vPROM: vSwitch Enhanced Programmable Measurement In SDN

An Wang*, Yang Guo†, Songqing Chen*, Fang Hao‡,
T.V. Lakshman‡, Doug Montgomery†, Kotikalapudi Sriram†

* George Mason University | Volgenau School of Engineering

† NIST National Institute of Standards and Technology U.S. Department of Commerce

‡ NOKIA Bell Labs

PROGRAMMABILITY IN SDN

- Data Plane

  OpenFlow provides an open protocol to program the flow table in different switches and routers

- Control Plane

1st generation

- Low-level programming interfaces
- Explicit resource control
- Monolithic control platform

2nd generation

- High-level abstractions
- Extensible packet model
- Modular programming framework

* Pyretic, Kinetic, Frenetic Ocaml, etc.

CHALLENGES IN PROGRAMMABLE MEASUREMENT

- Interfaces between monitoring and other applications
  rule overlapping and conflicts

- Continuous involvement of the controller may be required
  subflow collections

- Associating with forwarding entries in the flow table is neither flexible nor sufficient for supporting various monitoring applications
  forwarding and monitoring applications have different header fields of interest

vPROM PROGRAMMABLE FRAMEWORK

- Runs on instrumented Open vSwitches

  *decouples monitoring from forwarding and support user-defined monitoring capability**

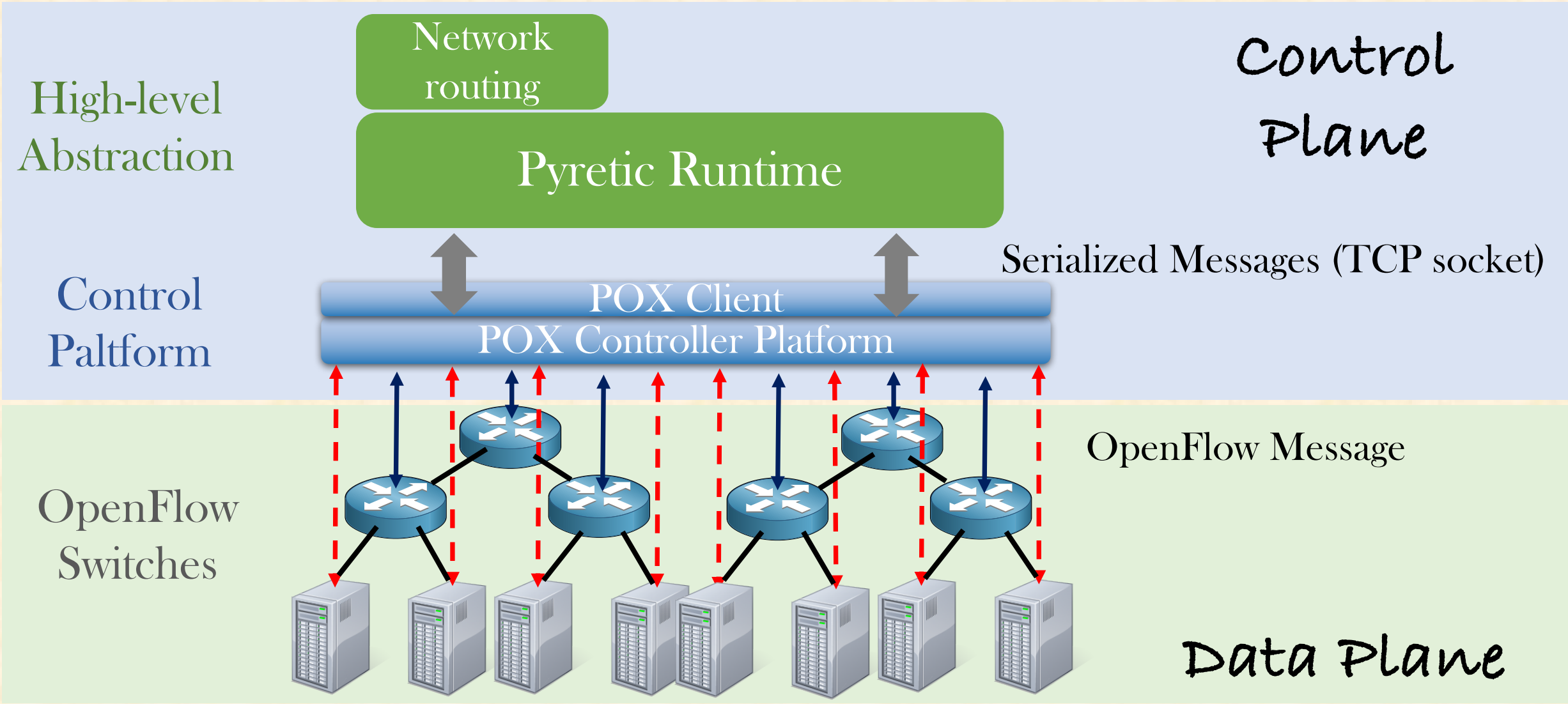
- Extend Pyretic to Pyretic+ to generate different rule sets; Extend OpenFlow to OpenFlow+ to allow applications to setup of monitoring rules

- Client to facilitate the communication between the Pyretic run-time system and the Ryu controller

