P4Tune: Enabling Programmability in a non-Programmable Network

Elie Kfoury, Jorge Crichigno
University of South Carolina
http://ce.sc.edu/cyberinfra
ekfoury@email.sc.edu, jcrichigno@cec.sc.edu

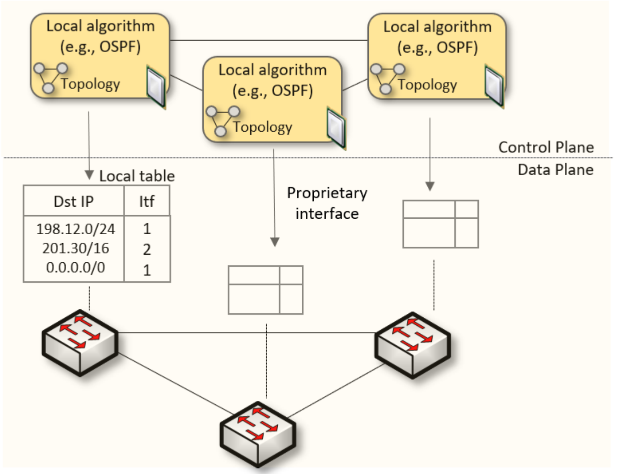
CI Engineering Lunch and Learn – Online
April 15, 2022

Agenda

- Non-programmable Networks
- Background on SDN and P4 programmable switches
- P4 switches adoption challenges
- P4Tune framework
- Use case 1: Dynamic buffer sizing
- Use case 2: Size-aware flow separation
- Use case 3: SYN flood mitigation
- Use case 4: DNS amplification
- Discussions
- Conclusion

Non-programmable Networks

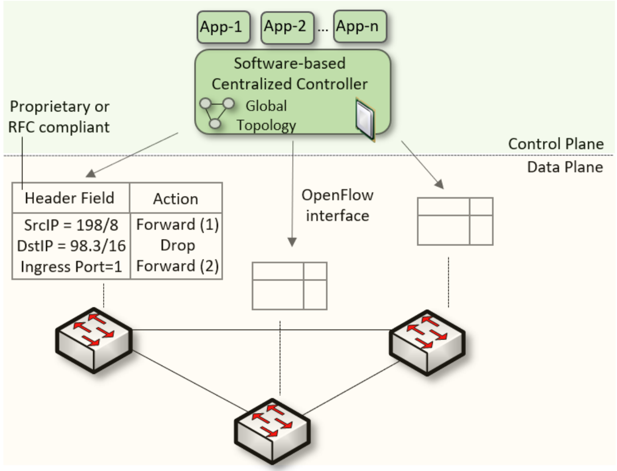
- Since the explosive growth of the Internet in the 1990s, the networking industry has been dominated by closed and proprietary hardware and software
- The interface between control and data planes has been historically proprietary
  - Vendor dependence: slow product cycles of vendor equipment, no innovation from network owners
  - A router is a monolithic unit built and internally accessed by the manufacturer only



Dst IP	Itf
198.12.0/24	1
201.30/16	2
0.0.0.0/0	1

SDN

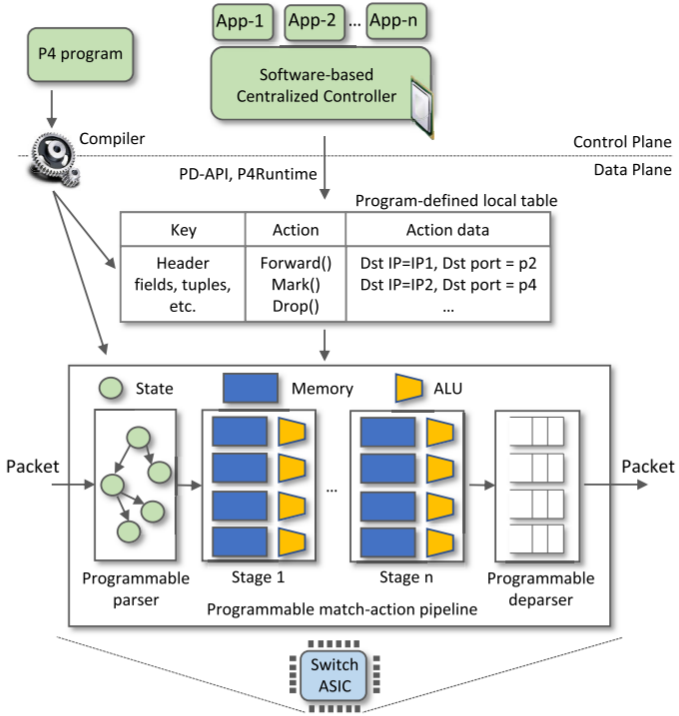
- Protocol ossification has been challenged first by SDN
- SDN explicitly separates the control and data planes, and implements the control plane intelligence as a software outside the switches
- The function of populating the forwarding table is now performed by the controller
- SDN is limited to the OpenFlow specifications



Header Field	Action
SrcIP = 198/8	Forward (1)
DstIP = 98.3/16	Drop
Ingress Port=1	Forward (2)

P4 Programmable Switches

- P4[1] programmable switches permit a programmer to program the data plane
  - ➢ Define and parse new protocols
  - ➢ Customize packet processing functions
  - ➢ Measure events occurring in the data plane with high precision
  - ➢ Offload applications to the data plane



Key	Action	Action data

Header fields, tuples, etc.	Forward()
	Mark()
	Drop()
		Dst IP=IP1, Dst port = p2
		Dst IP=IP2, Dst port = p4
		...

