SOFTWARE DEFINED NETWORKS

Jonathan Chu

Muhammad Salman Malik

Credits

Material Derived from:

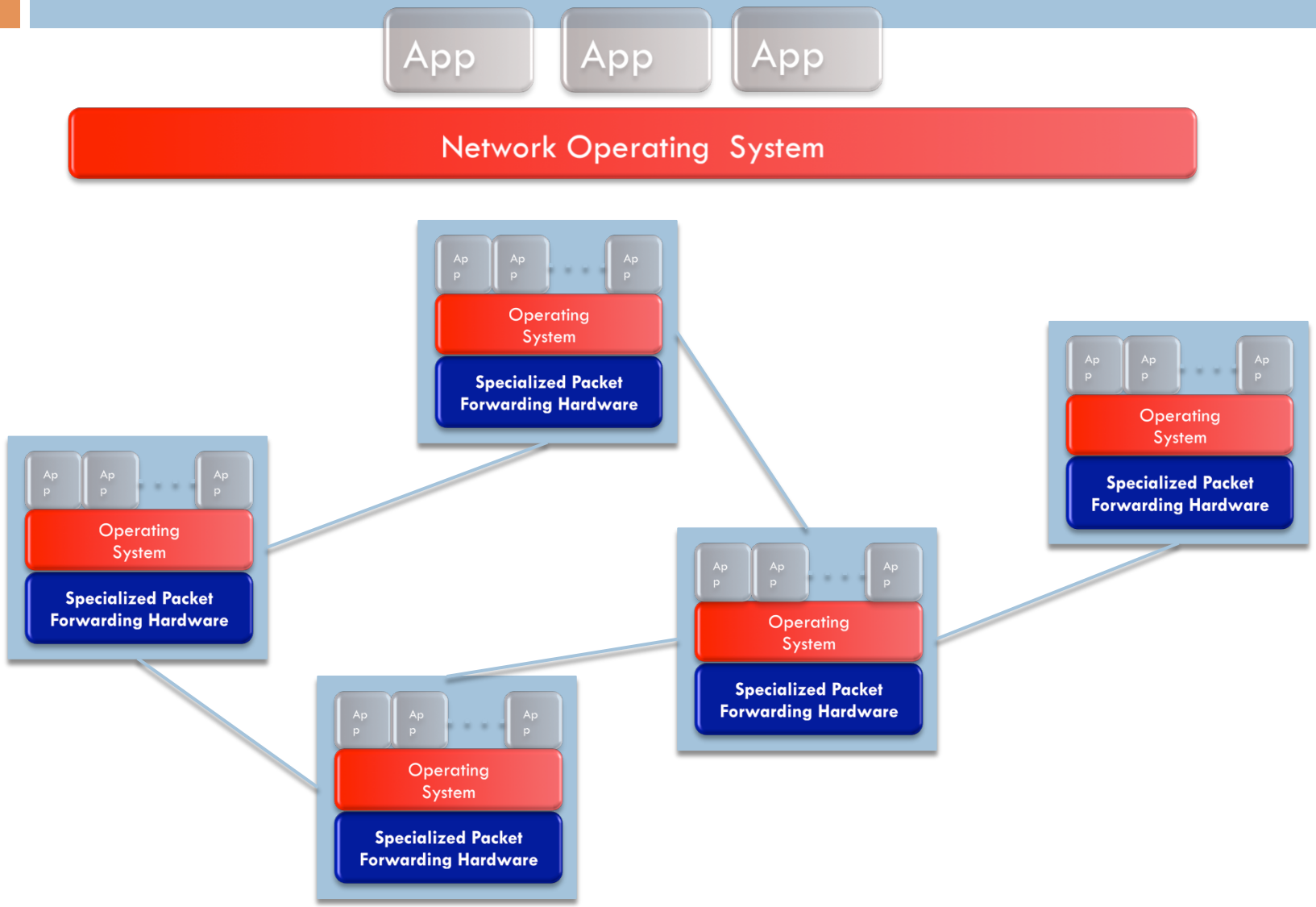
- Rob Sherwood, Saurav Das, Yiannis Yiakoumis AT&T Tech Talks October 2010

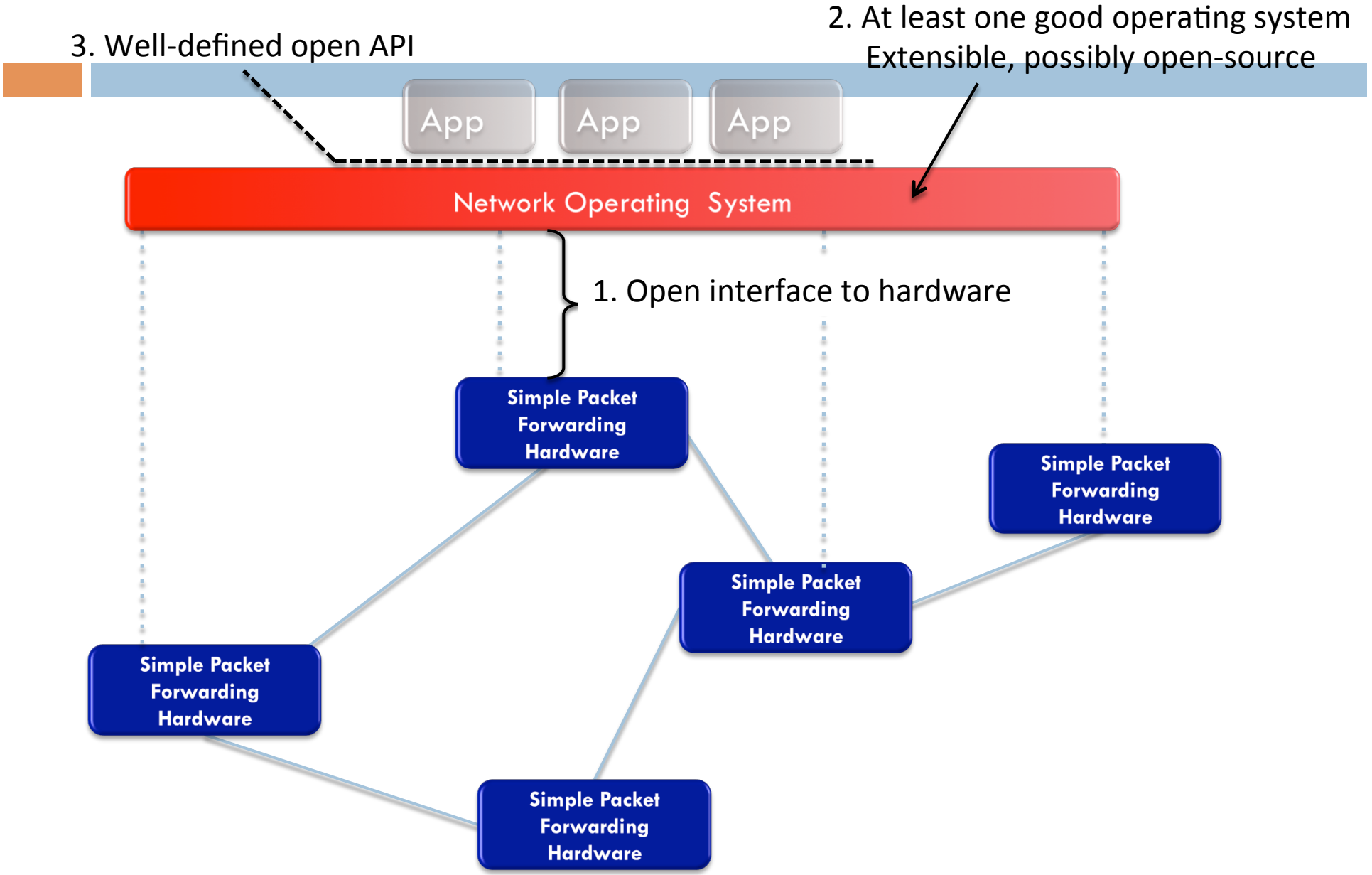
  (available at:*www.openflow.org/wk/images/1/17/OpenFlow_in_SPnetworks.ppt*)

- DevoFlow presentation by the author at SIGCOMM 2011

  (available at: *http://conferences.sigcomm.org/sigcomm/2011/slides/s254.pdf*)

Software Defined Networks




3. Well-defined open API
2. At least one good operating system
Extensible, possibly open-source
App
App
App
Network Operating System
1. Open interface to hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware
Simple Packet Forwarding Hardware

Why does it matter ?