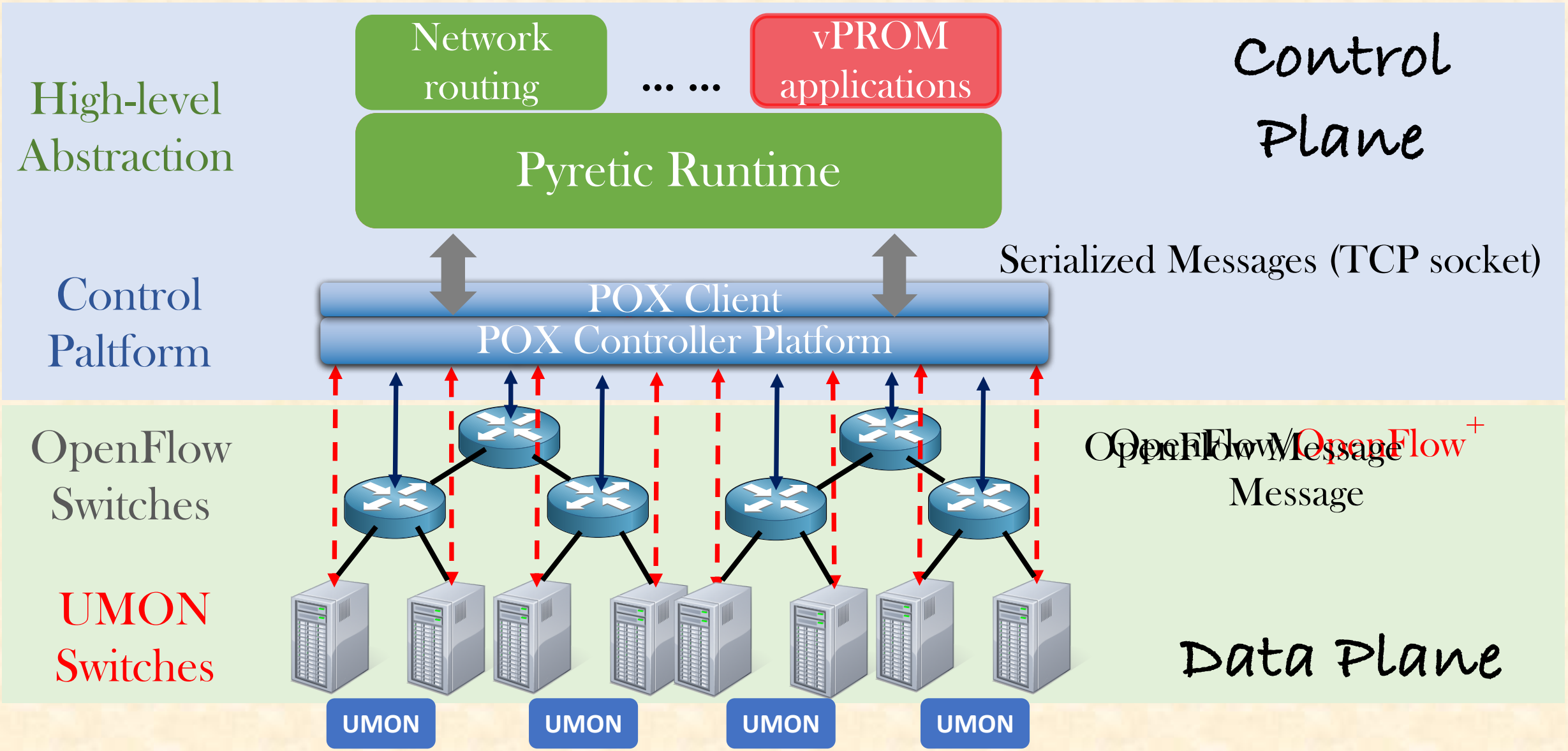
  Ryu supports OpenFlow 1.0 - 1.5 with access to over 40 fields

**UMON: Flexible and fine grained traffic monitoring in open vswitch, CoNEXT 2015*

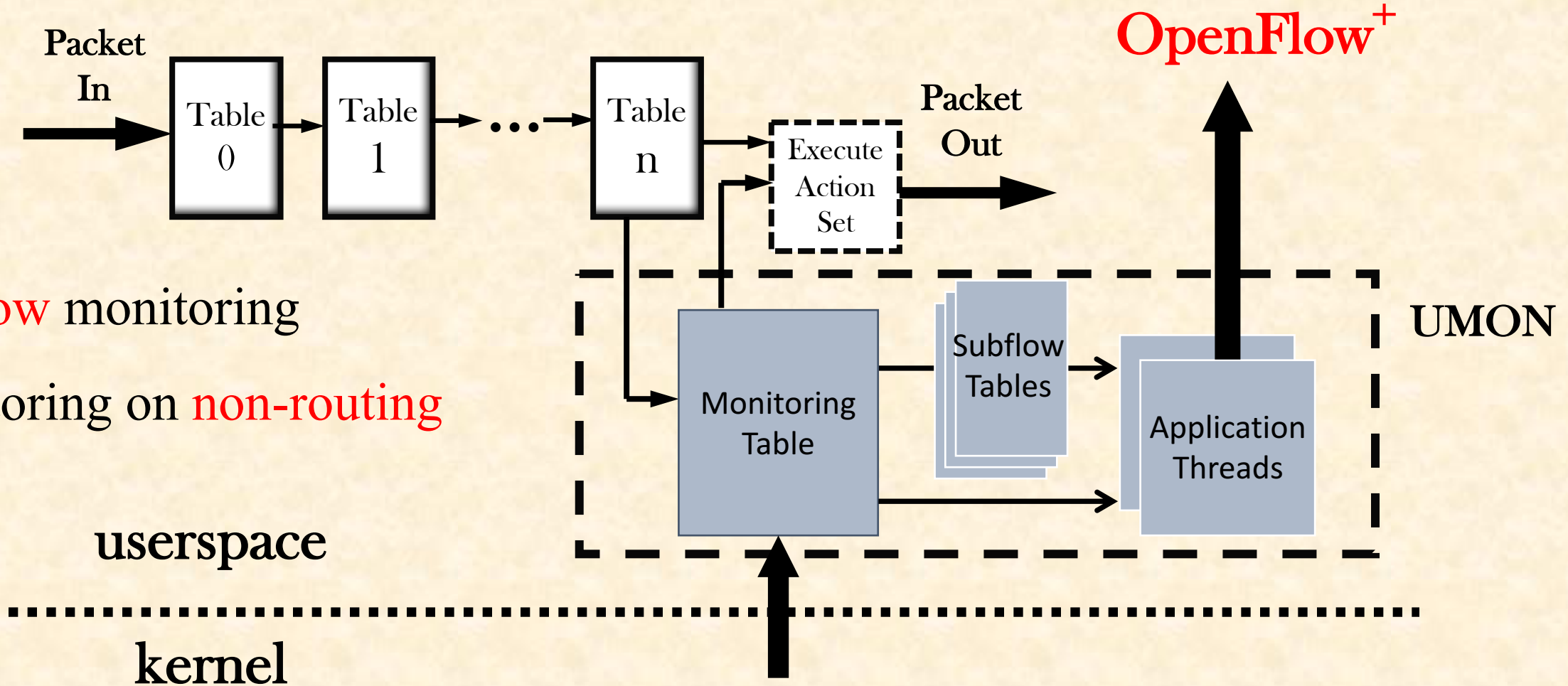
Pyretic architecture



vPROM ARCHITECTURE



UMON RECAP



① Subflow monitoring

② Monitoring on non-routing fields

periodic update monitoring table with kernel flow table info

vPROM COMPONENTS

1) Pyretic^{+} Language

Three query policies are defined to collect statistics of packets of each group

Syntax	Summary
packets(limit=n, group_by=[f1,f2,...])	callbacks on every packet received for up to *n* packets identical on fields *f1,f2,...*
count_packets(interval=t, group_by=[f1,f2,...])	counts every packet received. Callback every *t* seconds to provide count for each group
count_bytes(interval=t, group_by=[f1,f2,...])	counts every byte received. Callback every *t* seconds to provide count for each group