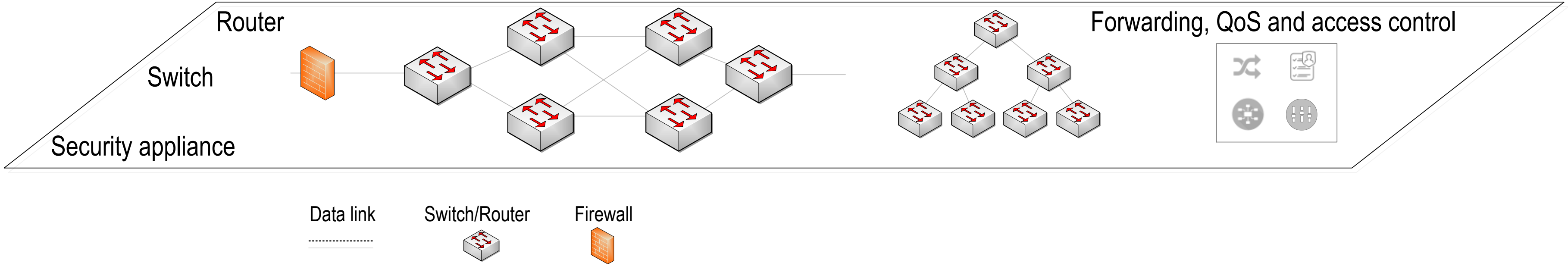
[1] P4 stands for stands for Programming Protocol-independent Packet Processors

P4 Switches Deployment Challenges

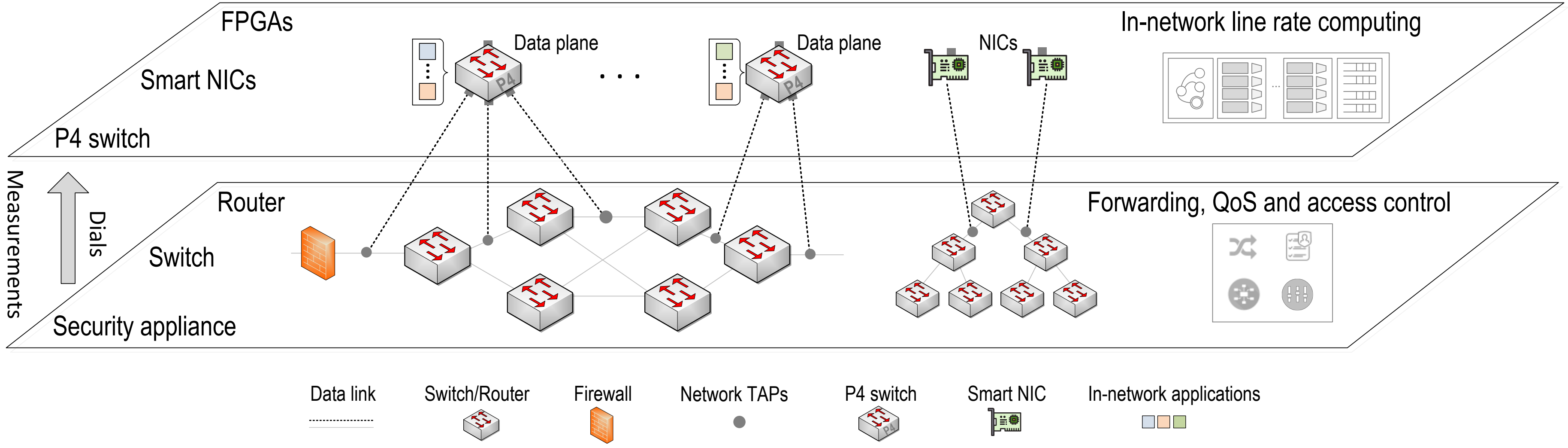
- Data plane programmability knowledge by operators
  - ➢ Operators only configure legacy devices (e.g., modify routing configuration, updating ACL)
  - ➢ Programming P4 targets is complex[1]
- Cost of replacing the existing infrastructure
  - ➢ Significant costs, time, and efforts spent in building the network and the existing equipment
  - ➢ Replacing these devices with P4 switches would incur significant costs
- Vendor support
  - ➢ The support in legacy devices is readily available
  - ➢ P4 switches are whiteboxes, with little to no support from vendors
- Network disruption
  - ➢ P4 programs might be potential sources of packet-processing error
  - ➢ Bugs can lead to network disruption, affecting the availability of the services

[1] The switch.p4 program, which contains the standard switch capabilities, has more than 10^{30} control paths

