable switch.

Why programmable switch?

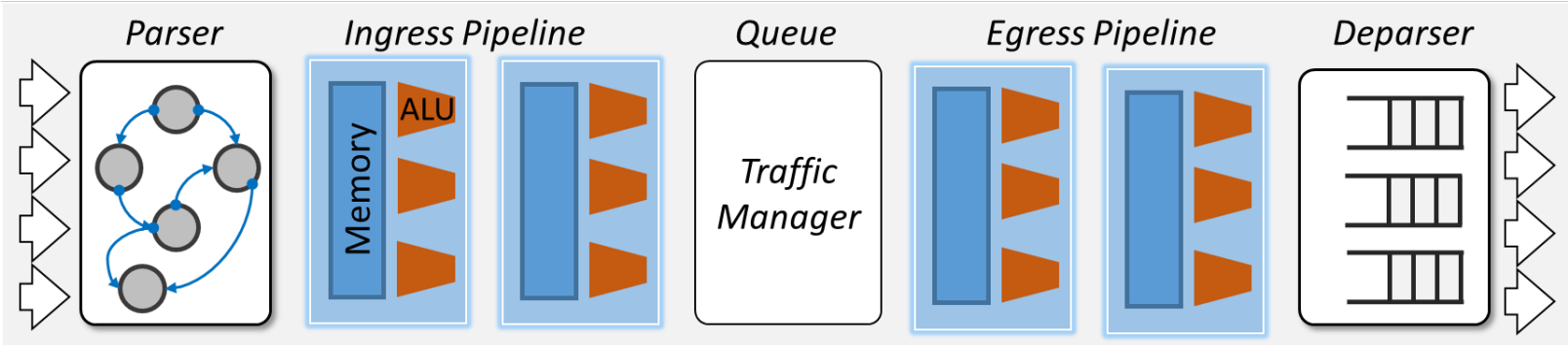
- Re-configurable packet processing
- High bandwidth (Tbps)

Enable agile customization of various networks with high bandwidth

Our Goal:

Faithfully emulate both network data plane and control plane.

- Data Plane: topology, metrics (delay, loss)
- Control Plane: routing (static/dynamic)

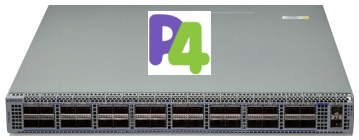


[RMT@SIGCOMM'13]

Talk Outline

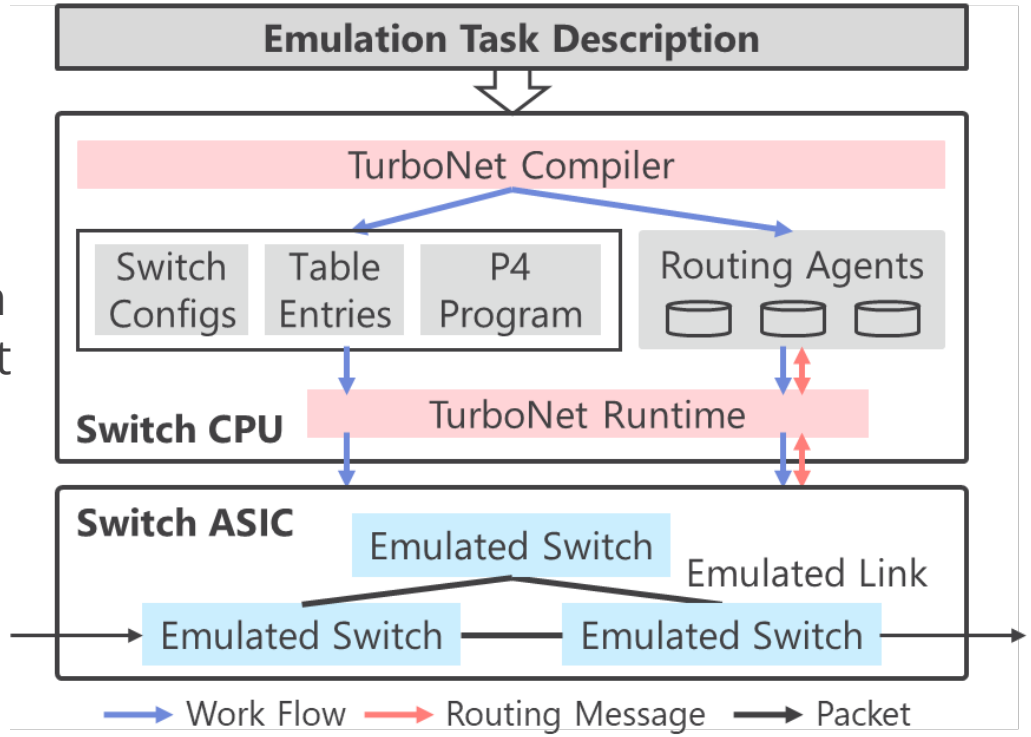
- ~~Motivation~~
- **1. TurboNet Overview: architecture and workflow**
- 2. Data Plane Emulation
- 3. Control Plane Emulation
- 4. Some Evaluation Results
- 5. Conclusion

TurboNet Overview



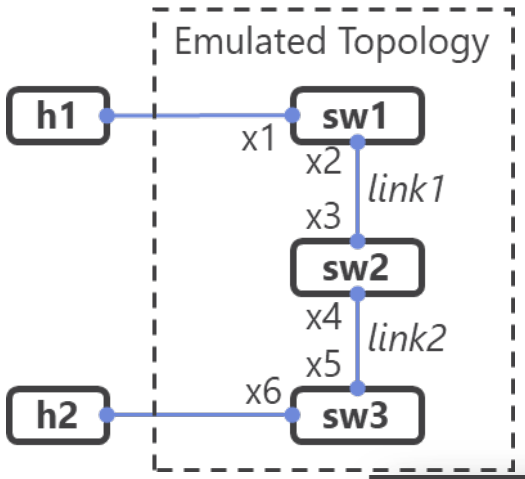
A programmable switch deployed with TurboNet

TurboNet Architecture



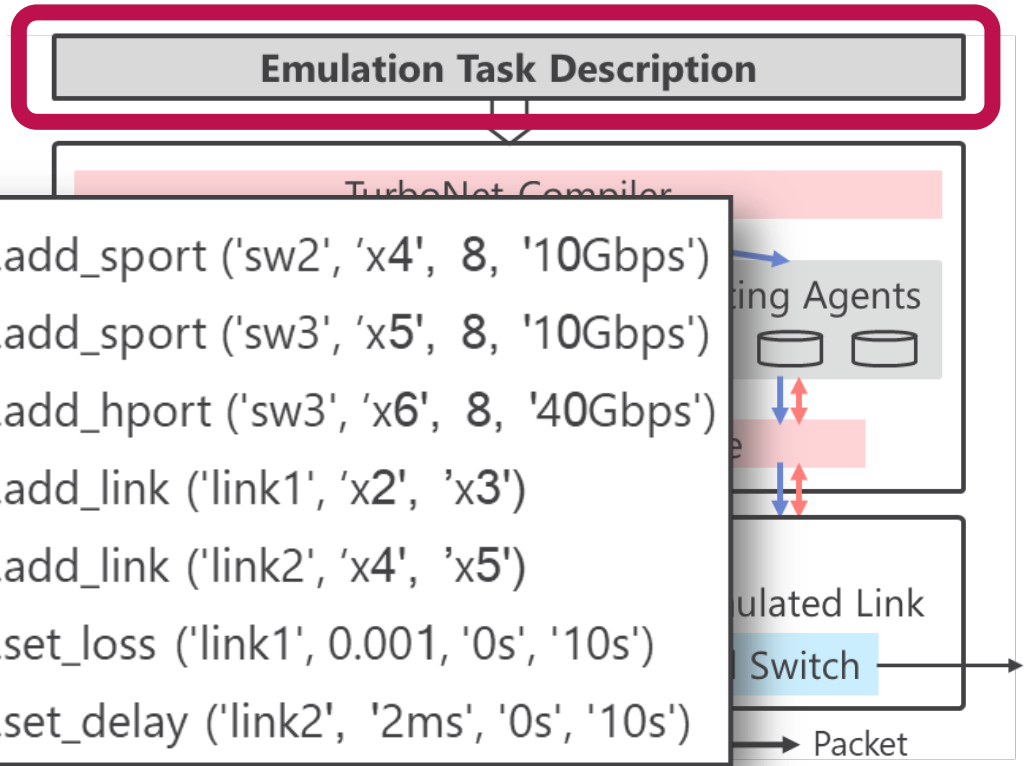
TurboNet Overview

TurboNet Workflow



1. Emulation Task Description

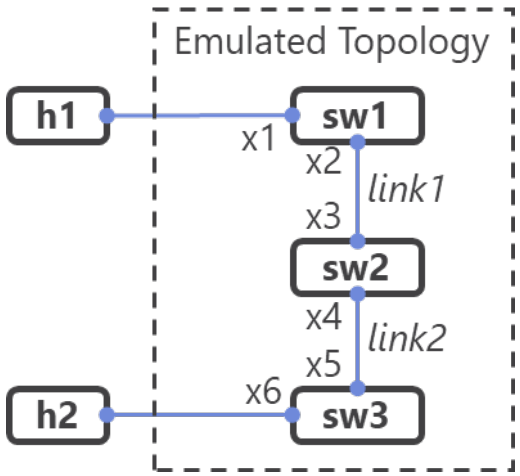
TurboNet Architecture




```
1  topo = Topo ()
2  topo.add_switch ('sw1')
3  topo.add_switch ('sw2')
4  topo.add_switch ('sw3')
5  topo.add_hport ('sw1', 'x1', 8, '40Gbps')
6  topo.add_sport ('sw1', 'x2', 8, '10Gbps')
7  topo.add_sport ('sw2', 'x3', 8, '10Gbps')
8  topo.add_sport ('sw2', 'x4', 8, '10Gbps')
9  topo.add_sport ('sw3', 'x5', 8, '10Gbps')
10 topo.add_hport ('sw3', 'x6', 8, '40Gbps')
11 topo.add_link ('link1', 'x2', 'x3')
12 topo.add_link ('link2', 'x4', 'x5')
13 topo.set_loss ('link1', 0.001, '0s', '10s')
14 topo.set_delay ('link2', '2ms', '0s', '10s')
```