- Easy to manage
- Support fine grained policies

Great !

But…

- What about Reliability ?
- What about performance?
- What about Scalability ?

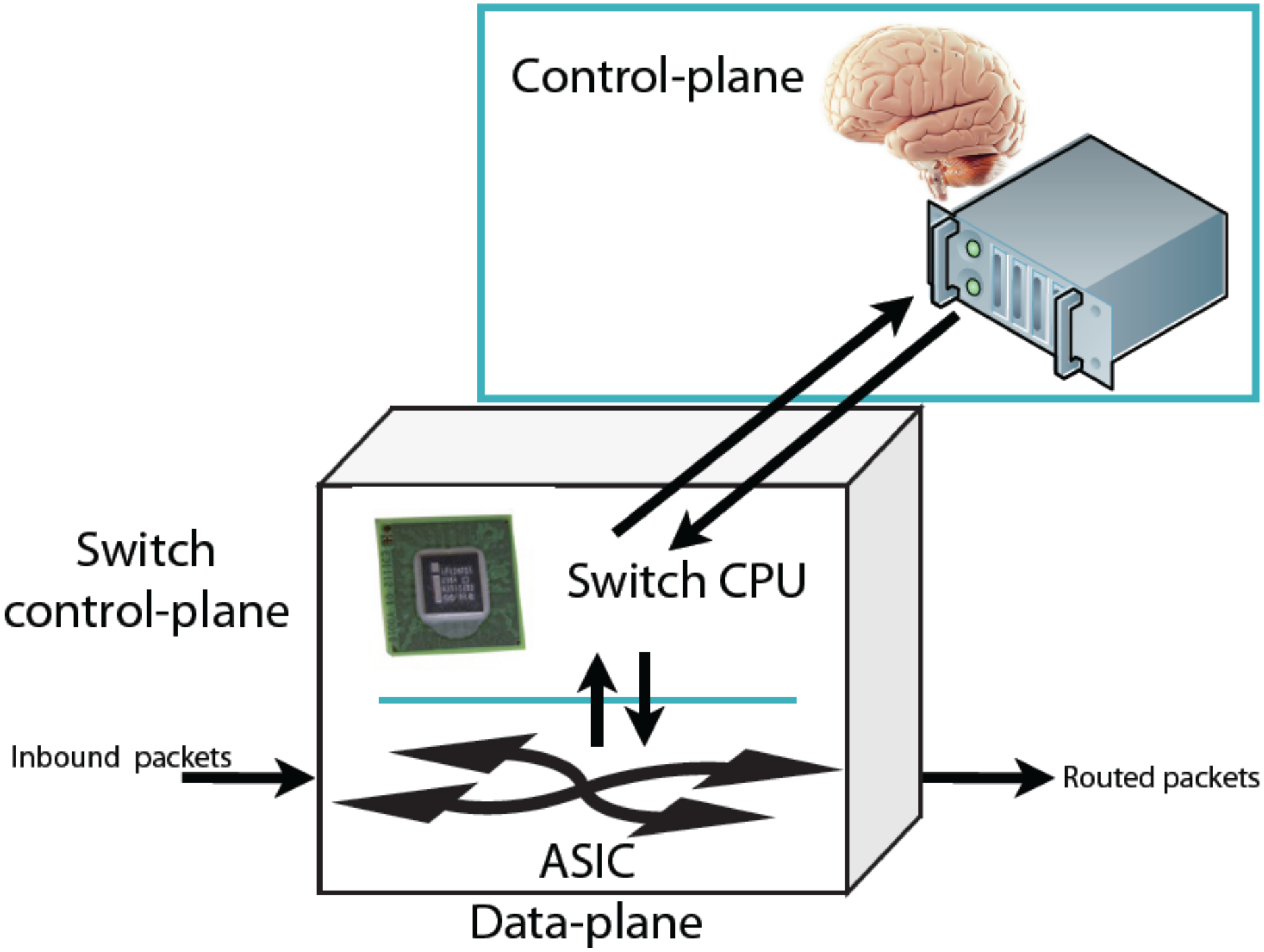
} DevoFlow, DIFANE

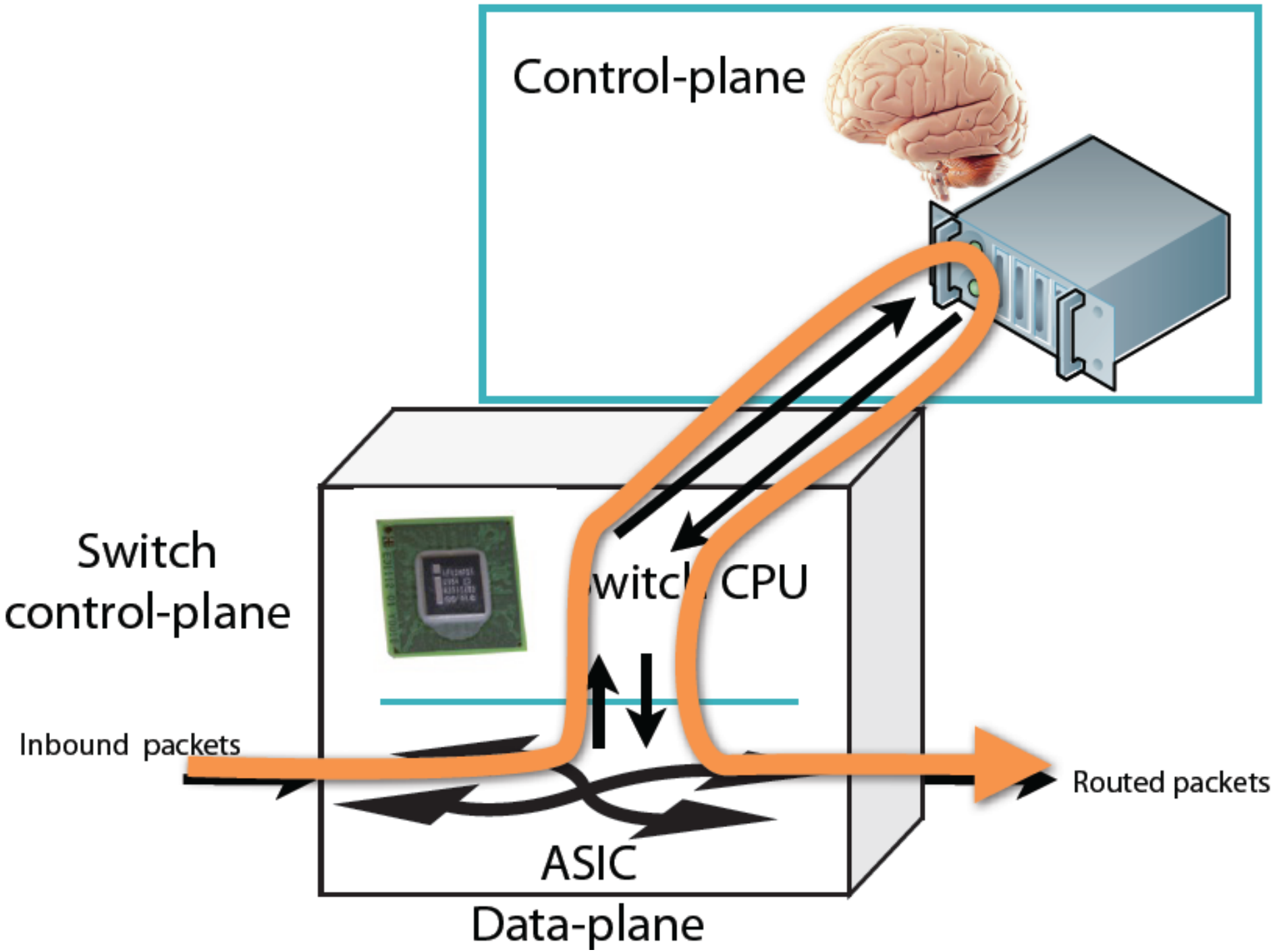
DEVOFLOW: SCALING FLOW MANAGEMENT FOR HIGH-PERFORMANCE NETWORKS

R. CURTIS, J. C. MOGUL, J. TOURRILHES, P. YALAGANDULA, P. SHARMA, AND S. BANERJEE. DEVOFLOW: SCALING FLOW MANAGEMENT FOR HIGH-

PERFORMANCE NETWORKS. IN ACM SIGCOMM,AUG. 2011.