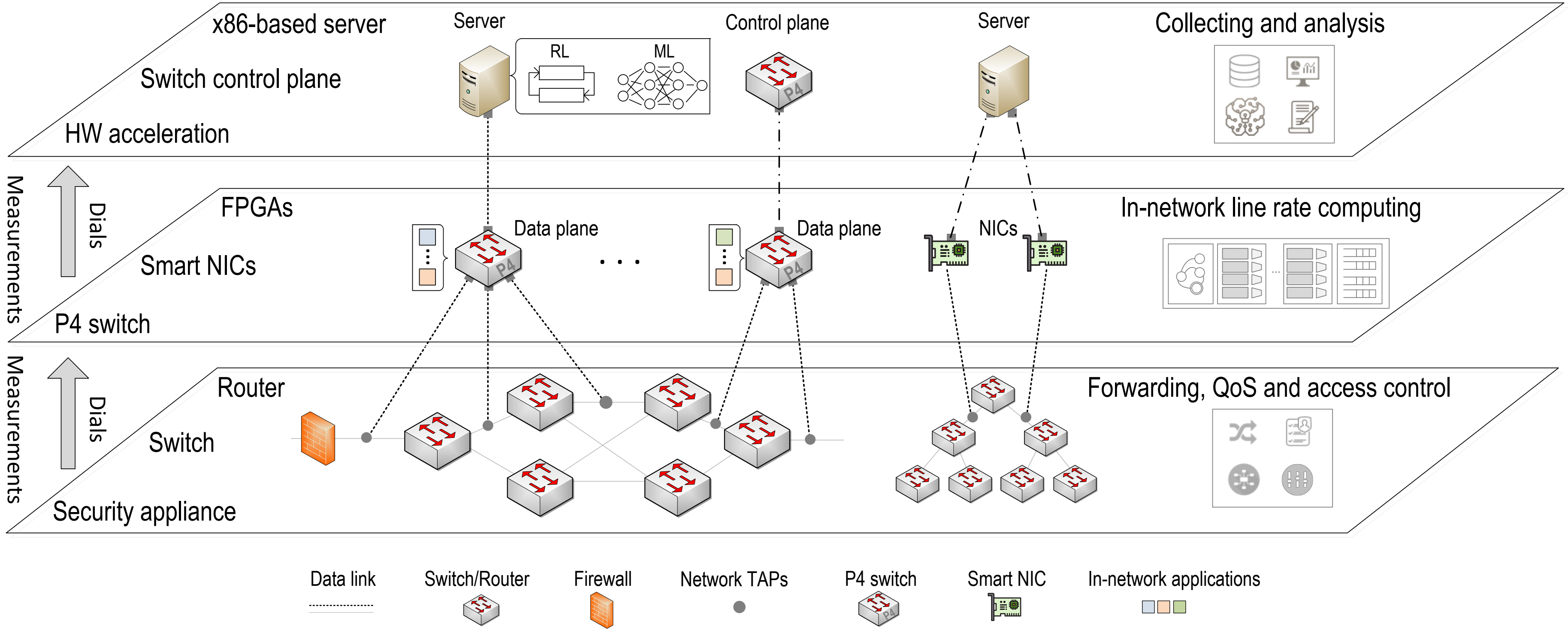
P4Tune Overview



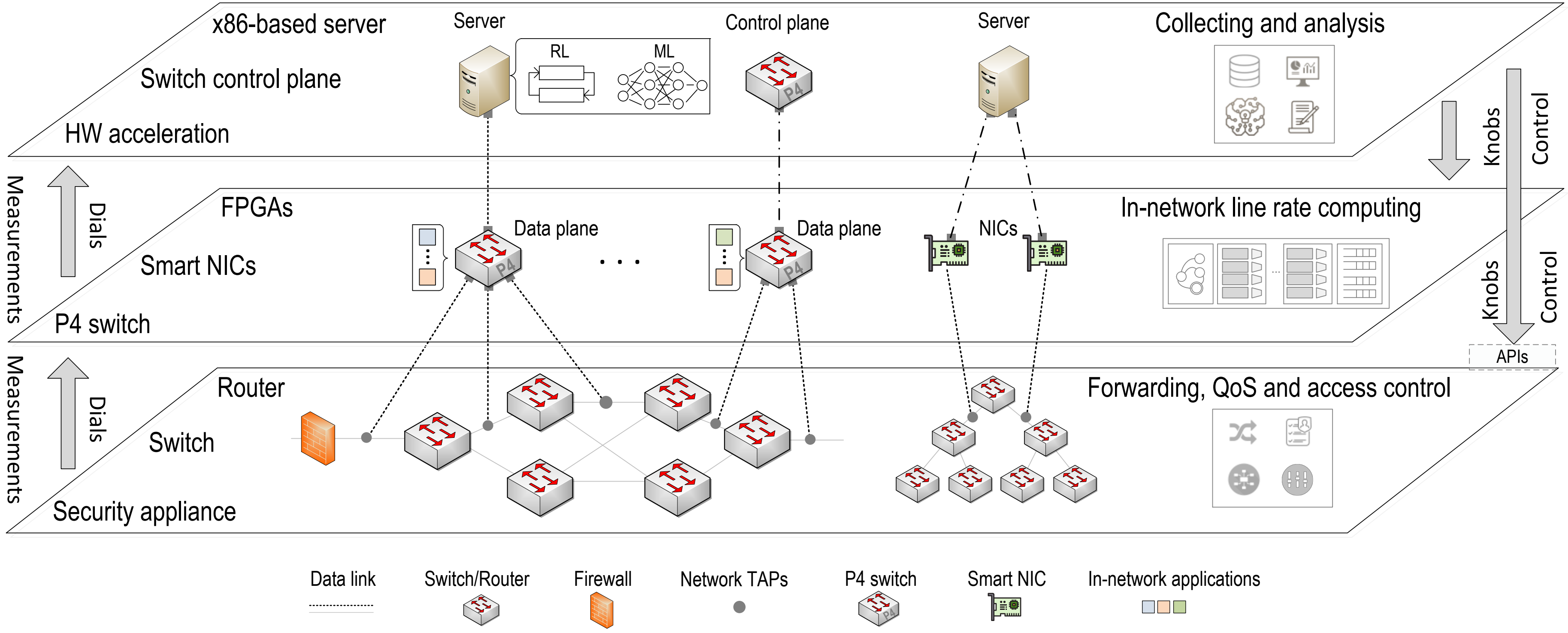
P4Tune Overview



P4Tune Overview



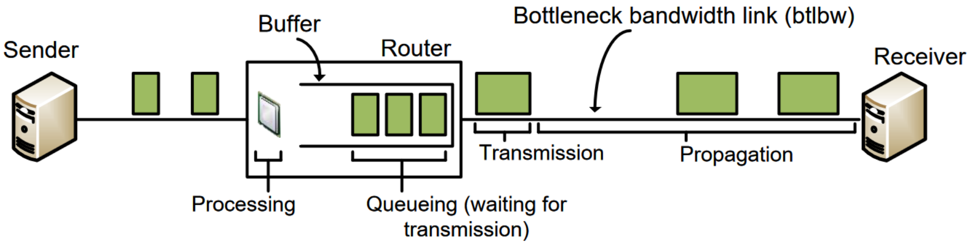
P4Tune Overview



Use Case 1: Dynamic Buffer Sizing

Buffer Sizing Problem

- Routers and switches have a memory referred to as packet buffer
- The size of the buffer impacts the network performance
  - Large buffers -> excessive delays, bufferbloat
  - Small buffers -> packet drops, potential low link utilization


Buffer
Bottleneck bandwidth link (btlbw)
Sender
Router
Receiver
Transmission
Propagation
Processing
Queueing (waiting for transmission)

Buffer Sizing Rules

- General rule-of-thumb: bandwidth-delay product (older rule)
  - Buffer = $C * RTT$
  - C is the capacity of the link and RTT is the average round-trip time (RTT)
- Stanford rule
  - Buffer = $\frac{C * RTT}{\sqrt{N}}$
  - N is the number of long (persistent over time) flows traversing the link