TurboNet Overview

TurboNet Workflow

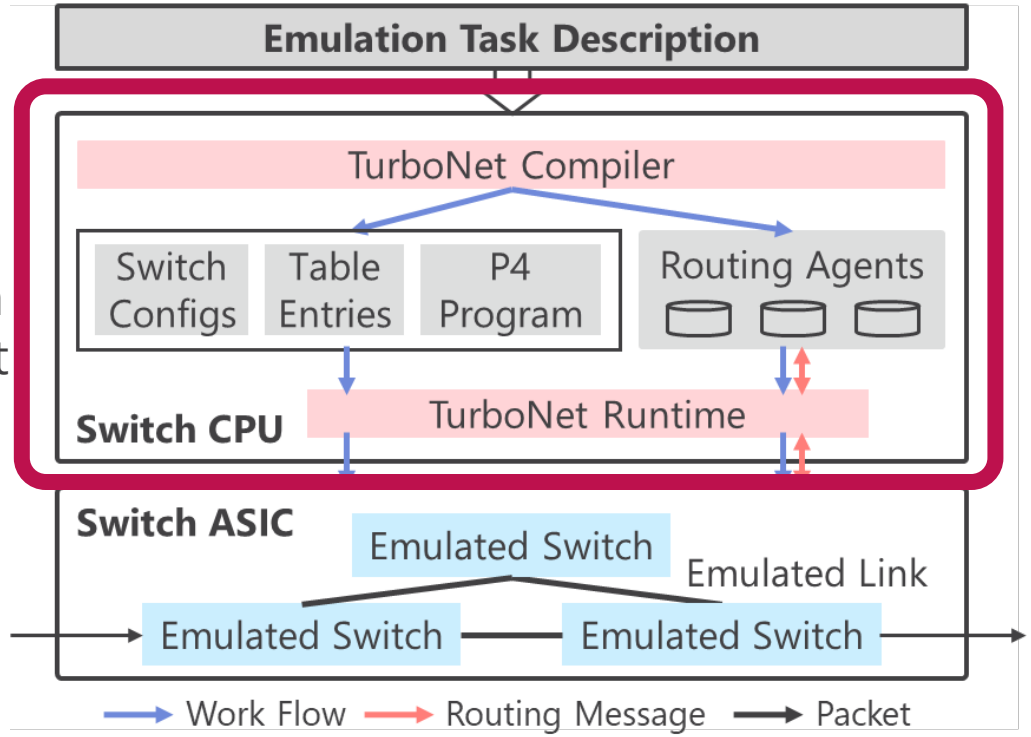


1. Emulation Task Description

2. Configuration Generation

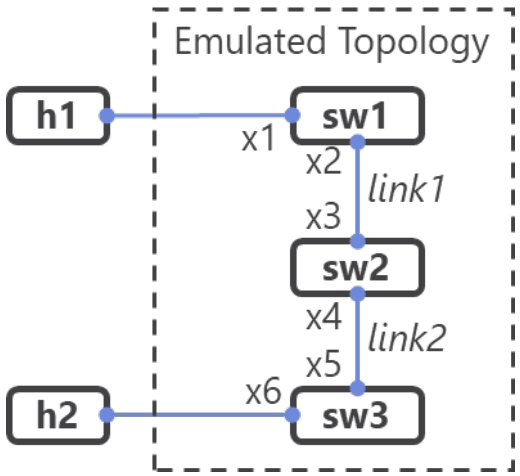
A programmable switch deployed with TurboNet

TurboNet Architecture



TurboNet Overview

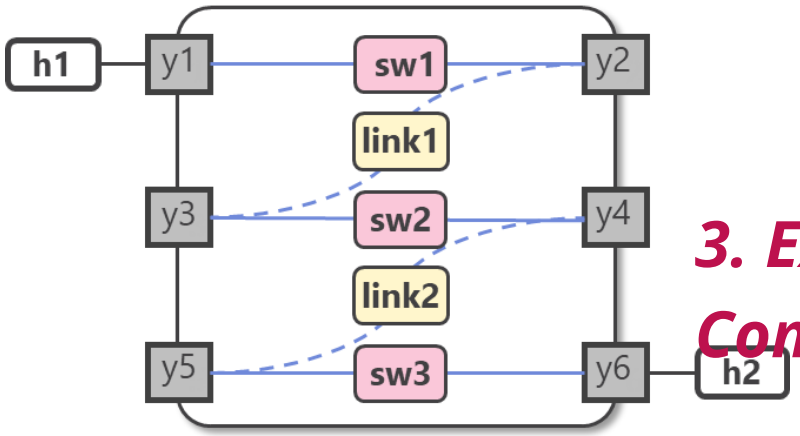
TurboNet Workflow



1. Emulation Task Description

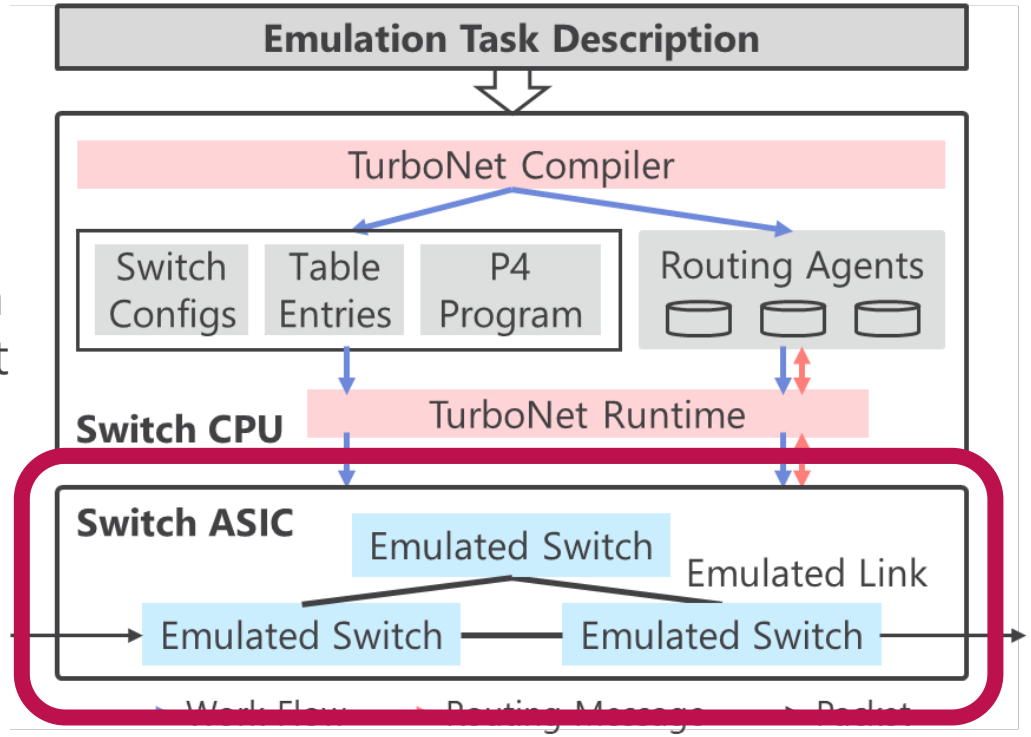
2. Configuration Generation

A programmable switch deployed with TurboNet



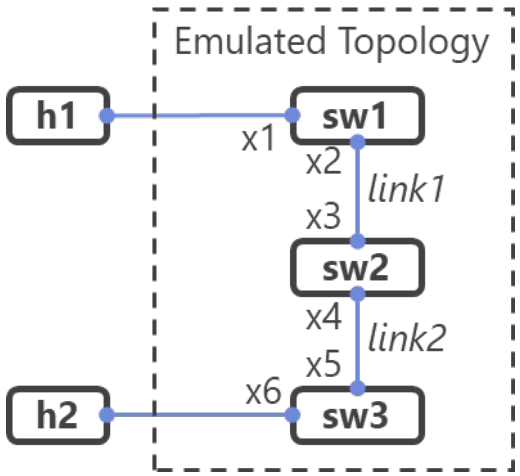
3. Experiment Conduction

TurboNet Architecture

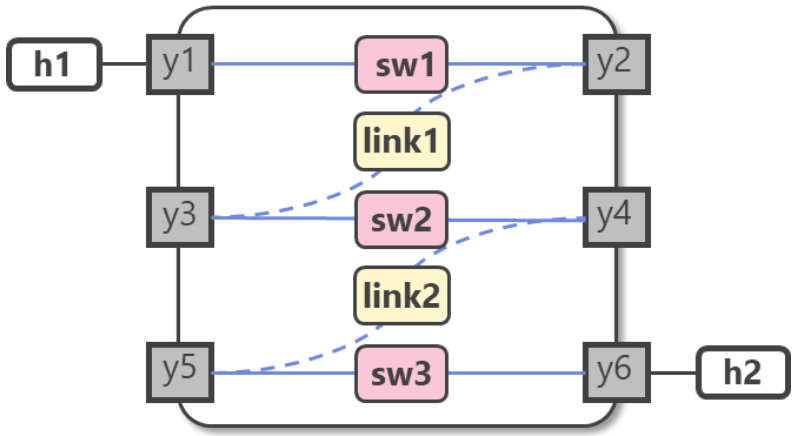


TurboNet Overview

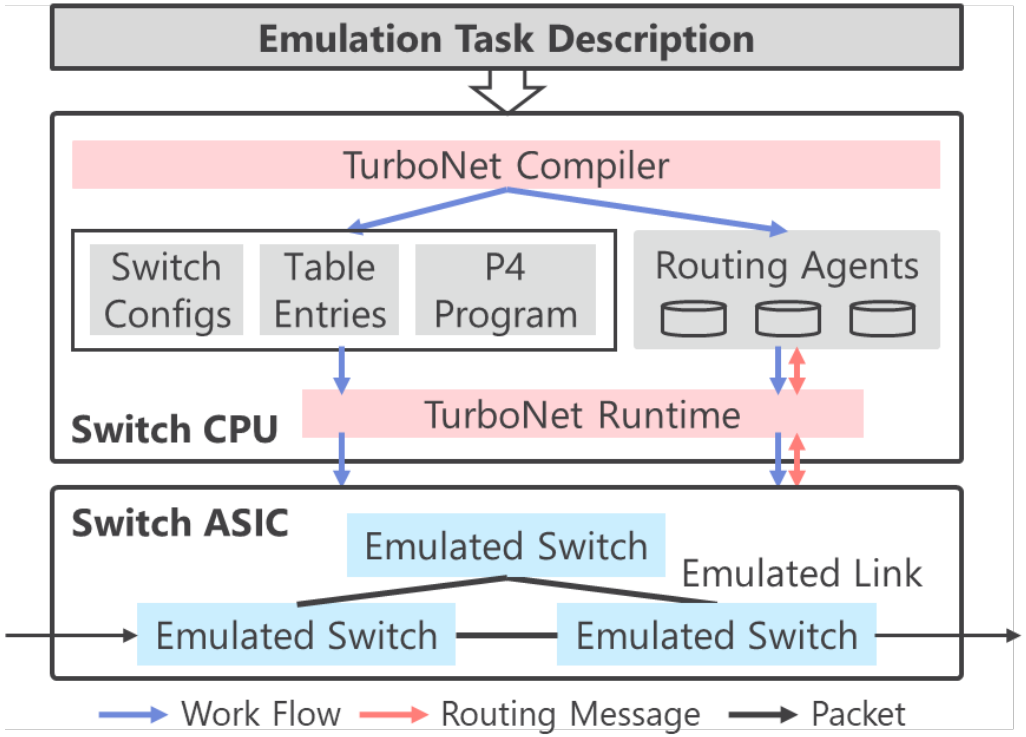
TurboNet Workflow



How?



