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

INFORMATION TECHNOLOGY LABORATORY

Establishing the Technical Basis for Trustworthy Networking

Programmable Measurement and Monitoring for Software Defined Networks

Yang Guo (yang.guo@nist.gov)

https://www.nist.gov/software-defined-virtual-networks

Technical Approach

- **Leverage open source networking and emerging AI to develop secure and resilient networks**

- **Develop novel network measurement technologies**
- **Automate network measurement and anomaly detection via network programmability**
- **Leverage AI for autonomous and secure networks**

Instrumenting Open vSwitch with Monitoring Capabilities

Motivation

- **Fine-grained and flexible network traffic monitoring is important for effective network management**
  - Traffic engineering, anomaly detection, network diagnosis, traffic matrix estimation, DDoS detection and mitigation, etc.
- **Scalability has been the main challenge**
  - High switching speed
  - Large number of flows
  - *Solution*: sampling, probabilistic based measurement, hardware enhanced measurement solutions, etc.
- **Open vSwitch (OVS) is a popular software switch**
  - Developed by *Nicira* as an edge switches for Data center
  - slower switching speed, smaller #flows, access to more CPU and memory resources

Motivation

- **Our Idea: instrument software switch to provide *user-defined traffic monitoring***

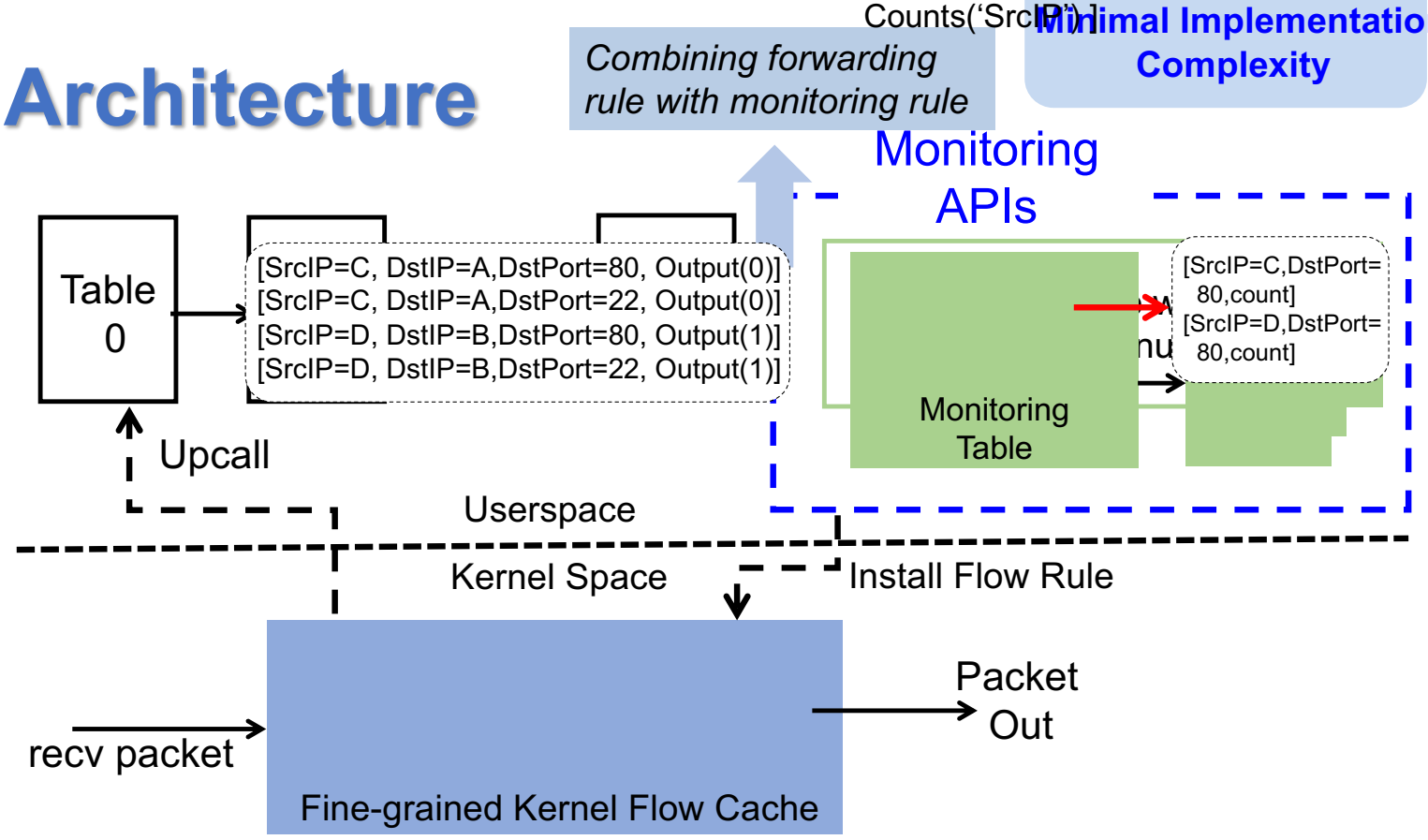
- **Why software switch?**
  - Slower switching speed
  - Access to more resources (both CPU and memory)
  - Sitting at the edge
  - Open source

- **What UMON aims to achieve?**
  - Monitor arbitrary fields
  - Programmable monitoring
  - Allow to push other management functions, such as anomaly detection, to the switches