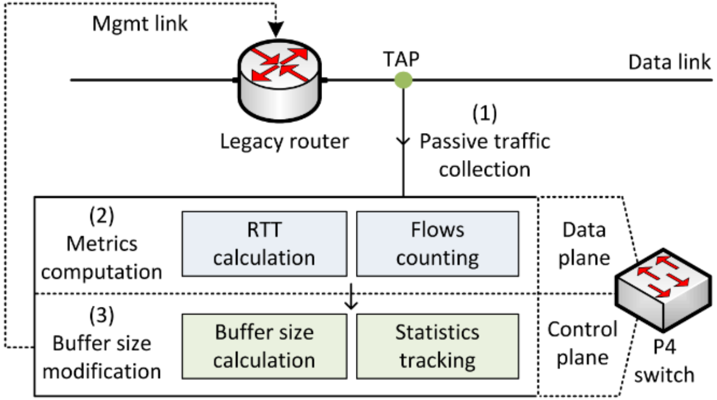
Stanford Rule Applicability

- Setting the router's buffer size to BDP/√N would require determining the current average RTT and the number of flows
- A general-purpose CPU cannot cope with high traffic rates
- Sampling techniques (e.g., NetFlow) are not accurate enough[1]

[1] Spang, Bruce, and Nick McKeown. "On estimating the number of flows." *Stanford Workshop on Buffer Sizing*. 2019.

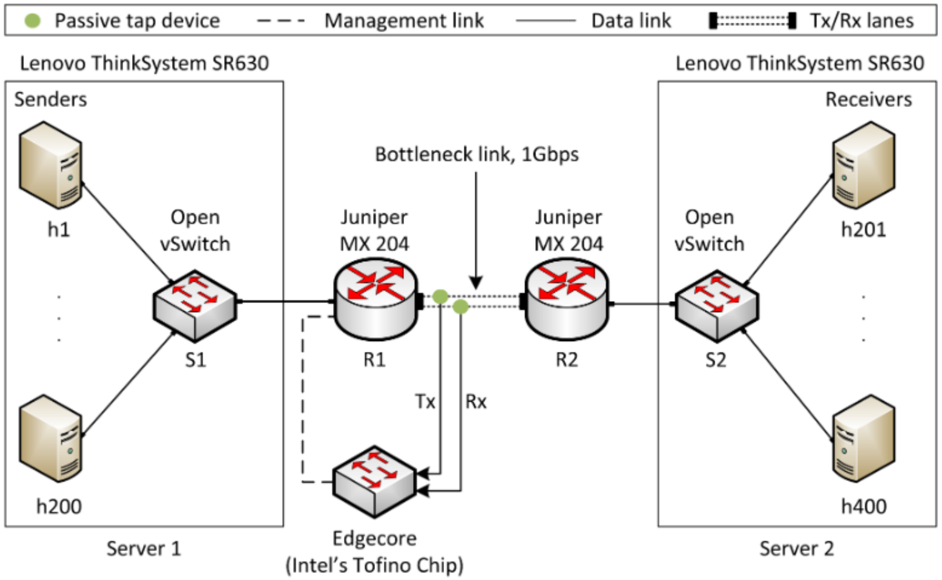
Proposed System

- Dynamically modify the buffer size of routers based on measurements collected on programmable switches
  1. Copy of the traffic is forwarded to a programmable switch by passively tapping router's ports
  2. The programmable switch identifies, tracks, and computes the RTT of long flows
  3. The programmable switch modifies the legacy router's buffer size



Implementation and Evaluation

- Different congestion control algorithms[1]
- iPerf3
- Default buffer size of the router is 200ms[2]



[1] Mishra et al. "The great Internet TCP congestion control census," ACM on Measurement and Analysis of Computing Systems, 2019

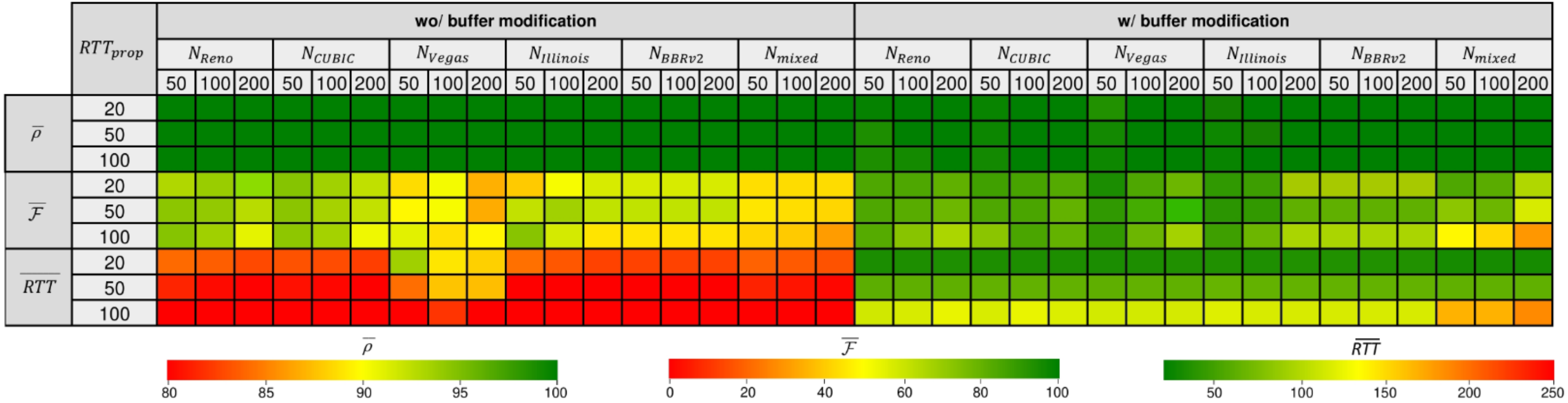
[2] N. McKeown et al. "Sizing router buffers (redux)," ACM SIGCOMM Computer Communication Review, vol. 49, no. 5