Salman Malik

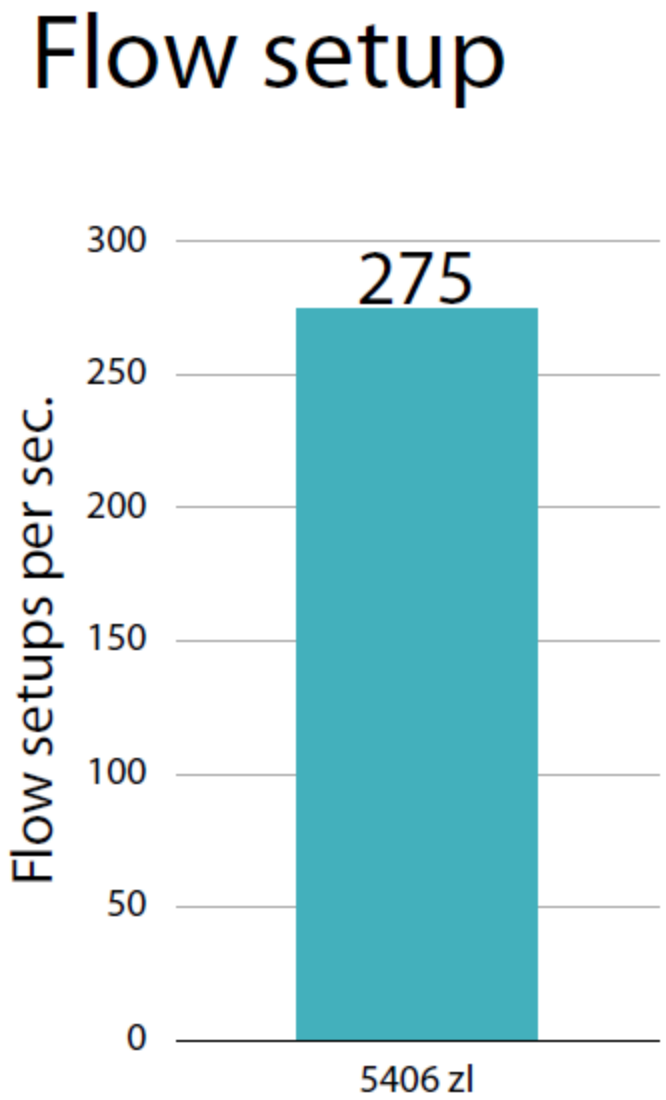

Control-plane
Switch
control-plane
Switch CPU
Inbound packets
Routed packets
ASIC
Data-plane


Control-plane
Switch
control-plane
CPU
Inbound packets
Routed packets
ASIC
Data-plane

Scaling problem: switches

- Inherent overheads
  - Bandwidth
    - OpenFlow creates too much control traffic
  - Latency
- Implementation-imposed overheads
  - Flow setup
  - Statistics gathering

Implementation Imposed Overhead



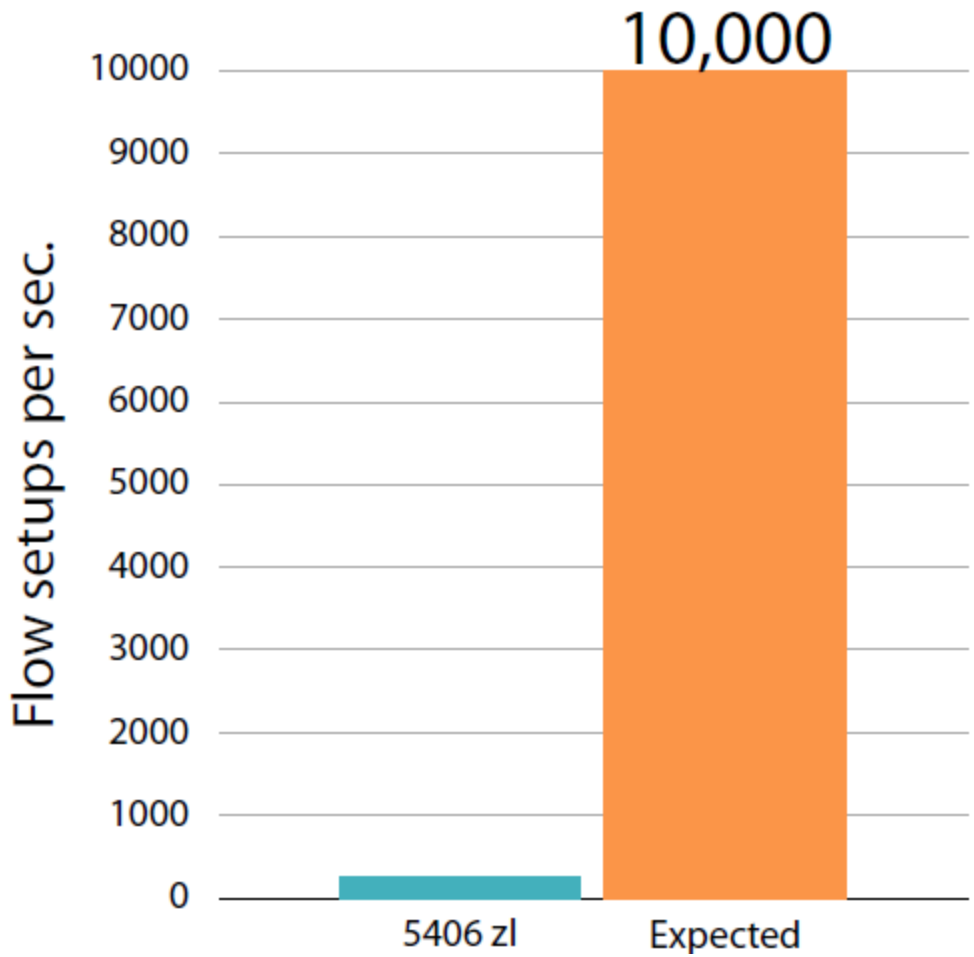
Switch and controller are connected by a fast medium, so why just 275 flows ? :O

Because:

-data-plane invokes control plane to send encapsulated packets to controller

-control data-path b/w ASIC and CPU is not frequently used in traditional operations

275 flow setups/sec seems reasonable ?