UMON Architecture

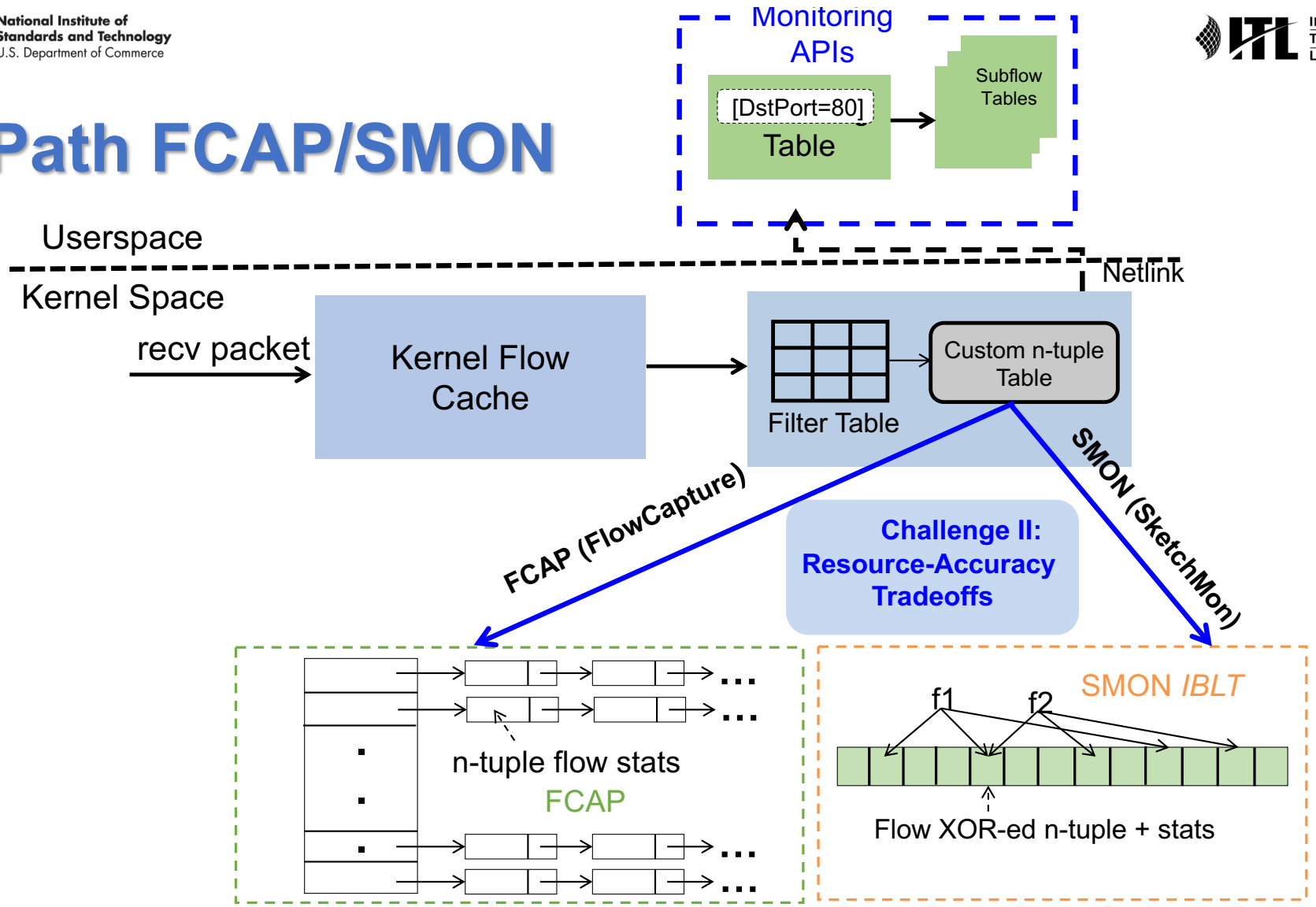
Challenge I: Minimal Implementation Complexity