- group_by defines the granularity of subsets of flows; To support TCP flagged packets monitoring, we introduce '*tcpflag*' to the group_by parameter
- new policy 'prtscan_detection' could activate/deactivate local port-scan detector

vPROM COMPONENTS

2) Pyretic^{+} Run-time System

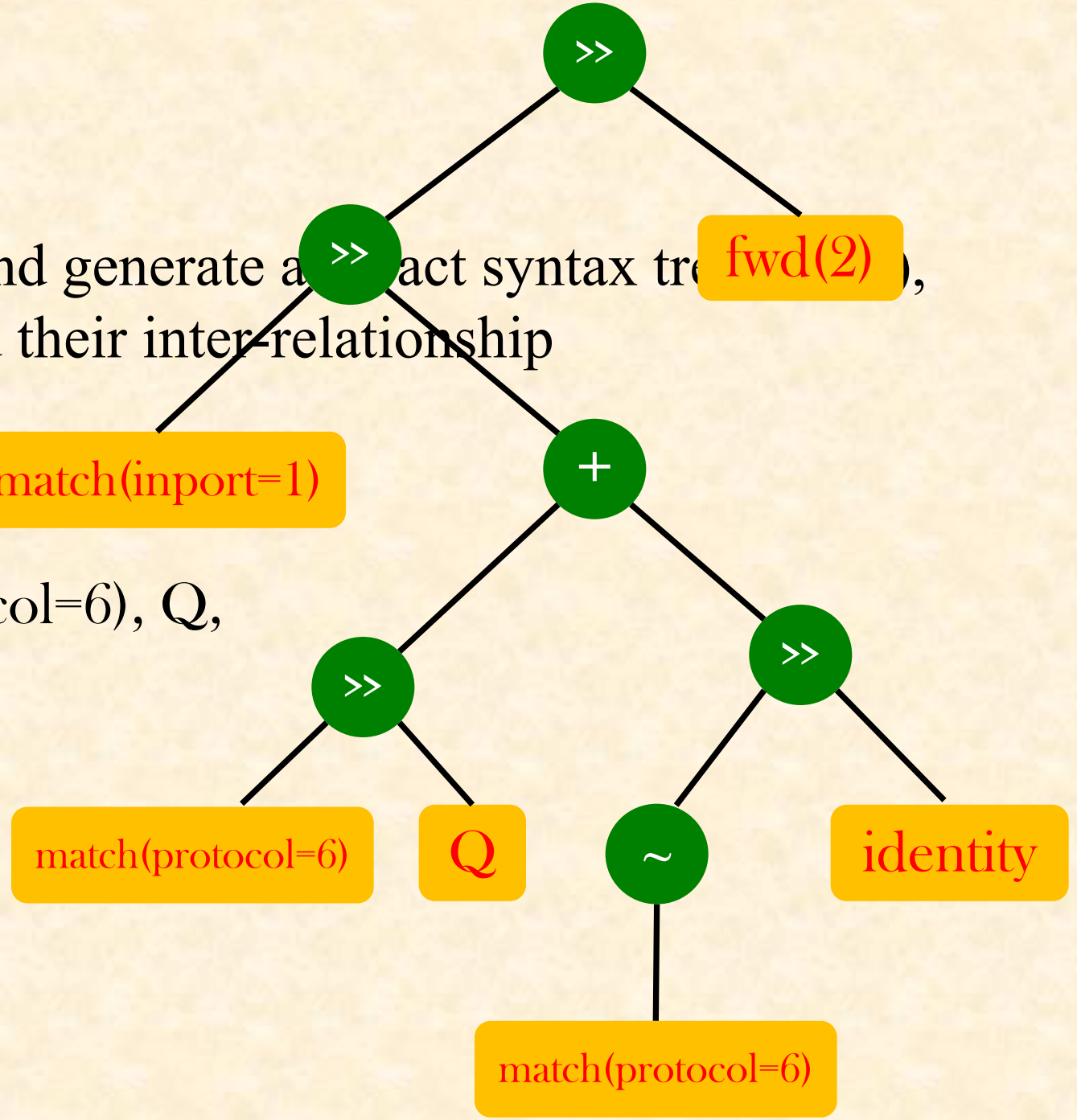
compiles application programs and generate a[illegible]act syntax tre[illegible], which represents the policies and their inter-relationship

e.g.

match(inport=1) >> if_(match(protocol=6), Q, identity) >> fwd(2)

Q = count_packets(interval=t, group_by=['srcip', 'dstip'])

Preorder Traversal



vPROM SYNTAX PARSER

A. Deriving Monitoring AST

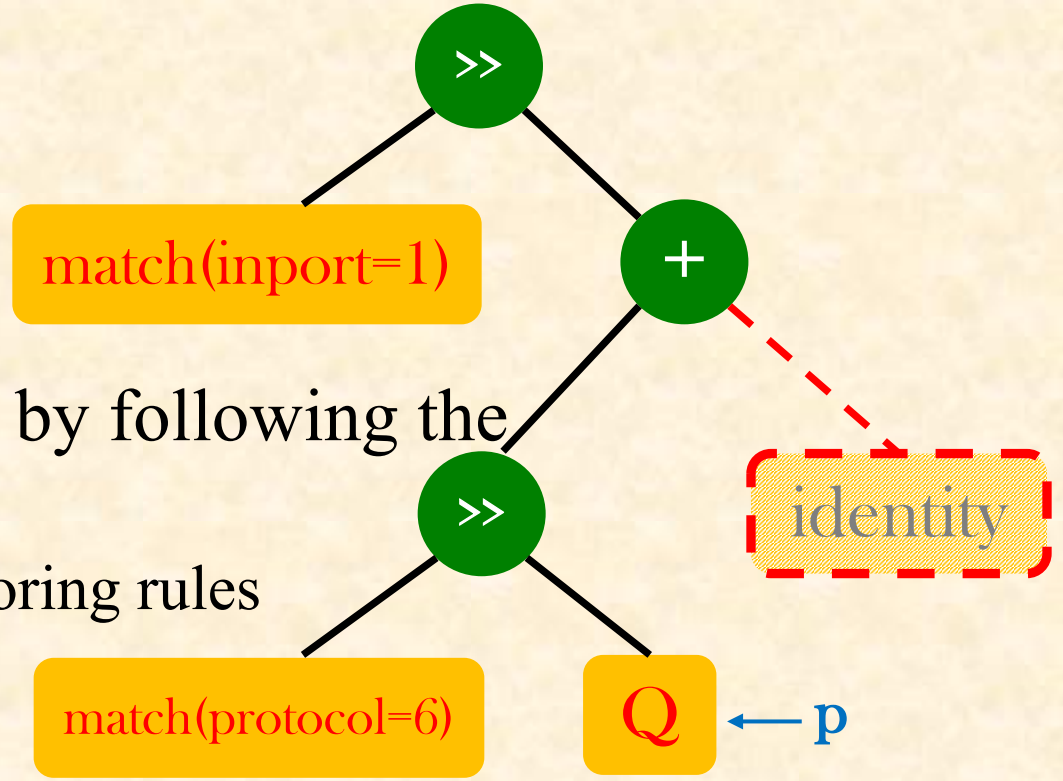
I. identify all the nodes of query policies