Implementation and Evaluation

- Two scenarios are considered:
  1. Default buffer size on the router, without any dynamic modification
  2. P4 switch measures and modifies the buffer size of the router

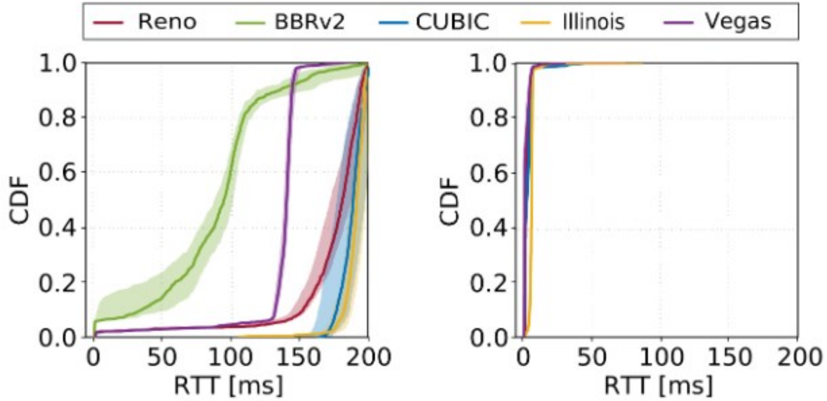
Results

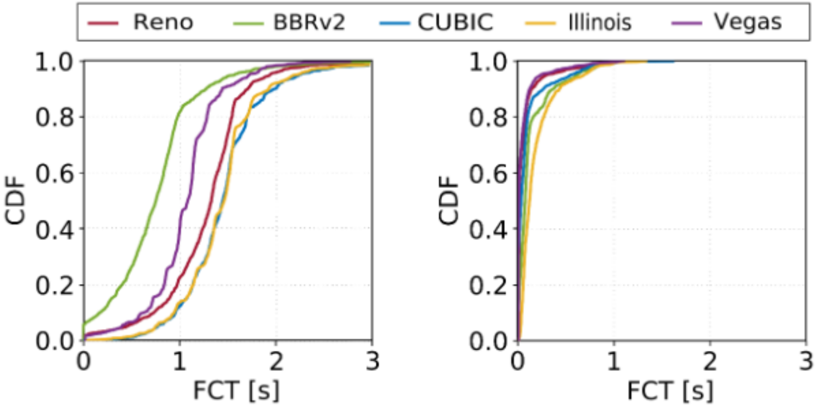
- Multiple long flows, CCAs, and propagation delays
- Average link utilization $(\overline{\rho})$
- Average fairness index $(\overline{\mathcal{F}})$
- Average RTT $(\overline{RTT})$



Results

- Performance of short flows sharing the bottleneck with long flows
- 1000 short flows are arriving according to a Poisson process
- Flow size distribution resembles a web search workload (10KB to 1MB)
- Background traffic: 200 long flows, propagation delay = 50ms

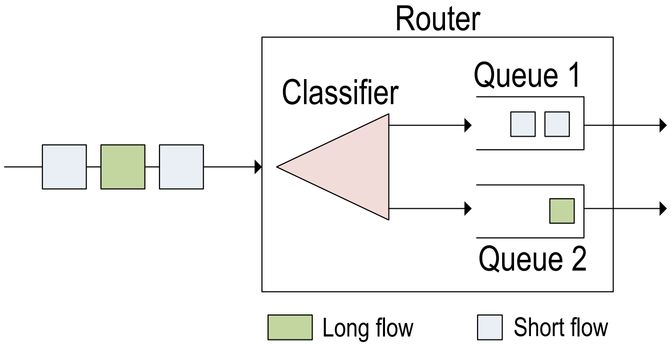




Use Case 2: Traffic Separation based on Flow Size

Size-Aware Flow Separation

- The FCT of short flows sharing a router queue with long flows is significantly impacted when the network is busy
- A possible solution to prevent the increase of FCT is to separate short flows from long flows



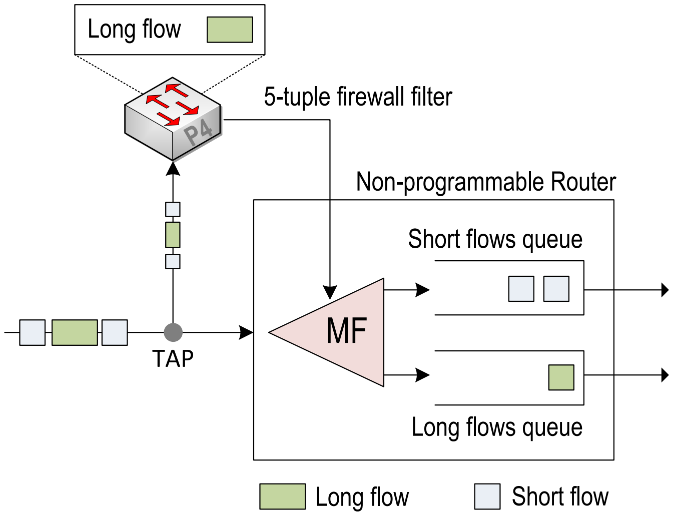
Classification in Legacy Devices

- Typical classifiers available in commercial routers:
  - Behavior aggregate (BA): Inspect the fixed-length fields in the packet header (e.g., DSCP)
  - Multifield classifier (MF): examines multiple fields in the packet (e.g., source/destination addresses/port, TCP flags, protocol, packet length) based on firewall filter rules
- Traffic rarely uses DSCP fields[1]
- Multifield classifier are used with hardcoded rules set by the operators

[1] Roddav et al. "On the Usage of DSCP and ECN Codepoints in Internet Backbone Traffic Traces for IPv4 and IPv6." *ISNCC 2019*