Combining forwarding rule with monitoring rule

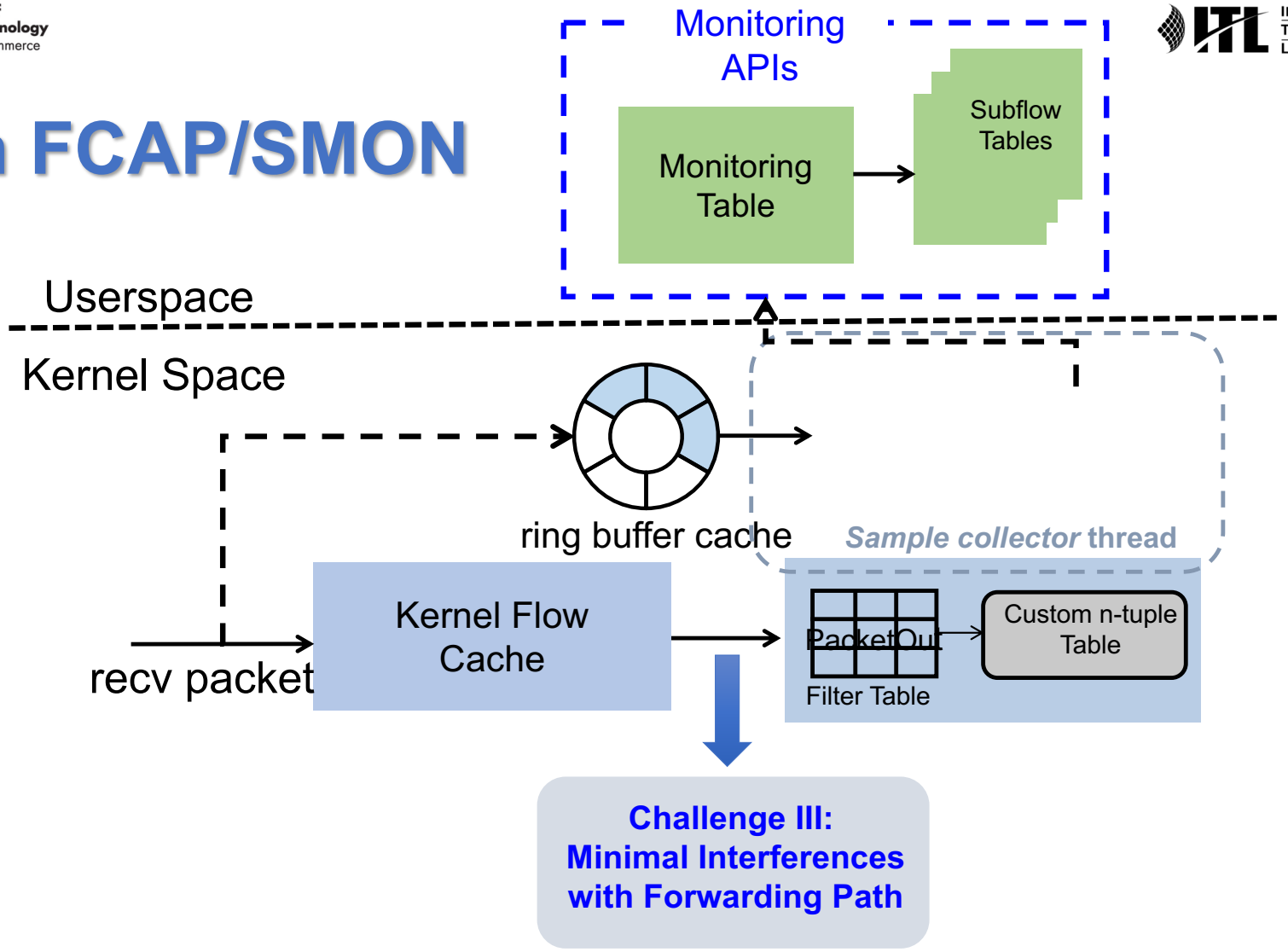


Kernel Flow Cache gets bloated during peak traffic
+
handlers and revalidators get heavily loaded

On-Path FCAP/SMON



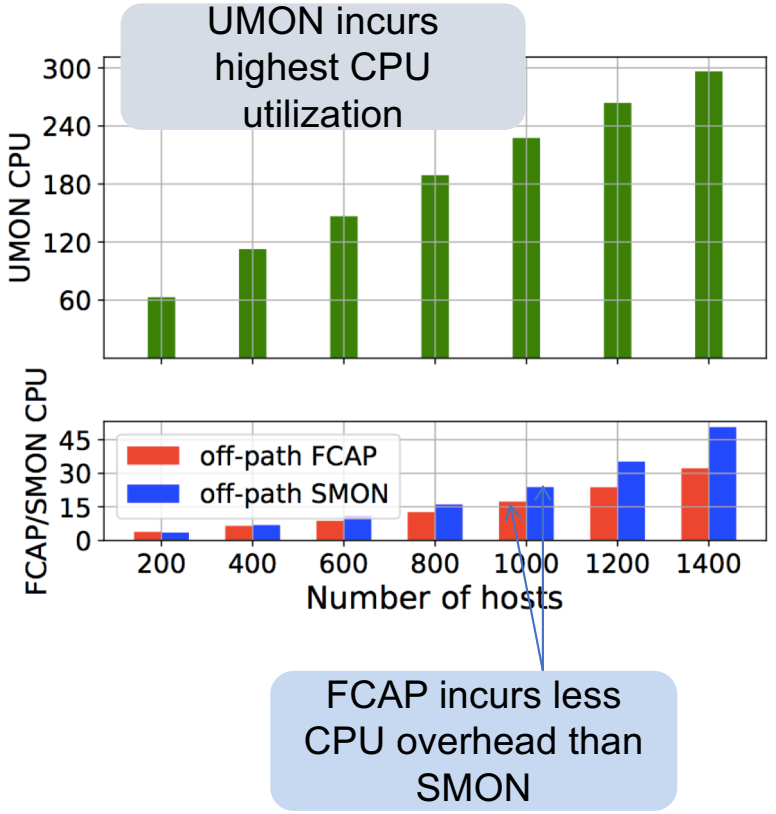
Off-Path FCAP/SMON



Evaluation

- **Testbed Setup**
  - Intel Xeon ***4-Core 3.20GHz CPU***; ***4GB*** memory
  - Host and OVS connected with ***10Gbps*** cables
  - ***Ryu*** SDN controller

- **Total CPU utilization of all related threads:**
  - *2 handlers* + *2 revalidators*
  - *collector* thread in the userspace
  - Custom *sample_collector* thread in the kernel module



Overall Comparison and Insights

Designs	*On-Path*		*Off-Path*		*UMON*
	SMON	*FCAP*	*SMON*	*FCAP*	
CPU Overhead	moderate	low	moderate	low	high
Memory Consumption	low	low	moderate	moderate	high
Measurement Accuracy	high	precise	high	precise	precise
Forwarding Latency	high	high	low	low	high
Implementation Complexity	high	high	high	high	low

- ***UMON:*** ***least*** implementation efforts; ***highest*** CPU overhead; ***highest*** *memory* consumption.
- ***Off-path designs:*** outperform on-path designs in terms of switching performance; higher memory usage.
- Hash table: ***more efficient*** than sketch, ***lower*** computational cost.