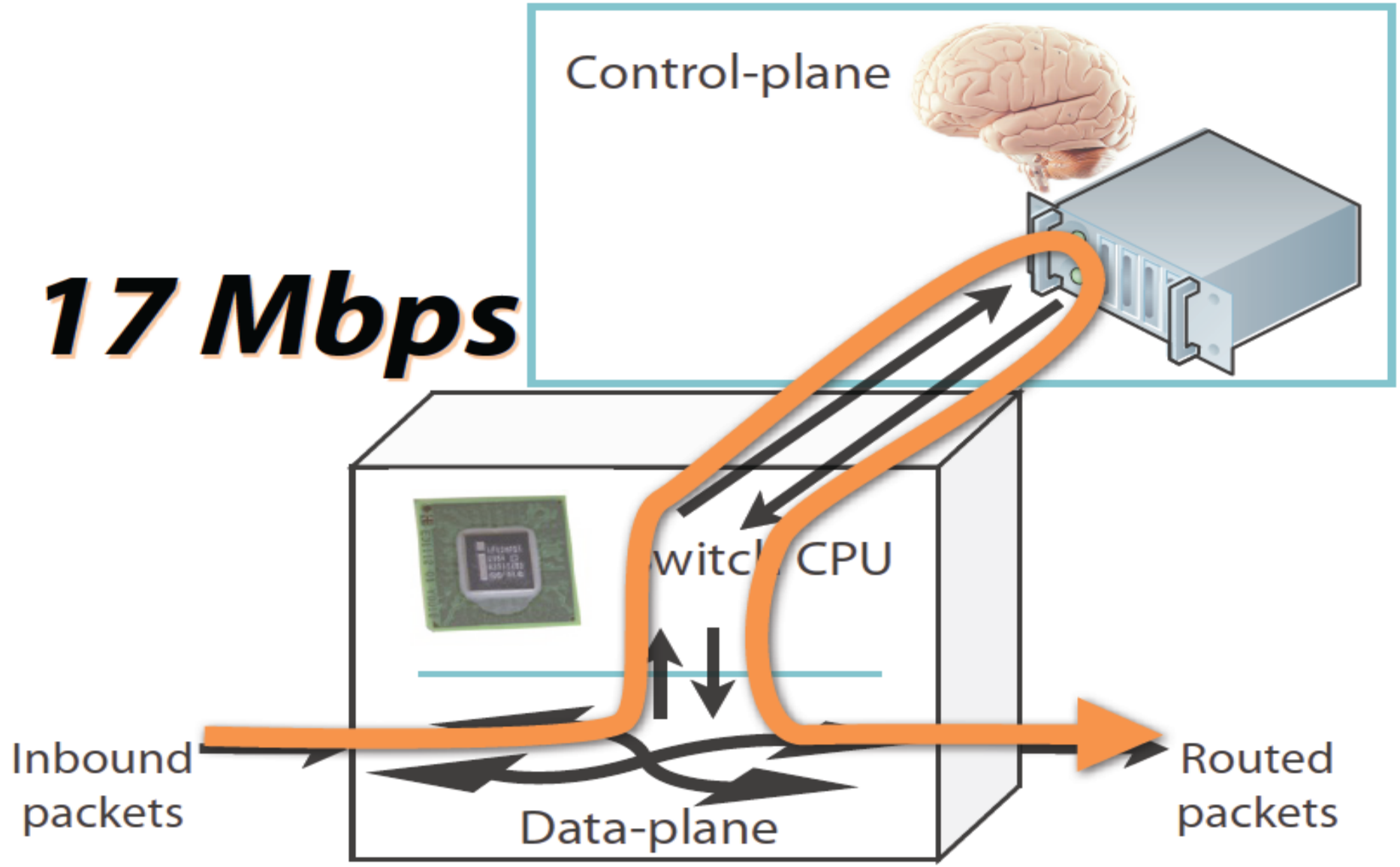


We can expect up to 10K flow arrivals / sec.

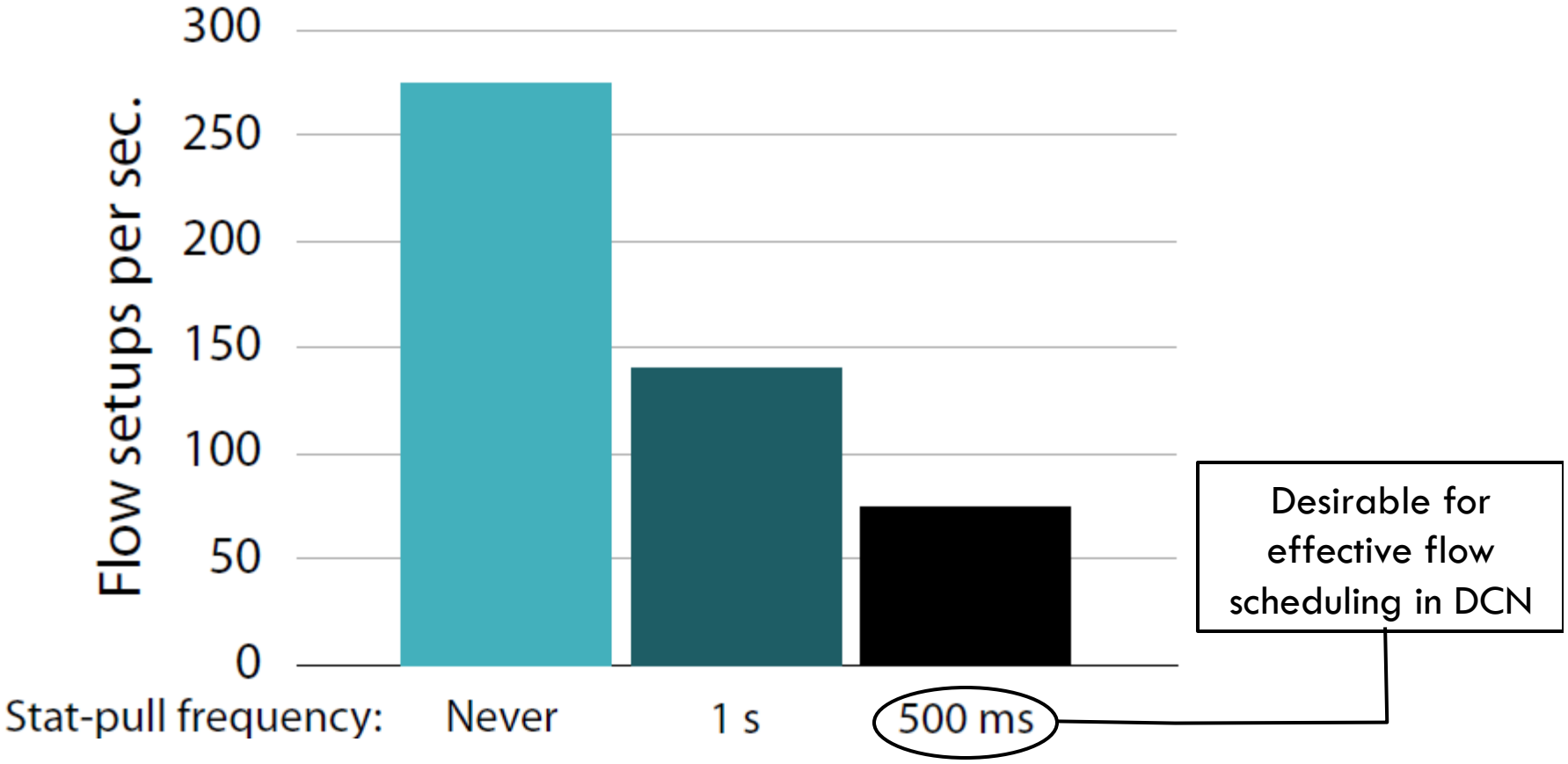
[Benson et al. IMC 2010]

40x difference!