II. for each node, find all its anterior nodes by following the parents nodes iteratively
the posterior nodes have no effect on the monitoring rules

III. for operator nodes that ∈ [‘*intersection*’, ‘*sequential*’, ‘*difference*’], all the nodes in its subtrees should be included

IV. monitoring AST is compiled into policy with a stack machine compiler that maintains a first-in, first-out (FIFO) stack

match(inport=1) >> match(protocol=6) >> Q



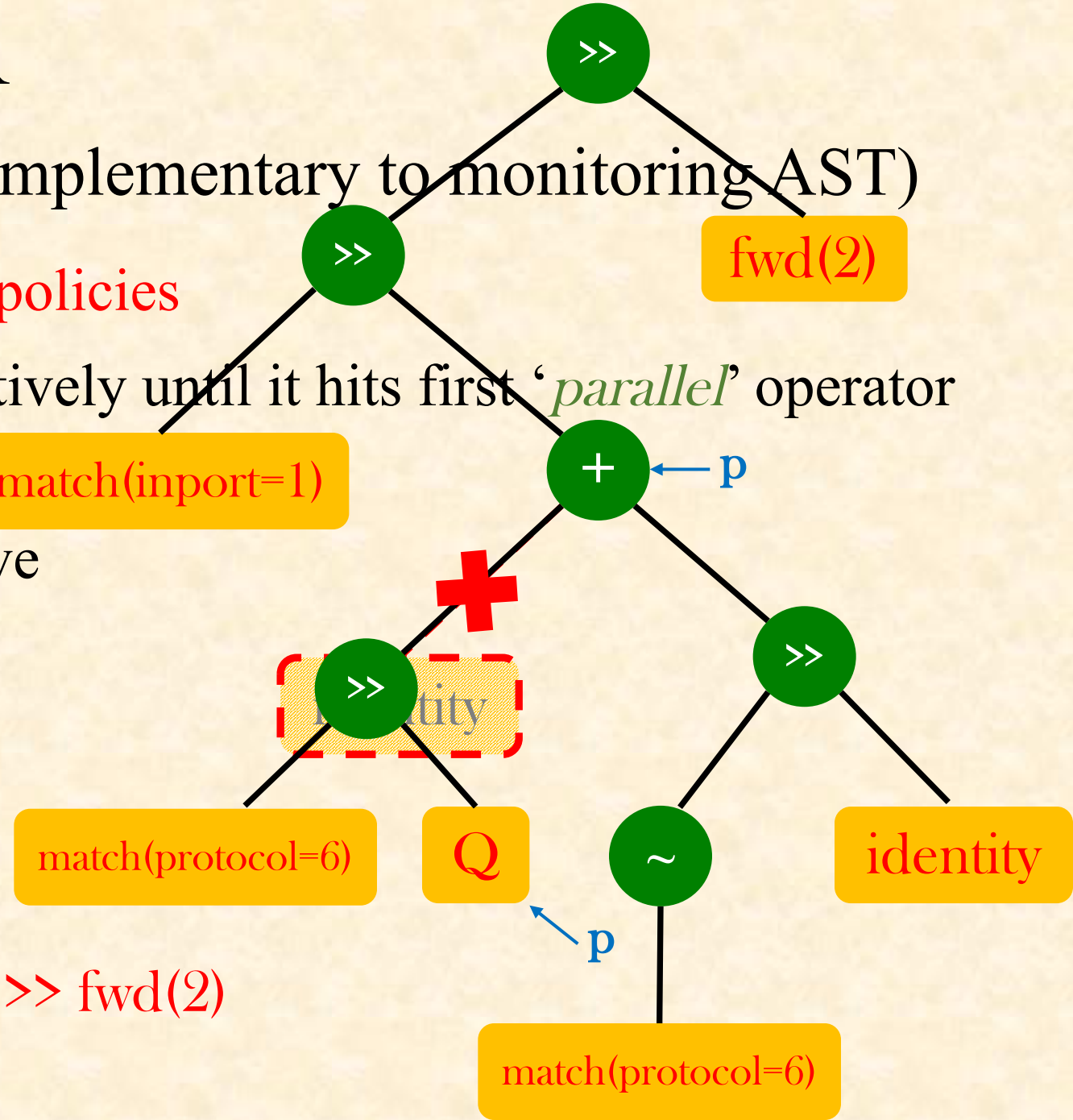
vPROM SYNTAX PARSER

B. Deriving Forwarding AST (complementary to monitoring AST)

I. identify all the nodes of query policies

II. for each node, go upward iteratively until it hits first '*parallel*' operator node

III. prune subtrees that are exclusive to the monitoring AST



match(inport=1) >> identity >> fwd(2)

vPROM COMPONENTS

3) OpenFlow^{+} Protocol

- Monitoring Table Management

ofp message type	ofp commands
OFPT_MONITOR_MOD	OFPMMC_ADD, OFPMMC_MODIFY, OFPMMC_DELETE, OFPMMC_MODIFY_STRICT, OFPMMC_DELETE_STRICT

- Stats Collection

  define a new multi-part message OFPMP_MONITOR_STATS with two types: OFPMR_ALL and OFPMR_EXACT