vPROM: vSwitch Enhanced Programmable Measurement

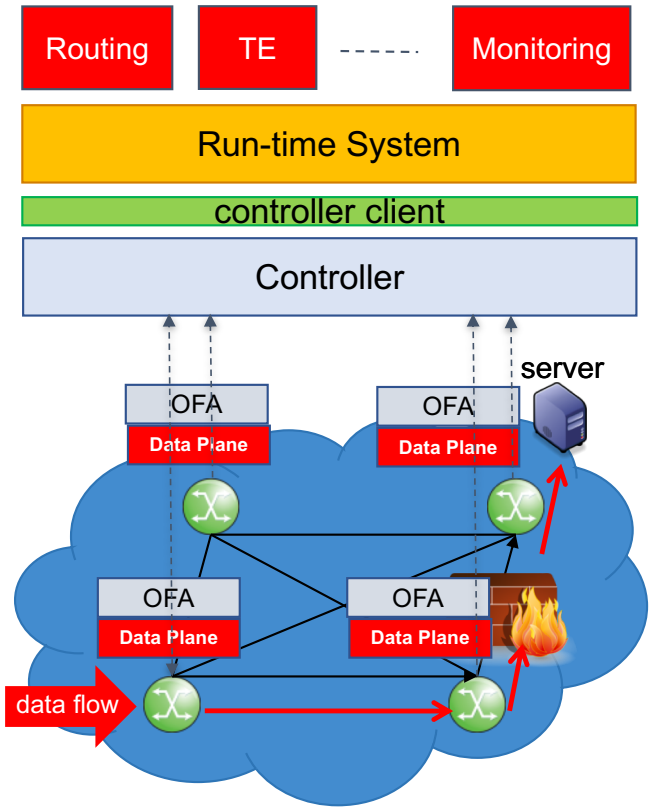
Software Defined Network Programmability

- Program the network with perception that underlying network is a single device

- High-level languages
  - e.g., *Frenetic*, *Pyretic*, *Ox*
  - High-level, unified abstractions
  - Compositional semantics

- Run-time system
  - Handles module interactions
  - Deals with asynchronous behavior

- Controller client
  - Shim between runtime system and controller



SDN based Programmable Measurement

- **Network measurement controlled and managed by a program written in networking program language**

- **Benefits:**
  - Automate the measurement process
  - Utilize software switches as measurement points across the networks
  - Acquire only necessary statistics
    - dynamically adjust what/where to measure
    - minimize resource usage

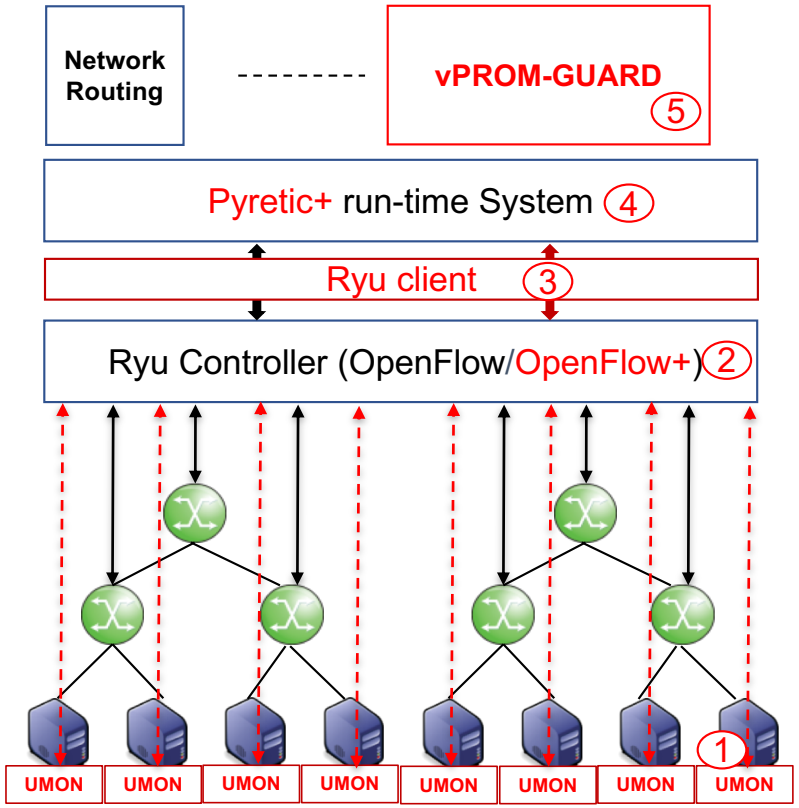
vPROM: vSwtich enhanced Programmable Measurement

- **Issues with vanilla SDN based programmable measurement**
  - Interaction between forwarding, monitoring, and other applications is complex
  - SDN controller is involved too frequently
  - Limited measurement resources, e.g., TCAM, at physical SDN switches
  - Packet and byte counts associated with flow forwarding entries are neither flexible nor sufficient

- **Key Ideas:** ***leverage instrumented UMON switch at network edge***
  - Extend *OpenFlow, run-time system*, and *network programming language* to have a unified system

vPROM Architecture

1. UMON: instrumented Open vSwitch
2. OpenFlow+: extended OpenFlow protocol support UMON
3. Ryu client
4. Pyretic+: extended Pyretic runtime system
5. vPROM-GUARD: DDoS and port detection vPROM application