Stat Collection Problem



- Flow setups and stat pulling compete for the same bandwidth



DevoFlow Design Principles

- Keep flows in the data-plane (as much as possible)
- Maintain just enough visibility for effective flow management

DevoFlow Mechanisms

- Control mechanism
  - ASIC clones wildcard rules as match rules for new incoming flows
- Stats gathering mechanism
  - Sampling based
  - Trigger based

Control Mechanism

wildcard rules

src	dst	src port	dst Port
*	129.100.1.5	*	*

exact-match rules

src	dst	src port	dst Port

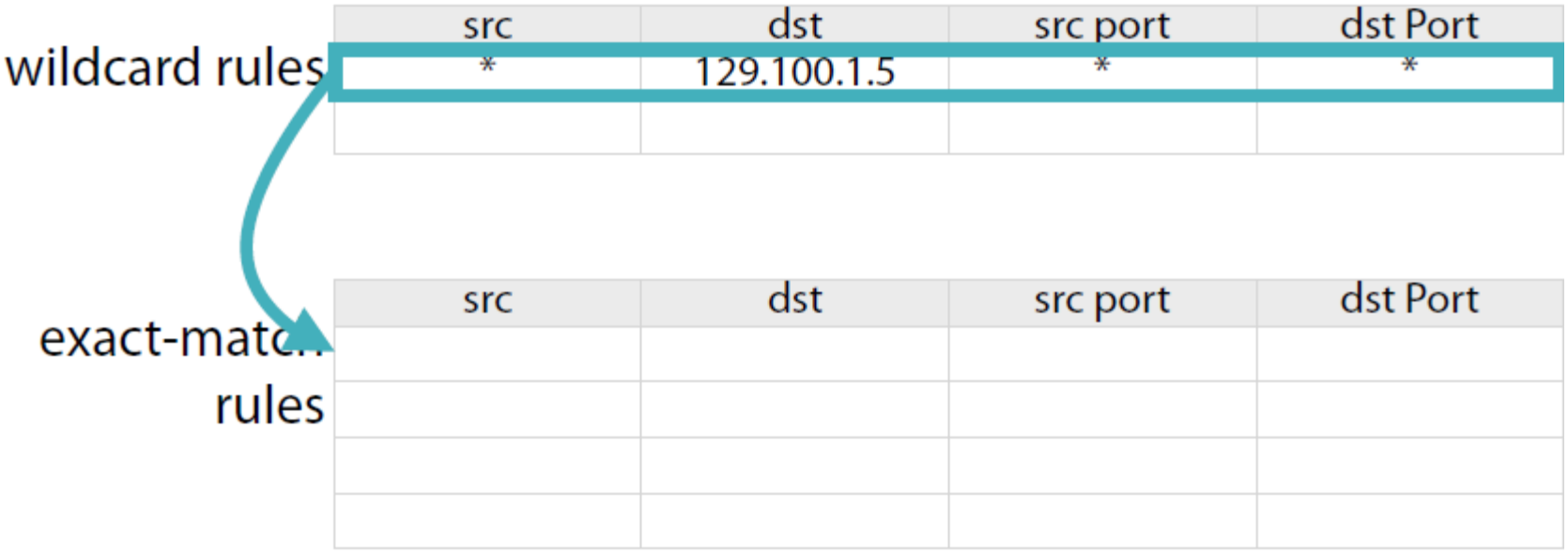
Control Mechanism

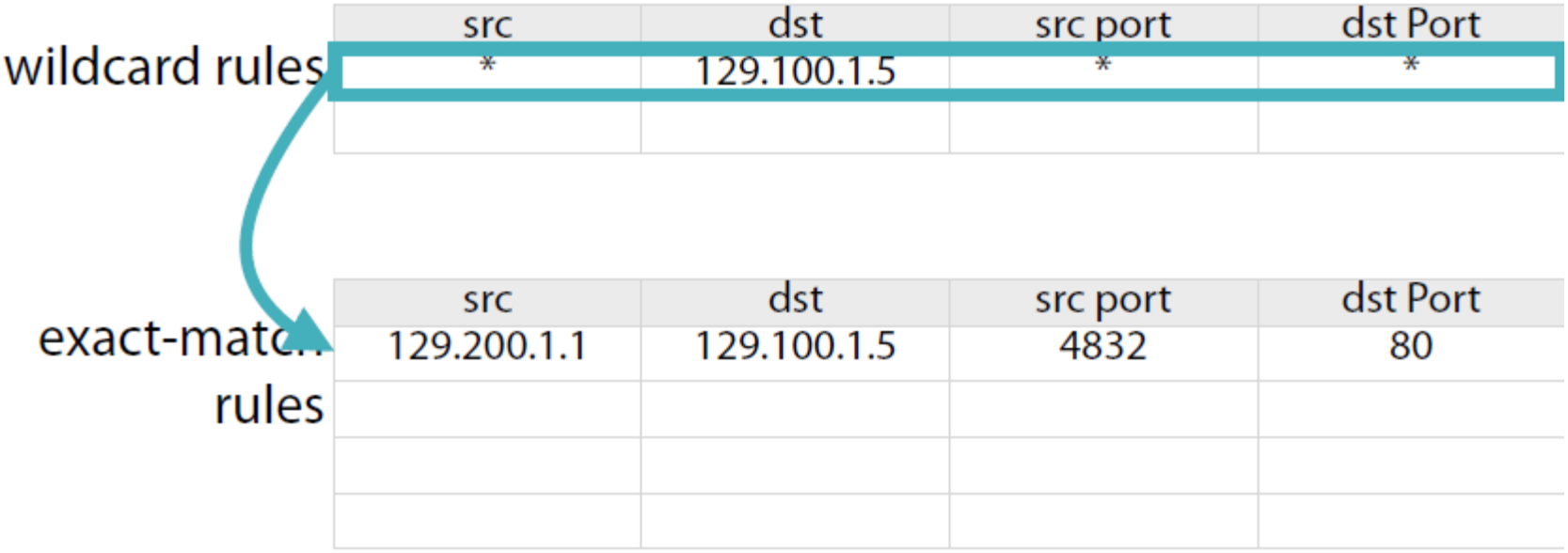
wildcard rules

src	dst	src port	dst Port
*	129.100.1.5	*	*

exact-match rules

src	dst	src port	dst Port


wildcard rules
src
dst
src port
dst Port
*
129.100.1.5
*
*
exact-match
rules
src
dst
src port
dst Port


wildcard rules
src
dst
src port
dst Port
*
129.100.1.5
*
*
exact-match rules
src
dst
src port
dst Port
129.200.1.1
129.100.1.5
4832
80

Stats gathering mechanism

- Sampling
  - Packet header is sent to the controller with 1/1000 probability
- Triggers and reports
  - Can set a threshold per value; when threshold is reached, flow is setup at central controller

Evaluation

- Flow level based network simulator
- Modeling Clos and HyperX Topology
- Load balancing as an example to compare performance