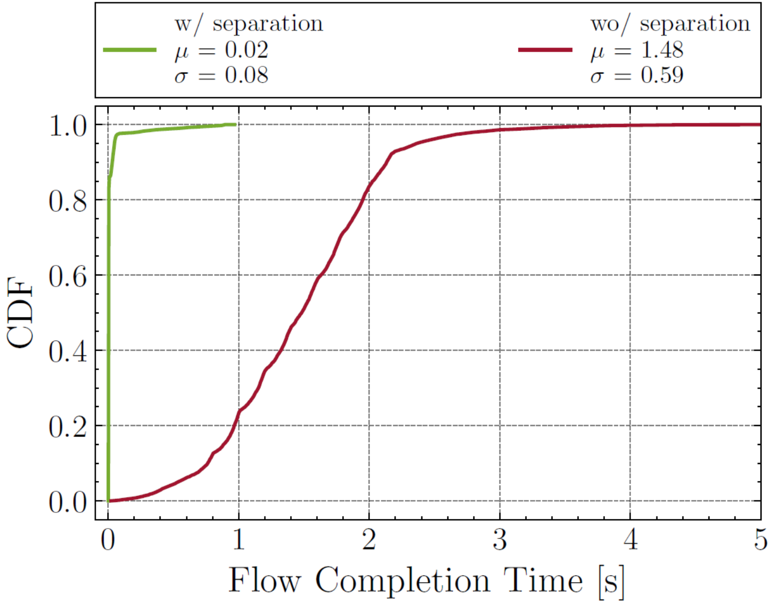
P4-Assisted Flow Classification

- P4 can identify large flows at line rate (e.g., count-min sketch to track packet counts)
- The 5-tuple of the large flows are created added as a firewall filter
- Flows in the firewall filter are assigned to a separate queue (Long flows queue)



Results

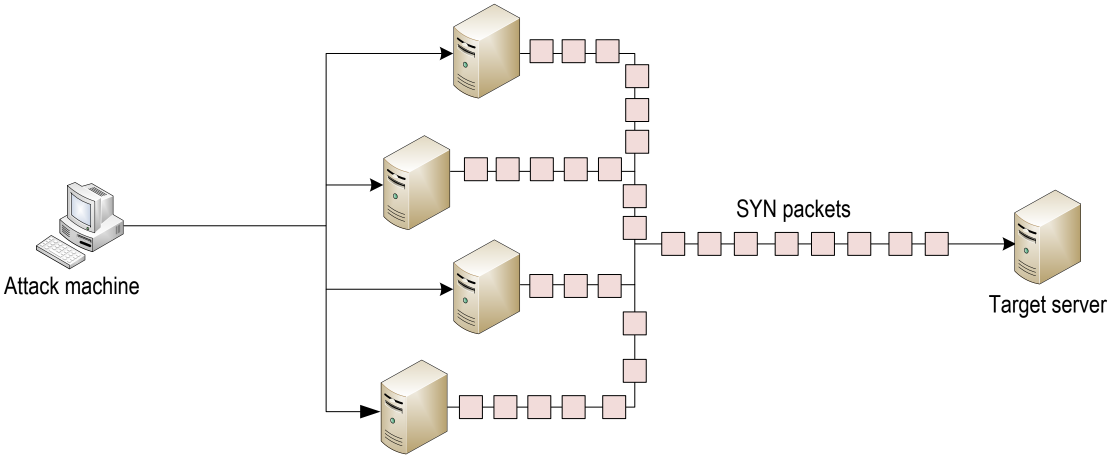
- Performance of short flows sharing the bottleneck with long flows
- 10,000 short flows are arriving according to a Poisson process
- Flow size distribution resembles a web search workload (10KB to 1MB)
- Background traffic: 10 long flows, random starting time over the test duration



Use Case 3: SYN Flood Detection and Mitigation

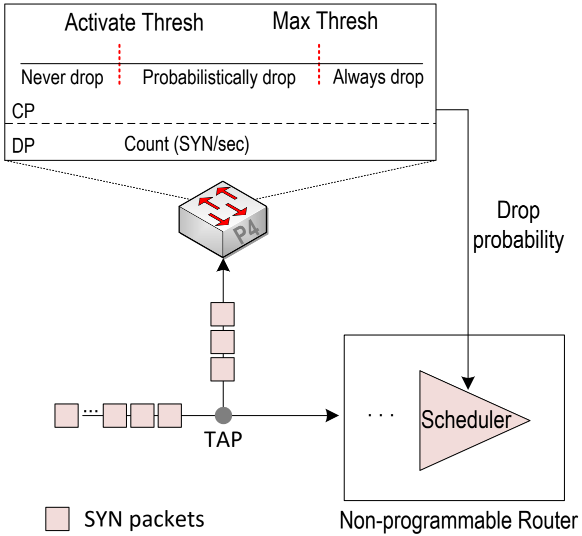
SYN Flood Attack

- Massive amount of TCP SYN requests with spoofed IP addresses are sent to the server
- These connections consume the server's resources, making it unresponsive to legitimate traffic



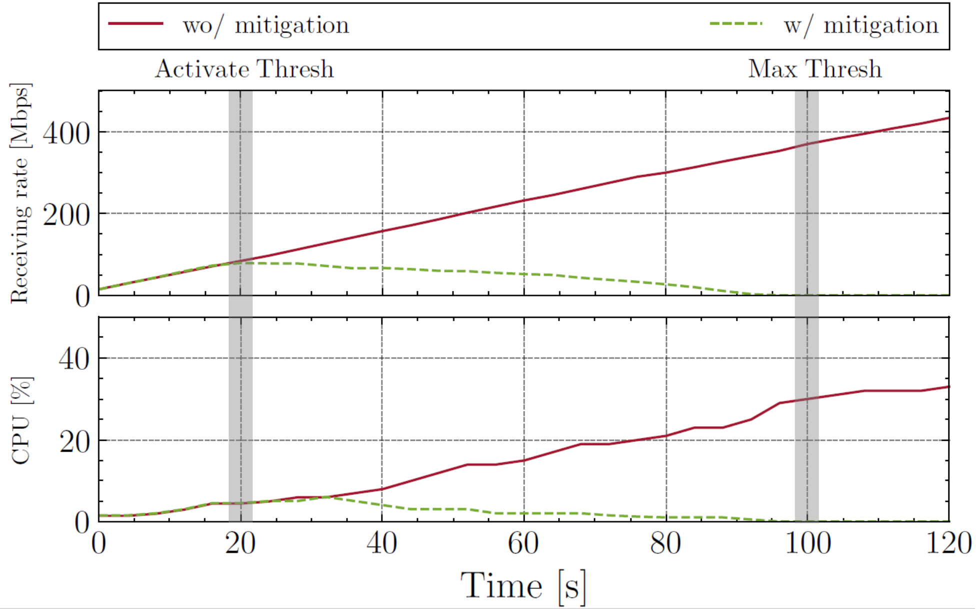
Detecting SYN Flood with P4

- Count the number of SYN packets per second in the programmable data plane
- Implement the Random Early Discard (RED) method
- Construct a rule that makes the router drops with a probability