- Application Thread Management

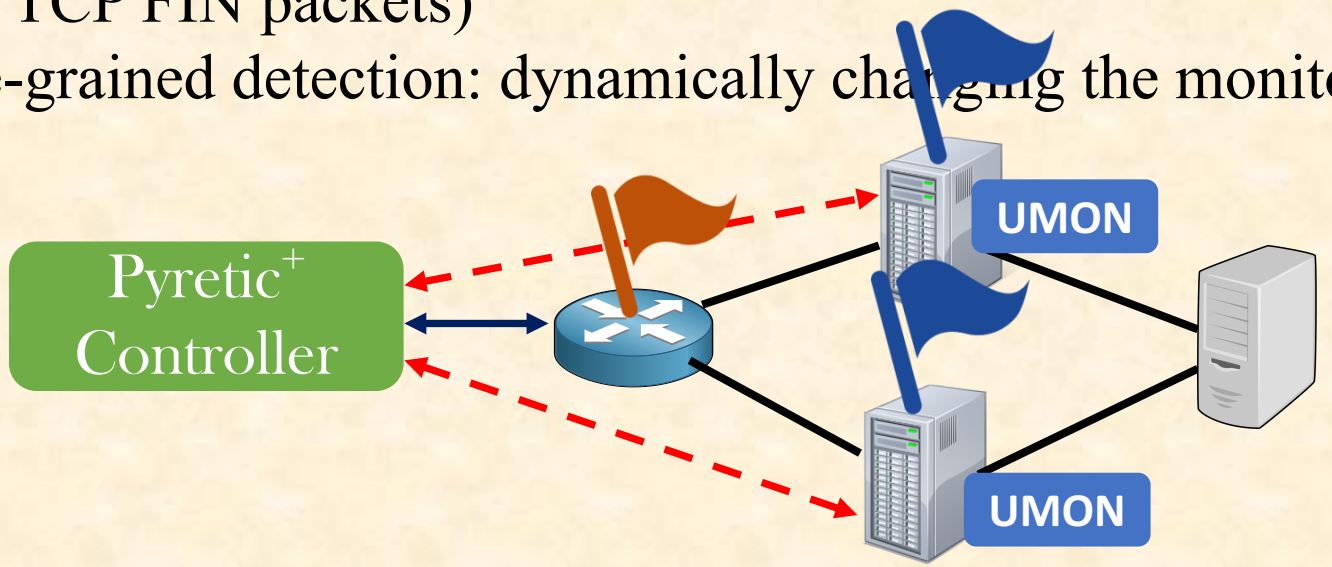
  define new action OFPAT_PRTSCAN_DETECTION for vertical and horizontal scanning detections

4) Ryu client

serialize/deserialize messages to Pyretic backend process; later release of OpenFlow protocol could be easily integrated to the client

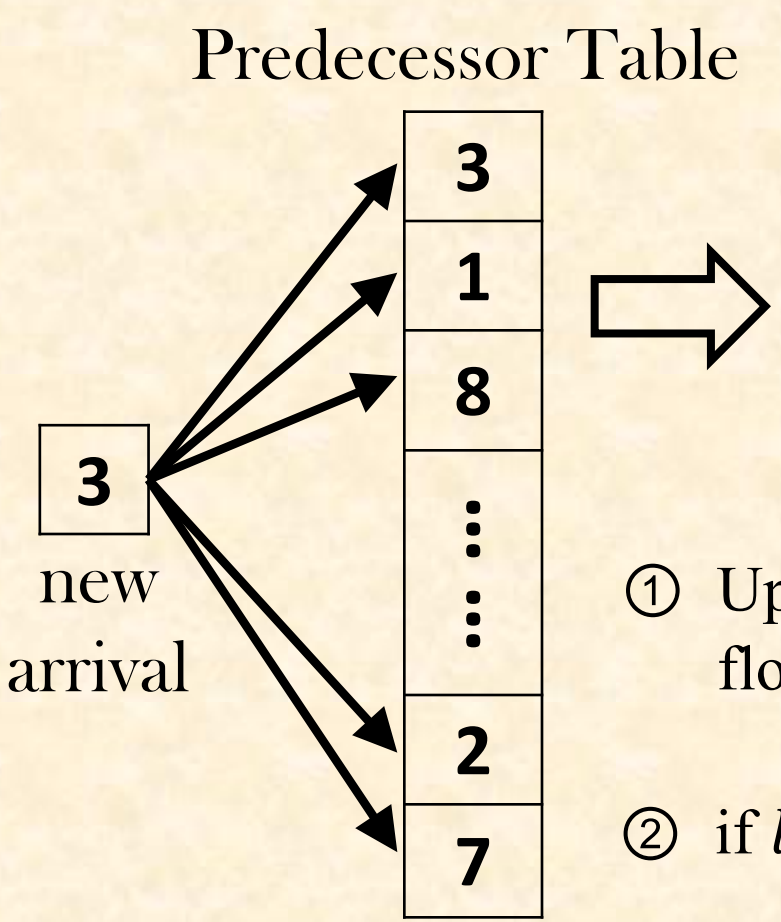
vPROM-GUARD

- vPROM could respond to the ever changing attack vectors dynamically
- vPROM-GUARD detects coarse-grained attack cues by default and switches to fine-grained detection when suspicious activities are detected
  - coarse-grained attack cues: big flows and CUSUM (imbalance between TCP SYN and TCP FIN packets)
  - fine-grained detection: dynamically changing the monitoring granularities



BIG FLOW DETECTION

We employ Coincidence Base Traffic Estimator (CATE*) mechanism



Coincidence Count Table

Flow id	Count
3	2
7	5
...	...

① Upon new arrival, iterate Predecessor Table to count flow appearance as l_f

② if $l_f > 0$ { $f \in$ CCT, update CCT with l_f ; otherwise, insert CCT with l_f

③ big flows have $p_f = \frac{\sqrt{M(N,f)}}{Nf} \geq 0.05$

flow id is defined as tuple of *dstip* and *protocol* in our case

*Fast, memory-efficient traffic estimation by coincidence counting, INFOCOM 2005

CUSUM (CHANGE POINT DETECTION)

TCP {SYN, SYNACK} and TCP {FIN, FINACK, RST} should be balanced in normal network environment, Cumulative Sum Method (CUSUM*) is utilized to detect deviations

Let q_i and p_i be the number of requests and responses in i-th measurement epoch

Then, normalized difference $\tilde{\delta} = \frac{(q_i - p_i)}{P_i}$, where $P_i = \alpha P_{i-1} + (1-\alpha)p_i$