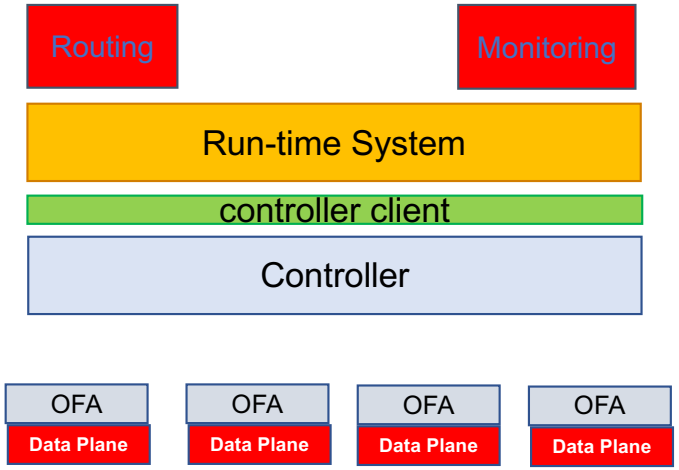


vPROM Example

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

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

match(ethtype=0X0800) & match(protocol = 6) >> Q
```



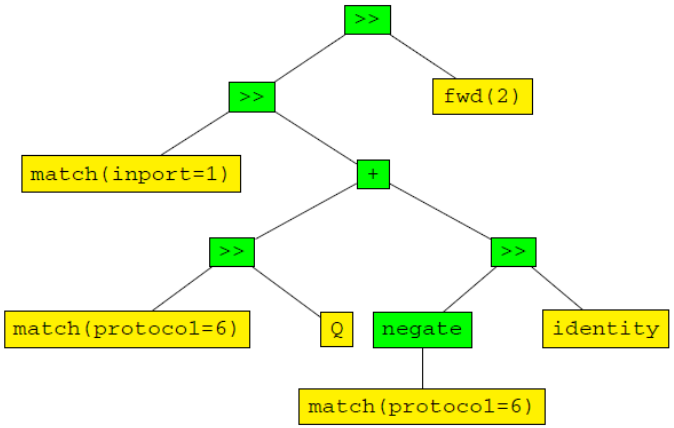