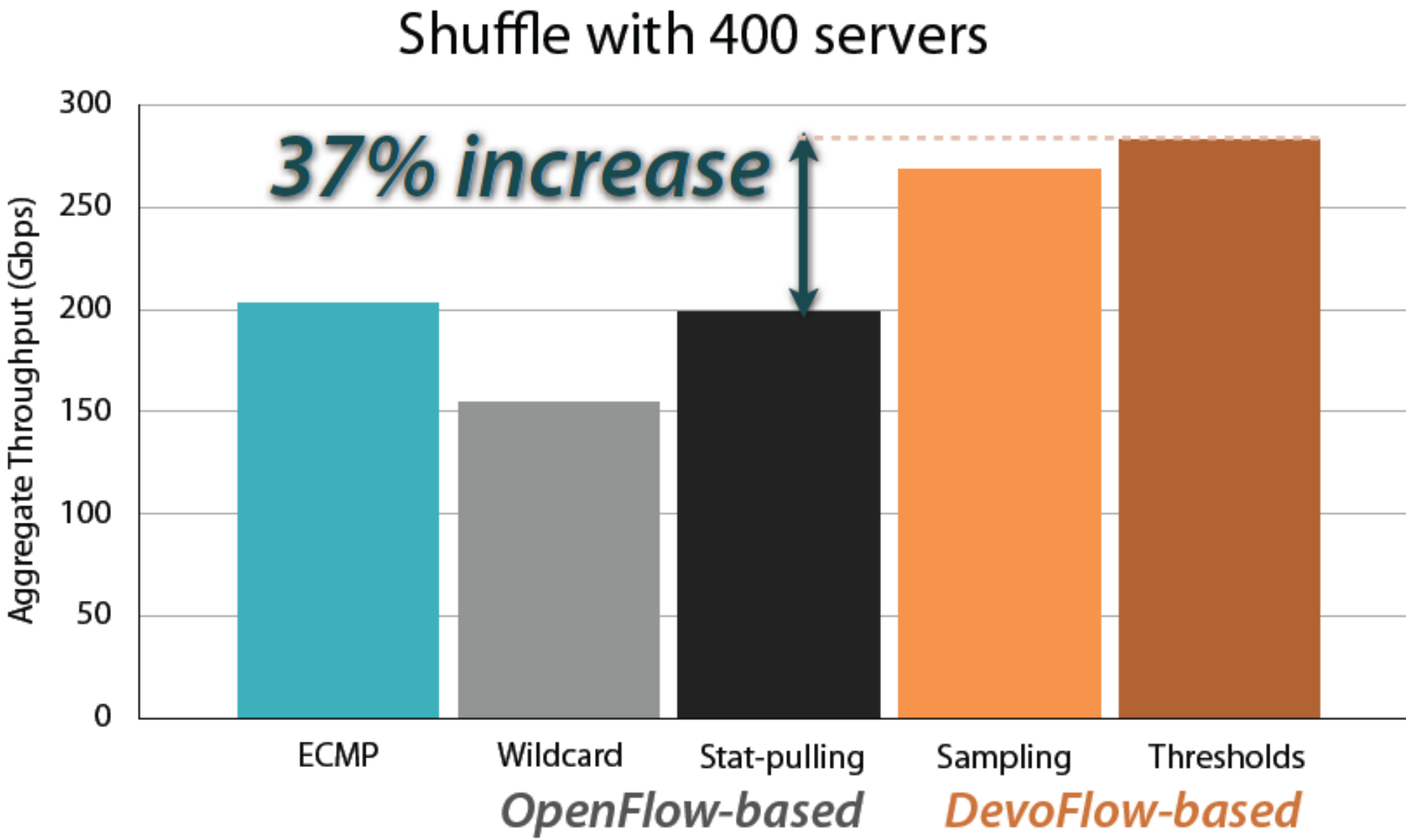
Workloads

- MapReduce Shuffle, 128 MB to all servers, five at a time
- Reverse-engineered MSR workload [Kandula et al. IMC 2009]
  - Based on two distributions: inter-arrival times and bytes per flow

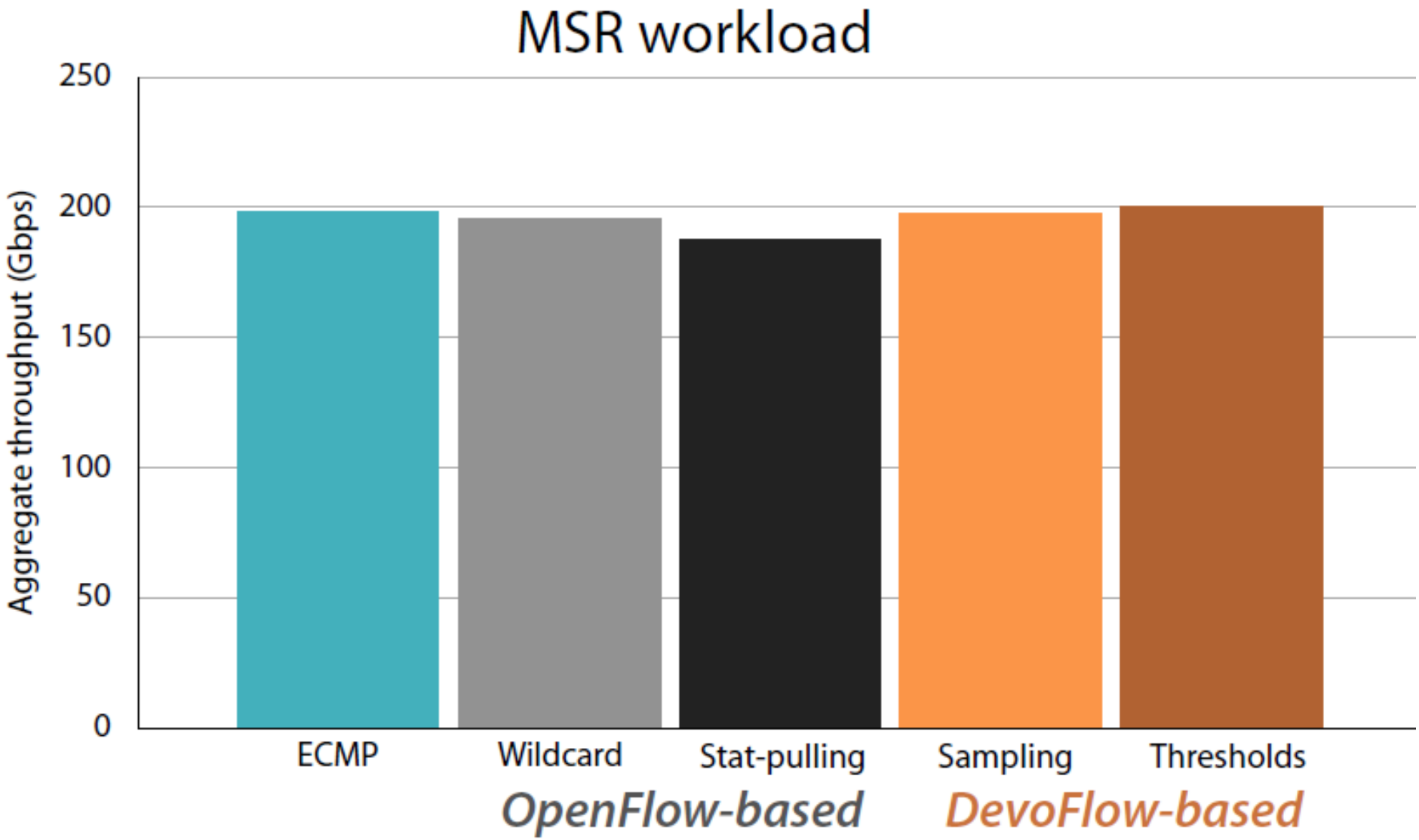
Schedulers

- ECMP
- OpenFlow
  - Course grained using wild cards
  - Fine grained using stat pulling
- DevoFlow
  - Stat via sampling
  - Trigger and reports at specified threshold of bytes