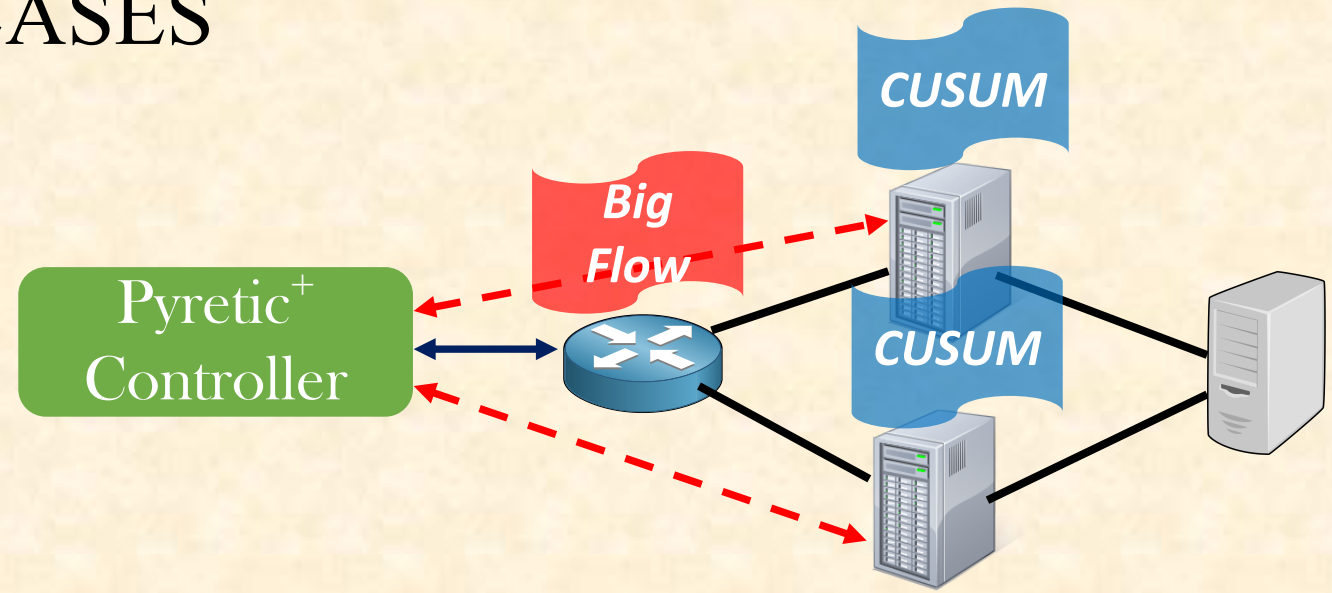
Cumulative sum $S_i = (S_{i-1} + \tilde{\delta} - t)^+$, t is a constant threshold and $(\cdot)^+$ takes positive value or zero

Potential attacks exists if $S_i > T$, with T being a tunable parameter

UMON keeps increasing the monitoring granularity until desired information of the attacker has been obtained

**Change-point monitoring for the detection of DoS attacks, TDSC 2004*

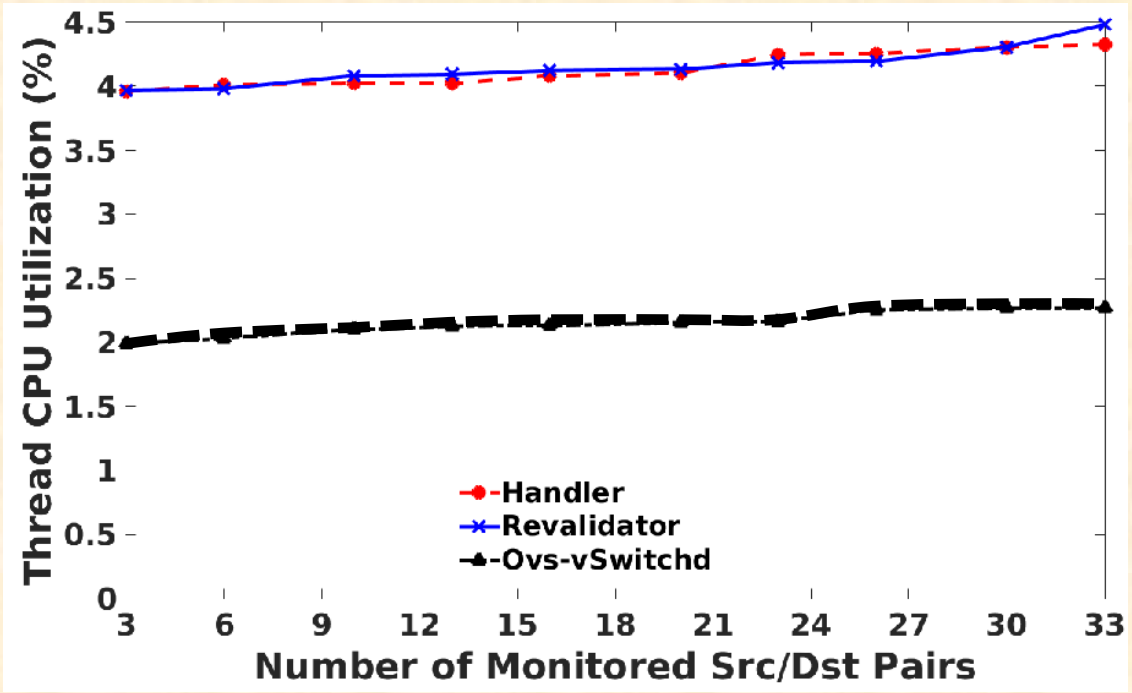
USE CASES



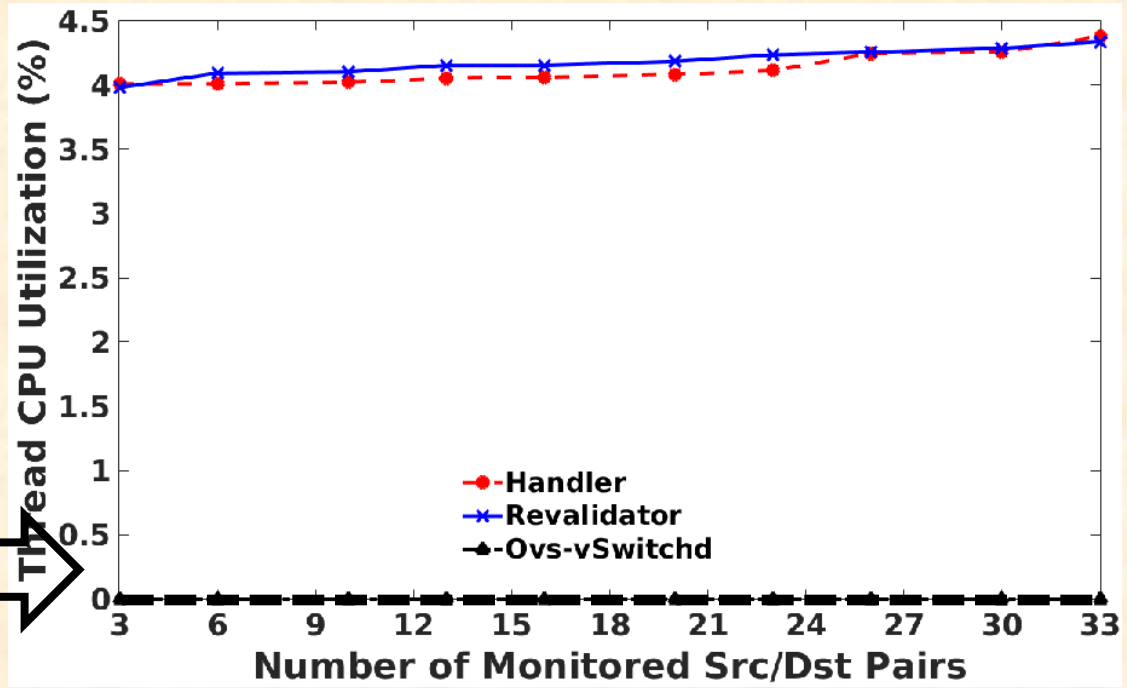
Flag Indicators	Potential Attacks
Big Flow + CUSUM	TCP SYN flooding attack
Big Flow	other types of DDoS attacks
CUSUM	vPROM-GUARD starts collecting subflows and detecting port scanning attacks