TurboNet Architecture



Talk Outline

- Motivation
- 1. TurboNet Overview
- **2. Data Plane Emulation**
- 3. Control Plane Emulation
- 4. Some Evaluation Results
- 5. Conclusion

Data Plane Emulation

Forwarding packets in a network with specified performance

- Topology Emulation: flexibly emulate various topologies in a programmable switch
- Metric Emulation: make the emulated network behave like real networks
  - e.g., link loss, link delay, …

Data Plane Emulation

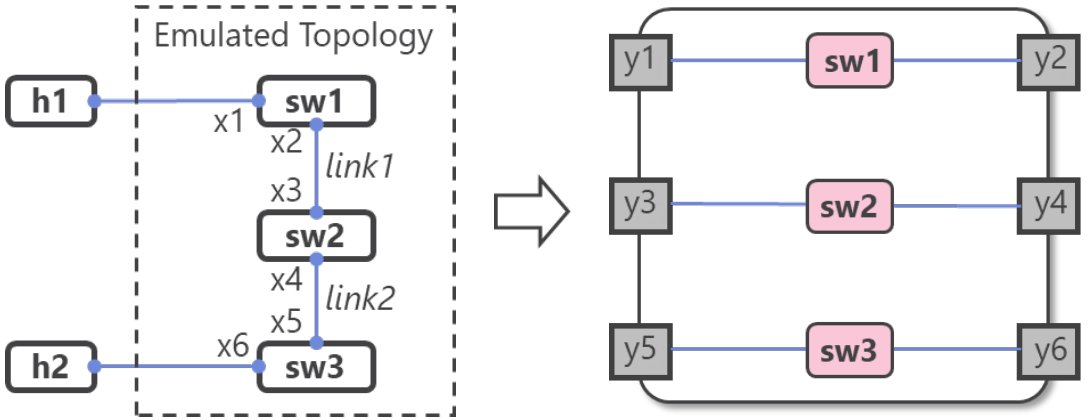
Forwarding packets in a network with specified performance

- Topology Emulation: flexibly emulate various topologies in a programmable switch
- Metric Emulation: make the emulated network behave like real networks
  - e.g., link loss, link delay, …

Topology Emulation

Key Idea: Split one programmable switch into multiple emulated

- Port Mapper



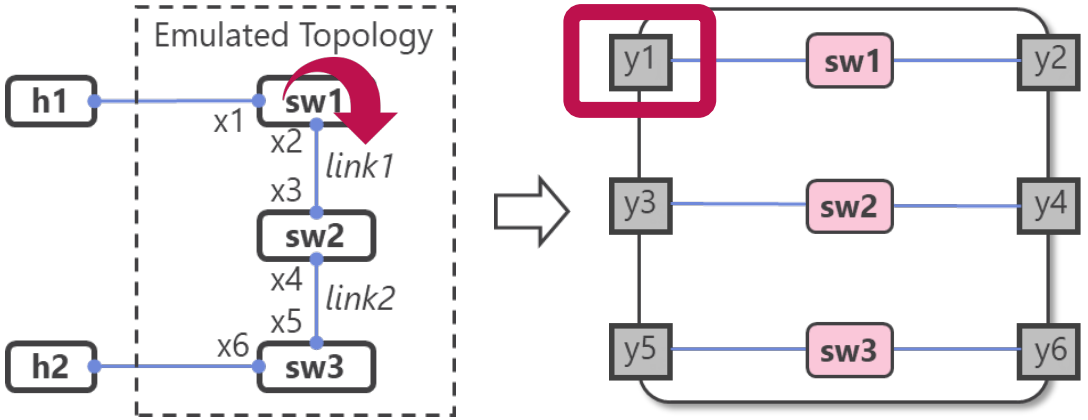
Map each port in the input topology to an individual port on the programmable switch (x1~x6 -> y1~y6)

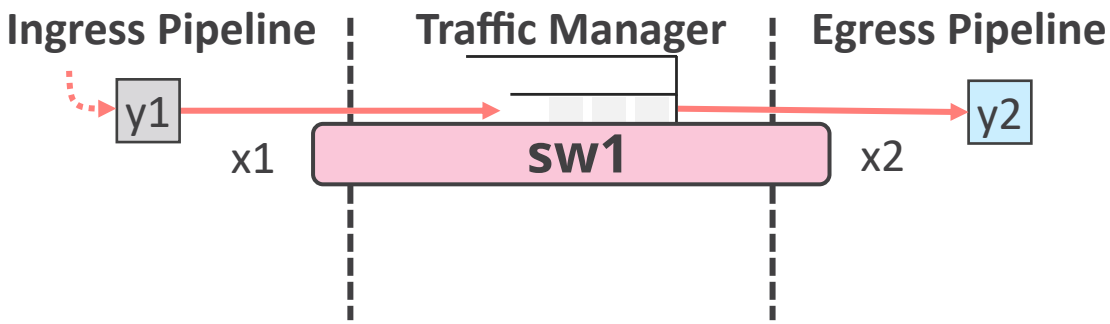
Switch Emulation

Topology Emulation

Key Idea: Split one programmable switch into multiple emulated

- Port Mapper





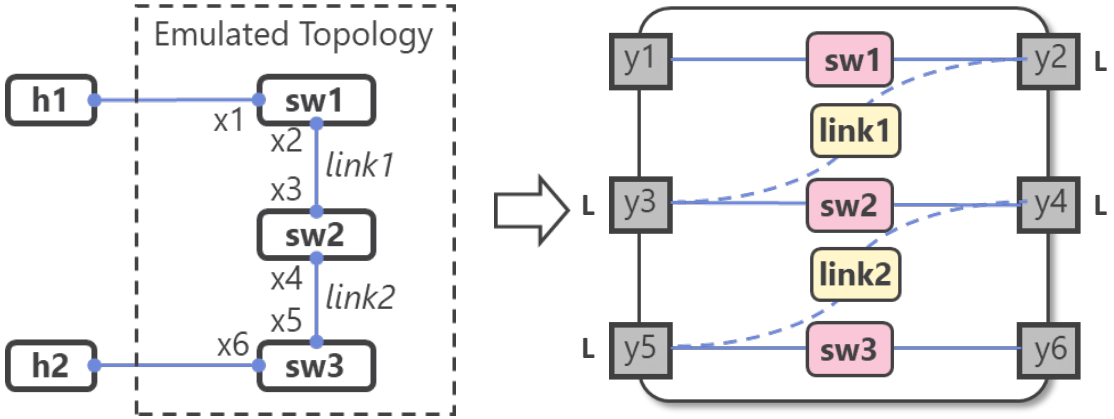
Map each port in the input topology to an individual port on the programmable switch (x1~x6 -> y1~y6)

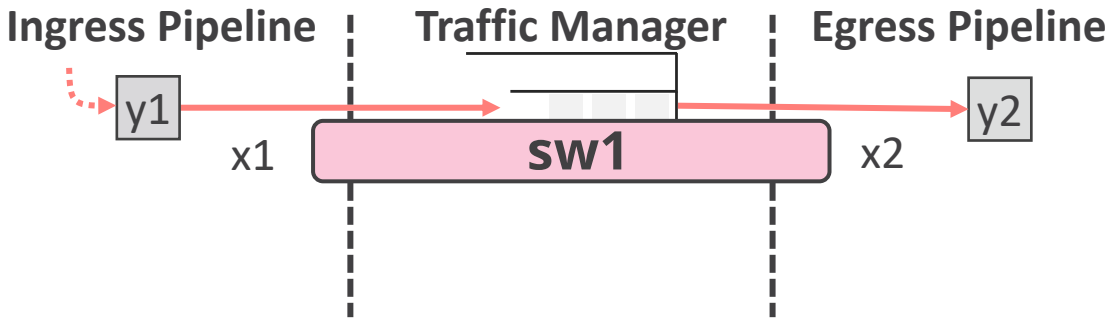
Switch Emulation

Topology Emulation

Key Idea: Split one programmable switch into multiple emulated

- Port Mapper





Map each port in the input topology to an individual port on the programmable switch (x1~x6 -> y1~y6)

Switch Emulation

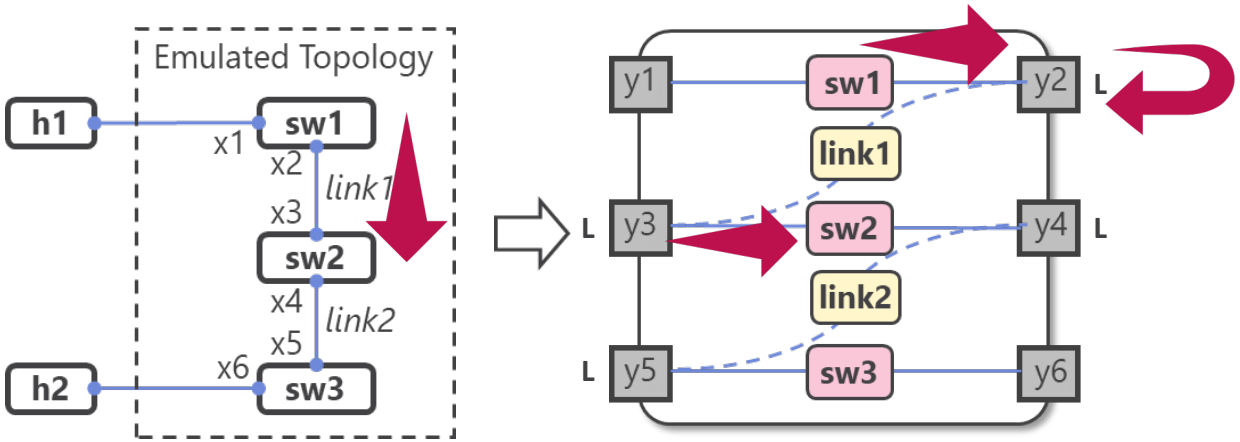
Link Emulation

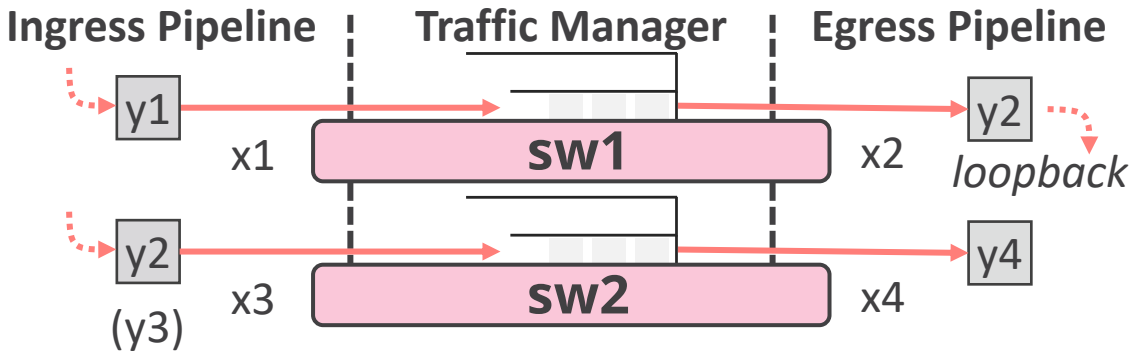
- ***Cables***
- ***Loopback***

Topology Emulation

Key Idea: Split one programmable switch into multiple emulated

- Port Mapper





Map each port in the input topology to an individual port on the programmable switch (x1~x6 -> y1~y6)