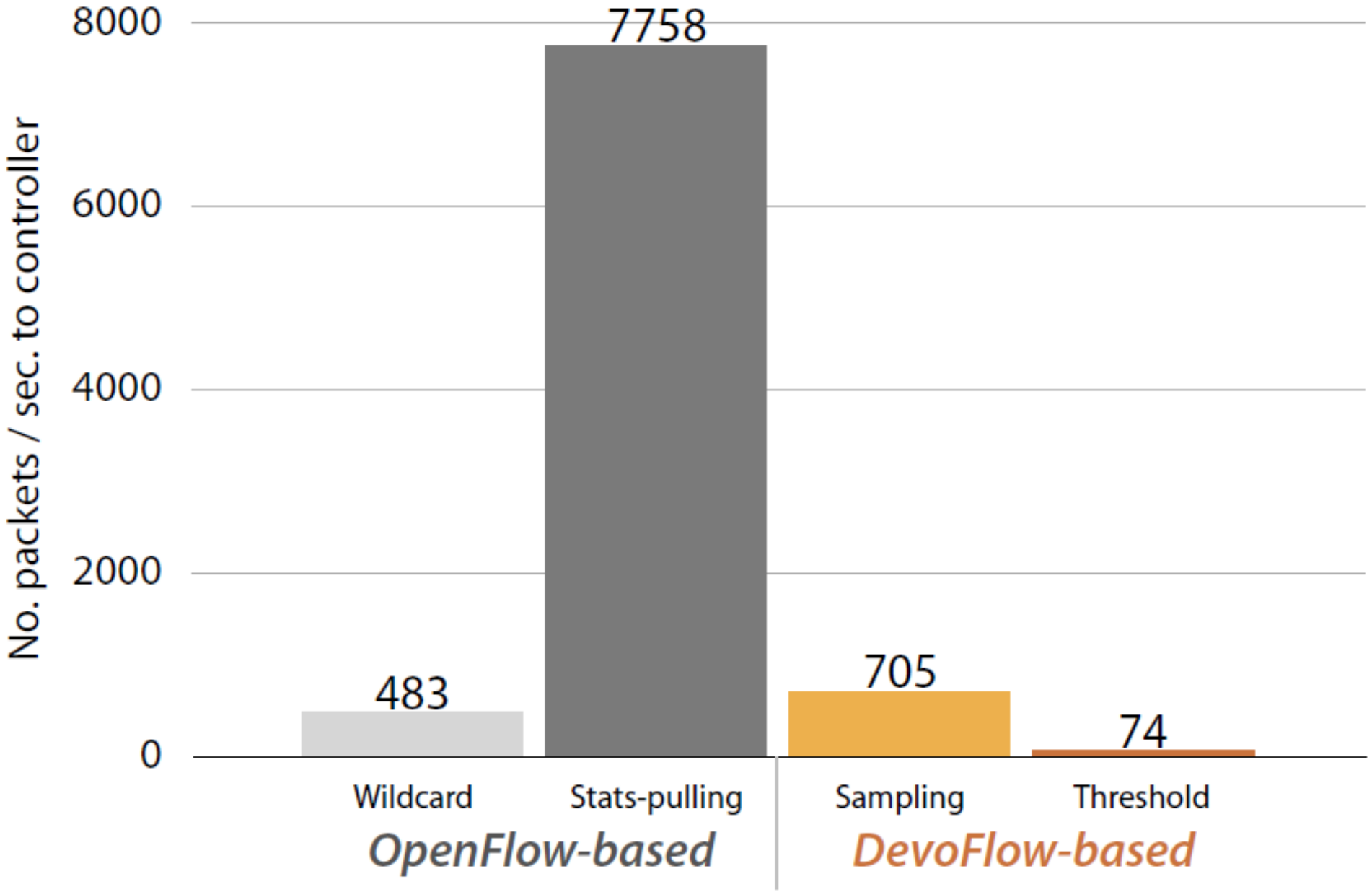
Performance: Clos topology



Performance: Clos topology



Overheads: Control traffic

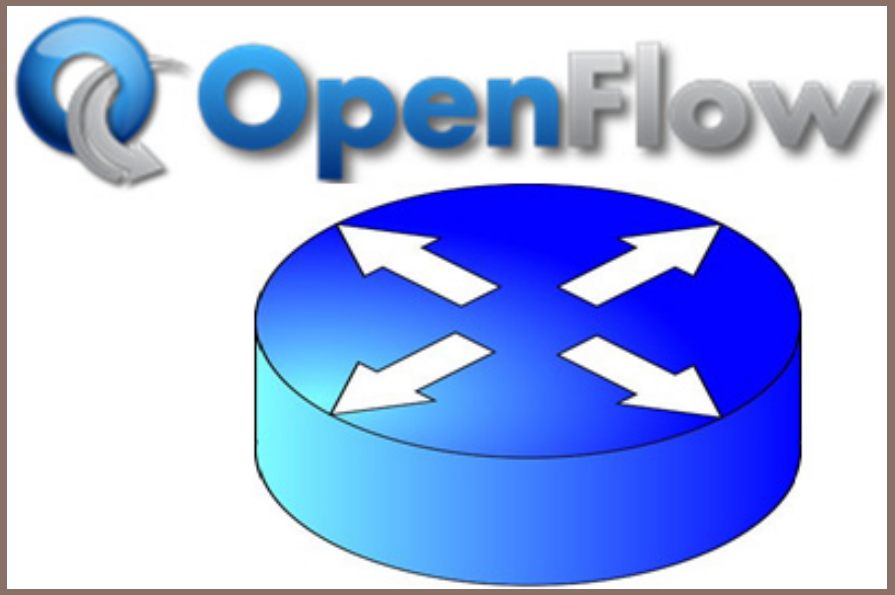


Conclusion

- OpenFlow imposes high overheads on switches
- Proposed DevoFlow to give tools to reduce reliance on the control-plane
- DevoFlow can reduce overheads by 10–50x for data center flow scheduling

SCALABLE FLOW-BASED NETWORKING WITH DIFANE

MINLAN YU, JENNIFER REXFORD, MICHAEL FREEDMAN, JIA WANG



Jonathan M. Chu

CS 538 Presentation

DIFANE

- DIFANE is a scalable network architecture based on using OpenFlow switches
- It's goal is to decrease RTT, increase Throughput, and to scale to large networks
- It distributes rules across multiple switches for load balancing in the network

Keeping packets in the data-plane

- DIFANE is designed so that packets never go to controller
- It distributes policy rules at the TCAM level through switches

Use of wildcard rules

- Instead of microrules, Difane uses wildcard rules which can handle a larger number of packets through fewer rules

- 3 sets of wildcard rules:
- Cache rules
- Authority rules
- Partition rules

Difane flow Management Architecture

1)An incoming packet will arrive at the ingress switch

2) If a rule for handling that packet is already cached into the ingress switch, it will head directly to its correct egress switch.

3) If there is no rule in place at the ingress switch, then the packet will go to an authority switch which will have the rule to forward it to the egress switch

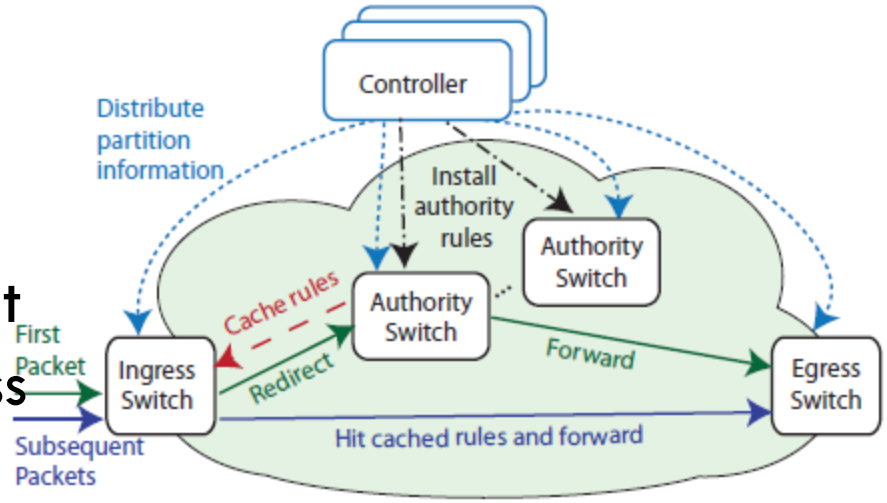


Figure 1: DIFANE flow management architecture. (Dashed lines are control messages. Straight lines are data traffic.)