```
in_port=1,priority=60000,actions=output:2
priority=59999,actions=drop
```

tcp,actions=subflow_collection:nw_src=0.0.0.0/32,nw_dst=0.0.0.0/32

Pyretic Run-time System

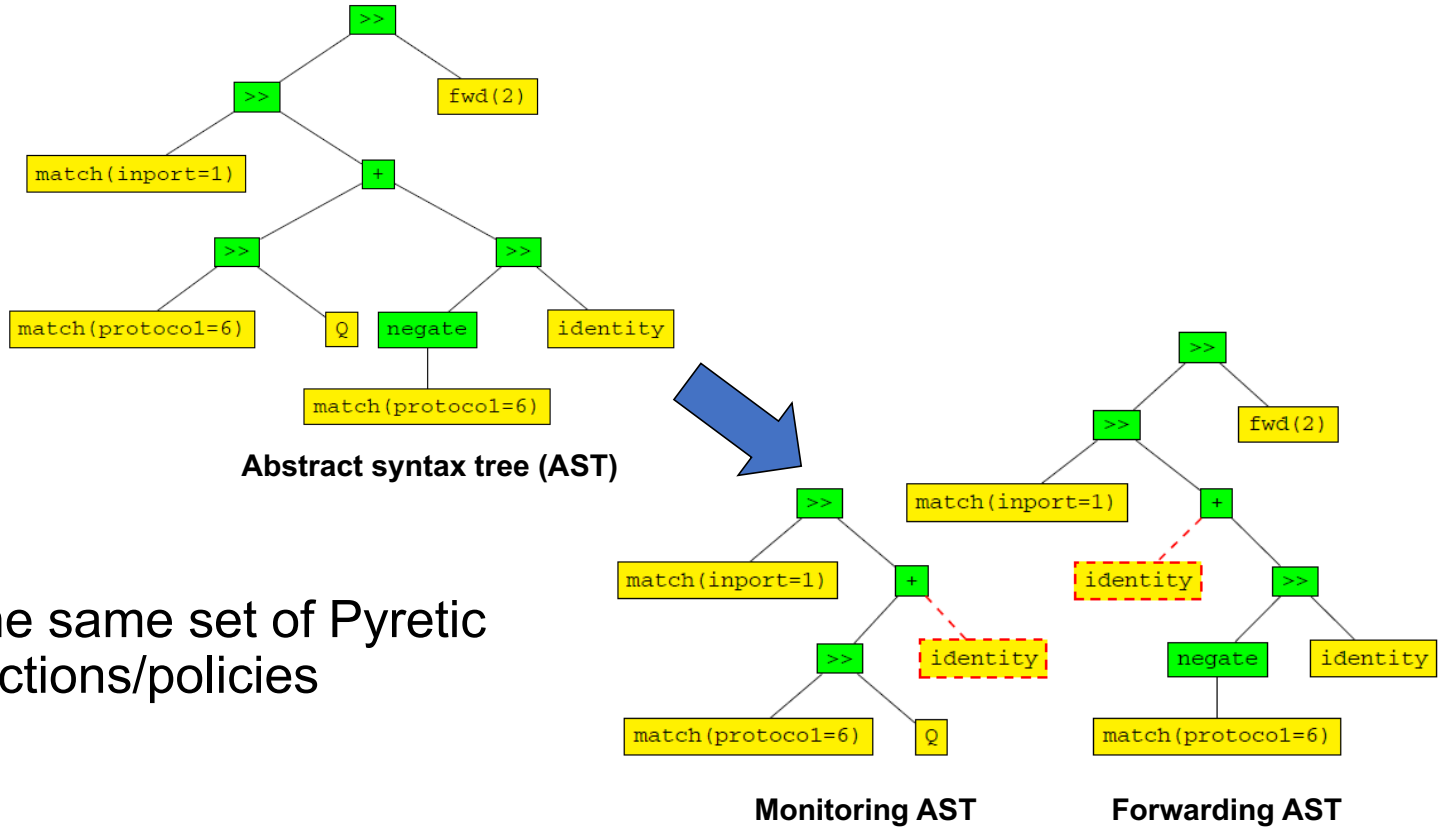
match(inport=1) >> if_(match(protocol=6), Q, identity) >> fwd(2)

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



Abstract syntax tree (AST)

Pyretic+ Run-time System



Goal: using the same set of Pyretic abstractions/policies

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,…]	Count 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**

OpenFlow+ Protocol

- **Monitoring Table Management**

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

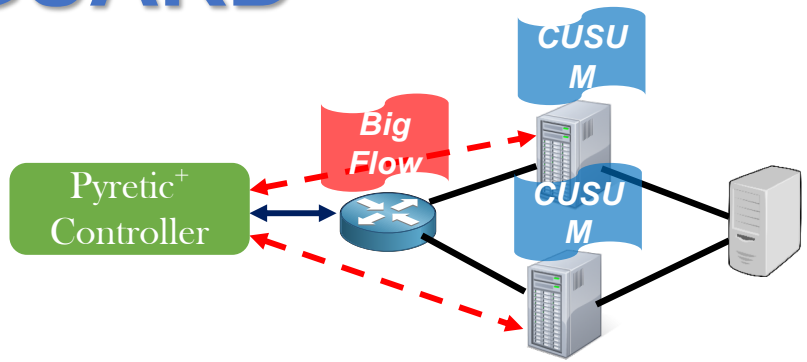
- **Stats Collection**
  - Define a new multi-part message OFPMP_MONITOR_STATS with two types: OFPMR_ALL and OFPMR_EXACT

- **Application Thread Management**
  - Define new action OFPAT_PRTSCAN_DETECTION for vertical and horizontal scanning detections

vPROM-GUARD: DDoS and Port Scan Detection

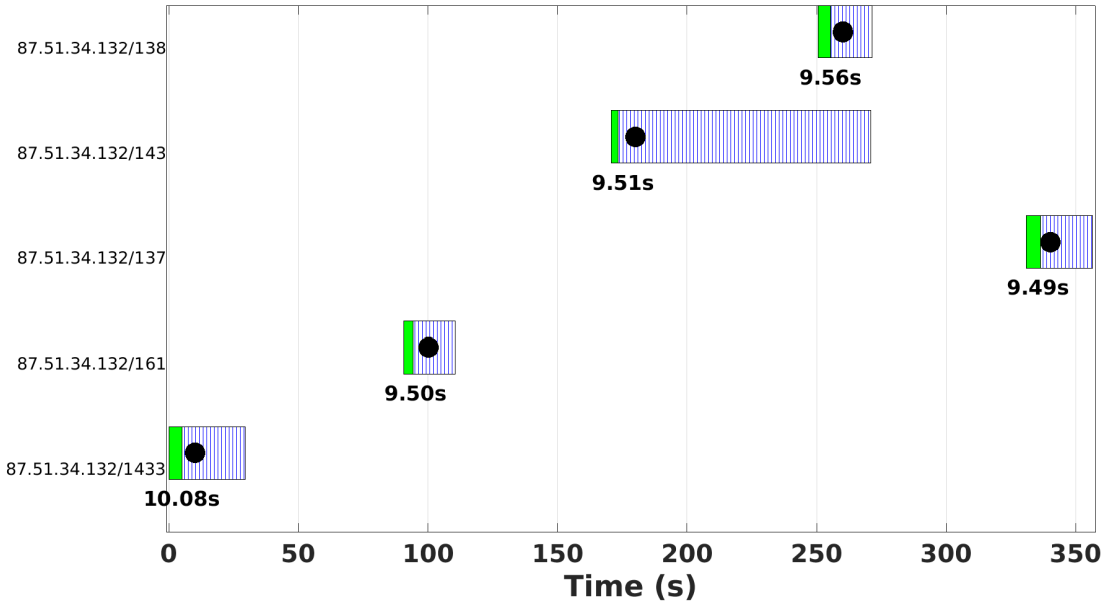
- Anomaly detection often requires low level feature, e.g., packet-level or micro-flow, measurement at line rate – challenging
- vPROM-GUARD
  - monitor *attack cues* at *coarse level* when in normal operations
    - Monitoring TCP signaling packets – TCP {SYN, SYN/ACK} and {SYN, FIN} are request-response pairs that should be balanced
    - Using Cumulative Sum Method to detect the deviation
  - when suspicious activities are detected, switch to a *full-blown fine grained network monitoring* and start *DDoS and port-scan detection* at both edge UMON vSwitches and at the central vPROM application
- Benefits:
  - Only alerted hosts conduct fine grained measurement
  - Local detection at edge mitigates the burden at central detection
  - False alarms are more tolerable