Switch Emulation

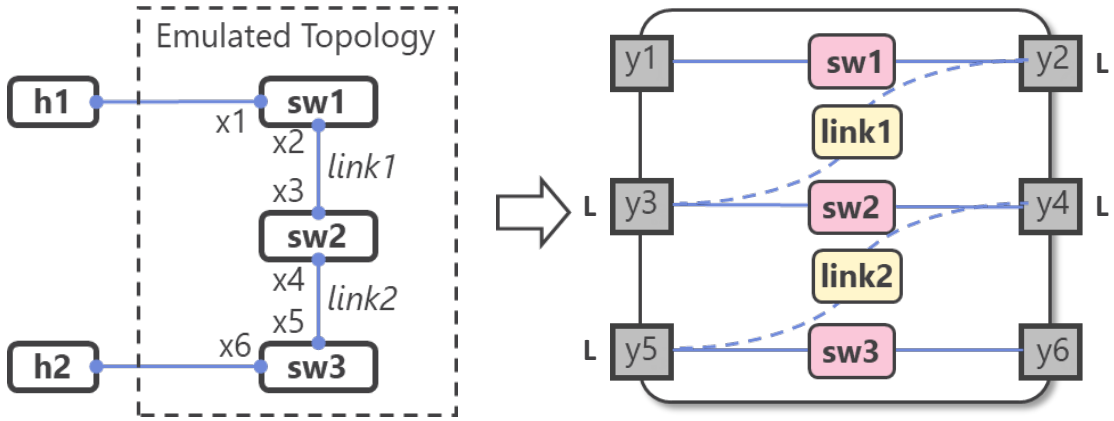
Link Emulation

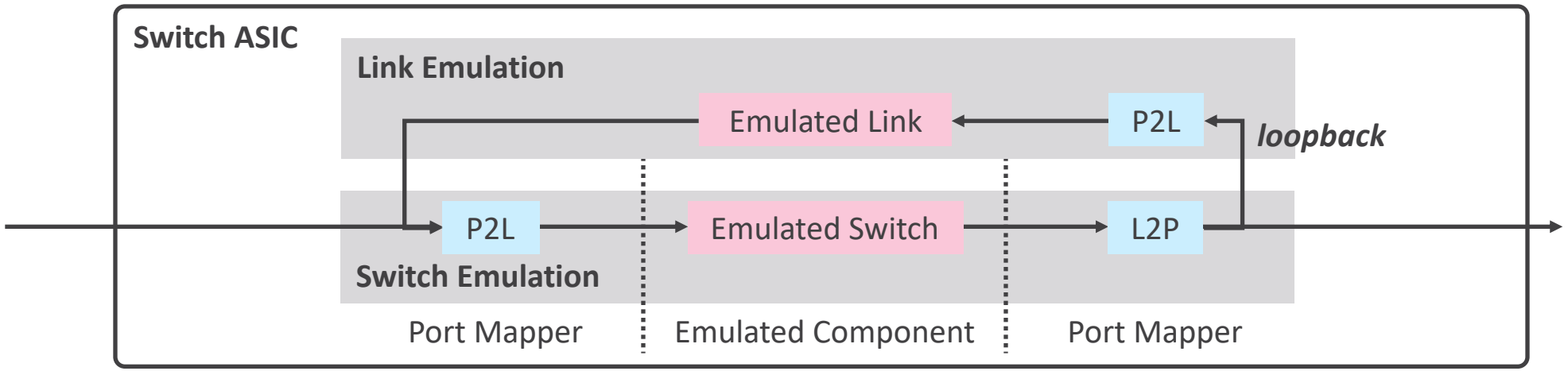
- ***Cables***
- ***Loopback***

Topology Emulation

Key Idea: Split one programmable switch into multiple emulated

- Port Mapper





Topology Emulation

Key Idea: Split one programmable switch into multiple emulated

- Port Mapper ***At most 64*4 ports?***

Barefoot Tofino switch
- 64 100Gbps port groups
- A port group:
  - 1x100G, 2x40G, 4x25G, 4x10G ports

Topology Emulation

Key Idea: Split one programmable switch into multiple emulated

- Port Mapper ***At most 64*4 ports?***

Challenge: How to emulate larger networks?

- Queue Mapper! ***At most 64*32 queues!***

Barefoot Tofino switch
- 64 100Gbps port groups
- A port group:
  - 1x100G, 2x40G, 4x25G, 4x10G ports
  - 32 queues

Topology Emulation

Key Idea: Split one programmable switch into multiple emulated

- Port Mapper ***At most 64*4 ports?***

Challenge: How to emulate larger networks?

- Queue Mapper! ***At most 64*32 queues!***

Problem: How to perform port/queue mapping?

- ***0-1 ILP problem***

Topology Emulation

Key Idea: Split one programmable switch into multiple emulated

- Port Mapper ***At most 64*4 ports?***

Challenge: How to emulate larger networks?

- Queue Mapper! ***At most 64*32 queues?***

Problem: How to perfo

- ***0-1 ILP problem***

Input
- Ports
- Bandwidth
- Queue number

(See more in our paper)

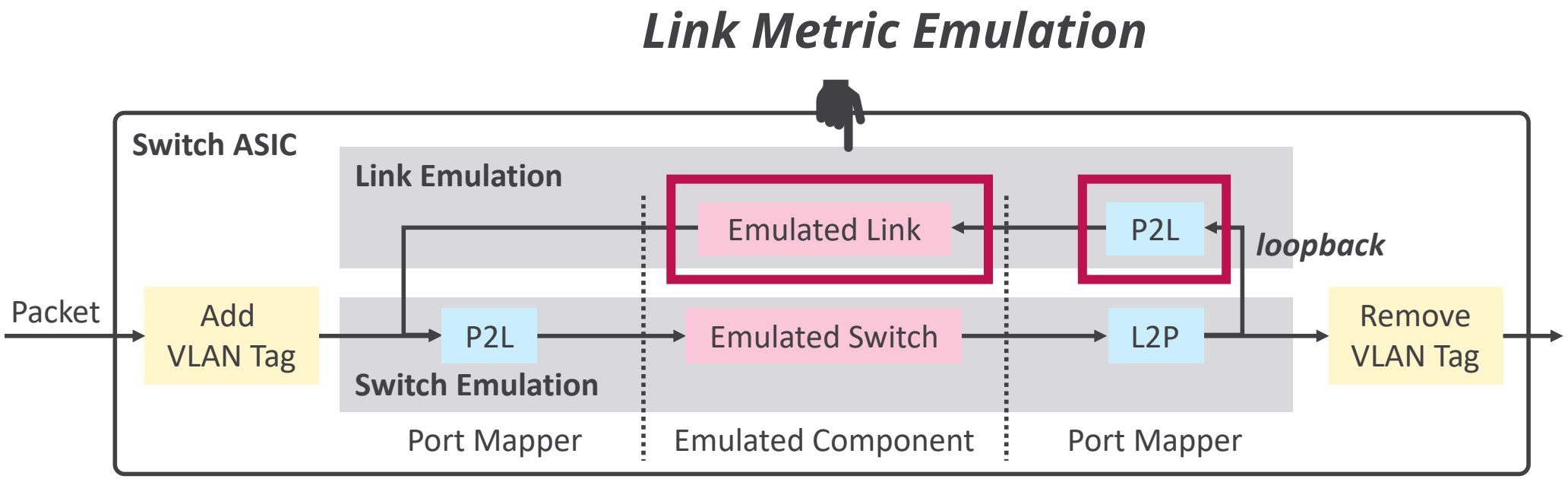
PM Problem	$x_i^{1,2,3,4}, x_{ik}^{1,2,3,4} = PM(S, b_k, q_k, P)$
Objective	$min: \sum_{i \in P} (x_i^1 + x_i^2 + x_i^3 + x_i^4)$
	$x_i^3 + x_i^4 \leq 1, \forall i \in P$
	$3 \sum x_{ik}^4 \leq 4 \cdot x_i^4, \forall i \in P$

QM Problem	$y_i, y_{ik}, x_i^{1,2,3,4}, x_{ik}^{1,2,3,4} = QM(S_1, S_2, b_k, q_k, P)$
Objective	$min: \sum_{i \in P} x_i^1 + x_i^2 + x_i^3 + x_i^4 + y_i$
Constraints	$C_1: x_i^1 + x_i^2 + x_i^3 + x_i^4 + y_i \leq 1, \forall i \in P$
	$C_2: x_i^{1,2,3,4}, x_{ik}^{1,2,3,4} = PM(S_2, b_k, q_k, P)$
	$C_3: \sum_{k \in S_1} b_k y_{ik} \leq 100 y_i, \forall i \in P$
	$C_4: \sum_{i \in P} y_{ik} = 1, \forall k \in S_1$
	$C_5: \sum_{k \in S_1} q_k y_{ik} \leq 32 y_i, \forall i \in P$

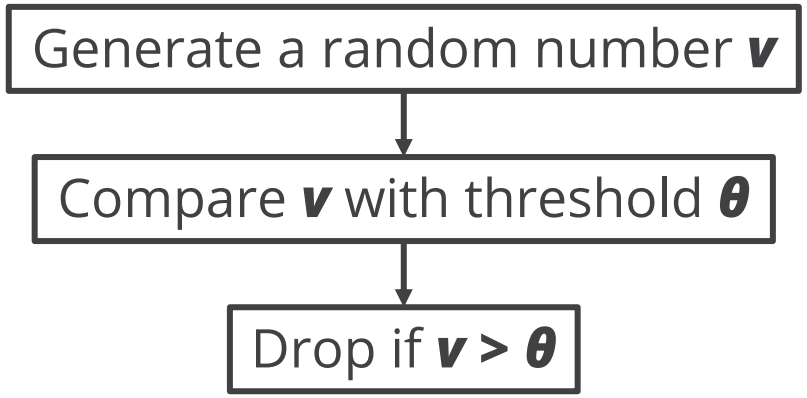
Data Plane Emulation

- Topology Emulation
- Metric Emulation: make the emulated network behave like real networks
  - Link loss
  - Link delay
  - Background traffic

Link Loss Emulation



Link Loss Emulation

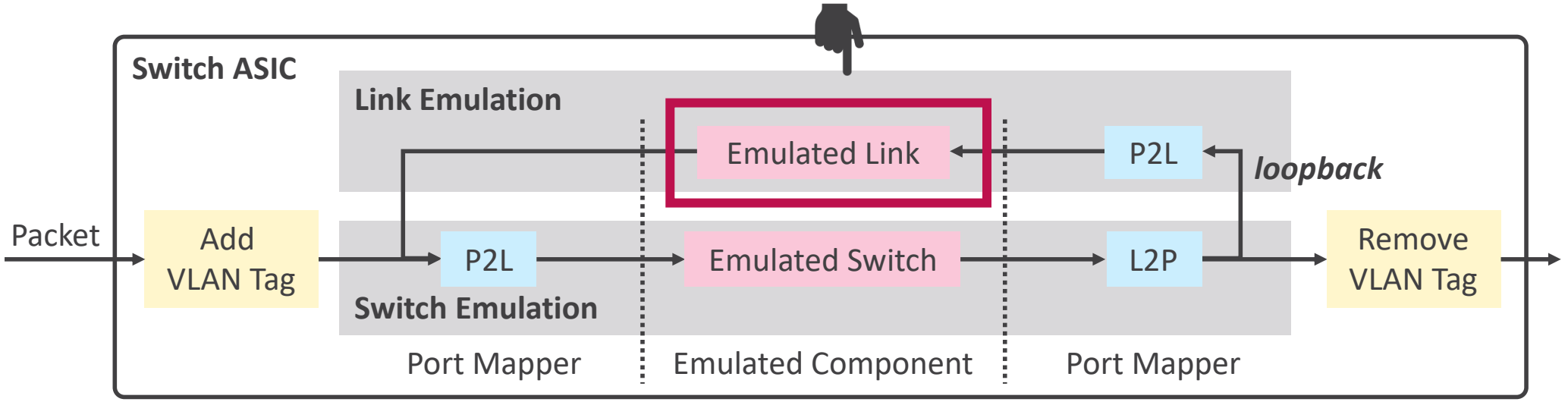


P4 primitive: modify_field_rng_uniform

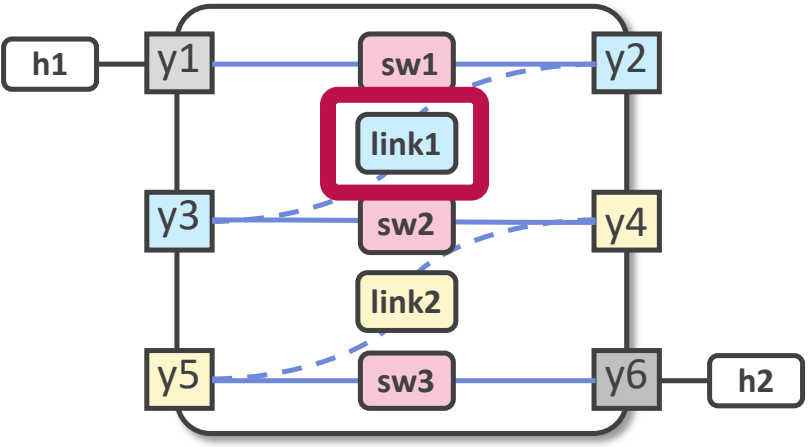
θ is decided by given loss rate **r: P(v > θ) = r**