EVALUATIONS

Open vSwitch 2.3.2 is instrumented to implement the schemes; Use Tcpreplay to replay data center traces* of ~65 minutes



Pyretic



vPROM

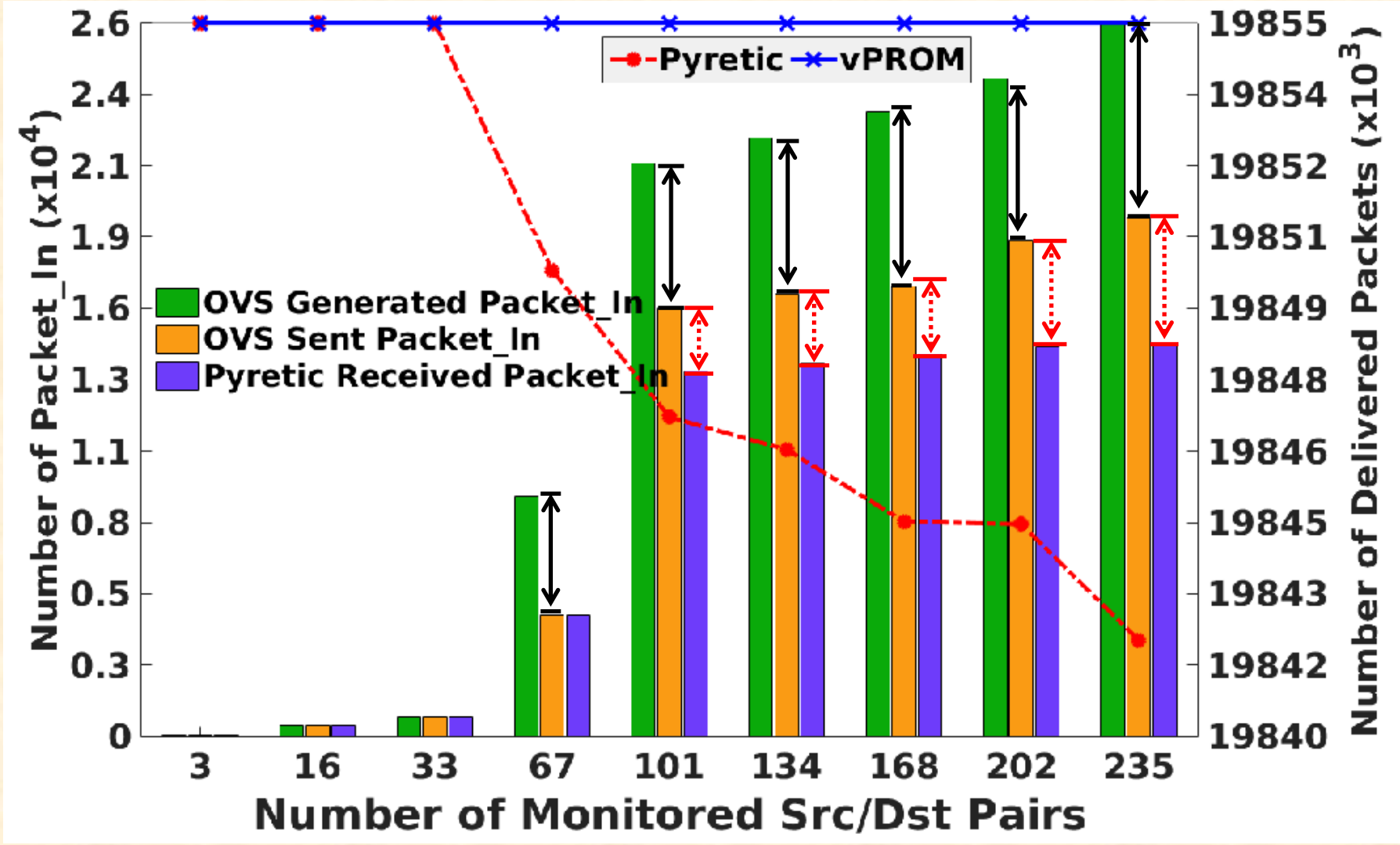
Network traffic characteristics of data centers in the wild, SIGCOMM 2010