Results

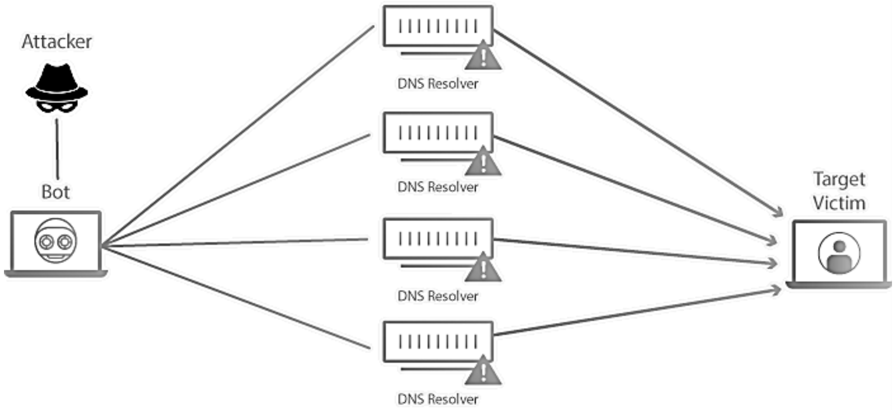
- SYN flood synthetically generated
- The attack rate increases every 2 seconds
- Rate measured at the receiver side (victim)
- SYN flood traffic was successfully mitigated



Use Case 4: DNS Amplification Detection and Mitigation

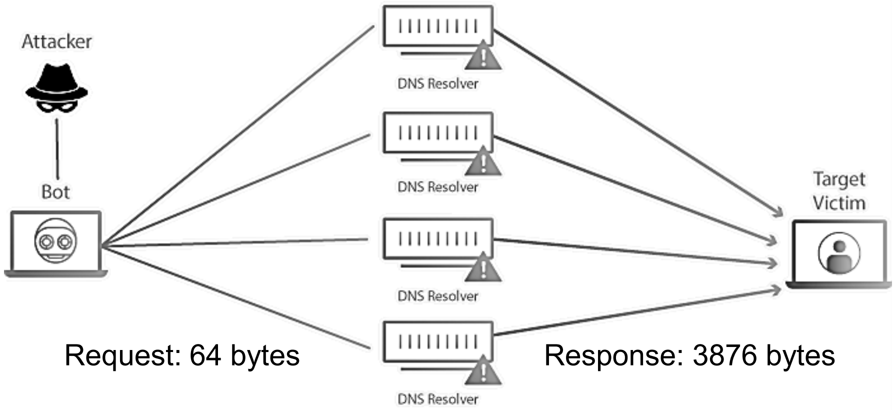
DNS Amplification

- An attack where a massive amount of DNS response packets is sent to a victim’s server
- Attacker sends requests with “ANY” keyword to gather as much zone information as possible to maximize the amplification effect



DNS Amplification

- An attack where a massive amount of DNS response packets is sent to a victim’s server
- Attacker sends requests with “ANY” keyword to gather as much zone information as possible to maximize the amplification effect