Drop packets according to a probability

Link Loss Emulation



Link Delay Emulation

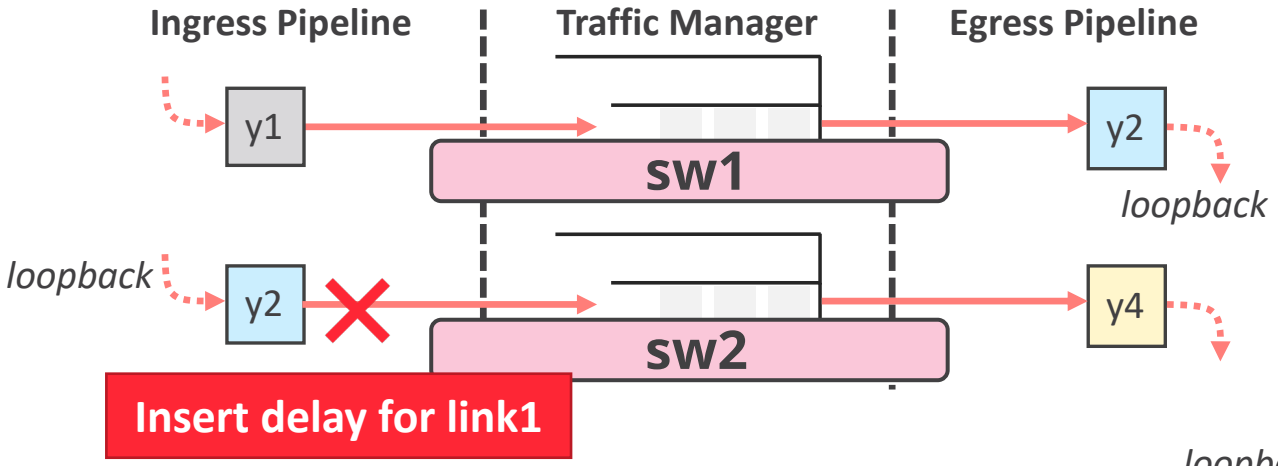


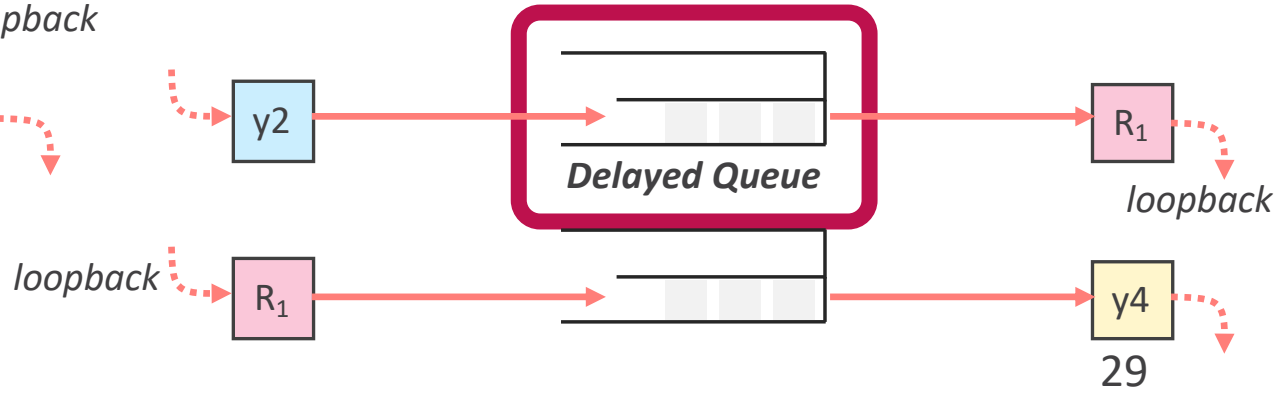
Challenge: How to keep packets in programmable switch?

Observation: queuing time is in proportion to queue depth.

To achieve target queuing time, maintain queue depth!

- Implement ***delayed queues*** with desired queuing time
- Send packets to delayed queues to emulate target delay

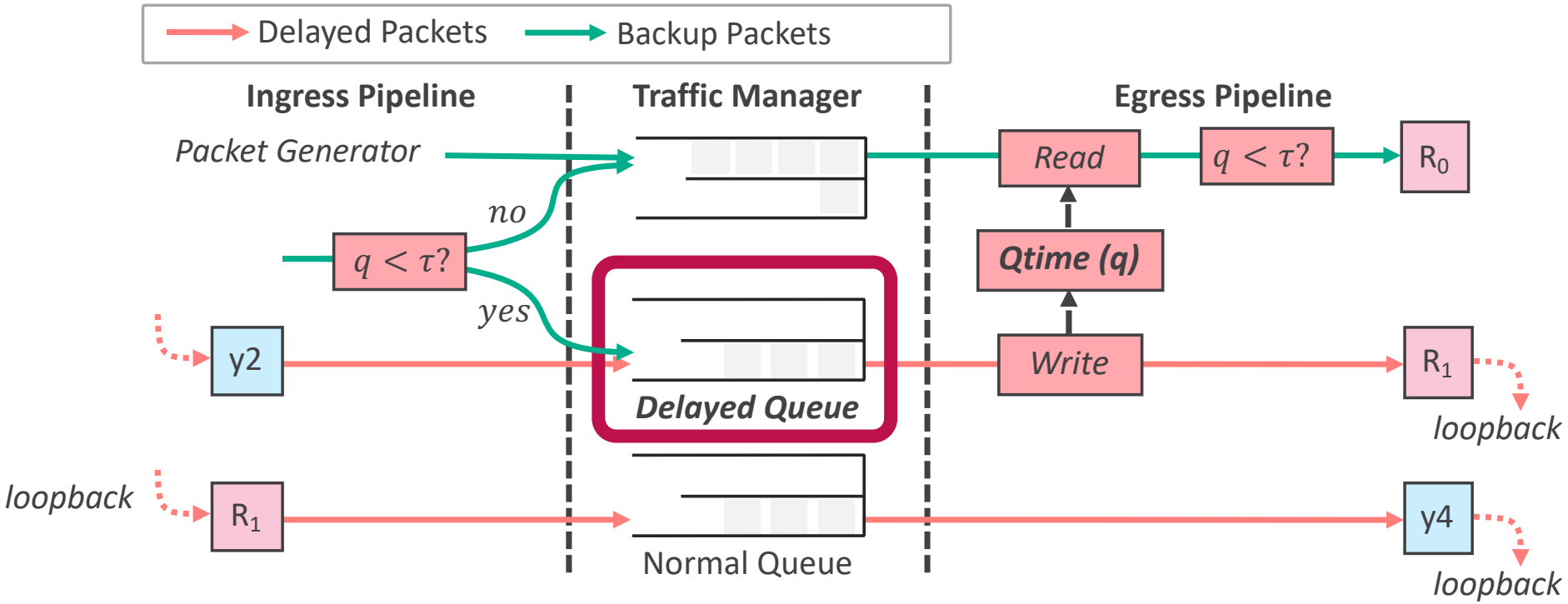




Link Delay Emulation

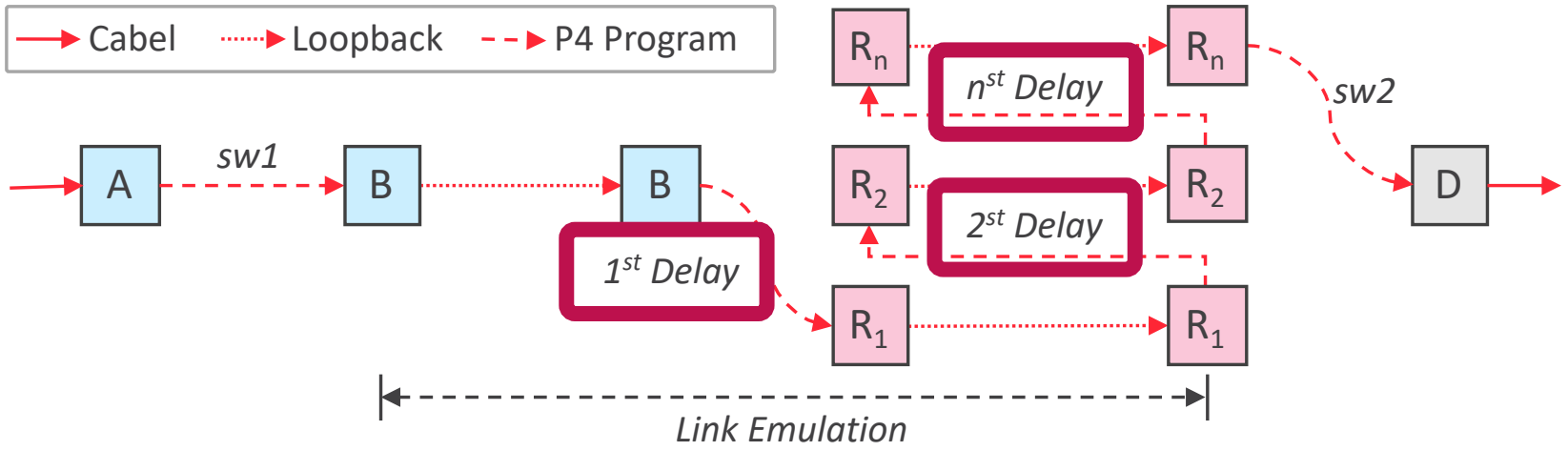
1. Implement delayed queues with desired queuing time

- dynamically inject packets



Link Delay Emulation

1. Implement delayed queues with desired queuing time
   - dynamically inject backup packets
2. Send packets to delayed queues to emulate target delay