vPROM-GUARD



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

vPROM-GUARD SYN Flood Attack Detection



- **Vertical line**: change-point monitoring issues a potential attack warning
- **Dot**: vPROM-GUARD actually detects the attack.
- ~10 seconds (2 polling periods)

Conclusions

- **Propose, design and prototype vSwitch Enhanced Programmable Measurement framework**
  - Instrument the edge software switch for measurement and anomaly detection
  - Automate the measurement process
  - Acquire only necessary statistics: minimize resource usage

- **Related work**
  - Network programmability (run-time system and network programming language) has been studied extensively
    - Frenetic, NetKAT, SDX, Kinetic, etc.
  - Flow-rule based measurement using physical SDN switches
    - Limited TCAM
  - Programmed measurement
    - Path query, intentional monitoring
    - Constant controller involvement

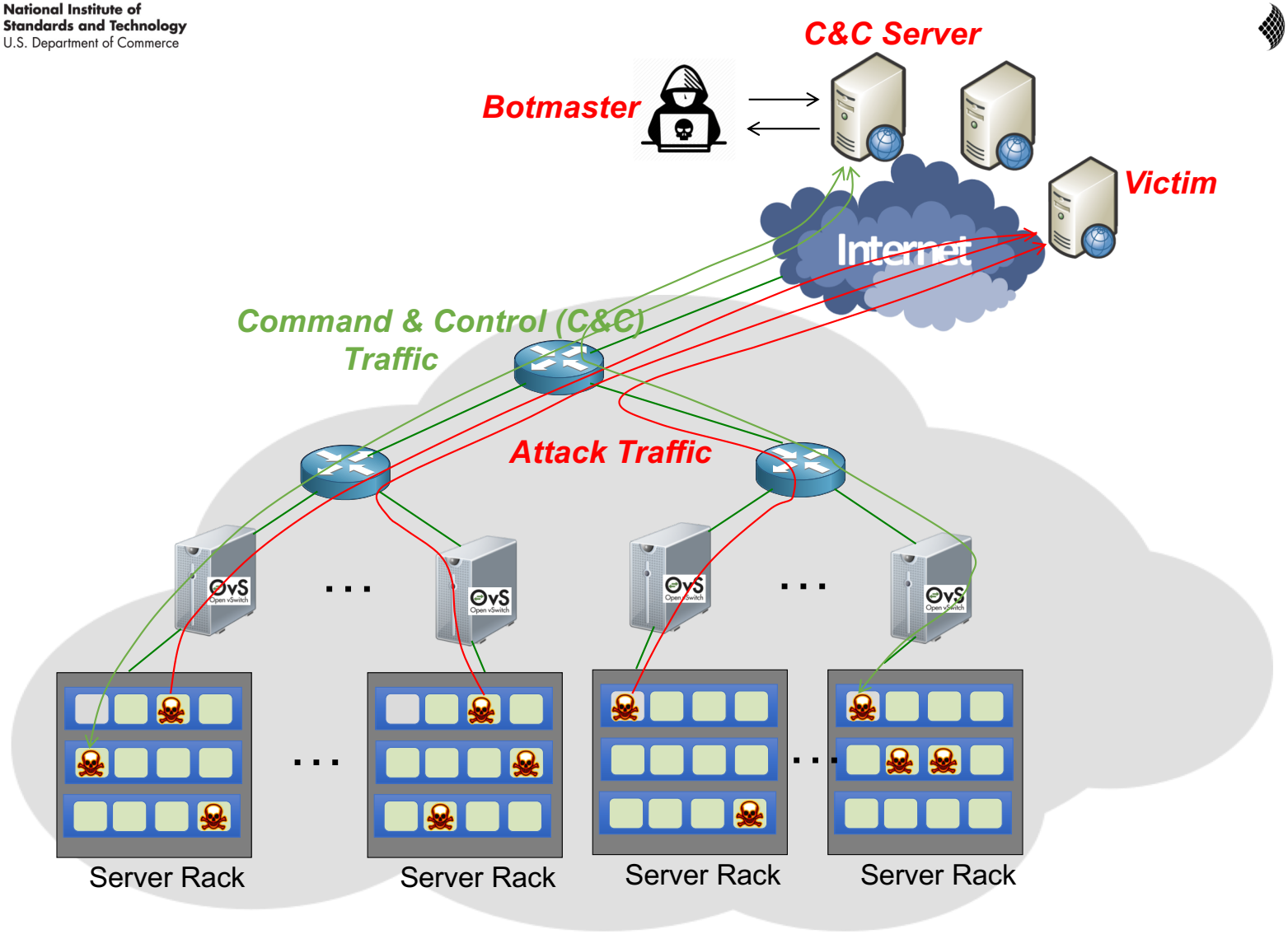
Accomplishment

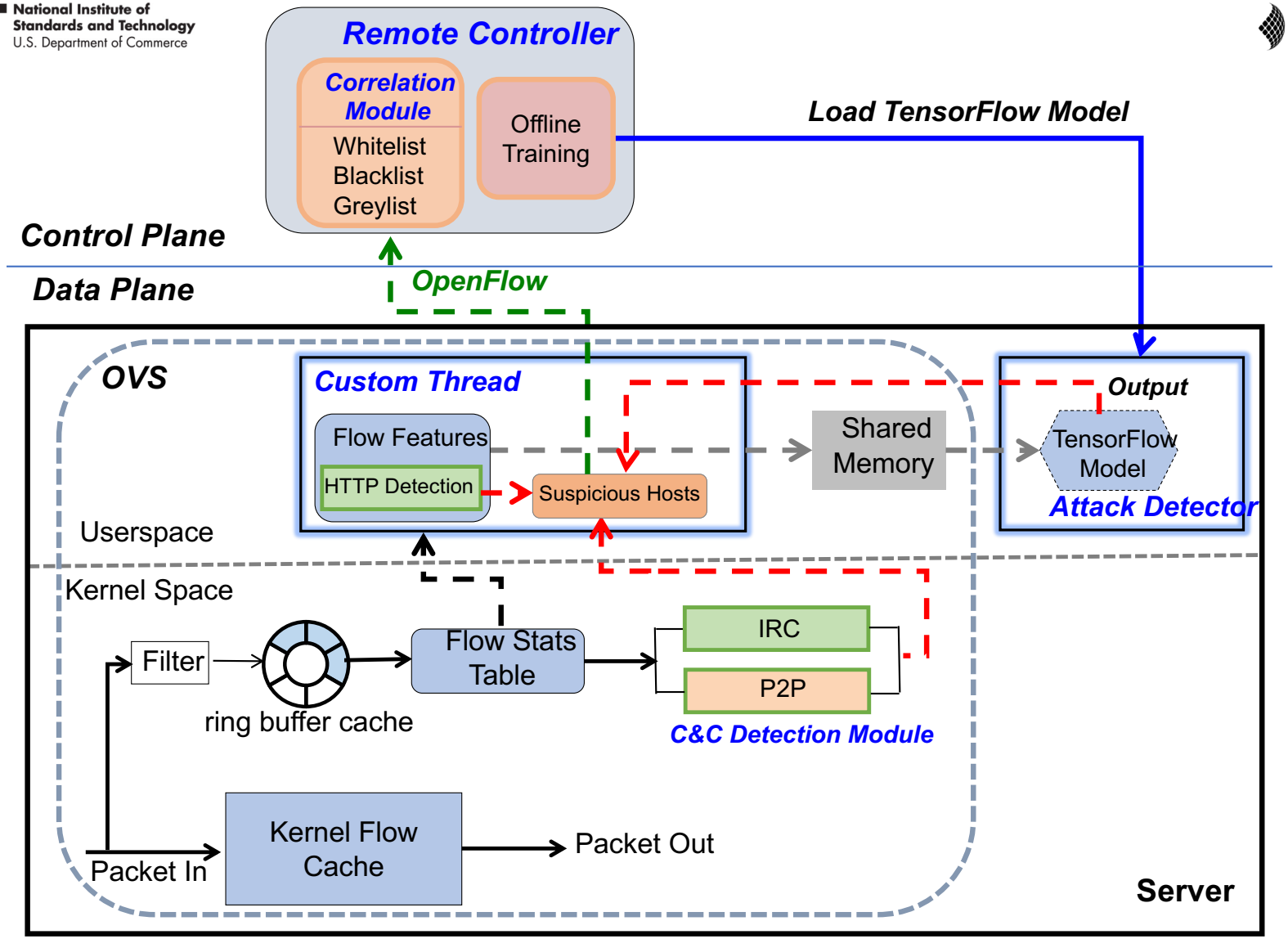
- **Publications:**
  - A. Wang, Y. Guo, F. Hao, T. Lakshman, S. Chen, "*UMON: Flexible and Fine Grained Traffic Monitoring in Open vSwitch*", ACM CoNEXT, 2015.
  - A. Wang, Y. Guo, S. Chen, F. Hao, T. Lakshman, D. Montgomery, K. Sriram, "*vPROM: VSwitch enhanced programmable measurement in SDN*", IEEE ICNP 2017.
  - Z. Zha, A. Wang, Y. Guo, D. Montgomery, S. Chen, "*Instrumenting Open vSwitch with Monitoring Capabilities: Design and Challenges*", ACM SOSR 2018.
  - Y. Guo, D. Montgomery, Programmable Measurement Framework in SDN, Workshop on SoSSDN 2016.
  - Yang Guo, Alexander L. Stolyar, Anwar Walid, "*Online VM Auto-Scaling Algorithms for Application Hosting in a Cloud*", IEEE Transactions on Cloud Computing (TCC), Accepted.

- **Open Source Code**
  - Instrumented Open vSwitch source code, *https://github.com/iOVS/iOVS*