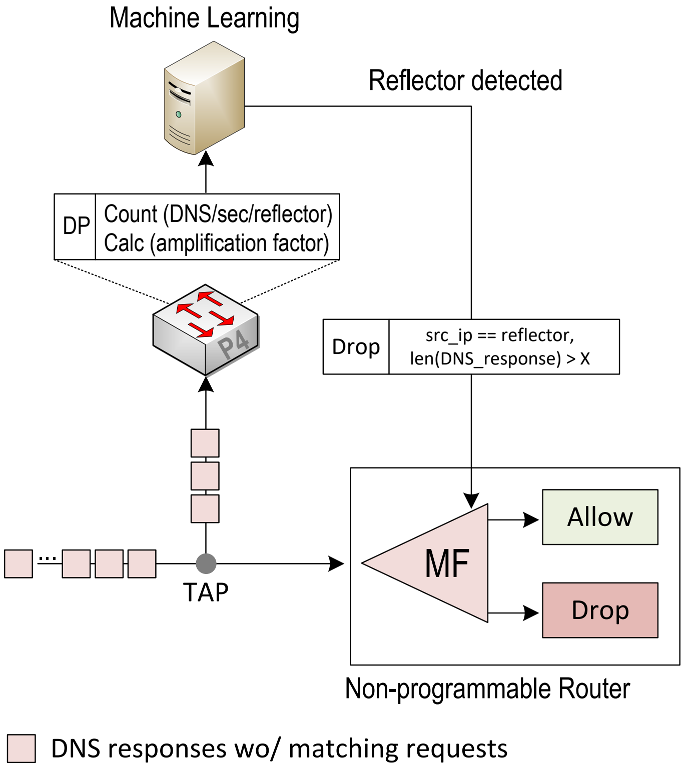


Request: 64 bytes

Response: 3876 bytes

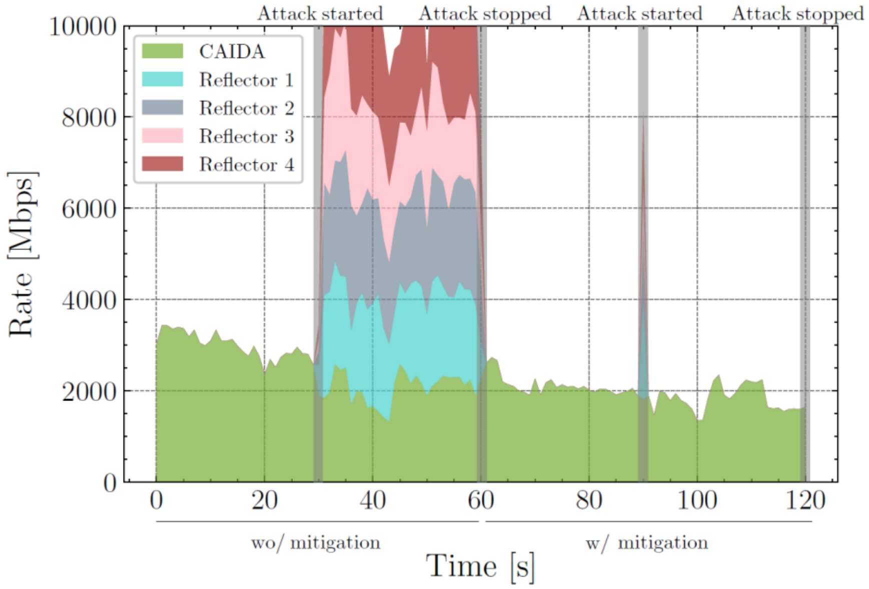
Detecting DNS Amplification with P4

- Count the number of DNS responses received without a DNS request/s/reflector
- Calculate the amplification factor
- Use machine learning to identify thresholds used for attack detection
- Install a rule that matches on the reflector IP and the DNS response packet length
- Allow/drop packet



Detecting DNS Amplification with P4

- CAIDA traffic replayed
- > 10Gbps DNS amplification attack generated
- Attack was mitigated in < 1s



Discussions

- P4Tune is cost-efficient as TAPs and programmable data planes are relatively cheap
- While P4Tune is not applying the configuration rules at line rate, the P4 switches are still performing packet processing at line rate
- P4Tune can be used in other applications including:
  - ➢ Traffic rerouting, load balancing
  - ➢ Traffic steering
  - ➢ Fine-grained measurements and telemetry
  - ➢ etc.
- P4Tune does not support applications that send feedback to the clients (e.g., HPCC)[1]

[1] Li, Yuliang, et al. "HPCC: High precision congestion control." Proceedings of the ACM Special Interest Group on Data Communication. 2019. 44-58.

Conclusion

- P4Tune, a cost-efficient architecture that uses passive programmable data planes to run custom packet processing on the traffic traversing the legacy network
- Configuration rules are constructed and pushed to the legacy devices
- The architecture creates a closed control loop
- Four use cases were implemented, namely, dynamic buffer sizing, flow separation, SYN flood mitigation, DNS amplification mitigation
- For future work, we plan to implement more applications using the framework and possibly test them in a production network

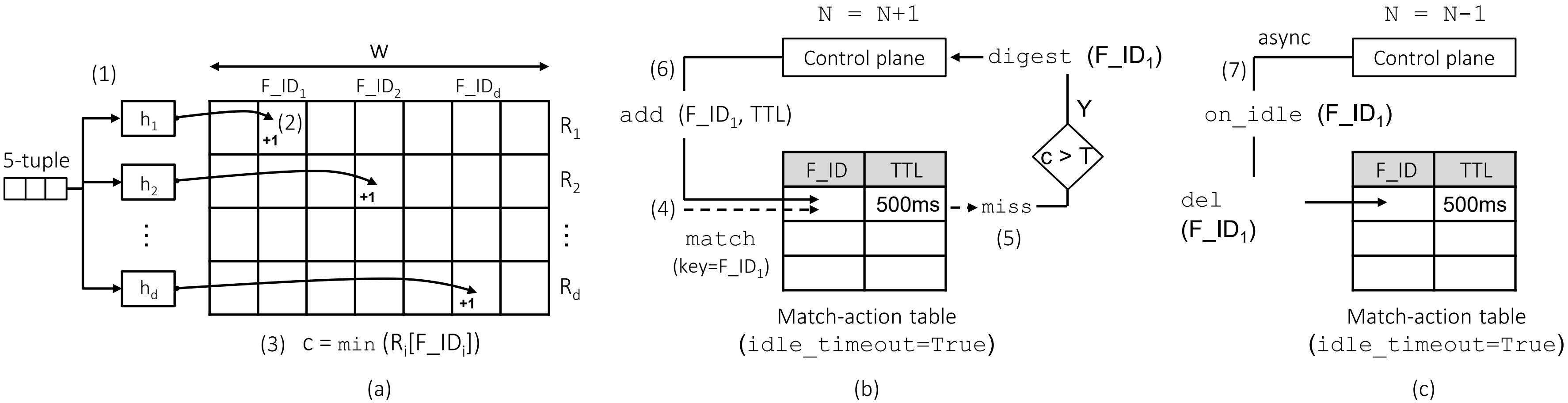
Acknowledgement

- Thanks to the National Science Foundation (NSF)

- This work was supported by NSF, Office of Advanced Cyberinfrastructure (OAC), award 2118311

UNIVERSITY OF
SOUTH CAROLINA

Identifying Long Flows in P4


(1)
w
F_ID$_1$
F_ID$_2$
F_ID$_d$
5-tuple
h$_1$
h$_2$
h$_d$
(2)
+1
+1
+1
R$_1$
R$_2$
R$_d$
(3) c = min (R$_i$[F_ID$_i$])
(a)
N = N+1
Control plane
digest (F_ID$_1$)
(6)
add (F_ID$_1$, TTL)
Y
c > T
F_ID
TTL
500ms
(4)
miss
match
(key=F_ID$_1$)
(5)
Match-action table
(idle_timeout=True)
(b)
N = N-1
async
Control plane
(7)
on_idle (F_ID$_1$)
del
(F_ID$_1$)
F_ID
TTL
500ms
Match-action table
(idle_timeout=True)
(c)