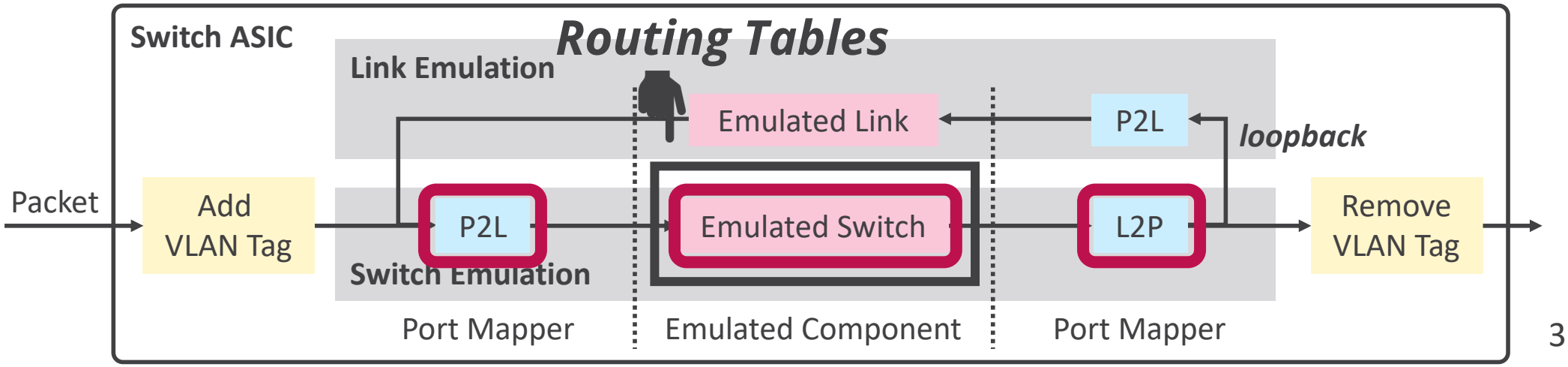
More details (resource, accuracy) in our paper

Talk Outline

- Motivation
- 1. TurboNet Overview
- 2. Data Plane Emulation
- **3. Control Plane Emulation**
- 4. Some Evaluation Results
- 5. Conclusion

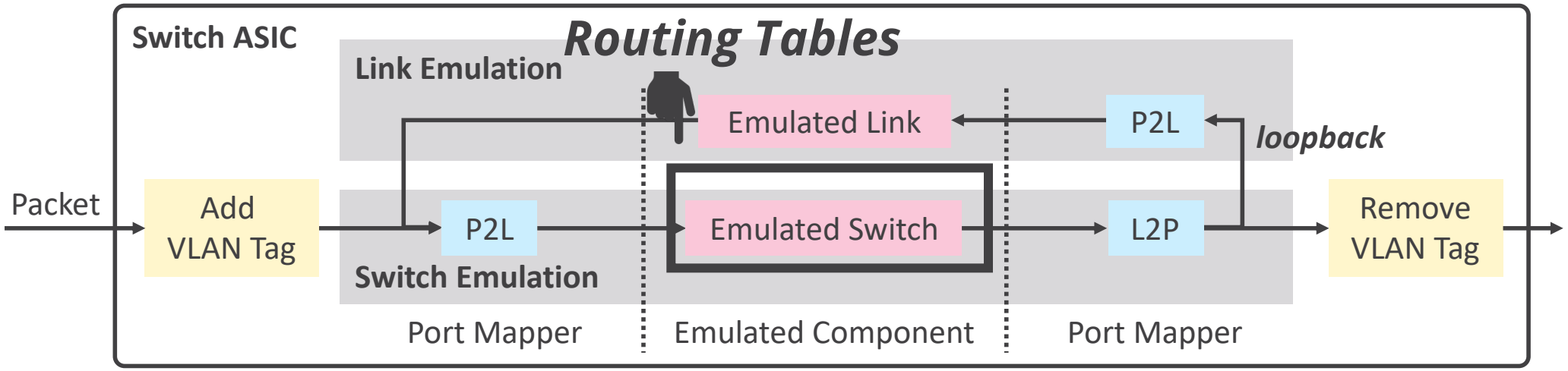
Control plane emulation

- Static Routing
- Dynamic routing



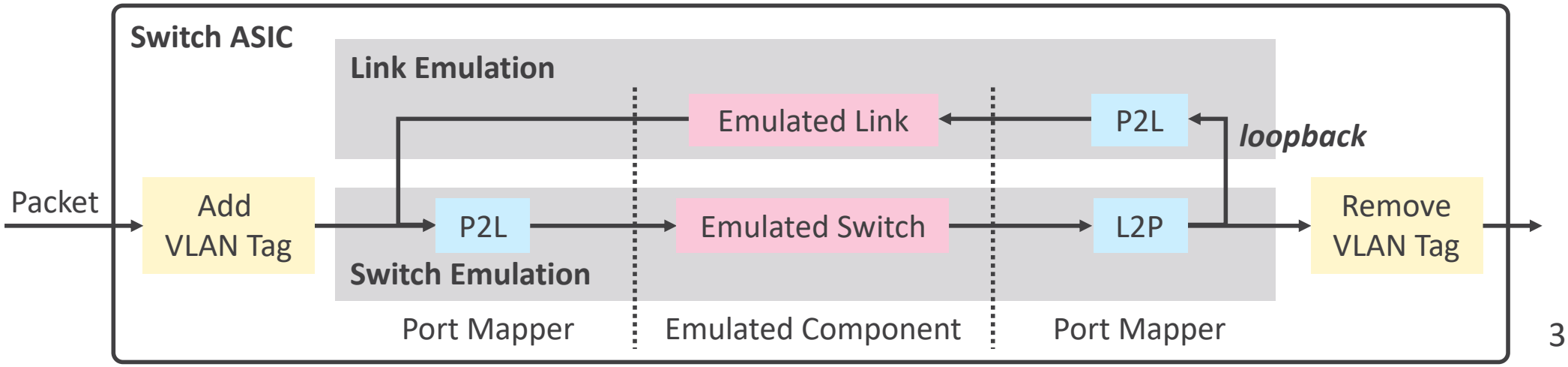
Control plane emulation

- Static Routing
- Dynamic routing



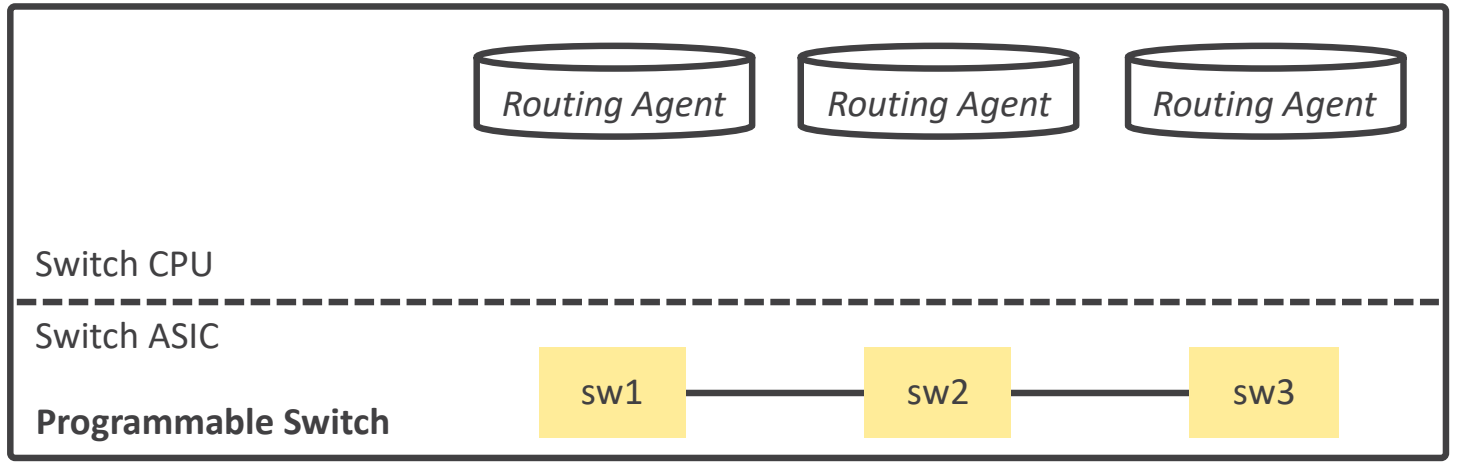
Control plane emulation

- Static Routing
- Dynamic routing
  - Distributed routing
  - Centralized routing



Control plane emulation

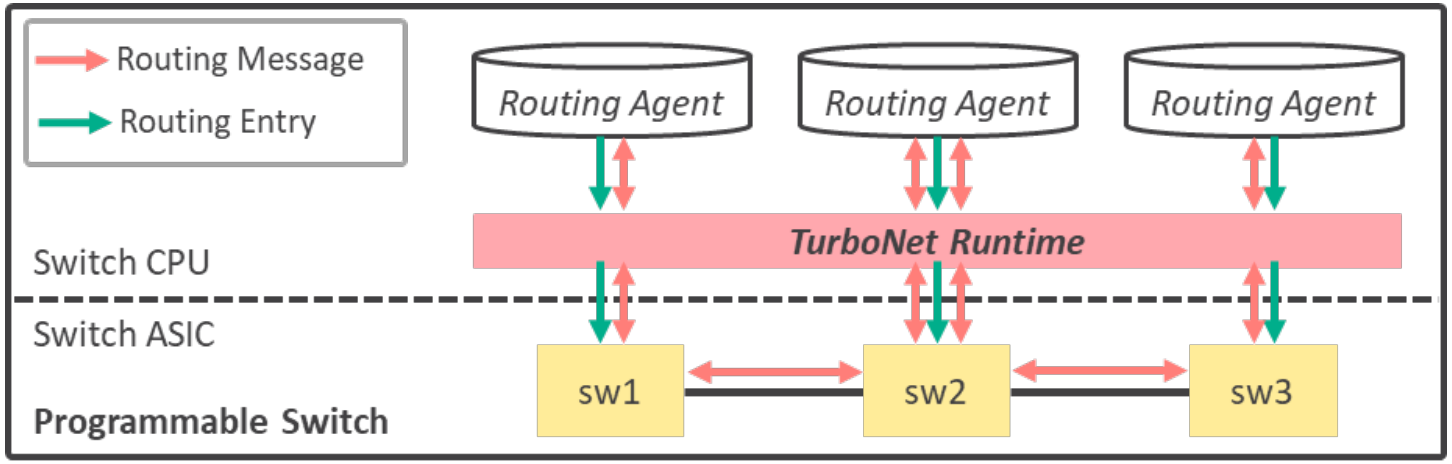
- Static Routing
- Dynamic routing
  - Distributed routing
  - Centralized routing



Distributed Routing

Control plane emulation

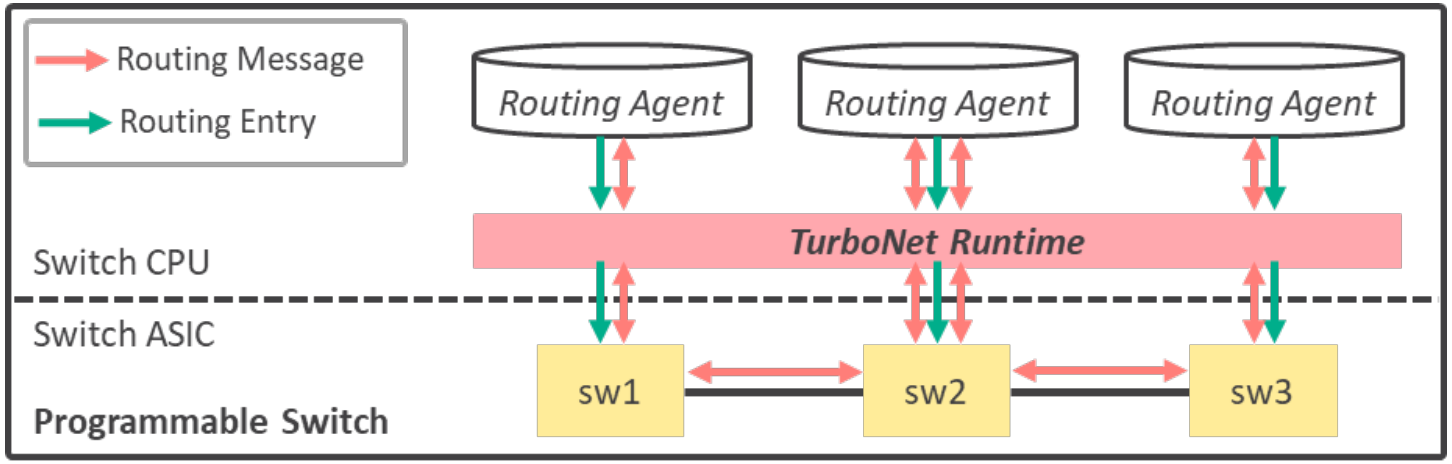
- Static Routing
- Dynamic routing
  - Distributed routing
  - Centralized routing