EVALUATIONS

CPU stress test by increasing the number of *srcip*/*dstip* pairs

Open vSwitch ofagent overflow

controller event queue overflow



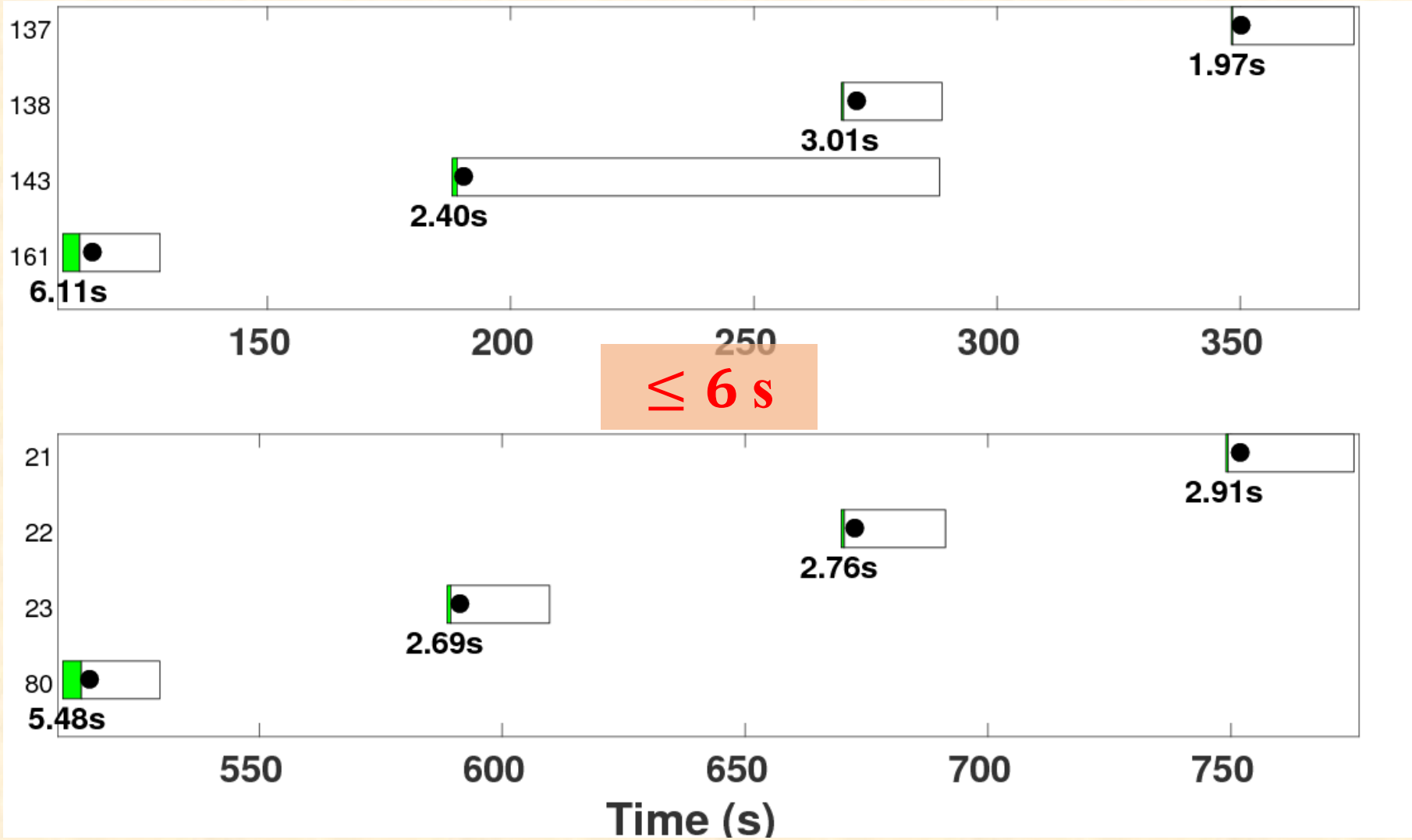
OVS generated > OVS sent > Pyretic received

EVALUATIONS

NUST SEECS trace* containing labeled SYN flood attacks

87.51.34.132

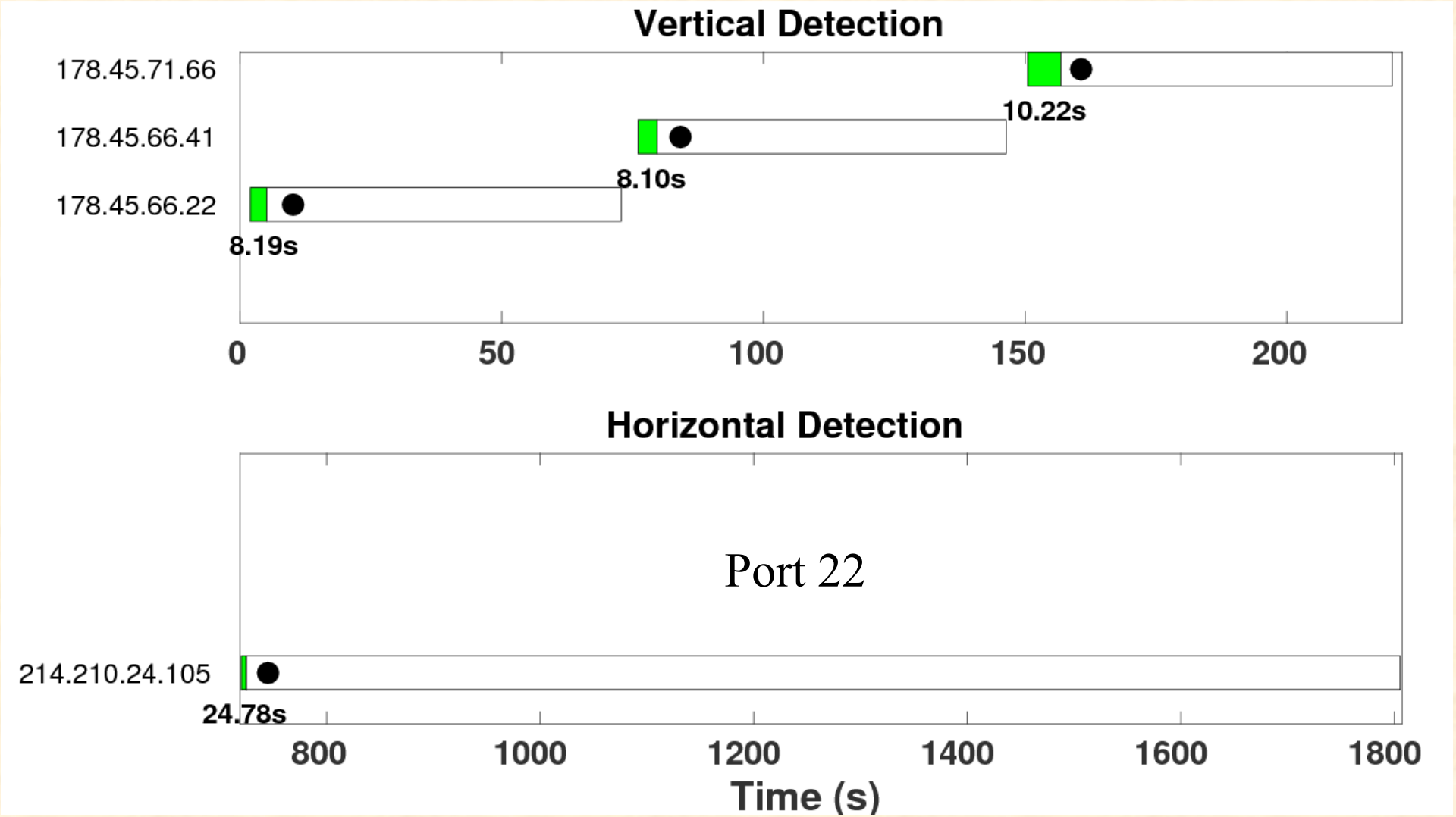
69.63.178.11



*On mitigating sampling-induced accuracy loss in traffic anomaly detection systems, ACM CCR 2010

EVALUATIONS

MAWILAB trace* containing labeled port scanning attacks



Traffic data repository at the wide project, USENIX ATC 2000

CONCLUSIONS

- We design and implement a vSwitch enhanced programmable measurement framework

- We extend the Pyretic platform to generate separate rule sets and corresponding APIs for monitoring and forwarding purposes, respectively

- Pyretic^{+} could detect DDoS and port scanning attacks effectively and efficiently

- More applications could be easily integrated with Pyretic^{+}

Thank You!

awang10@gmu.edu