Difane flow Management Architecture

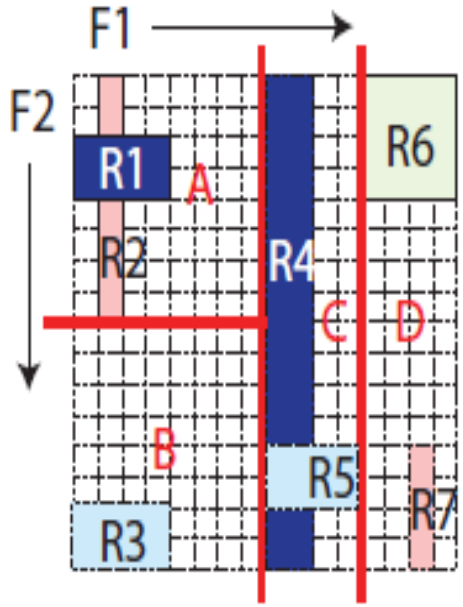
4) After forwarding the packet, the Authority switch will update the rule list in the ingress switch so that all future packets of that type will be correctly routed

5) All the rules are partitioned into the Authority switches beforehand

6) The controller partitions these rules using an algorithm

Rule Partioning and Allocation

- Partition the rule space such that authority switches do not have overlapping rules
- Each rule should only belong to one authority switch
- Uses a decision tree based partition algorithm



(a) Low-level rules and the partitio

Reducing Stretch

- Rules can be duplicated across multiple authority switches to reduce stretch
- Strech is small in testing, average stretch is 1.8 with 3 copies of each rule.

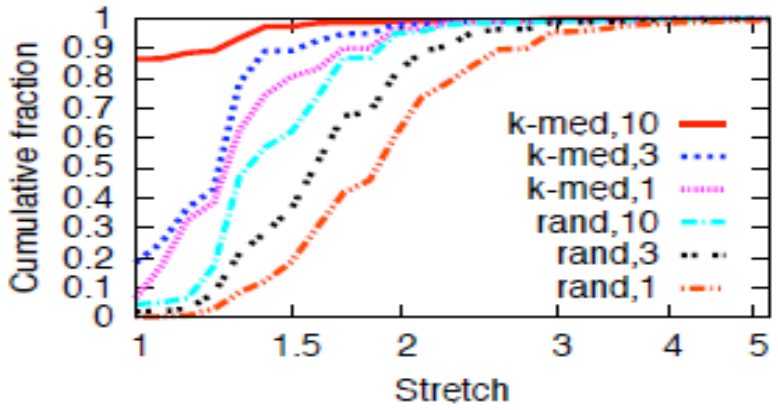


Figure 15: The stretch of campus network. (We place 1,3,10 authority switches for each set of the authority rules using k-median and random algorithms respectively.)

Eliminating Buffering

- A Typical OpenFlow system such as Nox, would buffer cache misses until the Controller could issue a command for how the missed packet should be handled.

- Instead of buffering, DIFANE automatically routes missed packets to Authority switches

Scaling to Large Networks

- Since rules are partitioned to multiple Authority Switches instead of to a central controller, Difane is scalable to large networks
- Few Authority Switches are needed even for large networks
- Wildcard rules are only kept at ingress switches for a limited time if the rule is not hit. Thus ingress switches do not need large memory

Handling change

- Authority switches can be later added by updating the partition rules

- In the event of an Authority switch failure, each ingress switch has backup partition rules to redirect traffic to another authority switch when the primary one fails

Implementation

- Used topologies and access-control rules from a large-scale campus network and three large backbone networks that are operated by tier-1 ISPs

- Utilized a Click-based OpenFlow switch

Comparison to NOX

1. DIFANE has lower round trip times than NOX

2. Difane has a higher throughput than NOX

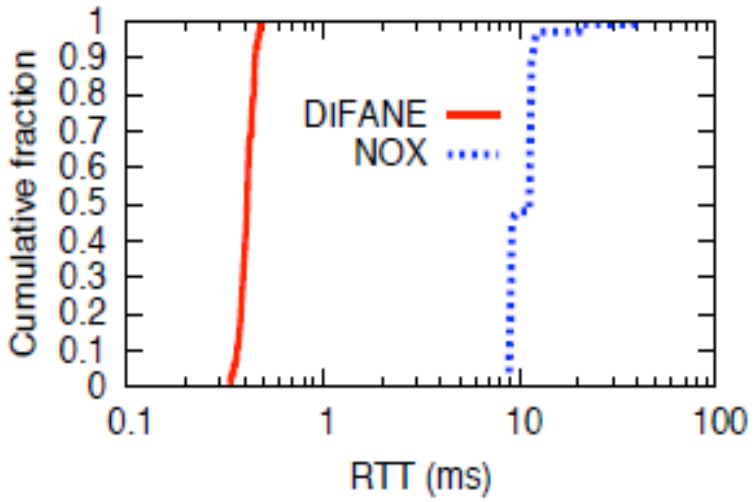


Figure 8: Delay comparison of DIFANE and NOX.

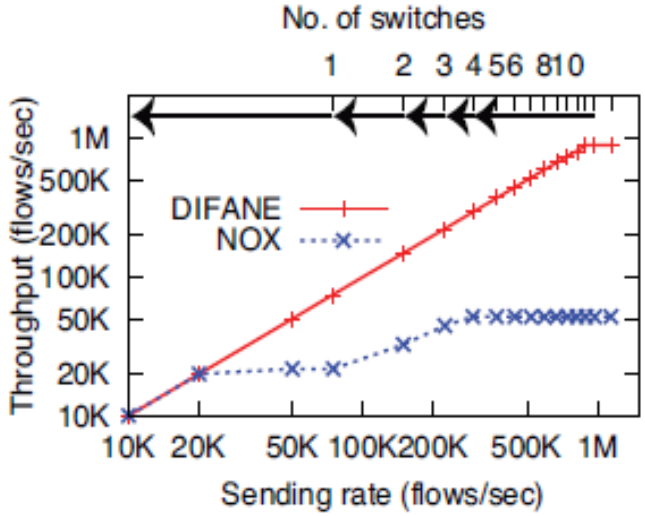


Figure 9: Throughput comparison of DIFANE and NOX.

Rules on Authority Switch

- By increasing the number of Authority Switches, one can reduce the number of TCAM entries per Authority Switch. Thus routers do not need to maintain much memory.

Network	# switches/routers	# Rules
Campus	~1700	30K
VPN	~1500	59K
IPTV	~3000	5M
IP	~2000	8M

Table 1: Network characteristics.

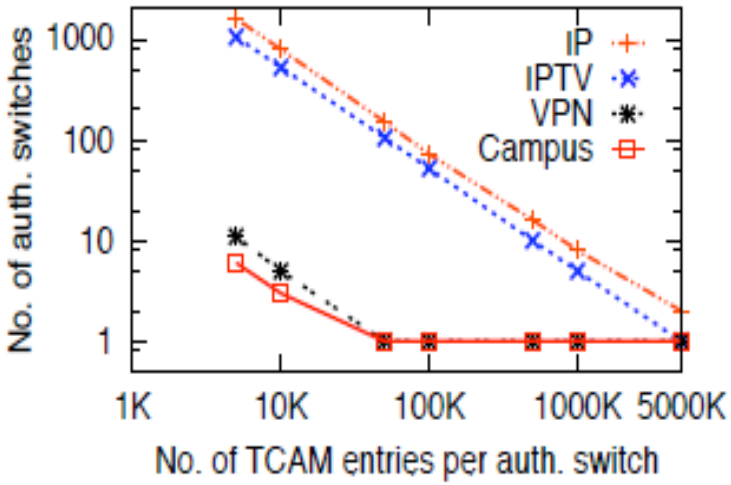


Figure 12: Evaluation on number of authority switches.

Results

- DIFANE is able to successfully distribute rules across multiple switches while lowering RTT, increasing Throughput, and being scalable to large networks
- By keeping all the packets moving in the data-plane, there is little delay time

DIFANE vs DevoFlow

- Both DIFANE and DevoFlow attempt to keep flows in the data-plane
- DIFANE sets up the switches to handle miss packets so they stay in data plane and never have to go to controller

- Devoflow limits the amounts of statistics gathering in the control plane level and reduces the number of flows that need to go to controller

Discussion

- What disadvantages might DIFANE have that a regular OpenFlow system such as NOX wouldn't have?
- In what type of a network would DIFANE work well in? In what type of a network would DIFANE not work well?
- How well do you think, DevoFlow can support security policies?
  - Hint: Is it possible to assert same granularity of control as OpenFlow/NOX provides?