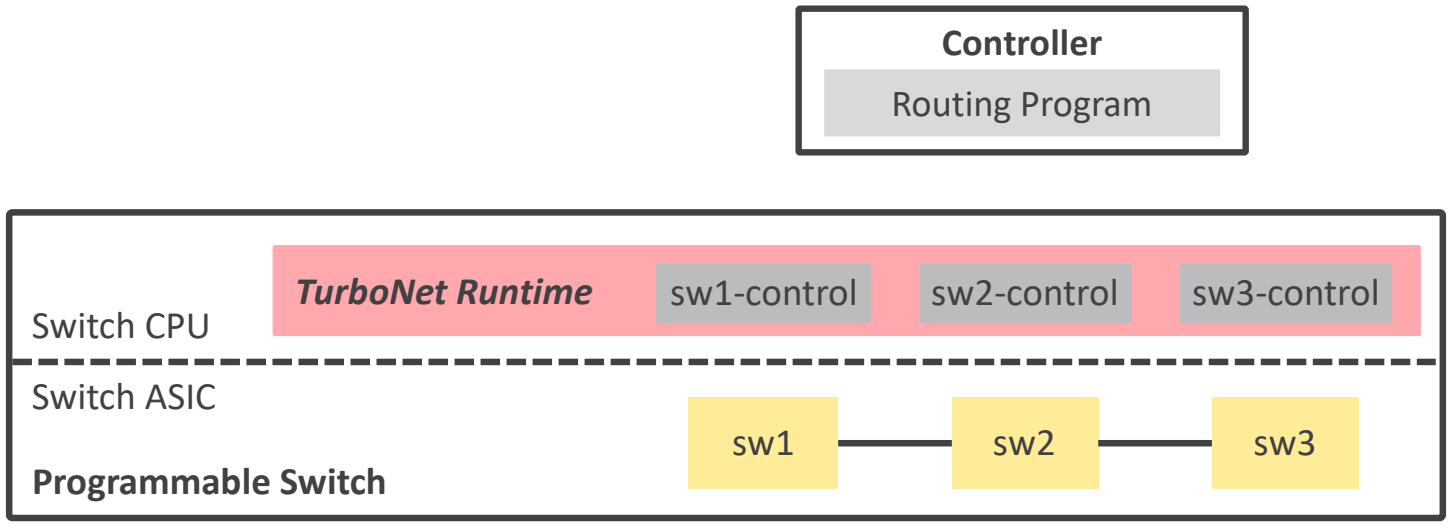
Distributed Routing

Control plane emulation

- Static Routing
- Dynamic routing
  - Distributed routing
  - Centralized routing



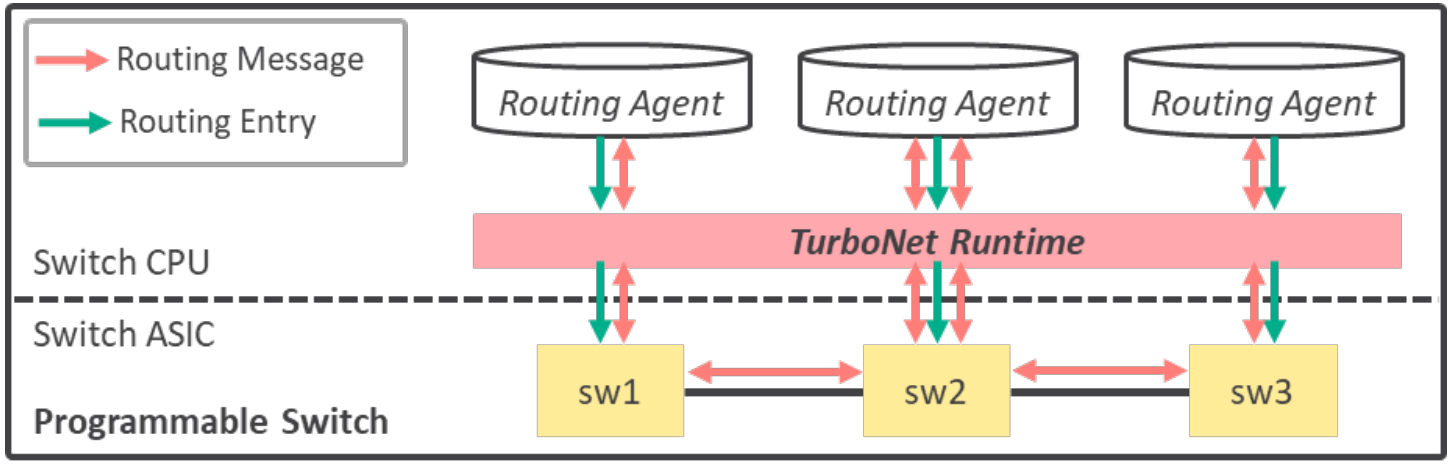
Distributed Routing



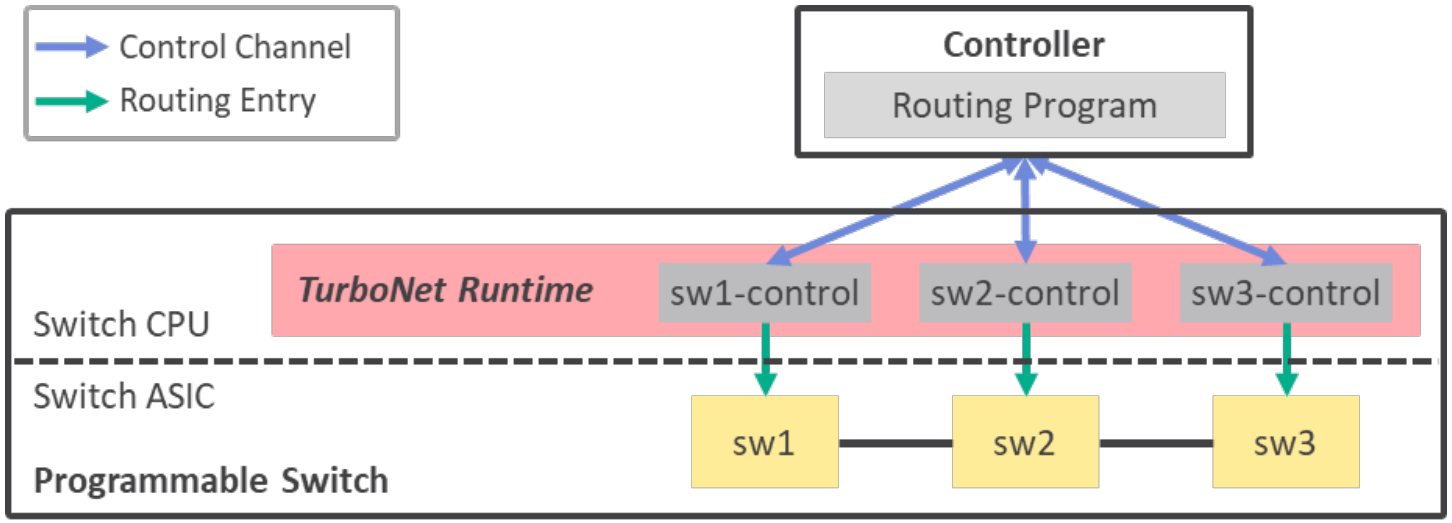
Centralized Routing

Control plane emulation

- Static Routing
- Dynamic routing
  - Distributed routing
  - Centralized routing



Distributed Routing



Centralized Routing

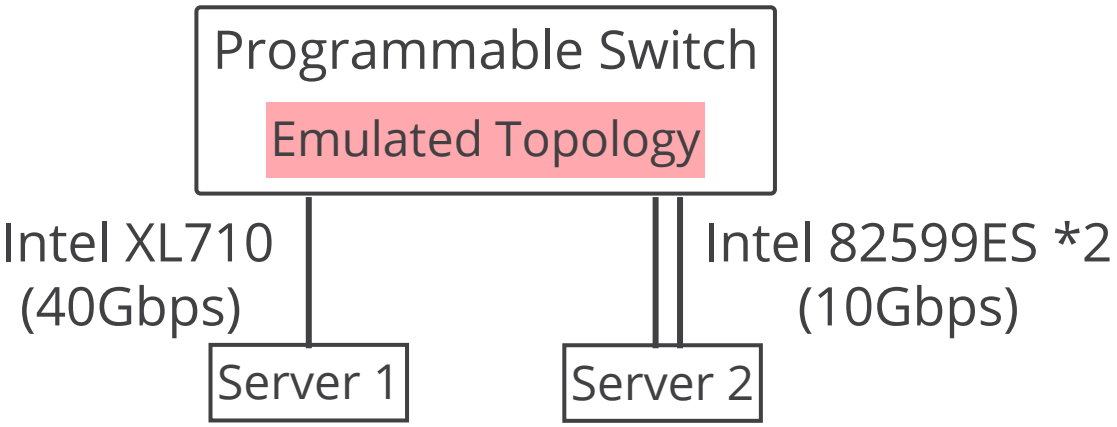
Talk Outline

- Motivation
- 1. TurboNet Overview
- 2. Data Plane Emulation
  - Topology Emulation
  - Metric Emulation: Background Traffic, Link Loss, Link Delay
- 3. Control Plane Emulation
- **4. Some Evaluation Results**
- 5. Conclusion

Evaluation Setup

Test-bed

- One Barefoot Tofino Switch
  - Intel Pentium 4-core 1.60GHz CPU
  - 8GiB memory
  - 2 pipelines (3.2Tbps)
- Two servers
  - 12-core Intel Xeon E5-2620 2.40GHz CPU



Topology Emulation Capability

Topology		TurboNet			
		2-Pipe (32*100G) Programmable Switch		4-Pipe (64*100G) Programmable Switch	
		PM	QM	PM	QM
Fat-Tree	$k = 4$	$Link \leq 25G$	$Link \leq 25G$	$Link \leq 40G$	$Link \leq 40G$
	$k = 6$	×	$Link \leq 8.3G$	×	$Link \leq 20G$
	$k = 8$	×	×	×	$Link \leq 6.2G$
VL2	$Link = 1G, 10G$	$D_A D_1 \leq 16$	$D_A D_1 \leq 20$	$D_A D_1 \leq 36$	$D_A D_1 \leq 44$
	$Link = 10G, 40G$	$D_A D_1 \leq 20$	$D_A D_1 \leq 20$	$D_A D_1 \leq 40$	$D_A D_1 \leq 40$
261 Internet Topologies		199 (76.2%)	259 (99.2%)	248 (95.0%)	260 (99.6%)