Ongoing Work and Future Direction (I)

- **Machine Learning Based Network Anomaly Detection**
  - Available AI based network anomaly detection suffers from multiple scaling issues
  - Aim to develop a Distributed ML based anomaly detection framework by leveraging vPROM framework
  - Conduct distributed monitoring and distributed ML based anomaly detection at the network edge as well as at a central location


C&C Server
Botmaster
Victim
Internet
Command & Control (C&C) Traffic
Attack Traffic
OvS
Server Rack
Server Rack
Server Rack
Server Rack


Remote Controller
Correlation Module
Whitelist
Blacklist
Greylist
Offline Training
Load TensorFlow Model
Control Plane
Data Plane
OpenFlow
OVS
Custom Thread
Flow Features
HTTP Detection
Suspicious Hosts
Shared Memory
Output
TensorFlow Model
Attack Detector
Userspace
Kernel Space
Filter
ring buffer cache
Flow Stats Table
IRC
P2P
C&C Detection Module
Kernel Flow Cache
Packet In
Packet Out
Server

Ongoing Work and Future Direction (II)

- **High-Speed Data Plane Measurement using Programmable Switches**
  - Programmable switch is designed to be programmable using high-level domain specific language, e.g. P4
    - Implement new functions at line speed
    - A uniform pipeline of programmable stages to process packet headers in rapid succession
    - Fast rollout of new network protocols
  - Challenges:
    - stringent time budget per pipeline stage (around 1ns)
    - limited amount of memory per pipeline stage
  - Coincidence Counting based Large Flow Detection in Data Plane

Questions and Discussion

- **For more information:**
  - Software Defined Virtual Networks
    - https://www.nist.gov/software-defined-virtual-networks
  - Advanced Network Technologies Division.
    - https://www.nist.gov/itl/antd
  - Information Technology Laboratory
    - https://www.nist.gov/itl