PM VS. QM

- Fat-tree: k=4 for PM, k=8 for QM

Note: A k-ary fat-tree has $k^3/4$ host-ports and k^3 switch-ports

Topology Emulation Capability

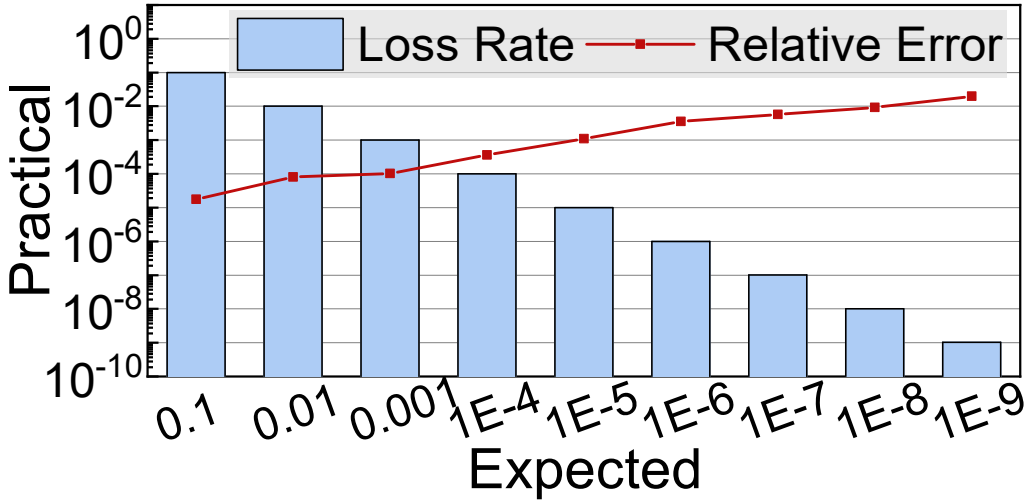
Topology		TurboNet			
		2-Pipe (32*100G) Programmable Switch		4-Pipe (64*100G) Programmable Switch	
		PM	QM	PM	QM
Fat-Tree	$k = 4$	$Link \leq 25G$	$Link \leq 25G$	$Link \leq 40G$	$Link \leq 40G$
	$k = 6$	×	$Link \leq 8.3G$	×	$Link \leq 20G$
	$k = 8$	×	×	×	$Link \leq 6.2G$
VL2	$Link = 1G, 10G$	$D_A D_1 \leq 16$	$D_A D_1 \leq 20$	$D_A D_1 \leq 36$	$D_A D_1 \leq 44$
	$Link = 10G, 40G$	$D_A D_1 \leq 20$	$D_A D_1 \leq 20$	$D_A D_1 \leq 40$	$D_A D_1 \leq 40$
261 Internet Topologies		199 (76.2%)	259 (99.2%)	248 (95.0%)	260 (99.6%)

PM VS. QM

- Fat-tree: k=4 for PM, k=8 for QM
- Internet topology: 60 more for QM

Note: A k-ary fat-tree has $k^3/4$ host-ports and k^3 switch-ports

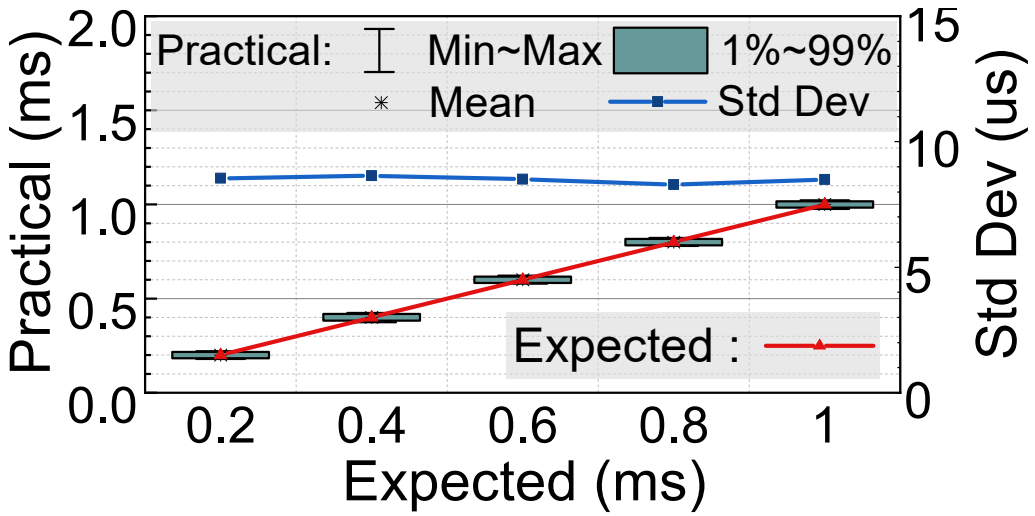
Link Loss Accuracy



Link loss accuracy
(32-bit random number generation)

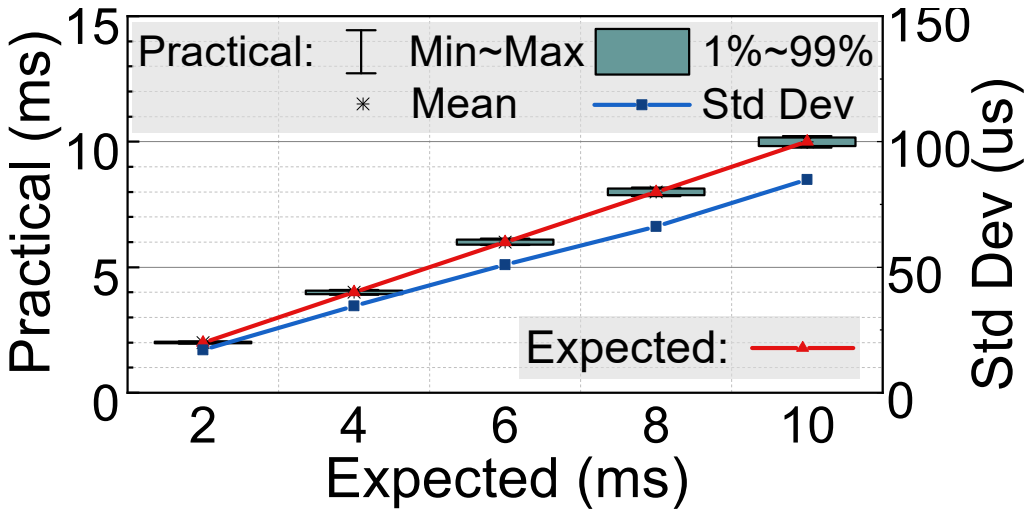
Expected loss rate changes from 0.1 to 10^{-8}
Relative error < 1%

Link Delay Accuracy



One-pass link delay
(A 10Gbps delayed queue)

Expected delay ranges from 0.2ms to 1ms
Standard deviation < 9μs

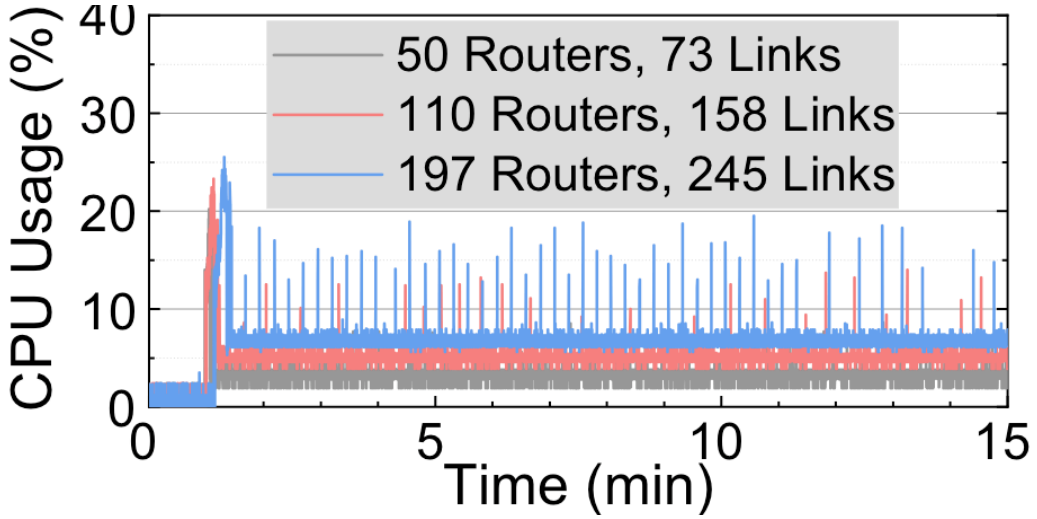


Multi-pass link delay
(A 1ms delayed queue)

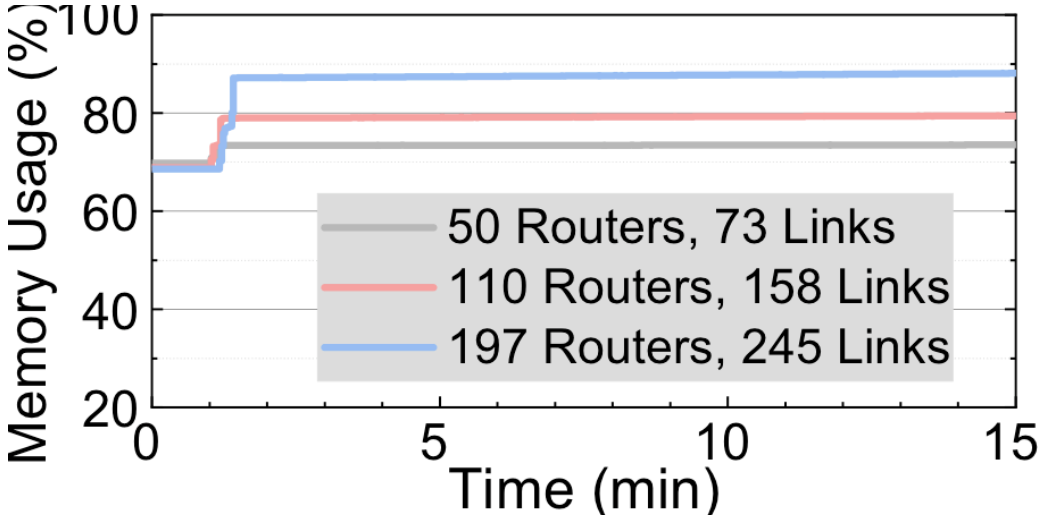
Expected delay ranges from 2ms to 10ms
Relative standard deviation < 0.9%

Control Plane Emulating Capability

Intel Pentium 4-core 1.60GHz CPU, 8GiB memory
All connected routers are BGP peers



CPU usage



Memory usage

TurboNet can easily support almost 200 BGP routing agents with 25% peak CPU usage.

Talk Outline

- Motivation
- 1. TurboNet Overview
- 2. Data Plane Emulation
  - Topology Emulation
  - Metric Emulation: Background Traffic, Link Loss, Link Delay
- 3. Control Plane Emulation
- **4. Some Evaluation Results**
- **5. Conclusion**

Conclusion

- **TurboNet is a network experiment platform that leverages one programmable switch to faithfully mimic networks.**
  - Data plane
  - Control plane

Thanks!